12-27-95 98:01

November 1995 • NREL/TP-442-6916

Combined Experiment Phase II Data Characterization

Mark S. Miller, Derek E. Shipley,
Teresa S. Young, Michael C. Robinson,
and Marvin W. Luttges
University of Colorado
Boulder, Colorado

David A. Simms
National Renewable Energy Laboratory
Golden, Colorado

1617 Cole Boulevard
Golden, Colorado 80401-3393
A national laboratory of the U.S. Department of Energy
Managed by Midwest Research Institute
for the U.S. Department of Energy
Under Contract No. DE-AC36-83CH10093

NREL/TP-442-6916 • UC Category: 1211 • DE96000473

Combined Experiment Phase II Data Characterization

Mark S. Miller, Derek E. Shipley,
Teresa S. Young, Michael C. Robinson,
and Marvin W. Luttges
University of Colorado
Boulder, Colorado

David A. Simms
National Renewable Energy Laboratory
Golden, Colorado

NREL Technical Monitor:
David A. Simms

National Renewable Energy Laboratory
1617 Cole Boulevard
Golden, Colorado 80401-3393
A national laboratory of the U.S. Department of Energy
Managed by Midwest Research Institute
for the U.S. Department of Energy
under contract No. DE-AC36-83CH10093

Prepared under Subcontract
No. XAO-2-12236-01-103983

November 1995

NOTICE

Available to DOE and DOE contractors from:
Office of Scientific and Technical Information (OSTI)
P.O. Box 62
Oak Ridge, TN 37831
Prices available by calling (615) 576-8401

Available to the public from:
National Technical Information Service (NTIS)
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
(703) 487-4650

COMBINED EXPERIMENT PHASE II DATA CHARACTERIZATION

Mark S. Miller, Derek E. Shipley, Teresa S. Young, Michael C. Robinson, Marvin W. Luttges
Department of Aerospace Engineering Sciences
University of Colorado, Boulder, CO 80309
and
David A. Simms
National Renewable Energy Laboratory, Golden, CO 80401

ABSTRACT

The National Renewable Energy Laboratory's "Combined Experiment" has yielded a large quantity of experimental data on the operation of a downwind horizontal axis wind turbine under field conditions. To fully utilize this valuable resource and identify particular episodes of interest, a number of databases were created that characterize individual data events and rotational cycles over a wide range of parameters. Each of the 59 five-minute data episodes collected during Phase II of the Combined Experiment have been characterized by the mean, minimum, maximum, and standard deviation of all data channels, except the blade surface pressures. Inflow condition, aerodynamic force coefficient, and minimum leading edge pressure coefficient databases have also been established, characterizing each of nearly 21,000 blade rotational cycles. In addition, a number of tools have been developed for searching these databases for particular episodes of interest. Due to their extensive size, only a portion of the episode characterization databases are included in an appendix, and examples of the cycle characterization databases are given. The search tools are discussed and the FORTRAN or C code for each is included in appendices.

INTRODUCTION

The National Renewable Energy Laboratory's (NREL's) "Combined Experiment" was established to provide a comprehensive test program focused on understanding the fundamental aerodynamics of rotating horizontal axis wind turbines (HAWTs). This fundamental aerodynamic understanding includes examining the difference between airfoil performance on an operational turbine and two-dimensional wind tunnel tests, and determining the effects of the aerodynamics upon the turbine structure. To this end, NREL fully instrumented a turbine to obtain a wide range of inflow, aerodynamic, structural, and operational data. Phase II of the Combined Experiment has been completed using untwisted, untapered blades. Nearly 300 minutes of high frequency data has been collected.

Because of the large amount of data collected during the experiment, characterizing the data is necessary to aid in the location of pertinent data. Two primary methods of characterization have been employed. First, the mean and standard deviation values for all data channels were calculated for every data episode. Each episode is five minutes in length. The statistical summary values aid researchers in determining what data is available over large periods of time. For instance, researchers using ADAMS/WT (Automatic Dynamic Analysis of Mechanical Systems/Wind Turbine) might find data episode characterizing useful. However, to understand factors such as the turbine's aerodynamics, a second, more in-depth method of characterizing the data is required. This method examines the data on an individual cycle basis. A cycle is defined as a single rotation of the instrumented blade from 0° azimuth, the instrumented blade straight up, to 360° azimuth. This second method characterized each rotational cycle by inflow conditions, aerodynamic force coefficients, and minimum leading edge pressure coefficient.

TEST SETUP

NREL's Combined Experiment horizontal axis wind turbine is a 10.1 meter diameter, three-bladed machine that rotates at a constant 72 revolutions per minute (RPM) and is capable of producing 20 kilowatts (kW) of power (Figure 1). The turbine is supported on a 0.4 meter (m) cylindrical tower at a height of 17 meters from the ground to the center of the hub. The blades used were rectangular, untwisted NREL S809 airfoil sections with an 0.457 meter chord.

One of the three blades was completely instrumented with pressure transducers (Figure 2) at four different span locations. Unsteady pressure data from all transducers were collected simultaneously. The data

sample rate (521 hertz [Hz]) was sufficient to capture the dynamic and transient pressure events elicited from time variant inlet flow conditions.

The inlet flow conditions were measured by the vertical plane array located 12 meters upwind of the turbine. The inlet flow magnitude and direction across the turbine diameter was measured using 11 prop-vane and 2 bi-vane anemometers. Eight of the prop-vane anemometers were arranged in a 4 meter radius circle approximately 16 meters above the ground. The remaining anemometers were spaced evenly inside the circle in a vertical line. The two bi-vane anemometers were mounted outside the circle on the horizontal axis.

Strain gages mounted on the blades and low-speed shaft were used to measure the rotor's mechanical loads. Torsional and edgewise bending moments were measured at the root of the instrumented blade and at two spanwise locations. Flapwise bending moments were measured at the roots of all three blades and at five spanwise locations on the instrumented blade (Figure 2). Mechanical turbine power was determined from measured low-speed shaft torque and rotor speed. Electrical power was measured via a power watt transducer connected to the three-phase, 480 volt, alternating current induction generator.

Data was collected over a wide range of nominal operating conditions for the turbine. Each data run consisted of a five-minute continuous data collection episode wherein all data were stored on time encoded tape. The turbine blade rotated at a nominal 1.2 Hz, thus, typically 360 consecutive blade rotation cycles are contained in each five-minute episode. A total of 59 episodes consisting of 239 channels of data were collected representing approximately 21,000 complete rotation cycles of the instrumented turbine blade. The 59-episode data set consists of 9 gigabytes of binary data and requires 30 optical disks for storage. For more information on the test setup and the data collected, refer to Butterfield et al. (1992) and Miller et al. (1994).

DATA EPISODE CHARACTERIZATION

Analyzing a data set of this size and complexity is no trivial task. The systematic identification of data episodes of interest demands detailed knowledge of the data recorded to each optical disk. Summary files were created for each of the 59 five-minute episodes that consist of the mean, standard deviation, minimum, and maximum for all 239 data channels. These summary files are recorded along with their respective data files on optical disk.

The summary file data is very helpful when investigating a single data episode, but even more useful is the ability to examine a certain parameter across all of the data episodes. Thus, databases were created which allow for examination of a single parameter. These databases are characterized based upon the summary statistics and contain mean, average, maximum, and minimum values for selected channels. All data channels, except for the blade surface pressures, have been characterized in this manner, but all are not included due to size constraints. Appendix A contains the tables most frequently used by the authors.

These databases can be utilized in a number of ways. For instance, using data from the yaw and velocity (disc-averaged wind speed) tables in Appendix A, Figure 3 gives an indication of the mean inflow conditions for all the data episodes. The inflow conditions, velocity and yaw, are the independent variables which drive the turbine's aerodynamics and, thus, its structural response. Figure 3 indicates that a significant portion, 45 of 59, of the data episodes have an average yaw between -9° and 3° and an average velocity between 5 and 17 meters per second (m/s). The low yaw angles are characteristic of downwind HAWTs since they tend to align themselves with the wind direction. Note, however, that there are large gaps in the matrix. For instance, there is only one data episode, d069021, with a mean yaw between 9° and 15°. Figure 4 depicts the same matrix, but the mean and standard deviation values for the inflow conditions have been inserted. The top chart provides mean velocity and standard deviation while the lower chart has mean yaw and standard deviation. As an example, looking at Figure 3, there is only one data episode, d072032, that meets the conditions of -15° to -9° yaw and 8 to 11 m/s in velocity. Figure 4 shows that this episode has an average velocity of 10.3 ± 3.0 m/s and an average yaw of -9.4° ± 21.3°. In boxes containing more than one data episode, all the data episodes are included in the average and standard deviation. In addition, the column and row averages are calculated along with a total for all the data episodes. The average velocity and yaw for all data episodes, 10.1 ± 4.2 m/s and -2.4° ± 3.3° respectively, also provide an indication of where most of the data are located.

ROTATIONAL CYCLE CHARACTERIZATION

The utility of these episode summary files in understanding turbine aerodynamics is severely limited due to the highly transient nature of the experimental data. As Figure 5 illustrates, the distribution of inflow velocity for two different episodes can be significantly different, even though they have similar means and standard deviations. Neither of these velocity distribution plots shows a stochastically normal distribution, nor are the shapes of the velocity distributions similar. Thus, to examine certain turbine aspects, a much finer resolution of data categorization was required.

Previous Combined Experiment investigators binned large amounts of the data on certain characteristics, such as inflow conditions (Huyer, 1993). Although this method provided some insight into turbine performance, to achieve the goal of understanding the turbine's fundamental aerodynamics and structural response, a more in-depth examination is required. Thorough examination of the Combined Experiment data requires categorization on a individual cycle basis. Thus, databases were created to contain average values and standard deviations of certain parameters for each of the 21,000 rotational cycles. Three separate databases were established based upon inflow conditions, aerodynamic force coefficients, and minimum leading-edge pressure coefficients.

Inflow Conditions Database

The inflow conditions, velocity and yaw, are extremely important when examining turbine data because these two independent variables drive the turbine's aerodynamics and have a significant effect upon structural responses. The program used to create the inflow conditions database, CYCCAT.C, is located in Appendix B along with the necessary header files. This program outputs cycle number along with the mean and standard deviation values of velocity (m/s), yaw (degrees [deg]), horizontal shear ((Δm/s)/m), and vertical shear ((Δm/s)/m) for each cycle within each data episode. The velocity value is obtained from the disc averaged wind speed. The disc averaged wind speed is calculated by averaging together readings of the eight prop-vane anemometers arranged in a 4 m radius circle on the Vertical Plane Array (VPA). The yaw value is created by subtracting the hub height wind direction on the VPA from the turbine yaw angle. Thus, yaw is the angle between the turbine and the wind direction. The horizontal shear is the difference between the value recorded by the prop-vane anemometers located at the 3 and 9 o'clock positions on the outer anemometer circle divided by the distance between them, 10.1 meters. The vertical shear is similar, except the anemometers at 12 and 6 o'clock are used. However, in reality the distance between the anemometers is 8.0 meters. Thus, the baseline data for horizontal and vertical shear is incorrect. The data can be corrected by multiplying the shear data by 10.1/8.0.

The program requires the user to input the name of the directory containing the data episode header file and the header file name. The program then creates the inflow conditions database. The program generates two output files for each data episode. For example, if data episode d073011 is being processed, the files d073011.sum and d073011.out are generated. The d073011.sum file contains the data in the form shown in Table 1. The d073011.out file contains only raw data, without the various titles (Table 2). For the Phase II data, the instrumented blade was at approximately 180° azimuth for all the data episodes. Thus, the first rotational cycle only contains information about half a rotational cycle. The inflow conditions database did not record these values, although the other databases do, because these half cycles were not of interest to the authors.

To simplify the explanation of the files, a * will be used in place of a data episode name. The primary usage of the *.sum files is for visual inspection of cycle inflow conditions. The *.out files are best suited for use with computational search tools. Using these files, the distribution of all available data on a individual cycle basis was generated (Figure 6). As earlier predicted, most of the cycles are located around 0° yaw and are between 5 and 17 m/s. However, using the data episode averaged data, it appeared that there was very little data for conditions where the yaw was more than 9°. Examining the data on a individual cycle basis, it is clear that the earlier conclusion is untrue because there is a significant amount of data for this range at all velocities.

A variety of search tools have been written to analyze the inflow conditions database. One of these programs, BESTCYC.F, is located in Appendix C. The program uses all the *.out files created by CYCCAT.C as input. As output, the program generates a file that contains the data episode, cycle number, and the mean and standard deviation values for velocity and yaw. The user is able to select from three different options. The first option simply finds

the mean and standard deviation in velocity and yaw for a user-provided range of cycles on a given episode. The second option locates a user-specified number of consecutive cycles on each episode having the lowest standard deviation in velocity or yaw. The third option allows the user to select a group of consecutive cycles based on the mean velocity and/or yaw limits defined by the user. The program then finds all the data that meets the input criteria. For example, this option allows the user to find all groups of 50 consecutive cycles that have a mean velocity between 9.5 and 10.5 m/s and a mean yaw between -2.5° and 2.5°. The user can then visually inspect the list or create a program to do it for them. This enables groups of cycles with low standard deviations in velocity and high standard deviations in yaw to be identified

Aerodynamic Force Coefficients Database

The authors decided through examination of the Combined Experiment data that the best means of determining aerodynamic performance was by the normal force coefficient, C_n, and the tangent force coefficient, C_t (Miller et al., 1994). The normal and tangent force coefficients were calculated for each of the four primary span locations (30%, 47%, 63%, and 80% span). These forces were created by integrating the pressure coefficient, C_p, values over the entire blade section during the initial Combined Experiment data processing. The program used to create the aerodynamic force coefficients database, AVCNCT.C, is located in Appendix D along with the necessary header files.

This program outputs cycle number along with the mean and standard deviation values of the normal and tangential aerodynamic force coefficients for the four primary span locations for each data episode. In addition, AVCNCT.C outputs the maximum value of both coefficients at the primary span locations for each cycle. This can be very useful in determining which cycles have significant aerodynamic forces that may be damaging the turbine structure.

The program requires the user to input the name of the directory containing the data episode header file and the header file name. After these have been entered, the program creates the aerodynamic force coefficients database. As in CYCCAT.C, each side of the 30 optical disks must be processed to create the entire database. However, the program is different from CYCCAT.C in several ways. The first major difference is that while creating the database, AVCNCT.C also checks and corrects errors in the aerodynamic force coefficients. Second, this program finds the maximum normal and tangent force coefficients at each of the four primary span locations for each individual cycle. Third, the program creates seven output files.

The program checks for errors on an individual data point basis. The exact reason for the few errors has not been determined, but they appear to be due to rare data glitches. The program calculates the standard deviation of the normal force coefficients for an entire cycle. Then, if a data point is more than two standard deviations away from the previous point, the data is considered to be anomalous. Anomalous points are corrected by replacing the anomalous value with the average of the two preceding points. Only the normal force coefficient data was examined. Therefore, if a normal force coefficient point is changed, the corresponding tangent force coefficient is also changed. This scheme was implemented because the tangent force values are very small, making this method of correction difficult and sometimes incorrect.

AVCNCT.C generates seven output files for each data episode. If data episode d073011 is being processed, d073011cn.sum, d073011ct.sum, d073011max.sum, d073011.err, d073011cn.out, d073011ct.out, and d073011max.out are created. The *cn.sum files contain all the normal force coefficient data (Table 3), the *ct.sum files contain all the tangent force coefficient data (Table 4), and the *max.sum contains all the maximum normal and tangent force coefficient data (Table 5). The *cn.out, *ct.out, and *max.out files simply contain the same data as the corresponding *.sum files without the titles. The *.err files show the value of the anomalous points and the value after being corrected (Table 6).

Several programs have been written to analyze the aerodynamic force coefficients database. One of these programs, CNCTMAX.F, is located in Appendix E. This program not only searches the aerodynamic force coefficients database, but examines the inflow conditions database as well. Thus, the program requires the *.out files created by CYCCAT.C and the *cn.out, *ct.out, and *max.out files created by AVCNCT.C. The program ranks every cycle by mean normal and tangent force coefficients in decreasing order for a user-specified span location. The user must input whether normal or tangent force coefficient is to be examined, the span location desired, the number of values to be

written to an output file, and the name of the output file. For instance, the program can be used to obtain the fifteen cycles with the highest tangent force coefficients at 47% span. The program outputs rank, data episode name, cycle number, mean force coefficient and standard deviation, the maximum force coefficient in that cycle, the mean angle of attack for the cycle (deg), the disc-averaged wind speed mean and standard deviation (m/s), and the yaw mean and standard deviation (deg). An example of the output is located in Table 7. The mean angle of attack for the cycle is calculated using geometric relations by methods of Shipley et al. (1994). The rest of the data is obtained from either the aerodynamic force coefficients or the inflow conditions database files.

A significant shortcoming of the present database is the inexact dynamic pressure that is used to create the normal and tangent force coefficients. This dynamic pressure is not yaw dependent and, thus, can be inaccurate during yawed conditions. However, a yaw dependent dynamic pressure can be derived using several different methods (Shipley et al., 1994). These methods were developed after the categorization program, AVCNCT.C, and the organizational program, CNCTMAX.F, were written.

Minimum Leading-Edge Pressure Coefficient Database

The minimum leading-edge pressure coefficient database was established to aid in the detection of dynamic stall events. One of the signatures of dynamic stall is a large suction peak or, in other words, a peak in the minimum leading-edge pressure coefficient. The program used to create the minimum leading-edge pressure coefficients database, DSTALLTAPE.F, is located in Appendix F. This program outputs the data episode and cycle number along with the minimum leading edge pressure coefficient for the cycle and the span location of the occurrence (30%, 47%, 63%, or 80% span). Also included is the disc-averaged wind speed (m/s), yaw (deg), and azimuth angle (deg) values at the precise moment when the minimum leading edge pressure coefficient occurred.

The program requires the user to input the name of the episode data file and output file name. In addition, the program allows the user to decide whether or not to renormalize the C_p values by a yaw dependent dynamic pressure. The dynamic pressure used is based upon a geometric model described in Shipley et al. (1994). An example of a database created by DSTALLFILE.F is located in Table 8. A header is included in the output files to indicate the contents. The authors' convention for naming the minimum leading edge pressure coefficient was maxcp?????r.dat. For example, for data episode d075022 the database was named maxcp75022r.dat. The 'r' in the name indicates that the C_p values were renormalized.

One of the programs created to analyze this database, DSTALL.F, is included in Appendix G. This program allows the user to limit the data in a number of ways: by minimum leading edge pressure coefficient, by span location, by velocity, by yaw, and by azimuth angle. These options permit the analysis of dynamic stall occurrences and magnitude under different operational conditions. For example, limiting the data below a given C_p allows the user to examine the data for all C_p spikes less than -8. Excluding an azimuthal region allows for a quantification of dynamic stall occurrence during different regions of the cycle. The program also allows for determination of occurrence of dynamic stall at more than one span location. Additionally, the program will display the number of cycles that fit the user-specified criteria. This feature allows the user to determine if the criteria is too general or too specific.

DSTALL.F is also able to present the data in several different formats. The data can be sorted by velocity, yaw, or minimum leading-edge pressure coefficient. The data can also be binned by velocity and yaw by specifying the upper and lower velocity and yaw bin limits along with the bin width. This final option averages all the C_p values of the cycles meeting the constraints set by the user. For instance, for a certain velocity range the user may wish to know the average C_p value for 30% span if the C_p value is limited to less than -10. The program generates an output file with the same format as the minimum leading-edge pressure coefficients database (Table 8).

The data obtained from the minimum leading-edge pressure coefficient database has a variety of applications. For example, the data could be used to obtain the frequency of dynamic stall occurrence under various inflow conditions (Figure 7). The criteria established to identify dynamic stall was that the C_p was less than -10.

CONCLUSION

In order to examine the massive amount of data collected during Phase II of the Combined

Experiment, the data has been characterized by data episode and by individual cycles. Also, a number of useful programs have been written not only to create the databases, but to examine them as well. Future researchers may find these programs useful when examining Phase II data or even for advanced phases of the Combined Experiment.

The various databases, the programs used to create them, and the search tools can be obtained from the authors.

REFERENCES

Butterfield, C.P., W.P. Musial, and D.A. Simms (1992): "Combined Experiment Phase I Final Report", NREL/TP-257-4655, National Renewable Energy Laboratory, Golden, CO.

Huyer, S. (1993): "Examination of Forced Unsteady Separated Flow Fields on a Rotating Wind Turbine Blade", NREL/TP-442-4864, National Renewable Energy Laboratory, Golden, CO.

Miller, M.S., D.E. Shipley, , M.C. Robinson, M.W. Luttges, and D.A. Simms (1994): "Determination of Data Reliability for Phase II of the Combined Experiment", NREL/TP-442-6914, National Renewable Energy Laboratory, Golden, CO.

Shipley, D.E., M.S. Miller, M.C. Robinson, M.W. Luttges, and D.A. Simms (1994): "Techniques for the Determination of Local Dynamic Pressure and Angle of Attack on a Horizontal Axis Wind Turbine", NREL/TP-442-7393, National Renewable Energy Laboratory, Golden, CO.

APPENDIX TABLE OF CONTENTS

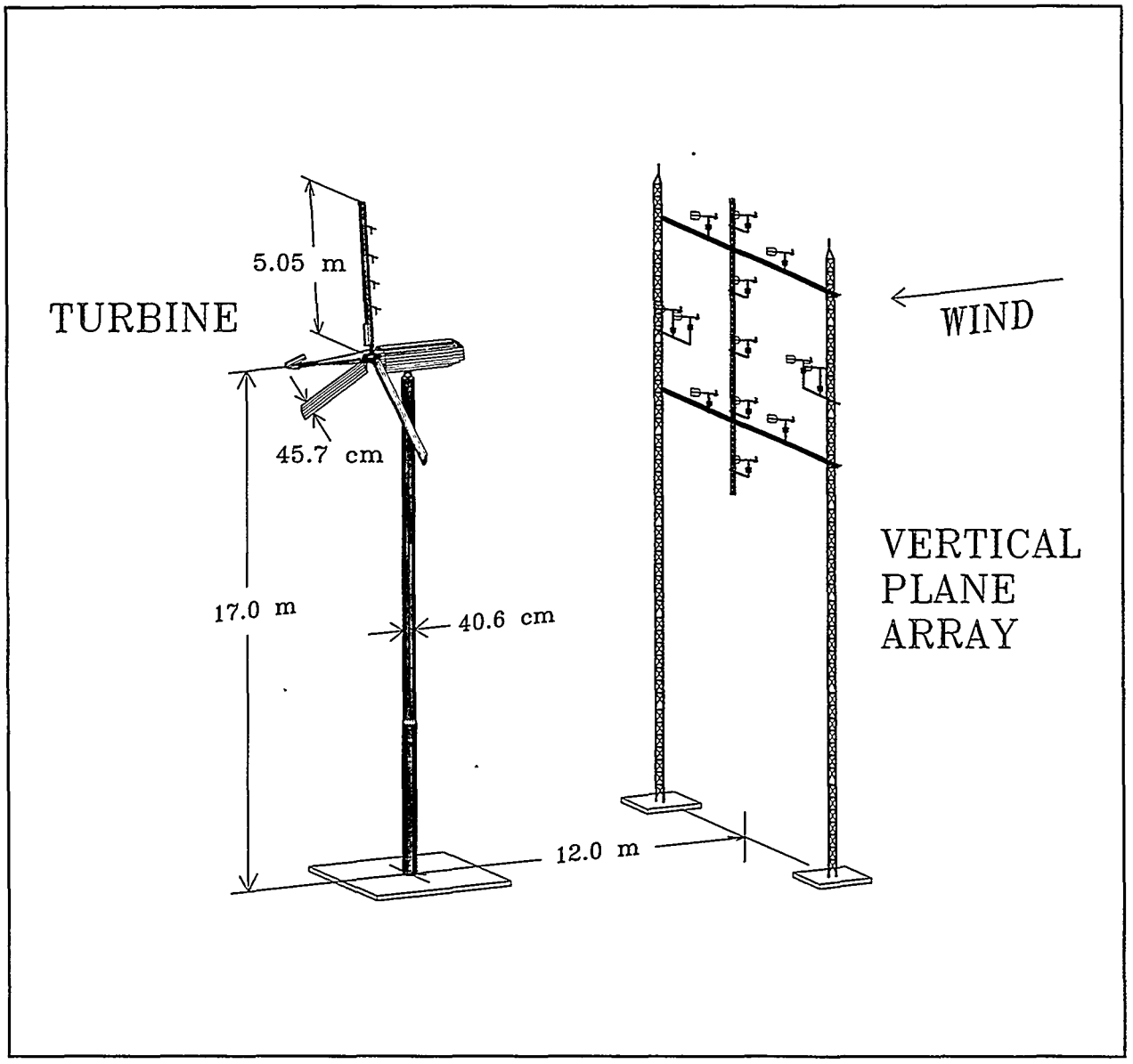

FIGURE 1. VIEW OF THE COMBINED EXPERIMENT TEST SITE INCLUDING THE GRUMMAN WIND STREAM 33 HORIZONTAL AXIS WIND TURBINE AND THE VERTICAL PLANE ARRAY.

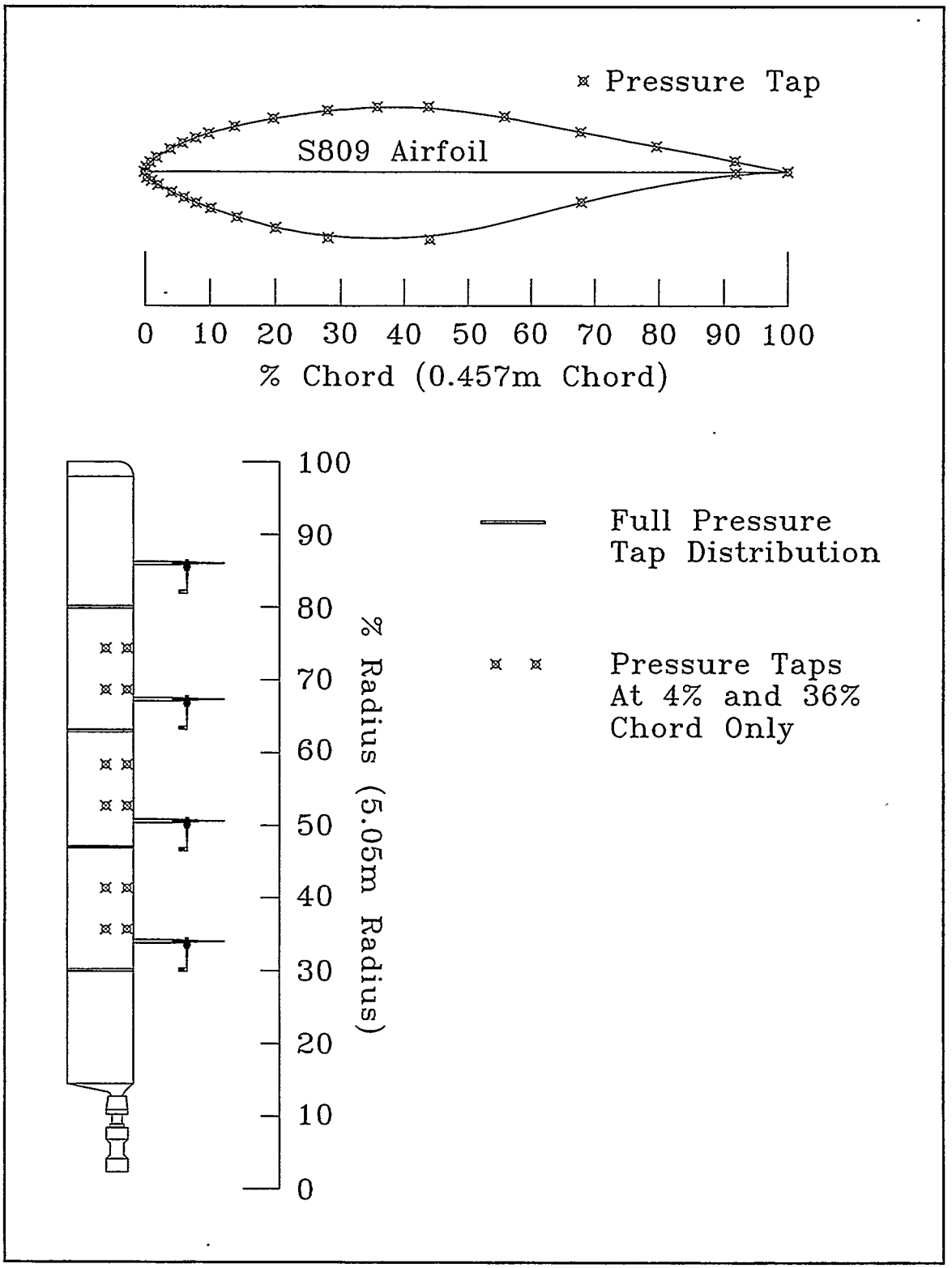

FIGURE 2. ROTOR BLADE CROSS SECTION AND LONGITUDINAL VIEWS SHOWING CHORDWISE PRESSURE TAP DISTRIBUTION AT EACH OF FOUR PRIMARY LOCATIONS (30%, 47%, 63%, AND 80% SPAN). DYNAMIC PRESSURE PROBES AND ANGLE OF ATTACK FLAGS ARE LOCATED JUST OUTBOARD OF SURFACE PRESSURE TAPS.

Average Velocity (m/s)

Yaw	< 5	5 to 8	8 to 11	11 to 14	14 to 17	> 17
-15 to -9		d069032 d069031	d072032			
-9 to -3	d067022	d073032 d075022 d070031 d073031 d065012	d069022 d072042 d066022 d067011 d073012	d072011 d072041 d073011 d071011 d066011 d068011	d072012 d068012 d068022 d071031	d072022 d071021 d071042
-3 to 3	d065022	d073042 d070041 d075012 d070042 d065011 d075021 d067021 d070022 d070021 d073022	d075011 d066021 d070012 d066032 d067012 d066031 d067031 d073021	d066012 d067032 d072031 d071012	d068021 d071041 d071032	
3 to 9		d073041 d070032 d065021 d070011			d072021	
9 to 15		d069021				

FIGURE 3. DISTRIBUTION OF FIVE-MINUTE DATA EPISODES BASED UPON AVERAGE INFLOW CONDITIONS, YAW, AND VELOCITY.

Average Velocity (m/s)

Average Yaw (degrees)	< 5	5 to 8	8 to 11	11 to 14	14 to 17	> 17
-15 to -9		d069032 d069031	d072032			
-9 to -3	d067022	d073032 d075022 d070031 d073031 d065012	d069022 d072042 d066022 d067011 d073012	d072011 d072041 d073011 d071011 d066011 d068011	d072012 d068012 d068022 d071031	d072022 d071021 d071042
-3 to 3	d065022	d073042 d070041 d075012 d070042 d065011 d075021 d067021 d070022 d070021 d073022	d075011 d066021 d070012 d066032 d067012 d066031 d067031 d073021	d066012 d067032 d072031 d071012	d068021 d071041 d071032	
3 to 9		d073041 d070032 d065021 d070011			d072021	
9 to 15		d069021				

FIGURE 3. DISTRIBUTION OF FIVE-MINUTE DATA EPISODES BASED UPON AVERAGE INFLOW CONDITIONS, YAW, AND VELOCITY.

Mean Velocity and Standard Deviation

Average Velocity (m/s)

Average Yaw (degrees)	<5	5 to 8	8 to 11	11 to 14	14 to 17	>17	Row Avg
-15 to -9	-	7.4 ± 1.7	10.3 ± 3.0	-	-	-	8.3 ± 2.5
-9 to -3	4.2 ± 1.4	6.4 ± 1.5	9.2 ± 1.9	12.8 ± 2.9	14.8 ± 2.8	17.8 ± 3.2	11.4 ± 4.7
-3 to 3	4.7 ± 1.5	7.3 ± 1.3	9.3 ± 1.6	12.4 ± 2.6	15.5 ± 2.8	-	9.5 ± 3.6
3 to 9	-	6.8 ± 1.5	-	-	14.2 ± 3.7	-	8.3 ± 3.6
9 to 15	-	7.7 ± 1.4	-	-	-	-	7.7 ± 1.4
	4.6 ± 1.5	7.0 ± 1.5	9.3 ± 1.8	12.6 ± 2.8	15.0 ± 2.9	17.8 ± 3.2	10.1 ± 4.2
	Column Averages						Total

Mean Yaw and Standard Deviation

Average Velocity (m/s)

Average Yaw (degrees)	<5	5 to 8	8 to 11	11 to 14	14 to 17	>17	Row Avg
-15 to -9	-	-15.4 ± 15.3	-9.4 ± 21.3	-	-	-	-13.5 ± 17.6
-9 to -3	-8.2 ± 20.2	-5.3 ± 14.4	-4.6 ± 11.8	-4.6 ± 10.9	-4.2 ± 11.6	-3.6 ± 12.7	-4.7 ± 12.7
-3 to 3	-0.3 ± 9.7	0.0 ± 11.3	-1.3 ± 11.6	-1.1 ± 12.9	-2.1 ± 11.9	-	-0.9 ± 11.6
3 to 9	-	3.8 ± 12.6	-	-	5.6 ± 13.7	-	4.2 ± 12.8
9 to 15	-	13.5 ± 20.9	-	-	-	-	13.5 ± 20.9
	-2.9 ± 14.6	-1.4 ± 14.6	-3.0 ± 12.7	-3.2 ± 11.8	-2.2 ± 12.4	-3.6 ± 12.7	-2.4 ± 13.3
	Column Averages						Total

FIGURE 4. DISTRIBUTION OF INFLOW DATA FOR ALL FIVE-MINUTE TEST EPISODES. MEANS AND STANDARD DEVIATIONS ARE GIVEN FOR FREESTREAM VELOCITY IN THE TOP CHART AND YAW IN THE BOTTOM CHART.

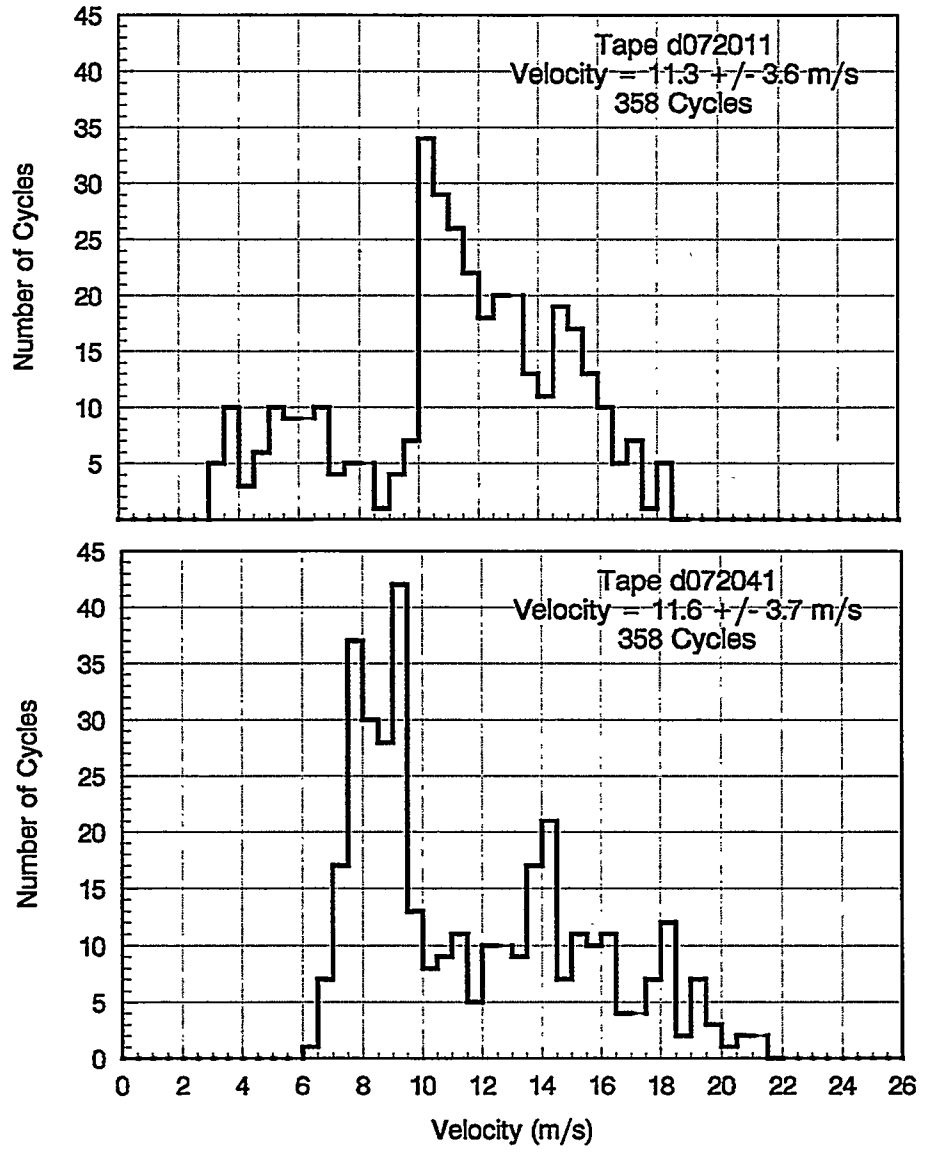

FIGURE 5. VELOCITY DISTRIBUTION HISTOGRAMS FOR SINGLE CYCLES OVER TWO SEPARATE FIVE-MINUTE DATA EPISODES WITH SIMILAR MEAN VELOCITIES.

TABLE 1. EXAMPLE OF INFLOW CONDITIONS DATABASE *.SUM FILE USING d075022.SUM. THESE FILES CONTAIN THE CYCLE NUMBER ALONG WITH THE CYCLE MEAN AND STANDARD DEVIATION VALUES FOR VELOCITY, YAW, HORIZONTAL SHEAR, AND VERTICAL SHEAR WITH APPROPRIATE TITLES.

Cyc 1	Vel: 6.392+-0.079	Yaw: 2.079+-2.916	HzShr: 0.113+-0.011	VShr: 0.057+-0.009
Cyc 2	Vel: 6.592+-0.042	Yaw: 0.763+-1.662	HzShr: 0.002+-0.052	VShr: 0.081+-0.013
Cyc 3	Vel: 6.449+-0.067	Yaw: -6.355+-4.692	HzShr: -0.177+-0.025	VShr: 0.066+-0.007
Cyc 4	Vel: 6.253+-0.034	Yaw: -5.536+-2.292	HzShr: -0.226+-0.024	VShr: 0.014+-0.031
Cyc 5	Vel: 6.248+-0.013	Yaw: -5.957+-2.507	HzShr: -0.206+-0.033	VShr: -0.034+-0.007
Cyc 6	Vel: 6.345+-0.048	Yaw: -11.910+-4.353	HzShr: -0.056+-0.044	VShr: -0.073+-0.016
Cyc 7	Vel: 6.421+-0.022	Yaw: -22.322+-0.821	HzShr: -0.011+-0.015	VShr: -0.065+-0.022
Cyc 8	Vel: 6.454+-0.021	Yaw: -21.511+-2.636	HzShr: 0.029+-0.010	VShr: 0.043+-0.022
Cyc 9	Vel: 6.386+-0.023	Yaw: -26.589+-2.396	HzShr: 0.003+-0.020	VShr: 0.090+-0.006
Cyc 10	Vel: 6.282+-0.044	Yaw: -23.241+-2.151	HzShr: -0.033+-0.012	VShr: 0.127+-0.024
Cyc 11	Vel: 6.285+-0.015	Yaw: -26.714+-2.447	HzShr: -0.005+-0.006	VShr: 0.111+-0.018
Cyc 12	Vel: 6.313+-0.016	Yaw: -27.362+-1.079	HzShr: -0.009+-0.022	VShr: 0.032+-0.017
Cyc 13	Vel: 6.240+-0.059	Yaw: -21.448+-0.814	HzShr: -0.080+-0.008	VShr: 0.054+-0.012
Cyc 14	Vel: 6.023+-0.052	Yaw: -21.377+-1.503	HzShr: -0.036+-0.019	VShr: 0.079+-0.006
Cyc 15	Vel: 6.094+-0.075	Yaw: -17.013+-2.115	HzShr: -0.023+-0.009	VShr: 0.090+-0.018
Cyc 16	Vel: 6.332+-0.047	Yaw: -16.108+-2.447	HzShr: -0.057+-0.012	VShr: 0.085+-0.012
Cyc 17	Vel: 6.486+-0.047	Yaw: -17.043+-1.484	HzShr: -0.097+-0.004	VShr: 0.066+-0.023
Cyc 18	Vel: 6.510+-0.038	Yaw: -18.867+-1.409	HzShr: -0.096+-0.012	VShr: -0.013+-0.010
Cyc 19	Vel: 6.382+-0.036	Yaw: -20.165+-1.478	HzShr: -0.095+-0.019	VShr: -0.035+-0.011
Cyc 20	Vel: 6.290+-0.041	Yaw: -19.863+-0.857	HzShr: -0.053+-0.008	VShr: -0.006+-0.021

TABLE 2. EXAMPLE OF INFLOW CONDITIONS DATABASE *.OUT FILE USING d075022.OUT. THESE FILES CONTAIN THE CYCLE NUMBER ALONG WITH THE CYCLE MEAN AND STANDARD DEVIATION VALUES FOR VELOCITY, YAW, HORIZONTAL SHEAR, AND VERTICAL SHEAR WITHOUT APPROPRIATE TITLES.

2	6.592	0.042	0.763	1.662	0.002	0.052	0.081	0.013
3	6.449	0.067	-6.355	4.692	-0.177	0.025	0.066	0.007
4	6.253	0.034	-5.536	2.292	-0.226	0.024	0.014	0.031
5	6.248	0.013	-5.957	2.507	-0.206	0.033	-0.034	0.007
6	6.345	0.048	-11.910	4.353	-0.056	0.044	-0.073	0.016
7	6.421	0.022	-22.322	0.821	-0.011	0.015	-0.065	0.022
8	6.454	0.021	-21.511	2.636	0.029	0.010	0.043	0.022
9	6.386	0.023	-26.589	2.396	0.003	0.020	0.090	0.006
10	6.282	0.044	-23.241	2.151	-0.033	0.012	0.127	0.024
11	6.285	0.015	-26.714	2.447	-0.005	0.006	0.111	0.018
12	6.313	0.016	-27.362	1.079	-0.009	0.022	0.032	0.017
13	6.240	0.059	-21.448	0.814	-0.080	0.008	0.054	0.012
14	6.023	0.052	-21.377	1.503	-0.036	0.019	0.079	0.006
15	6.094	0.075	-17.013	2.115	-0.023	0.009	0.090	0.018
16	6.332	0.047	-16.108	2.447	-0.057	0.012	0.085	0.012
17	6.486	0.047	-17.043	1.484	-0.097	0.004	0.066	0.023
18	6.510	0.038	-18.867	1.409	-0.096	0.012	-0.013	0.010
19	6.382	0.036	-20.165	1.478	-0.095	0.019	-0.035	0.011
20	6.290	0.041	-19.863	0.857	-0.053	0.008	-0.006	0.021

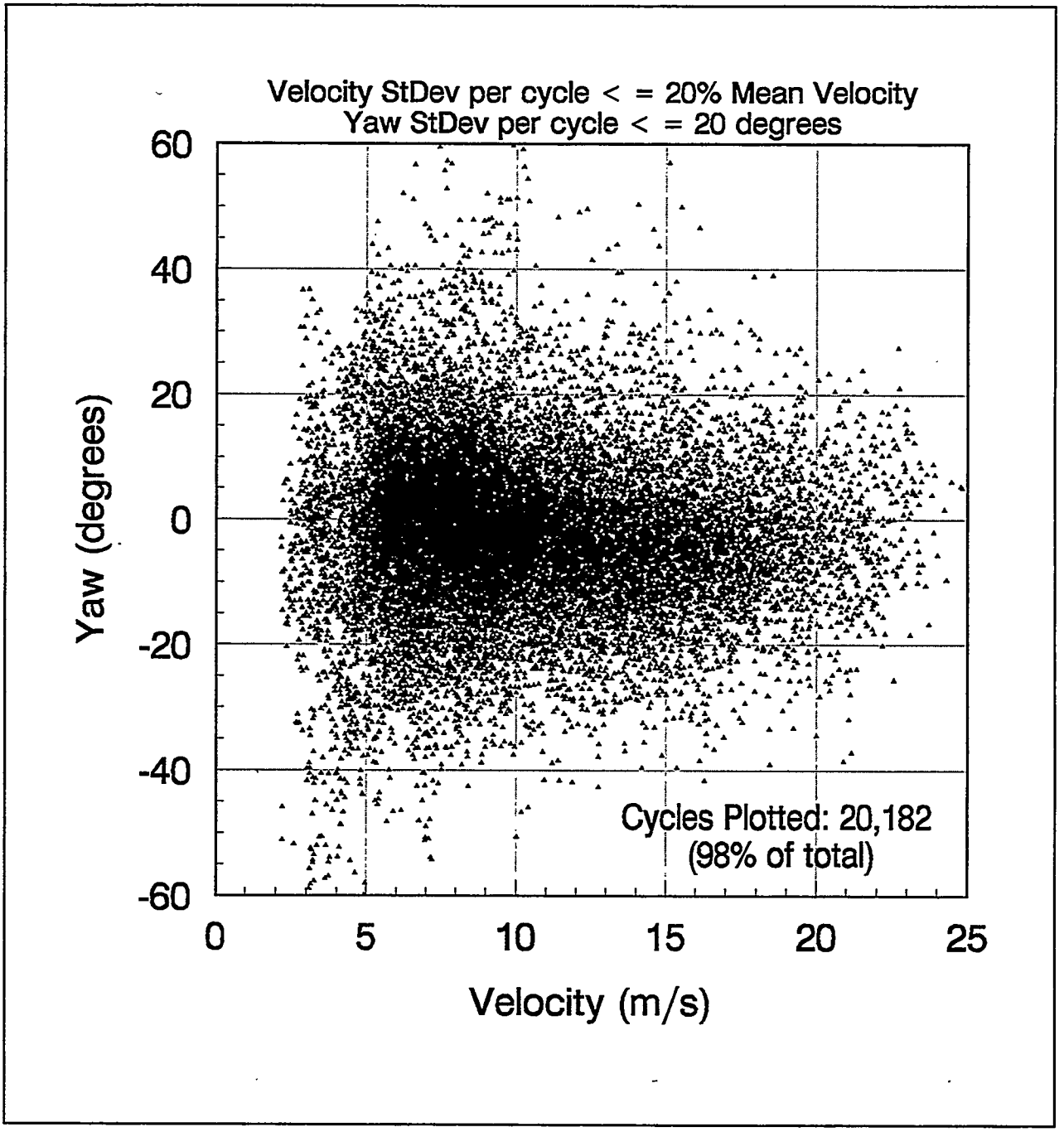

FIGURE 6. DISTRIBUTION OF AVERAGE INFLOW CONDITIONS FOR EACH INDIVIDUAL CYCLE. THE DATA FROM ALL 59 DATA EPISODES IS PLOTTED.

TABLE 3. EXAMPLE OF AERODYNAMIC FORCE COEFFICIENTS DATABASE *CN.SUM FILE USING d075021CN.SUM. THESE FILES CONTAIN THE CYCLE NUMBER ALONG WITH THE CYCLE MEAN AND STANDARD DEVIATION VALUES FOR THE NORMAL COEFFICIENT AT 30%, 47%, 63%, AND 80% SPAN WITH APPROPRIATE TITLES.

cyc: 1	30%: 1.724+- 0.357	47%: 0.939+- 0.134	63%: 0.670+- 0.090	80%: 0.340+- 0.092
cyc: 2	30%: 1.490+- 0.238	47%: 0.898+- 0.125	63%: 0.617+- 0.087	80%: 0.314+- 0.085
cyc: 3	30%: 1.115+- 0.166	47%: 0.778+- 0.110	63%: 0.551+- 0.114	80%: 0.281+- 0.091
cyc: 4	30%: 1.308+- 0.209	47%: 0.846+- 0.146	63%: 0.560+- 0.118	80%: 0.281+- 0.107
cyc: 5	30%: 1.446+- 0.194	47%: 0.890+- 0.115	63%: 0.708+- 0.101	80%: 0.409+- 0.077
cyc: 6	30%: 1.508+- 0.201	47%: 0.932+- 0.123	63%: 0.770+- 0.102	80%: 0.455+- 0.087
cyc: 7	30%: 1.455+- 0.283	47%: 0.903+- 0.161	63%: 0.713+- 0.088	80%: 0.421+- 0.087
cyc: 8	30%: 1.539+- 0.250	47%: 0.895+- 0.088	63%: 0.688+- 0.093	80%: 0.374+- 0.084
cyc: 9	30%: 1.425+- 0.184	47%: 0.843+- 0.100	63%: 0.638+- 0.079	80%: 0.348+- 0.075
cyc: 10	30%: 1.330+- 0.140	47%: 0.841+- 0.111	63%: 0.659+- 0.109	80%: 0.379+- 0.112
cyc: 11	30%: 1.488+- 0.301	47%: 0.884+- 0.114	63%: 0.689+- 0.123	80%: 0.413+- 0.106
cyc: 12	30%: 1.463+- 0.192	47%: 0.882+- 0.122	63%: 0.685+- 0.103	80%: 0.377+- 0.116
cyc: 13	30%: 1.233+- 0.112	47%: 0.819+- 0.109	63%: 0.598+- 0.091	80%: 0.339+- 0.079
cyc: 14	30%: 1.173+- 0.226	47%: 0.785+- 0.104	63%: 0.532+- 0.076	80%: 0.266+- 0.083
cyc: 15	30%: 1.471+- 0.227	47%: 0.878+- 0.180	63%: 0.655+- 0.112	80%: 0.342+- 0.093
cyc: 16	30%: 1.429+- 0.236	47%: 0.891+- 0.114	63%: 0.702+- 0.105	80%: 0.399+- 0.095
cyc: 17	30%: 1.401+- 0.255	47%: 0.905+- 0.116	63%: 0.701+- 0.112	80%: 0.400+- 0.115
cyc: 18	30%: 1.474+- 0.230	47%: 0.919+- 0.111	63%: 0.725+- 0.170	80%: 0.412+- 0.146
cyc: 19	30%: 1.479+- 0.228	47%: 0.895+- 0.134	63%: 0.670+- 0.161	80%: 0.388+- 0.151
cyc: 20	30%: 1.621+- 0.323	47%: 0.937+- 0.174	63%: 0.761+- 0.082	80%: 0.456+- 0.099

TABLE 4. EXAMPLE OF AERODYNAMIC FORCE COEFFICIENTS DATABASE *CT.SUM FILE USING d075021CT.SUM. THESE FILES CONTAIN THE CYCLE NUMBER ALONG WITH THE CYCLE MEAN AND STANDARD DEVIATION VALUES FOR THE NORMAL COEFFICIENT AT 30%, 47%, 63%, AND 80% SPAN WITH APPROPRIATE TITLES.

cyc: 1	30%: 0.096+- 0.075	47%: 0.135+- 0.029	63%: 0.052+- 0.015	80%: 0.001+- 0.005
cyc: 2	30%: 0.084+- 0.073	47%: 0.121+- 0.024	63%: 0.040+- 0.014	80%:-0.002+- 0.006
cyc: 3	30%: 0.121+- 0.036	47%: 0.099+- 0.020	63%: 0.035+- 0.019	80%:-0.002+- 0.007
cyc: 4	30%: 0.119+- 0.056	47%: 0.110+- 0.034	63%: 0.035+- 0.017	80%:-0.002+- 0.007
cyc: 5	30%: 0.070+- 0.056	47%: 0.133+- 0.026	63%: 0.064+- 0.020	80%: 0.009+- 0.008
cyc: 6	30%: 0.065+- 0.073	47%: 0.143+- 0.028	63%: 0.079+- 0.020	80%: 0.013+- 0.010
cyc: 7	30%: 0.032+- 0.078	47%: 0.138+- 0.033	63%: 0.064+- 0.018	80%: 0.009+- 0.008
cyc: 8	30%: 0.021+- 0.077	47%: 0.134+- 0.020	63%: 0.060+- 0.016	80%: 0.004+- 0.007
cyc: 9	30%: 0.050+- 0.064	47%: 0.122+- 0.025	63%: 0.050+- 0.015	80%: 0.003+- 0.007
cyc: 10	30%: 0.102+- 0.049	47%: 0.122+- 0.025	63%: 0.053+- 0.019	80%: 0.005+- 0.009
cyc: 11	30%: 0.079+- 0.066	47%: 0.132+- 0.030	63%: 0.057+- 0.026	80%: 0.010+- 0.011
cyc: 12	30%: 0.054+- 0.072	47%: 0.126+- 0.027	63%: 0.058+- 0.020	80%: 0.007+- 0.011
cyc: 13	30%: 0.100+- 0.047	47%: 0.106+- 0.023	63%: 0.046+- 0.019	80%: 0.003+- 0.006
cyc: 14	30%: 0.096+- 0.054	47%: 0.097+- 0.029	63%: 0.031+- 0.013	80%:-0.003+- 0.006
cyc: 15	30%: 0.053+- 0.075	47%: 0.121+- 0.042	63%: 0.052+- 0.022	80%: 0.003+- 0.008
cyc: 16	30%: 0.032+- 0.075	47%: 0.133+- 0.027	63%: 0.063+- 0.021	80%: 0.008+- 0.009
cyc: 17	30%: 0.046+- 0.066	47%: 0.134+- 0.028	63%: 0.064+- 0.024	80%: 0.009+- 0.012
cyc: 18	30%: 0.046+- 0.069	47%: 0.133+- 0.031	63%: 0.072+- 0.034	80%: 0.013+- 0.015
cyc: 19	30%: 0.021+- 0.082	47%: 0.109+- 0.042	63%: 0.061+- 0.030	80%: 0.011+- 0.015
cyc: 20	30%: 0.008+- 0.090	47%: 0.102+- 0.068	63%: 0.080+- 0.022	80%: 0.017+- 0.015

TABLE 5. EXAMPLE OF AERODYNAMIC FORCE COEFFICIENT DATABASE *MAX.SUM FILE USING d075021MAX.SUM. THESE FILES CONTAIN THE CYCLE NUMBER ALONG WITH THE CYCLE'S MAXIMUM NORMAL AND TANGENT COEFFICIENT AT 30%, 47%, 63%, AND 80% SPAN WITH APPROPRIATE TITLES.

cyc: 1	cn30%: 2.279	cn47%: 1.136	cn63%: 0.865	cn80%: 0.500	ct30%: 0.258	ct47%: 0.183	ct63%: 0.074	ct80%: 0.014
cyc: 2	cn30%: 2.048	cn47%: 1.153	cn63%: 0.873	cn80%: 0.535	ct30%: 0.251	ct47%: 0.160	ct63%: 0.081	ct80%: 0.016
cyc: 3	cn30%: 1.381	cn47%: 1.030	cn63%: 0.942	cn80%: 0.543	ct30%: 0.205	ct47%: 0.144	ct63%: 0.102	ct80%: 0.025
cyc: 4	cn30%: 1.705	cn47%: 1.164	cn63%: 0.857	cn80%: 0.542	ct30%: 0.244	ct47%: 0.196	ct63%: 0.076	ct80%: 0.022
cyc: 5	cn30%: 1.820	cn47%: 1.096	cn63%: 0.926	cn80%: 0.565	ct30%: 0.220	ct47%: 0.187	ct63%: 0.108	ct80%: 0.031
cyc: 6	cn30%: 1.911	cn47%: 1.127	cn63%: 1.005	cn80%: 0.578	ct30%: 0.244	ct47%: 0.195	ct63%: 0.122	ct80%: 0.036
cyc: 7	cn30%: 1.773	cn47%: 1.119	cn63%: 0.930	cn80%: 0.613	ct30%: 0.277	ct47%: 0.192	ct63%: 0.111	ct80%: 0.035
cyc: 8	cn30%: 1.977	cn47%: 1.131	cn63%: 0.917	cn80%: 0.563	ct30%: 0.242	ct47%: 0.193	ct63%: 0.101	ct80%: 0.022
cyc: 9	cn30%: 1.841	cn47%: 0.998	cn63%: 0.829	cn80%: 0.555	ct30%: 0.209	ct47%: 0.175	ct63%: 0.087	ct80%: 0.023
cyc: 10	cn30%: 1.786	cn47%: 1.077	cn63%: 0.883	cn80%: 0.589	ct30%: 0.203	ct47%: 0.178	ct63%: 0.099	ct80%: 0.030
cyc: 11	cn30%: 2.201	cn47%: 1.213	cn63%: 0.959	cn80%: 0.611	ct30%: 0.242	ct47%: 0.227	ct63%: 0.117	ct80%: 0.037
cyc: 12	cn30%: 2.246	cn47%: 1.153	cn63%: 1.017	cn80%: 0.616	ct30%: 0.211	ct47%: 0.191	ct63%: 0.118	ct80%: 0.041
cyc: 13	cn30%: 1.568	cn47%: 1.079	cn63%: 0.815	cn80%: 0.480	ct30%: 0.192	ct47%: 0.164	ct63%: 0.095	ct80%: 0.021
cyc: 14	cn30%: 1.654	cn47%: 1.083	cn63%: 0.706	cn80%: 0.390	ct30%: 0.210	ct47%: 0.170	ct63%: 0.072	ct80%: 0.012
cyc: 15	cn30%: 1.800	cn47%: 1.268	cn63%: 0.894	cn80%: 0.503	ct30%: 0.262	ct47%: 0.196	ct63%: 0.094	ct80%: 0.021

TABLE 6. EXAMPLE OF AERODYNAMIC FORCE COEFFICIENT DATABASE *.ERR FILE USING d075021.ERR. THESE FILES CONTAIN THE ANOMALOUS AND THE CORRECTED POINTS ALONG WITH THE CYCLE NUMBER AND THE SPAN LOCATION OF THE ERROR.

Anomalous points listing from d075021

change cn 1.293289 to 0.680574
change ct -0.059162 to 0.068494
in cyc 4, at 63% span

change cn -1.541332 to 0.769346
change ct 0.556902 to 0.074663
change cn 12.129753 to 0.771591
change ct -0.175177 to 0.076658
in cyc 11, at 63% span

TABLE 7. EXAMPLE OF CNCTMAX.F OUTPUT FILE BASED ON THE TANGENT FORCE COEFFICIENT AT 47% SPAN. THIS FILE CONTAINS RANK, TAPE NAME, CYCLE NUMBER, THE CYCLE'S MEAN ANGLE OF ATTACK, MAXIMUM TANGENT FORCE FOR THE CYCLE, AS WELL AS THE MEAN AND STANDARD DEVIATION VALUES FOR TANGENT FORCE, VELOCITY, AND YAW.

Rank	Tape	Cyc	Mean Ct	Ct sd	Max Ct	AOA	Mean Vel	Vsd	Mean Yaw	Ysd
1	d071032	41	0.898	0.704	1.48	30.518	17.590	0.453	-15.778	3.698
2	d073032	107	0.407	0.074	0.89	2.194	5.319	0.078	-4.860	1.970
3	d073022	143	0.164	0.033	0.22	8.648	7.829	0.069	6.595	5.317
4	d072042	155	0.160	0.055	0.27	9.406	8.129	0.095	-7.337	1.650
5	d073021	312	0.160	0.026	0.21	12.166	9.063	0.068	-2.150	0.331
6	d072032	267	0.159	0.078	0.32	8.706	8.233	0.282	22.100	9.522
7	d072022	273	0.155	0.122	0.34	19.954	14.208	0.077	-39.543	5.686
8	d073012	66	0.155	0.045	0.27	9.962	8.273	0.314	3.241	2.502
9	d073022	248	0.154	0.024	0.20	10.807	8.615	0.101	7.973	1.055
10	d072042	210	0.154	0.040	0.22	12.196	9.085	0.109	-3.716	2.394
11	d072042	258	0.154	0.037	0.25	14.025	9.774	0.061	-5.761	1.122
12	d073022	247	0.151	0.035	0.20	11.654	8.908	0.037	7.215	1.339
13	d071011	335	0.151	0.044	0.24	10.535	8.492	0.099	5.558	1.247
14	d073022	211	0.150	0.033	0.21	11.293	8.816	0.044	9.813	1.760
15	d072021	330	0.150	0.024	0.19	10.301	8.411	0.025	5.813	0.657

TABLE 8. EXAMPLE OF MINIMUM LEADING-EDGE PRESSURE COEFFICIENTS DATABASE USING MAXCP65011R.DAT. THESE FILES CONTAIN THE TAPE AND CYCLE NUMBER ALONG WITH THE MINIMUM LEADING-EDGE PRESSURE COEFFICIENT FOR EACH SPAN LOCATION AND THE INSTANTANEOUS VELOCITY, YAW, AND AZIMUTH VALUES AT WHICH THE PRESSURE COEFFICIENT OCCURRED. THE HEADER FOR THE FILE IS ALSO INCLUDED.

The maximum leading edge suction in each cycle is listed below
The Cp's were re-normalized using a yaw dependent dynamic pressure
The entire cycle was evaluated for max leading edge suction
All wind velocities were evaluated for max leading edge suction
All yaws were evaluated for max leading edge suction

Tape	Cycle	Span	Velocity	Yaw	Max Cp	Azimuth
d065011	2	30	5.832	5.453	-5.5472	278.4697
d065011	2	47	5.843	5.133	-3.4009	344.9312
d065011	2	63	5.832	5.368	-0.9282	294.8799
d065011	2	80	5.832	5.453	0.3685	282.5722
d065011	3	30	5.834	4.860	-6.0285	296.9729
d065011	3	47	5.843	4.963	-3.1553	0.5209
d065011	3	63	5.843	4.963	-0.7646	4.6235
d065011	3	80	5.840	5.559	0.4137	24.3158
d065011	4	30	5.823	-3.720	-6.4344	233.3655
d065011	4	47	5.787	-2.232	-3.5243	171.3513
d065011	4	63	5.831	-2.314	-0.5690	330.9015
d065011	4	80	5.810	-3.369	0.3410	182.1178
d065011	5	30	5.817	0.395	-5.7351	287.4024
d065011	5	47	5.778	0.927	-3.1869	82.4892
d065011	5	63	5.778	0.927	-0.6048	82.4892
d065011	5	80	5.814	0.203	0.5778	345.2203

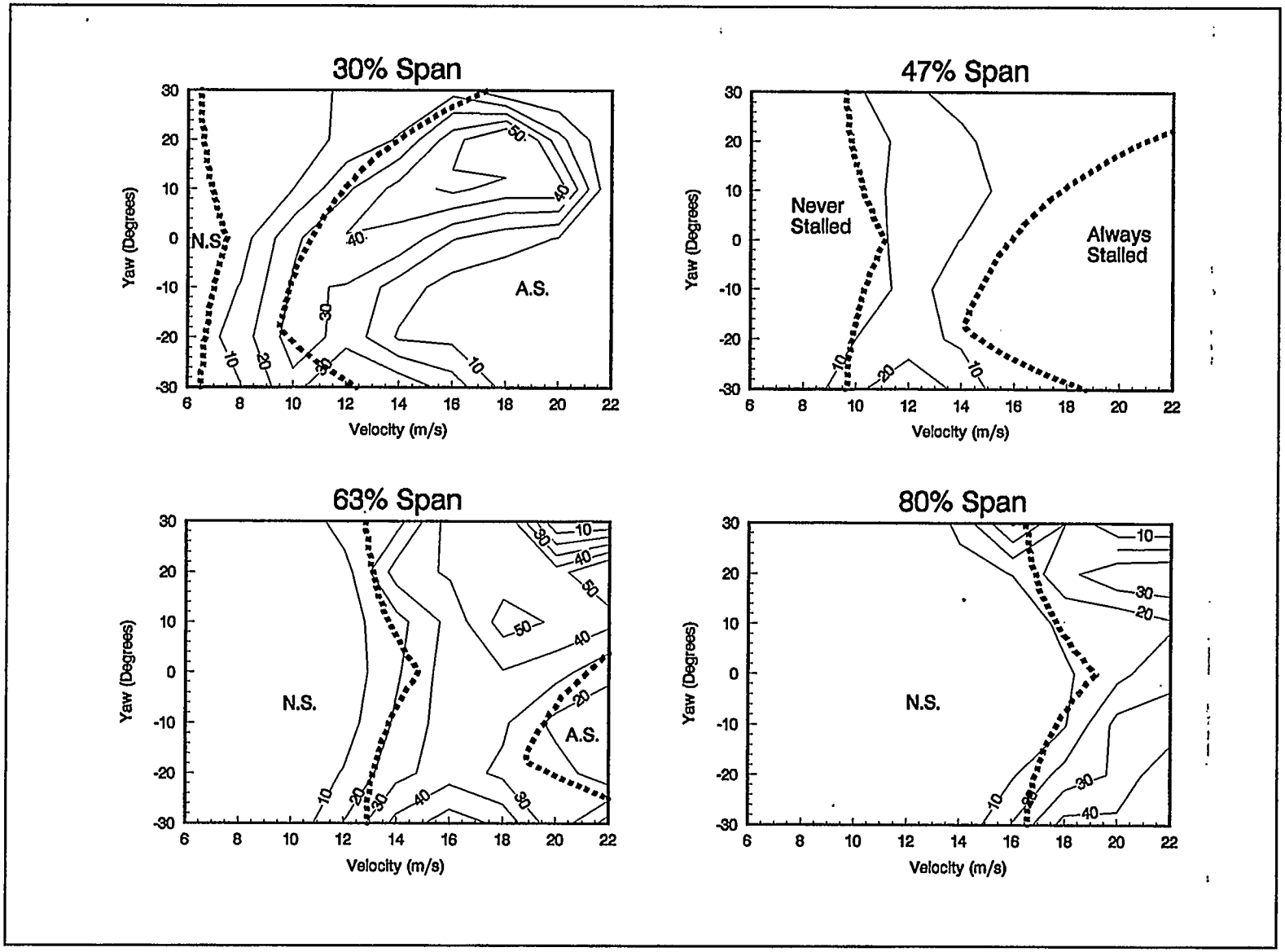

FIGURE 7. FREQUENCY OF DYNAMIC STALL OCCURRENCE AT FOUR SPAN LOCATIONS UNDER VARIOUS INFLOW CONDITIONS OF YAW AND WIND VELOCITY. CONTOURS REPRESENT PERCENTAGE OF CYCLES IN EACH BIN HAVING A PEAK Cp < -10.

Appendix A

Single Parameter Tables for Entire Data Episodes

I. Disc Averaged Wind Speed (m/s), Channel 734
II. Yaw (deg), Channel 307 - Channel 210
III. Measured Dynamic Pressure at 86% Span (psi), Channel 458
IV. Measured Dynamic Pressure at 67.3% Span (psi), Channel 934
V. Measured Dynamic Pressure at 50.6% Span (psi), Channel 862
VI. Measured Dynamic Pressure at 34% Span (psi), Channel 832
VII. Instrumented Blade Pitch Angle (deg), Channel 461
VIII. Root Flap Bending Moment (N*m), Blade 1 (Instrumented), Channel 402
IX. Root Flap Bending Moment (N*m), Blade 2, Channel 403
X. Root Flap Bending Moment (N*m), Blade 3, Channel 404
XI. Root Edge Bending Moment (N*m), Channel 414
XII. Low Speed Shaft Torque (N*m), Channel 423
XIII. Generator Power (kW), Channel 308
XIV. Normal Force Coefficient, Cn, at 80% Span, Channel 714
XV. Normal Force Coefficient, Cn, at 63% Span, Channel 715
XVI. Normal Force Coefficient, Cn, at 47% Span, Channel 716
XVII. Normal Force Coefficient, Cn, at 30% Span, Channel 717
XVIII. Tangent Force Coefficient, Ct, at 80% Span, Channel 718
XIX. Tangent Force Coefficient, Ct, at 63% Span, Channel 719
XX. Tangent Force Coefficient, Ct, at 47% Span, Channel 720
XXI. Tangent Force Coefficient, Ct, at 30% Span, Channel 721
XXII. Angle of Attack at 82% Span (deg), Channel 935
XXIII. Angle of Attack at 63.3% Span (deg), Channel 936
XXIV. Angle of Attack at 46.6% Span (deg), Channel 937
XXV. Angle of Attack at 30% Span (deg), Channel 938

734: Disc Averaged Wind Speed (m/s)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
4.246	1.382	2.169	8.622	67	22	298.959	135 16 4 30.791	135 16 9 29.750
4.429	1.319	2.371	6.924	65	22	299.370	115 9 47 40.501	115 9 52 39.869
4.960	1.704	2.199	10.630	73	42	299.532	153 12 12 50.225	153 12 17 49.755
5.279	1.054	2.717	7.752	73	32	273.569	153 12 2 26.020	153 12 6 59.586
5.785	1.264	2.672	8.471	73	41	298.510	153 12 7 50.870	153 12 12 49.378
5.865	1.104	3.752	8.930	75	22	289.576	206 10 33 5.072	206 10 37 54.646
6.087	1.104	3.920	8.078	70	31	299.305	135 21 45 20.167	135 21 50 19.470
6.231	0.780	4.621	8.684	70	41	299.230	135 21 57 .617	135 22 1 59.845
6.300	1.540	3.989	10.270	70	32	299.151	135 21 50 20.307	135 21 55 19.458
6.786	1.379	3.818	10.690	75	12	299.929	206 10 14 10.124	206 10 19 10.051
6.925	1.450	4.109	11.250	73	31	298.996	153 11 57 20.314	153 12 2 19.306
7.114	2.118	3.146	10.500	69	32	293.222	135 20 35 46.647	135 20 40 39.869
7.242	0.951	4.866	10.140	70	42	293.820	135 22 2 5.718	135 22 6 59.537
7.351	1.296	4.568	9.772	65	21	299.355	115 9 42 40.316	115 9 47 39.669
7.362	1.565	4.824	11.780	65	11	299.439	115 9 31 15.541	115 9 36 14.980
7.377	1.116	5.124	9.983	75	21	299.812	206 10 28 .205	206 10 33 .013
7.423	1.223	5.191	9.900	67	21	299.545	135 15 59 30.414	135 16 4 29.957
7.561	1.350	5.131	10.930	70	22	254.371	135 21 39 45.583	135 21 43 59.952
7.568	0.768	5.573	9.771	70	21	298.798	135 21 34 40.919	135 21 39 39.715
7.583	1.055	5.188	10.370	69	31	299.679	135 20 30 40.268	135 20 35 39.945
7.663	1.442	4.781	10.610	69	21	254.521	135 20 19 45.149	135 20 23 59.668
7.665	1.663	3.800	11.720	65	12	299.224	115 9 36 15.822	115 9 41 15.046
7.680	0.969	5.483	10.070	70	11	299.601	135 21 23 35.137	135 21 28 34.737
7.822	1.539	5.253	12.040	73	22	292.729	153 11 51 46.543	153 11 56 39.272
8.177	1.856	4.193	11.770	69	22	224.076	135 20 24 45.731	135 20 28 29.803
8.204	1.126	5.687	10.810	75	11	299.439	206 10 9 10.688	206 10 14 10.124
8.397	1.610	4.820	12.720	72	42	289.056	138 17 14 30.447	138 17 19 19.501
8.417	1.096	5.725	11.060	66	21	299.234	135 14 2 40.759	135 14 7 39.991
8.436	1.107	5.341	11.030	70	12	299.255	135 21 28 35.569	135 21 33 34.824
8.657	1.251	5.765	11.850	66	22	299.046	135 14 7 40.831	135 14 12 39.875
8.962	1.440	6.255	12.670	66	32	298.883	135 14 19 10.642	135 14 24 9.523
9.009	1.132	6.505	12.440	67	12	298.846	135 15 52 40.559	135 15 57 39.403
9.428	1.246	6.756	13.110	67	11	299.499	135 15 47 40.226	135 15 52 39.723
10.280	2.989	4.858	18.430	72	32	261.823	138 17 4 6.242	138 17 8 28.064
10.320	1.431	6.213	15.070	67	31	299.288	135 16 11 .492	135 16 15 59.778
10.320	1.867	5.962	14.940	66	31	299.449	135 14 14 10.599	135 14 19 10.046
10.620	1.202	6.866	13.910	73	21	299.151	153 11 46 40.756	153 11 51 39.904
10.950	1.814	7.030	14.910	73	12	300.008	153 11 40 50.094	153 11 45 50.101
11.170	1.481	7.551	15.120	66	12	297.491	135 13 53 45.999	135 13 58 43.489
11.320	3.595	3.187	18.790	72	11	299.750	138 16 33 30.441	138 16 38 30.191
11.360	1.489	8.023	15.320	67	32	299.149	135 16 16 .526	135 16 20 59.672
11.600	3.738	6.329	21.840	72	41	298.986	138 17 9 30.796	138 17 14 29.779
13.060	2.237	7.341	18.260	73	11	298.742	153 11 35 50.765	153 11 40 49.504
13.270	2.329	8.381	18.910	71	11	299.566	138 15 9 10.495	138 15 14 10.059
13.480	1.820	9.897	18.610	66	11	299.144	135 13 48 40.309	135 13 53 39.449
13.530	2.667	7.223	19.360	72	31	299.134	138 16 59 .620	138 17 3 59.754
13.580	3.077	6.973	24.310	71	12	299.574	138 15 14 10.641	138 15 19 10.211
13.860	2.118	8.705	19.340	68	11	299.142	135 17 7 28.499	135 17 12 27.639
14.120	2.167	8.988	19.800	68	21	359.543	135 17 19 16.509	135 17 25 16.050
14.180	3.663	6.942	24.510	72	21	298.848	138 16 46 22.829	138 16 51 21.675
14.340	3.537	8.039	24.090	72	12	299.134	138 16 38 30.898	138 16 43 30.029
14.460	2.441	8.383	19.670	68	12	295.419	135 17 12 33.107	135 17 17 28.524
14.680	2.075	9.396	20.020	68	22	358.295	135 17 25 22.344	135 17 31 20.639
15.590	2.803	9.178	23.460	71	31	298.721	138 15 31 10.531	138 15 36 9.250
15.920	2.348	10.320	23.120	71	41	299.130	138 15 42 40.824	138 15 47 39.951
16.870	3.070	10.260	24.050	71	32	299.142	138 15 36 10.629	138 15 41 9.770
17.280	3.476	8.016	25.170	72	22	299.146	138 16 51 22.508	138 16 56 21.652
17.350	3.528	8.318	24.920	71	21	299.176	138 15 20 15.474	138 15 25 14.650
18.740	2.429	11.660	25.960	71	42	287.025	138 15 47 46.530	138 15 52 33.554

307 - 210: Yaw (deg)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-18.707	16.069	-49.147	11.988	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-12.173	13.060	-44.320	13.943	69	31	299.679	135 20 30 40.268	135 20 35 39.945
-9.380	21.110	-42.529	84.554	72	32	261.823	138 17 04 06.242	138 17 08 28.064
-8.764	17.191	-59.887	29.948	73	32	273.569	153 12 02 26.020	153 12 06 59.586
-8.217	20.059	-61.922	36.723	67	22	298.959	135 16 04 30.791	135 16 09 29.750
-6.011	9.681	-33.208	29.643	73	12	300.008	153 11 40 50.094	153 11 45 50.101
-5.836	13.320	-44.902	39.618	72	11	299.750	138 16 33 30.441	138 16 38 30.191
-5.391	14.606	-38.581	32.006	72	41	298.986	138 17 09 30.796	138 17 14 29.779
-4.946	12.819	-60.038	39.482	68	12	295.419	135 17 12 33.107	135 17 17 28.524
-4.823	14.525	-44.750	34.976	65	12	299.224	115 09 36 15.822	115 09 41 15.046
-4.701	10.666	-36.548	19.310	66	22	299.046	135 14 07 40.831	135 14 12 39.875
-4.651	7.759	-30.148	24.278	66	11	299.144	135 13 48 40.309	135 13 53 39.449
-4.643	12.266	-41.561	18.184	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-4.605	12.035	-41.894	49.570	71	31	298.721	138 15 31 10.531	138 15 36 09.250
-4.505	14.273	-54.215	25.399	73	31	298.996	153 11 57 20.314	153 12 02 19.306
-4.476	15.345	-45.495	39.416	72	42	289.056	138 17 14 30.447	138 17 19 19.501
-4.310	12.474	-39.142	32.795	69	22	224.076	135 20 24 45.731	135 20 28 29.803
-4.294	6.179	-21.155	16.362	68	11	299.142	135 17 07 28.499	135 17 12 27.639
-4.009	9.937	-32.995	19.905	72	12	299.134	138 16 38 30.898	138 16 43 30.029
-3.936	7.511	-23.169	26.774	73	11	298.742	153 11 35 50.765	153 11 40 49.504
-3.908	11.488	-42.591	16.975	75	22	289.576	206 10 33 05.072	206 10 37 54.646
-3.762	14.484	-39.543	39.034	72	22	299.146	138 16 51 22.508	138 16 56 21.652
-3.733	11.372	-39.556	31.808	71	21	299.176	138 15 20 15.474	138 15 25 14.650
-3.564	10.271	-45.881	29.253	71	11	299.566	138 15 09 10.495	138 15 14 10.059
-3.408	9.636	-24.546	26.585	68	22	358.295	135 17 25 22.344	135 17 31 20.639
-3.343	10.161	-29.387	33.554	71	42	287.025	138 15 47 46.530	138 15 52 33.554
-3.226	7.591	-25.176	20.263	67	11	299.499	135 15 47 40.226	135 15 52 39.723
-2.582	11.633	-34.405	24.499	70	21	298.798	135 21 34 40.919	135 21 39 39.715
-2.534	13.779	-37.591	43.729	66	21	299.234	135 14 02 40.759	135 14 07 39.991
-2.523	10.808	-37.826	28.707	71	32	299.142	138 15 36 10.629	138 15 41 09.770
-2.521	8.241	-28.071	28.271	66	31	299.449	135 14 14 10.599	135 14 19 10.046
-2.505	8.253	-23.971	31.304	73	21	299.151	153 11 46 40.756	153 11 51 39.904
-2.160	9.656	-27.078	27.449	66	12	297.491	135 13 53 45.999	135 13 58 43.489
-2.040	12.161	-29.877	50.383	68	21	359.543	135 17 19 16.509	135 17 25 16.050
-2.004	9.790	-29.317	22.145	66	32	298.883	135 14 19 10.642	135 14 24 09.523
-1.928	9.425	-29.117	26.427	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-1.866	10.681	-29.271	29.085	71	12	299.574	138 15 14 10.641	138 15 19 10.211
-1.842	11.090	-27.800	31.569	71	41	299.130	138 15 42 40.824	138 15 47 39.951
-1.547	10.153	-30.393	24.790	67	32	299.149	135 16 16 00.526	135 16 20 59.672
-1.342	11.246	-37.513	28.784	67	12	298.846	135 15 52 40.559	135 15 57 39.403
-1.288	9.404	-34.605	17.311	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-0.999	9.157	-33.522	18.586	75	21	299.812	206 10 28 00.205	206 10 33 00.013
-0.831	7.695	-19.384	21.201	65	22	299.370	115 09 47 40.501	115 09 52 39.869
-0.309	12.009	-34.133	38.153	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.154	8.769	-24.567	32.897	75	11	299.439	206 10 09 10.688	206 10 14 10.124
0.334	10.821	-39.895	24.731	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.346	15.173	-36.284	52.843	67	31	299.288	135 16 11 00.492	135 16 15 59.778
0.955	11.421	-26.618	47.750	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.991	17.477	-41.634	38.036	72	31	299.134	138 16 59 00.620	138 17 03 59.754
1.301	12.758	-48.193	43.275	70	42	293.820	135 22 02 05.718	135 22 06 59.537
1.310	10.500	-26.725	26.606	67	21	299.545	135 15 59 30.414	135 16 04 29.957
1.424	11.874	-40.536	47.525	65	11	299.439	115 09 31 15.541	115 09 36 14.980
2.277	10.789	-23.440	39.917	70	41	299.230	135 21 57 00.617	135 22 01 59.845
3.090	14.333	-33.696	33.616	70	32	299.151	135 21 50 20.307	135 21 55 19.458
3.791	15.042	-27.614	37.010	73	41	298.510	153 12 07 50.870	153 12 12 49.378
4.119	9.754	-21.976	56.875	65	21	299.355	115 09 42 40.316	115 09 47 39.669
4.309	8.214	-21.973	29.424	70	11	299.601	135 21 23 35.137	135 21 28 34.737
5.564	13.342	-33.205	57.035	72	21	298.848	138 16 46 22.829	138 16 51 21.675
13.549	20.688	-30.570	66.412	69	21	254.521	135 20 19 45.149	135 20 23 59.668

458: Total Pressure Probe at 86% Span (psi)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.073	0.008	-0.890	1.212	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.074	0.005	-0.402	0.100	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.074	0.005	-0.405	0.102	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.074	0.005	-0.403	0.831	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.075	0.008	-0.403	0.834	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.075	0.005	-0.404	0.099	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.078	0.005	-0.396	0.111	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.079	0.011	-0.465	1.256	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.079	0.007	-0.386	0.113	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.079	0.005	-0.399	0.103	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.080	0.010	-0.489	0.967	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.080	0.007	-0.372	0.759	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.080	0.024	-0.626	1.180	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.080	0.006	-0.381	0.111	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.080	0.009	-0.977	0.718	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.081	0.025	-0.951	0.981	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.081	0.006	0.058	0.113	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.081	0.015	-0.464	0.765	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.081	0.007	-0.372	0.113	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.081	0.010	-0.995	1.263	75	21	299.812	206 10 28 .205	206 10 33 .013
0.081	0.007	-0.387	0.770	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.081	0.006	-0.403	0.871	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.081	0.017	-0.507	0.933	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.081	0.007	-0.227	0.114	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.082	0.023	-0.494	0.979	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.082	0.008	-0.438	0.766	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.082	0.020	-0.510	0.925	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.082	0.011	-0.434	0.993	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.082	0.016	-0.486	1.289	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.082	0.009	-0.190	0.781	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.082	0.022	-0.483	0.986	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.082	0.008	-0.461	0.130	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.083	0.014	-1.007	1.265	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.083	0.007	-0.225	0.120	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.083	0.007	-0.224	0.995	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.083	0.017	-0.484	1.344	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.084	0.015	-0.938	0.983	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.084	0.007	-0.225	0.782	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.084	0.013	-0.432	1.258	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.085	0.014	-0.452	1.308	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.086	0.021	-0.502	0.995	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.086	0.014	-0.907	0.988	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.086	0.020	-0.505	1.038	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.086	0.018	-0.489	0.980	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.086	0.018	-0.500	0.901	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.086	0.022	-0.652	0.996	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.088	0.018	-0.510	0.751	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.088	0.022	-0.966	0.995	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.088	0.015	-0.913	1.010	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.089	0.010	-0.937	0.140	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.089	0.013	-0.429	0.172	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.090	0.015	-1.013	0.776	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.090	0.013	-0.282	0.155	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.092	0.020	-0.511	0.780	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.093	0.018	-0.489	0.890	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.093	0.019	-0.959	0.905	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.093	0.018	-0.496	0.896	71	42	287.025	138 15 47 46.530	138 15 52 33.554
0.094	0.020	-0.930	0.900	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.096	0.013	-0.554	1.003	66	11	299.144	135 13 48 40.309	135 13 53 39.449

934: Total Pressure Probe at 67.3% Span (psi)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.049	0.006	-0.516	0.625	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.049	0.005	-0.536	0.074	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.049	0.005	-1.077	0.074	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.050	0.004	0.032	0.072	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.050	0.005	0.029	0.076	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.050	0.006	-0.529	0.626	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.050	0.007	0.022	0.643	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.050	0.006	-0.774	0.258	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.050	0.005	0.030	0.241	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.050	0.005	-0.887	0.541	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.050	0.006	-0.581	0.088	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.050	0.005	0.022	0.076	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.051	0.005	0.022	0.082	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.051	0.007	0.024	0.083	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.051	0.007	-0.569	1.127	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.051	0.005	0.018	0.259	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.051	0.008	-1.097	1.189	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.051	0.006	-0.529	0.705	75	21	299.812	206 10 28 .205	206 10 33 .013
0.051	0.007	0.027	0.087	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.051	0.006	0.025	0.086	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.051	0.006	-1.025	0.537	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.051	0.006	-0.589	0.562	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.051	0.006	-1.082	0.113	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.051	0.006	0.023	0.567	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.052	0.010	0.018	0.099	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.052	0.005	0.027	0.081	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.052	0.006	-0.611	0.084	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.052	0.006	0.028	0.080	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.052	0.006	-0.882	0.554	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.053	0.008	-0.129	0.083	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.053	0.007	-0.845	0.091	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.053	0.007	-1.080	0.083	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.053	0.008	-0.002	0.091	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.054	0.007	0.026	1.187	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.054	0.007	-0.593	0.088	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.054	0.006	-0.843	0.573	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.054	0.012	0.009	0.113	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.055	0.013	-0.612	0.116	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.055	0.013	-0.553	0.578	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.055	0.012	-0.003	0.157	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.056	0.007	-0.597	0.091	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.056	0.010	0.010	0.555	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.057	0.008	0.020	0.100	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.057	0.007	0.018	1.170	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.057	0.013	-0.879	0.126	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.058	0.013	0.012	0.115	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.058	0.010	-1.051	0.541	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.059	0.009	-0.623	0.104	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.059	0.010	0.004	0.104	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.059	0.009	-0.833	0.112	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.059	0.010	0.006	0.548	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.059	0.012	-0.554	0.590	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.059	0.012	-0.580	0.574	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.059	0.010	0.017	0.584	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.060	0.011	0.019	0.204	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.060	0.008	0.025	0.128	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.061	0.007	-0.052	0.093	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.063	0.016	0.003	0.947	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.064	0.013	-0.559	0.618	71	42	287.025	138 15 47 46.530	138 15 52 33.554

862: Total Pressure Probe at 50.6% Span (psi)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.028	0.005	0.009	0.051	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.028	0.003	-0.185	0.242	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.028	0.003	-0.181	0.242	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.028	0.005	0.012	0.053	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.029	0.005	-0.392	0.247	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.029	0.003	-0.183	0.046	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.029	0.004	-0.188	0.054	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.029	0.004	-0.185	0.244	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.029	0.005	-0.393	0.243	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.030	0.004	-0.184	0.053	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.030	0.005	-0.185	0.345	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.030	0.004	-0.003	0.238	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.030	0.005	-0.185	0.246	75	21	299.812	206 10 28 .205	206 10 33 .013
0.030	0.004	-0.185	0.054	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.030	0.004	-0.393	0.242	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.030	0.004	-0.003	0.055	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.030	0.005	-0.197	0.248	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.030	0.005	-0.187	0.242	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.031	0.004	0.009	0.054	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.031	0.004	0.007	0.059	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.031	0.008	-0.189	0.245	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.031	0.005	-0.186	0.242	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.031	0.006	-0.185	0.247	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.031	0.006	-0.180	0.242	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.031	0.005	-0.288	0.248	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.031	0.005	-0.183	0.242	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.031	0.005	-0.180	0.243	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.031	0.004	0.011	0.243	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.031	0.006	-0.383	0.242	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.032	0.004	0.010	0.058	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.032	0.005	-0.178	0.242	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.032	0.005	0.006	0.242	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.032	0.004	-0.183	0.058	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.033	0.006	-0.240	0.242	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.033	0.007	-0.013	0.244	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.033	0.005	-0.188	0.251	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.034	0.009	0.001	0.079	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.034	0.006	-0.179	0.242	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.034	0.006	-0.190	0.242	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.035	0.006	-0.184	0.246	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.035	0.009	-0.185	0.249	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.035	0.011	-0.193	0.244	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.036	0.008	-0.188	0.246	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.037	0.009	-0.193	0.261	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.037	0.006	-0.175	0.246	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.037	0.006	-0.389	0.250	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.037	0.011	-0.375	0.253	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.038	0.007	-0.280	0.246	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.038	0.010	-0.184	0.245	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.038	0.009	-0.007	0.095	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.039	0.009	-0.209	0.253	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.039	0.009	-0.389	0.250	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.039	0.008	-0.207	0.447	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.040	0.010	-0.378	0.251	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.042	0.010	-0.391	0.252	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.043	0.010	-0.178	0.254	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.044	0.013	-0.186	0.247	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.045	0.013	-0.364	0.444	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.045	0.011	-0.189	0.253	71	42	287.025	138 15 47 46.530	138 15 52 33.554

832: Total Pressure Probe at 34% Span (psi)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.013	0.002	-0.181	0.025	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.013	0.003	-0.416	0.236	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.013	0.003	-0.003	0.029	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.014	0.004	-0.417	0.242	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.014	0.003	-0.417	0.240	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.014	0.003	0.002	0.241	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.014	0.003	-0.204	0.237	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.014	0.003	-0.206	0.028	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.014	0.004	-0.204	0.454	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.015	0.003	-0.019	0.031	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.015	0.003	-0.004	0.230	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.015	0.004	-0.200	0.403	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.015	0.003	-0.207	0.233	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.015	0.006	-0.207	0.356	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.015	0.003	-0.022	0.031	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.015	0.004	-0.210	0.031	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.015	0.004	-0.203	0.353	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.015	0.004	-0.201	0.239	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.016	0.003	0.001	0.031	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.016	0.004	-0.209	0.236	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.016	0.004	-0.202	0.244	75	21	299.812	206 10 28 .205	206 10 33 .013
0.016	0.003	0.000	0.035	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.016	0.005	-0.201	0.233	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.016	0.005	-0.209	0.234	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.016	0.004	-0.202	0.231	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.016	0.005	-0.384	0.240	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.016	0.004	-0.200	0.234	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.016	0.004	-0.199	0.315	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.017	0.003	-0.006	0.236	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.017	0.004	-0.203	0.235	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.017	0.003	-0.001	0.039	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.017	0.004	-0.207	0.238	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.017	0.003	-0.001	0.231	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.018	0.005	-0.027	0.045	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.018	0.005	-0.207	0.447	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.018	0.004	-0.308	0.242	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.018	0.008	-0.013	0.055	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.019	0.005	-0.203	0.236	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.019	0.004	-0.009	0.234	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.020	0.005	-0.203	0.244	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.020	0.008	-0.203	0.239	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.020	0.009	-0.207	0.243	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.021	0.005	-0.206	0.237	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.021	0.008	-0.235	0.241	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.021	0.005	-0.204	0.242	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.022	0.008	-0.205	0.251	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.022	0.009	-0.204	0.242	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.022	0.006	-0.417	0.443	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.023	0.009	-0.203	0.239	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.023	0.008	-0.096	0.069	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.024	0.007	-0.425	0.238	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.024	0.007	-0.252	0.055	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.025	0.007	-0.317	0.242	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.025	0.009	-0.202	0.290	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.026	0.009	-0.206	0.248	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.028	0.011	-0.211	0.241	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.028	0.010	-0.209	0.448	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.028	0.011	-0.387	0.453	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.030	0.010	-0.287	0.245	71	42	287.025	138 15 47 46.530	138 15 52 33.554

461: Instrumented Blade Pitch Angle (deg)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
8.611	0.558	7.012	15.090	65	11	299.439	115 9 31 15.541	115 9 36 14.980
8.812	0.493	-75.220	40.290	65	22	299.370	115 9 47 40.501	115 9 52 39.869
8.874	0.427	-75.300	15.180	65	21	299.355	115 9 42 40.316	115 9 47 39.669
8.880	1.479	-75.260	40.370	65	12	299.224	115 9 36 15.822	115 9 41 15.046
11.190	0.850	-76.300	41.460	72	42	289.056	138 17 14 30.447	138 17 19 19.501
11.240	1.076	-76.300	41.460	72	32	261.823	138 17 4 6.242	138 17 8 28.064
11.320	1.162	-95.020	41.460	72	41	298.986	138 17 9 30.796	138 17 14 29.779
11.320	0.405	-76.300	13.050	72	11	299.750	138 16 33 30.441	138 16 38 30.191
11.420	0.722	-76.340	41.810	72	31	299.134	138 16 59 .620	138 17 3 59.754
11.460	0.756	-76.130	41.680	72	12	299.134	138 16 38 30.898	138 16 43 30.029
11.480	0.651	-76.690	13.050	72	21	298.848	138 16 46 22.829	138 16 51 21.675
11.620	0.921	-103.600	41.810	66	32	298.883	135 14 19 10.642	135 14 24 9.523
11.650	0.805	-76.780	13.050	66	22	299.046	135 14 7 40.831	135 14 12 39.875
11.660	0.711	-96.020	41.720	66	21	299.234	135 14 2 40.759	135 14 7 39.991
11.660	0.691	-76.560	41.760	72	22	299.146	138 16 51 22.508	138 16 56 21.652
11.700	0.884	-76.740	41.550	66	31	299.449	135 14 14 10.599	135 14 19 10.046
11.800	0.134	11.180	12.400	67	22	298.959	135 16 4 30.791	135 16 9 29.750
11.800	0.648	-76.740	41.460	66	12	297.491	135 13 53 45.999	135 13 58 43.489
11.820	0.484	-76.690	41.630	75	12	299.929	206 10 14 10.124	206 10 19 10.051
11.830	0.846	-76.740	41.630	66	11	299.144	135 13 48 40.309	135 13 53 39.449
11.850	0.523	-76.690	14.270	75	11	299.439	206 10 9 10.688	206 10 14 10.124
11.880	0.183	11.230	13.050	67	21	299.545	135 15 59 30.414	135 16 4 29.957
11.920	0.359	-76.610	12.700	73	42	299.532	153 12 12 50.225	153 12 17 49.755
11.930	0.168	11.440	13.090	73	41	298.510	153 12 7 50.870	153 12 12 49.378
11.930	0.160	11.400	12.570	73	32	273.569	153 12 2 26.020	153 12 6 59.586
11.940	0.319	-76.610	12.620	67	12	298.846	135 15 52 40.559	135 15 57 39.403
11.940	0.342	-76.740	12.750	67	31	299.288	135 16 11 .492	135 16 15 59.778
11.940	0.242	11.230	13.050	67	11	299.499	135 15 47 40.226	135 15 52 39.723
11.970	1.312	-76.910	48.580	71	11	299.566	138 15 9 10.495	138 15 14 10.059
11.990	0.187	11.440	13.090	73	31	298.996	153 11 57 20.314	153 12 2 19.306
12.000	0.350	-76.740	12.750	67	32	299.149	135 16 16 .526	135 16 20 59.672
12.010	0.161	11.400	12.620	70	41	299.230	135 21 57 .617	135 22 1 59.845
12.020	0.308	-76.650	12.830	73	22	292.729	153 11 51 46.543	153 11 56 39.272
12.020	0.278	-76.780	12.620	70	31	299.305	135 21 45 20.167	135 21 50 19.470
12.030	0.179	11.400	12.750	70	32	299.151	135 21 50 20.307	135 21 55 19.458
12.030	1.198	-76.910	42.070	70	22	254.371	135 21 39 45.583	135 21 43 59.952
12.030	0.197	11.440	12.700	70	21	298.798	135 21 34 40.919	135 21 39 39.715
12.030	0.180	11.400	12.660	70	42	293.820	135 22 2 5.718	135 22 6 59.537
12.050	0.377	-76.740	12.700	70	11	299.601	135 21 23 35.137	135 21 28 34.737
12.050	0.407	-76.650	13.010	68	21	359.543	135 17 19 16.509	135 17 25 16.050
12.060	1.420	-76.780	42.020	68	22	358.295	135 17 25 22.344	135 17 31 20.639
12.060	0.282	11.050	12.880	68	11	299.142	135 17 7 28.499	135 17 12 27.639
12.060	0.224	11.400	12.750	70	12	299.255	135 21 28 35.569	135 21 33 34.824
12.070	0.294	11.100	12.830	68	12	295.419	135 17 12 33.107	135 17 17 28.524
12.080	1.111	-76.820	41.940	71	12	299.574	138 15 14 10.641	138 15 19 10.211
12.140	0.254	11.490	12.880	73	21	299.151	153 11 46 40.756	153 11 51 39.904
12.170	0.431	-47.460	13.960	73	12	300.008	153 11 40 50.094	153 11 45 50.101
12.180	1.509	-77.040	57.010	69	32	293.222	135 20 35 46.647	135 20 40 39.869
12.190	1.585	-77.000	65.960	69	21	254.521	135 20 19 45.149	135 20 23 59.668
12.220	1.092	-76.950	42.020	69	31	299.679	135 20 30 40.268	135 20 35 39.945
12.230	1.764	-94.460	61.270	69	22	224.076	135 20 24 45.731	135 20 28 29.803
12.250	0.979	-77.130	42.240	71	31	298.721	138 15 31 10.531	138 15 36 9.250
12.270	0.271	11.400	13.310	73	11	298.742	153 11 35 50.765	153 11 40 49.504
12.290	0.506	-76.690	13.400	71	41	299.130	138 15 42 40.824	138 15 47 39.951
12.340	0.915	-77.080	42.070	71	21	299.176	138 15 20 15.474	138 15 25 14.650
12.350	0.744	-77.130	42.240	71	32	299.142	138 15 36 10.629	138 15 41 9.770
12.480	0.789	-77.040	42.020	71	42	287.025	138 15 47 46.530	138 15 52 33.554
12.920	0.835	-49.720	14.870	75	21	299.812	206 10 28 .205	206 10 33 .013
13.400	0.559	-74.820	14.440	75	22	289.576	206 10 33 5.072	206 10 37 54.646

402: Root Flap Bending Moment, Blade 1 (Instrumented) (N*m)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-1255.0	376.20	-2115.0	427.6	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-1134.0	358.50	-2222.0	435.5	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-1043.0	471.10	-2642.0	759.1	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-1036.0	359.70	-2264.0	2730.0	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-881.8	406.10	-2184.0	2318.0	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-871.9	336.10	-1907.0	3037.0	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-852.0	365.60	-2090.0	3051.0	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-809.1	321.60	-2156.0	412.5	70	41	299.230	135 21 57 .617	135 22 1 59.845
-792.4	633.40	-3229.0	3151.0	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-786.0	458.50	-1892.0	801.1	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-727.4	407.20	-3422.0	3151.0	75	21	299.812	206 10 28 .205	206 10 33 .013
-712.9	405.30	-2556.0	1490.0	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-656.6	453.10	-2102.0	2936.0	73	31	298.996	153 11 57 20.314	153 12 2 19.306
-640.0	490.40	-2600.0	3153.0	69	21	254.521	135 20 19 45.149	135 20 23 59.668
-606.3	426.10	-2555.0	3168.0	69	31	299.679	135 20 30 40.268	135 20 35 39.945
-579.9	372.70	-2040.0	709.2	70	42	293.820	135 22 2 5.718	135 22 6 59.537
-538.1	382.30	-1677.0	751.7	67	21	299.545	135 15 59 30.414	135 16 4 29.957
-511.7	335.40	-1693.0	596.3	70	21	298.798	135 21 34 40.919	135 21 39 39.715
-507.6	449.90	-3408.0	2964.0	70	22	254.371	135 21 39 45.583	135 21 43 59.952
-463.4	359.10	-1597.0	643.1	70	11	299.601	135 21 23 35.137	135 21 28 34.737
-424.8	568.50	-2488.0	3127.0	69	22	224.076	135 20 24 45.731	135 20 28 29.803
-413.2	456.40	-1890.0	2720.0	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-401.4	366.20	-3315.0	1953.0	75	11	299.439	206 10 9 10.688	206 10 14 10.124
-309.6	434.10	-1808.0	1046.0	70	12	299.255	135 21 28 35.569	135 21 33 34.824
-302.1	453.40	-3407.0	3180.0	66	21	299.234	135 14 2 40.759	135 14 7 39.991
-252.3	375.70	-1429.0	1407.0	65	21	299.355	115 9 42 40.316	115 9 47 39.669
-246.0	501.10	-2252.0	3224.0	72	42	289.056	138 17 14 30.447	138 17 19 19.501
-235.5	456.80	-1684.0	1531.0	65	11	299.439	115 9 31 15.541	115 9 36 14.980
-220.9	448.10	-2024.0	3169.0	66	22	299.046	135 14 7 40.831	135 14 12 39.875
-161.8	436.30	-1715.0	1495.0	67	12	298.846	135 15 52 40.559	135 15 57 39.403
-152.0	471.30	-2599.0	3145.0	65	12	299.224	115 9 36 15.822	115 9 41 15.046
-141.1	448.80	-3407.0	3175.0	66	32	298.883	135 14 19 10.642	135 14 24 9.523
-63.9	423.00	-1512.0	1679.0	67	11	299.499	135 15 47 40.226	135 15 52 39.723
15.4	757.40	-3358.0	3177.0	72	32	261.823	138 17 4 6.242	138 17 8 28.064
70.1	529.30	-1933.0	2438.0	67	31	299.288	135 16 11 .492	135 16 15 59.778
125.9	502.00	-2429.0	3169.0	66	31	299.449	135 14 14 10.599	135 14 19 10.046
193.0	411.90	-1250.0	1652.0	73	21	299.151	153 11 46 40.756	153 11 51 39.904
255.5	515.40	-2460.0	2402.0	73	12	300.008	153 11 40 50.094	153 11 45 50.101
290.7	468.90	-1422.0	2053.0	67	32	299.149	135 16 16 .526	135 16 20 59.672
296.4	473.80	-2856.0	3193.0	66	12	297.491	135 13 53 45.999	135 13 58 43.489
322.7	770.60	-1999.0	3237.0	72	41	298.986	138 17 9 30.796	138 17 14 29.779
331.0	795.70	-2130.0	3237.0	72	11	299.750	138 16 33 30.441	138 16 38 30.191
610.3	542.80	-1045.0	2926.0	73	11	298.742	153 11 35 50.765	153 11 40 49.504
639.7	613.30	-2254.0	3237.0	72	31	299.134	138 16 59 .620	138 17 3 59.754
672.7	592.80	-2291.0	3278.0	71	11	299.566	138 15 9 10.495	138 15 14 10.059
685.3	532.80	-993.8	2992.0	68	11	299.142	135 17 7 28.499	135 17 12 27.639
697.2	654.30	-2315.0	3278.0	71	12	299.574	138 15 14 10.641	138 15 19 10.211
705.7	503.50	-2674.0	3040.0	66	11	299.144	135 13 48 40.309	135 13 53 39.449
712.6	628.00	-1629.0	3126.0	68	21	359.543	135 17 19 16.509	135 17 25 16.050
722.3	683.60	-2177.0	3126.0	68	12	295.419	135 17 12 33.107	135 17 17 28.524
745.8	730.00	-2423.0	3237.0	72	21	298.848	138 16 46 22.829	138 16 51 21.675
799.6	585.10	-3393.0	3126.0	68	22	358.295	135 17 25 22.344	135 17 31 20.639
809.3	683.70	-1806.0	3237.0	72	12	299.134	138 16 38 30.898	138 16 43 30.029
1014.0	699.10	-2727.0	3278.0	71	31	298.721	138 15 31 10.531	138 15 36 9.250
1071.0	628.10	-2098.0	3278.0	71	41	299.130	138 15 42 40.824	138 15 47 39.951
1177.0	688.60	-2465.0	3278.0	71	32	299.142	138 15 36 10.629	138 15 41 9.770
1205.0	770.90	-2544.0	3237.0	72	22	299.146	138 16 51 22.508	138 16 56 21.652
1240.0	794.40	-2725.0	3278.0	71	21	299.176	138 15 20 15.474	138 15 25 14.650
1457.0	700.40	-2178.0	3278.0	71	42	287.025	138 15 47 46.530	138 15 52 33.554

403: Root Flap Bending Moment, Blade 2 (N*m)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-1365.0	375.00	-2221.0	226.1	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-1191.0	469.20	-2550.0	1657.0	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-1182.0	357.30	-2352.0	1817.0	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-1104.0	346.80	-2159.0	2145.0	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-1030.0	404.30	-2318.0	2088.0	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-939.8	362.80	-2095.0	188.4	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-904.7	333.20	-1686.0	2122.0	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-895.9	313.90	-2175.0	573.7	70	41	299.230	135 21 57 .617	135 22 1 59.845
-875.0	451.80	-2047.0	890.1	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-837.0	402.40	-2050.0	2007.0	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-795.0	624.60	-2435.0	2335.0	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-794.8	449.50	-2386.0	1857.0	73	31	298.996	153 11 57 20.314	153 12 2 19.306
-696.5	376.80	-2305.0	2034.0	75	21	299.812	206 10 28 .205	206 10 33 .013
-668.2	362.30	-2183.0	566.6	70	42	293.820	135 22 2 5.718	135 22 6 59.537
-636.0	367.60	-1894.0	627.0	67	21	299.545	135 15 59 30.414	135 16 4 29.957
-635.7	476.70	-2435.0	2428.0	69	21	254.521	135 20 19 45.149	135 20 23 59.668
-598.8	415.20	-2435.0	2415.0	69	31	299.679	135 20 30 40.268	135 20 35 39.945
-597.3	323.70	-1902.0	873.5	70	21	298.798	135 21 34 40.919	135 21 39 39.715
-594.0	438.40	-2498.0	2194.0	70	22	254.371	135 21 39 45.583	135 21 43 59.952
-552.8	346.30	-1770.0	529.7	70	11	299.601	135 21 23 35.137	135 21 28 34.737
-550.9	445.00	-1973.0	1058.0	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-527.8	354.40	-1880.0	2164.0	75	11	299.439	206 10 9 10.688	206 10 14 10.124
-424.7	557.10	-2073.0	2348.0	69	22	224.076	135 20 24 45.731	135 20 28 29.803
-401.8	424.80	-1944.0	1066.0	70	12	299.255	135 21 28 35.569	135 21 33 34.824
-350.0	489.50	-1807.0	2316.0	72	42	289.056	138 17 14 30.447	138 17 19 19.501
-340.4	442.80	-2467.0	2059.0	66	21	299.234	135 14 2 40.759	135 14 7 39.991
-289.9	349.10	-1485.0	1281.0	65	21	299.355	115 9 42 40.316	115 9 47 39.669
-289.9	435.10	-1758.0	2009.0	65	11	299.439	115 9 31 15.541	115 9 36 14.980
-262.1	416.60	-1782.0	1957.0	67	12	298.846	135 15 52 40.559	135 15 57 39.403
-262.0	431.50	-1693.0	2379.0	66	22	299.046	135 14 7 40.831	135 14 12 39.875
-198.6	445.30	-1767.0	2160.0	65	12	299.224	115 9 36 15.822	115 9 41 15.046
-187.9	425.00	-2394.0	2149.0	66	32	298.883	135 14 19 10.642	135 14 24 9.523
-170.0	398.70	-1583.0	1970.0	67	11	299.499	135 15 47 40.226	135 15 52 39.723
-100.1	731.70	-2470.0	2401.0	72	32	261.823	138 17 4 6.242	138 17 8 28.064
-39.7	514.50	-2104.0	2012.0	67	31	299.288	135 16 11 .492	135 16 15 59.778
43.0	385.20	-1604.0	1541.0	73	21	299.151	153 11 46 40.756	153 11 51 39.904
71.2	475.10	-1762.0	2360.0	66	31	299.449	135 14 14 10.599	135 14 19 10.046
99.6	487.70	-1789.0	2001.0	73	12	300.008	153 11 40 50.094	153 11 45 50.101
173.0	442.20	-1575.0	1839.0	67	32	299.149	135 16 16 .526	135 16 20 59.672
184.8	762.00	-2406.0	2401.0	72	11	299.750	138 16 33 30.441	138 16 38 30.191
185.9	710.90	-1832.0	2401.0	72	41	298.986	138 17 9 30.796	138 17 14 29.779
246.3	450.10	-1975.0	2370.0	66	12	297.491	135 13 53 45.999	135 13 58 43.489
422.0	505.70	-1193.0	2321.0	73	11	298.742	153 11 35 50.765	153 11 40 49.504
496.7	589.50	-1612.0	2401.0	72	31	299.134	138 16 59 .620	138 17 3 59.754
514.1	551.80	-2215.0	2423.0	71	11	299.566	138 15 9 10.495	138 15 14 10.059
531.9	616.60	-2368.0	2423.0	71	12	299.574	138 15 14 10.641	138 15 19 10.211
585.8	685.40	-2021.0	2401.0	72	21	298.848	138 16 46 22.829	138 16 51 21.675
639.1	476.50	-1557.0	2404.0	66	11	299.144	135 13 48 40.309	135 13 53 39.449
647.1	630.90	-1637.0	2401.0	72	12	299.134	138 16 38 30.898	138 16 43 30.029
664.2	494.40	-913.5	2457.0	68	11	299.142	135 17 7 28.499	135 17 12 27.639
697.4	600.80	-1440.0	2457.0	68	21	359.543	135 17 19 16.509	135 17 25 16.050
700.6	644.50	-2364.0	2457.0	68	12	295.419	135 17 12 33.107	135 17 17 28.524
773.1	548.00	-1894.0	2457.0	68	22	358.295	135 17 25 22.344	135 17 31 20.639
803.6	658.00	-2279.0	2423.0	71	31	298.721	138 15 31 10.531	138 15 36 9.250
851.4	598.40	-1653.0	2423.0	71	41	299.130	138 15 42 40.824	138 15 47 39.951
946.3	650.40	-2448.0	2423.0	71	32	299.142	138 15 36 10.629	138 15 41 9.770
999.9	722.10	-2025.0	2401.0	72	22	299.146	138 16 51 22.508	138 16 56 21.652
1002.0	741.50	-2159.0	2423.0	71	21	299.176	138 15 20 15.474	138 15 25 14.650
1186.0	657.70	-1964.0	2423.0	71	42	287.025	138 15 47 46.530	138 15 52 33.554

404: Root Flap Bending Moment, Blade 3 (N*m)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-1217.0	387.60	-2086.0	236.7	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-1072.0	478.80	-2388.0	801.4	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-1058.0	367.60	-2275.0	1514.0	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-1028.0	365.60	-2057.0	1809.0	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-903.8	414.60	-2388.0	1636.0	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-862.0	631.80	-2460.0	2017.0	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-831.9	370.60	-1946.0	212.6	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-793.7	341.20	-1572.0	2071.0	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-789.2	326.50	-2130.0	664.9	70	41	299.230	135 21 57 .617	135 22 1 59.845
-767.0	460.80	-1973.0	908.7	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-707.7	479.80	-2460.0	2015.0	69	21	254.521	135 20 19 45.149	135 20 23 59.668
-673.2	453.40	-2041.0	1730.0	73	31	298.996	153 11 57 20.314	153 12 2 19.306
-668.0	426.20	-2460.0	1958.0	69	31	299.679	135 20 30 40.268	135 20 35 39.945
-621.3	405.20	-2211.0	2107.0	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-618.6	402.10	-2067.0	1926.0	75	21	299.812	206 10 28 .205	206 10 33 .013
-563.1	367.80	-1984.0	1572.0	70	42	293.820	135 22 2 5.718	135 22 6 59.537
-495.3	331.80	-1626.0	716.6	70	21	298.798	135 21 34 40.919	135 21 39 39.715
-491.8	562.90	-2235.0	2023.0	69	22	224.076	135 20 24 45.731	135 20 28 29.803
-488.7	440.90	-1790.0	2120.0	70	22	254.371	135 21 39 45.583	135 21 43 59.952
-485.0	378.70	-1664.0	753.9	67	21	299.545	135 15 59 30.414	135 16 4 29.957
-454.2	352.60	-1634.0	739.6	70	11	299.601	135 21 23 35.137	135 21 28 34.737
-430.1	449.00	-1746.0	1904.0	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-319.5	356.40	-2256.0	2146.0	75	11	299.439	206 10 9 10.688	206 10 14 10.124
-317.1	490.30	-1926.0	2093.0	72	42	289.056	138 17 14 30.447	138 17 19 19.501
-307.9	444.00	-2300.0	2080.0	66	21	299.234	135 14 2 40.759	135 14 7 39.991
-301.9	429.90	-1840.0	1729.0	70	12	299.255	135 21 28 35.569	135 21 33 34.824
-226.8	434.50	-1936.0	2088.0	66	22	299.046	135 14 7 40.831	135 14 12 39.875
-186.4	351.90	-1320.0	1691.0	65	21	299.355	115 9 42 40.316	115 9 47 39.669
-182.3	435.40	-1706.0	1291.0	65	11	299.439	115 9 31 15.541	115 9 36 14.980
-158.9	428.80	-1644.0	2097.0	66	32	298.883	135 14 19 10.642	135 14 24 9.523
-117.0	422.70	-1712.0	1512.0	67	12	298.846	135 15 52 40.559	135 15 57 39.403
-95.7	445.90	-1724.0	2093.0	65	12	299.224	115 9 36 15.822	115 9 41 15.046
-71.6	729.00	-2401.0	2095.0	72	32	261.823	138 17 4 6.242	138 17 8 28.064
-27.2	403.90	-1286.0	1531.0	67	11	299.499	135 15 47 40.226	135 15 52 39.723
97.9	515.10	-2244.0	2158.0	67	31	299.288	135 16 11 .492	135 16 15 59.778
100.2	478.20	-1958.0	2061.0	66	31	299.449	135 14 14 10.599	135 14 19 10.046
159.1	396.20	-1305.0	1743.0	73	21	299.151	153 11 46 40.756	153 11 51 39.904
210.3	709.90	-2169.0	2095.0	72	41	298.986	138 17 9 30.796	138 17 14 29.779
213.6	489.00	-2242.0	2108.0	73	12	300.008	153 11 40 50.094	153 11 45 50.101
221.9	756.60	-2353.0	2095.0	72	11	299.750	138 16 33 30.441	138 16 38 30.191
266.1	461.80	-1638.0	2009.0	66	12	297.491	135 13 53 45.999	135 13 58 43.489
303.6	450.30	-1208.0	2158.0	67	32	299.149	135 16 16 .526	135 16 20 59.672
512.8	578.50	-1605.0	2095.0	72	31	299.134	138 16 59 .620	138 17 3 59.754
535.9	508.80	-1318.0	2108.0	73	11	298.742	153 11 35 50.765	153 11 40 49.504
539.7	549.80	-1922.0	2120.0	71	11	299.566	138 15 9 10.495	138 15 14 10.059
562.9	615.90	-2068.0	2120.0	71	12	299.574	138 15 14 10.641	138 15 19 10.211
605.2	667.10	-2011.0	2095.0	72	21	298.848	138 16 46 22.829	138 16 51 21.675
635.0	470.80	-1564.0	2099.0	66	11	299.144	135 13 48 40.309	135 13 53 39.449
663.6	620.00	-1702.0	2095.0	72	12	299.134	138 16 38 30.898	138 16 43 30.029
731.8	492.20	-935.2	2188.0	68	11	299.142	135 17 7 28.499	135 17 12 27.639
757.8	598.50	-1663.0	2188.0	68	21	359.543	135 17 19 16.509	135 17 25 16.050
768.2	646.60	-2079.0	2188.0	68	12	295.419	135 17 12 33.107	135 17 17 28.524
833.8	640.90	-2377.0	2120.0	71	31	298.721	138 15 31 10.531	138 15 36 9.250
835.9	549.60	-2042.0	2188.0	68	22	358.295	135 17 25 22.344	135 17 31 20.639
881.6	583.90	-2242.0	2120.0	71	41	299.130	138 15 42 40.824	138 15 47 39.951
989.7	628.10	-1823.0	2120.0	71	32	299.142	138 15 36 10.629	138 15 41 9.770
1017.0	697.30	-1986.0	2095.0	72	22	299.146	138 16 51 22.508	138 16 56 21.652
1040.0	714.70	-2377.0	2120.0	71	21	299.176	138 15 20 15.474	138 15 25 14.650
1226.0	610.00	-1733.0	2120.0	71	42	287.025	138 15 47 46.530	138 15 52 33.554

414: Root Edge Bending Moment (N*m)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
158.8	697.0	-1341	3027	65	22	299.370	115 9 47 40.501	115 9 52 39.869
189.8	698.9	-2111	2933	75	12	299.929	206 10 14 10.124	206 10 19 10.051
197.7	690.7	-2218	2698	75	22	289.576	206 10 33 5.072	206 10 37 54.646
198.7	691.6	-1685	3154	73	32	273.569	153 12 2 26.020	153 12 6 59.586
204.8	694.5	-1130	1608	73	41	298.510	153 12 7 50.870	153 12 12 49.378
207.6	692.3	-1061	1868	73	42	299.532	153 12 12 50.225	153 12 17 49.755
209.5	709.4	-2335	3053	66	21	299.234	135 14 2 40.759	135 14 7 39.991
210.5	688.1	-973	1492	67	22	298.959	135 16 4 30.791	135 16 9 29.750
212.0	699.2	-2221	2182	75	21	299.812	206 10 28 .205	206 10 33 .013
219.3	690.6	-1004	1640	70	41	299.230	135 21 57 .617	135 22 1 59.845
221.4	691.3	-1001	1550	70	31	299.305	135 21 45 20.167	135 21 50 19.470
222.3	700.9	-1881	2650	73	31	298.996	153 11 57 20.314	153 12 2 19.306
222.6	708.3	-3159	3016	75	11	299.439	206 10 9 10.688	206 10 14 10.124
222.8	714.1	-1764	2924	66	22	299.046	135 14 7 40.831	135 14 12 39.875
226.8	710.2	-1162	2750	65	11	299.439	115 9 31 15.541	115 9 36 14.980
227.4	709.2	-1206	1836	65	21	299.355	115 9 42 40.316	115 9 47 39.669
229.3	694.4	-1097	1840	70	32	299.151	135 21 50 20.307	135 21 55 19.458
235.2	714.2	-2187	3144	66	32	298.883	135 14 19 10.642	135 14 24 9.523
239.9	701.5	-1074	1786	67	21	299.545	135 15 59 30.414	135 16 4 29.957
240.8	698.2	-1105	1780	70	42	293.820	135 22 2 5.718	135 22 6 59.537
243.9	712.9	-2190	3145	72	42	289.056	138 17 14 30.447	138 17 19 19.501
245.9	698.3	-1109	1796	70	21	298.798	135 21 34 40.919	135 21 39 39.715
247.7	713.9	-2307	3203	65	12	299.224	115 9 36 15.822	115 9 41 15.046
249.2	706.5	-1234	1970	73	22	292.729	153 11 51 46.543	153 11 56 39.272
252.5	703.7	-2355	2756	70	22	254.371	135 21 39 45.583	135 21 43 59.952
253.7	701.9	-1097	1788	70	11	299.601	135 21 23 35.137	135 21 28 34.737
254.5	700.9	-2405	2813	69	21	254.521	135 20 19 45.149	135 20 23 59.668
254.9	702.3	-3208	3166	69	31	299.679	135 20 30 40.268	135 20 35 39.945
259.8	703.2	-2249	2810	69	32	293.222	135 20 35 46.647	135 20 40 39.869
275.4	707.6	-1114	1843	70	12	299.255	135 21 28 35.569	135 21 33 34.824
279.2	710.8	-2772	3092	69	22	224.076	135 20 24 45.731	135 20 28 29.803
280.1	720.3	-2052	3041	66	31	299.449	135 14 14 10.599	135 14 19 10.046
291.3	716.6	-1473	2035	67	12	298.846	135 15 52 40.559	135 15 57 39.403
294.1	734.7	-2058	3164	72	32	261.823	138 17 4 6.242	138 17 8 28.064
306.2	719.4	-1204	2085	67	11	299.499	135 15 47 40.226	135 15 52 39.723
312.2	754.0	-2597	3061	71	12	299.574	138 15 14 10.641	138 15 19 10.211
312.5	725.3	-2175	3085	66	12	297.491	135 13 53 45.999	135 13 58 43.489
313.5	753.5	-2534	3092	71	11	299.566	138 15 9 10.495	138 15 14 10.059
331.3	728.5	-1393	2052	67	31	299.288	135 16 11 .492	135 16 15 59.778
331.8	743.2	-2365	3107	72	41	298.986	138 17 9 30.796	138 17 14 29.779
340.9	724.2	-2236	2175	73	21	299.151	153 11 46 40.756	153 11 51 39.904
345.4	751.1	-2128	3176	72	11	299.750	138 16 33 30.441	138 16 38 30.191
354.8	726.3	-2340	2424	73	12	300.008	153 11 40 50.094	153 11 45 50.101
359.5	767.2	-2142	2913	71	31	298.721	138 15 31 10.531	138 15 36 9.250
364.5	789.6	-2887	3092	71	21	299.176	138 15 20 15.474	138 15 25 14.650
364.6	732.9	-1315	2502	68	11	299.142	135 17 7 28.499	135 17 12 27.639
364.9	772.3	-2062	3078	71	32	299.142	138 15 36 10.629	138 15 41 9.770
365.2	752.2	-1680	2666	68	12	295.419	135 17 12 33.107	135 17 17 28.524
367.3	753.0	-1768	2597	68	21	359.543	135 17 19 16.509	135 17 25 16.050
369.5	729.9	-1321	2137	67	32	299.149	135 16 16 .526	135 16 20 59.672
371.1	772.0	-2005	3045	71	41	299.130	138 15 42 40.824	138 15 47 39.951
383.4	751.8	-2861	3093	68	22	358.295	135 17 25 22.344	135 17 31 20.639
383.5	793.9	-2003	3092	71	42	287.025	138 15 47 46.530	138 15 52 33.554
387.5	730.1	-1899	3100	66	11	299.144	135 13 48 40.309	135 13 53 39.449
392.7	739.0	-2179	3118	72	31	299.134	138 16 59 .620	138 17 3 59.754
399.9	745.9	-2593	2662	72	21	298.848	138 16 46 22.829	138 16 51 21.675
407.3	757.0	-2193	2972	72	12	299.134	138 16 38 30.898	138 16 43 30.029
416.9	734.2	-1667	2965	73	11	298.742	153 11 35 50.765	153 11 40 49.504
443.8	796.0	-2135	3126	72	22	299.146	138 16 51 22.508	138 16 56 21.652

423: Low Speed Shaft Torque (N*m)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-277.6	235.8	-2454	739	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-228.5	223.6	-3202	5925	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-209.1	324.5	-1441	4559	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-189.3	237.2	-2399	5769	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-102.1	263.8	-1219	5709	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-100.0	226.3	-3543	5895	65	22	299.370	115 9 47 40.501	115 9 52 39.869
7.3	258.3	-615	5747	70	31	299.305	135 21 45 20.167	135 21 50 19.470
31.7	192.6	-361	829	70	41	299.230	135 21 57 .617	135 22 1 59.845
78.0	357.5	-483	5321	70	32	299.151	135 21 50 20.307	135 21 55 19.458
117.2	357.5	-3759	2957	73	31	298.996	153 11 57 20.314	153 12 2 19.306
132.0	535.9	-3775	5906	69	32	293.222	135 20 35 46.647	135 20 40 39.869
141.5	320.2	-3645	5236	75	12	299.929	206 10 14 10.124	206 10 19 10.051
193.9	307.2	-4358	4038	75	21	299.812	206 10 28 .205	206 10 33 .013
233.0	405.8	-4071	5996	69	21	254.521	135 20 19 45.149	135 20 23 59.668
257.3	343.2	-6258	5948	69	31	299.679	135 20 30 40.268	135 20 35 39.945
261.6	256.1	-474	2752	70	42	293.820	135 22 2 5.718	135 22 6 59.537
321.4	294.2	-336	2458	67	21	299.545	135 15 59 30.414	135 16 4 29.957
328.1	205.7	-247	2895	70	21	298.798	135 21 34 40.919	135 21 39 39.715
352.7	370.9	-3934	3554	70	22	254.371	135 21 39 45.583	135 21 43 59.952
372.5	405.9	-395	2756	73	22	292.729	153 11 51 46.543	153 11 56 39.272
383.1	258.8	-253	2847	70	11	299.601	135 21 23 35.137	135 21 28 34.737
438.1	373.1	-290	2726	65	11	299.439	115 9 31 15.541	115 9 36 14.980
438.8	290.7	-188	2840	65	21	299.355	115 9 42 40.316	115 9 47 39.669
459.2	283.9	-4867	5895	75	11	299.439	206 10 9 10.688	206 10 14 10.124
471.7	507.4	-5551	5888	69	22	224.076	135 20 24 45.731	135 20 28 29.803
488.9	328.4	-5784	4519	66	21	299.234	135 14 2 40.759	135 14 7 39.991
546.7	401.6	-4067	3655	65	12	299.224	115 9 36 15.822	115 9 41 15.046
550.4	434.2	-4032	5963	72	42	289.056	138 17 14 30.447	138 17 19 19.501
561.6	325.6	-370	2943	70	12	299.255	135 21 28 35.569	135 21 33 34.824
586.3	361.9	-3763	2857	66	22	299.046	135 14 7 40.831	135 14 12 39.875
672.5	398.1	-5710	3516	66	32	298.883	135 14 19 10.642	135 14 24 9.523
747.2	325.3	-52	2578	67	12	298.846	135 15 52 40.559	135 15 57 39.403
859.1	336.7	-31	2797	67	11	299.499	135 15 47 40.226	135 15 52 39.723
869.1	694.8	-4098	3507	72	32	261.823	138 17 4 6.242	138 17 8 28.064
996.4	461.6	-4020	3765	66	31	299.449	135 14 14 10.599	135 14 19 10.046
1030.0	417.6	-330	2734	67	31	299.288	135 16 11 .492	135 16 15 59.778
1086.0	318.9	75	2007	73	21	299.151	153 11 46 40.756	153 11 51 39.904
1167.0	459.0	-3843	2366	73	12	300.008	153 11 40 50.094	153 11 45 50.101
1174.0	715.4	-5530	4103	72	41	298.986	138 17 9 30.796	138 17 14 29.779
1224.0	724.8	-501	5981	72	11	299.750	138 16 33 30.441	138 16 38 30.191
1232.0	388.9	-4140	3417	66	12	297.491	135 13 53 45.999	135 13 58 43.489
1302.0	366.9	275	2749	67	32	299.149	135 16 16 .526	135 16 20 59.672
1570.0	502.5	-3742	3899	72	31	299.134	138 16 59 .620	138 17 3 59.754
1583.0	450.5	108	2624	73	11	298.742	153 11 35 50.765	153 11 40 49.504
1595.0	452.0	-4417	3943	71	11	299.566	138 15 9 10.495	138 15 14 10.059
1620.0	505.0	-4423	3898	71	12	299.574	138 15 14 10.641	138 15 19 10.211
1642.0	596.8	-4392	2914	72	21	298.848	138 16 46 22.829	138 16 51 21.675
1708.0	510.9	-4062	4082	72	12	299.134	138 16 38 30.898	138 16 43 30.029
1723.0	381.8	-4338	5074	66	11	299.144	135 13 48 40.309	135 13 53 39.449
1782.0	391.7	501	2757	68	11	299.142	135 17 7 28.499	135 17 12 27.639
1813.0	498.7	-236	2865	68	12	295.419	135 17 12 33.107	135 17 17 28.524
1817.0	390.7	-3462	2912	68	21	359.543	135 17 19 16.509	135 17 25 16.050
1910.0	369.9	-4432	4182	68	22	358.295	135 17 25 22.344	135 17 31 20.639
1963.0	470.1	-4474	4754	71	31	298.721	138 15 31 10.531	138 15 36 9.250
2034.0	345.0	-4022	3254	71	41	299.130	138 15 42 40.824	138 15 47 39.951
2073.0	438.3	-4508	4229	72	22	299.146	138 16 51 22.508	138 16 56 21.652
2096.0	417.9	-4435	4308	71	32	299.142	138 15 36 10.629	138 15 41 9.770
2132.0	519.0	-4606	4248	71	21	299.176	138 15 20 15.474	138 15 25 14.650
2336.0	301.3	-4525	4317	71	42	287.025	138 15 47 46.530	138 15 52 33.554

308: Generator Power (kW)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-3.278	1.724	-5.728	3.813	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-2.667	1.716	-5.635	3.340	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-2.585	1.569	-5.713	1.759	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-2.312	2.354	-5.830	6.754	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-1.602	1.588	-4.184	2.215	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-1.561	1.904	-20.790	3.574	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-1.138	1.872	-5.409	3.078	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-0.907	1.382	-3.575	4.482	70	41	299.230	135 21 57 .617	135 22 1 59.845
-0.617	2.571	-4.551	7.077	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-0.436	2.293	-4.971	18.690	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-0.324	3.788	-6.276	7.673	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-0.115	2.553	-4.698	8.841	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.047	2.182	-22.060	6.383	75	21	299.812	206 10 28 .205	206 10 33 .013
0.326	2.771	-5.945	7.322	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.610	2.367	-4.638	7.478	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.761	1.822	-4.531	6.824	70	42	293.820	135 22 2 5.718	135 22 6 59.537
1.071	2.083	-3.523	5.842	67	21	299.545	135 15 59 30.414	135 16 4 29.957
1.146	1.448	-2.814	20.130	70	21	298.798	135 21 34 40.919	135 21 39 39.715
1.324	2.569	-3.731	9.262	70	22	254.371	135 21 39 45.583	135 21 43 59.952
1.413	1.809	-2.970	6.355	70	11	299.601	135 21 23 35.137	135 21 28 34.737
1.626	2.872	-3.781	20.780	73	22	292.729	153 11 51 46.543	153 11 56 39.272
1.691	1.950	-3.567	19.570	75	11	299.439	206 10 9 10.688	206 10 14 10.124
1.749	2.613	-3.287	9.297	65	11	299.439	115 9 31 15.541	115 9 36 14.980
2.069	3.514	-4.579	9.136	69	22	224.076	135 20 24 45.731	135 20 28 29.803
2.154	2.002	-2.155	7.931	65	21	299.355	115 9 42 40.316	115 9 47 39.669
2.569	2.266	-3.024	9.072	66	21	299.234	135 14 2 40.759	135 14 7 39.991
2.688	2.701	-3.911	9.668	65	12	299.224	115 9 36 15.822	115 9 41 15.046
2.738	2.301	-3.868	8.443	70	12	299.255	135 21 28 35.569	135 21 33 34.824
2.982	3.020	-23.690	11.230	72	42	289.056	138 17 14 30.447	138 17 19 19.501
3.315	2.504	-3.590	11.300	66	22	299.046	135 14 7 40.831	135 14 12 39.875
4.024	2.711	-1.600	11.530	66	32	298.883	135 14 19 10.642	135 14 24 9.523
4.027	2.253	-1.377	11.790	67	12	298.846	135 15 52 40.559	135 15 57 39.403
4.765	2.334	-1.162	18.430	67	11	299.499	135 15 47 40.226	135 15 52 39.723
5.136	4.778	-5.648	16.550	72	32	261.823	138 17 4 6.242	138 17 8 28.064
6.109	2.874	-3.367	13.780	67	31	299.288	135 16 11 .492	135 16 15 59.778
6.214	3.169	-1.385	14.070	66	31	299.449	135 14 14 10.599	135 14 19 10.046
6.546	2.212	-0.367	12.430	73	21	299.151	153 11 46 40.756	153 11 51 39.904
6.946	3.189	-1.440	14.050	73	12	300.008	153 11 40 50.094	153 11 45 50.101
7.246	4.822	-0.868	20.790	72	41	298.986	138 17 9 30.796	138 17 14 29.779
7.385	5.030	-4.790	17.470	72	11	299.750	138 16 33 30.441	138 16 38 30.191
7.554	2.645	-0.780	13.810	66	12	297.491	135 13 53 45.999	135 13 58 43.489
7.979	2.479	1.003	14.500	67	32	299.149	135 16 16 .526	135 16 20 59.672
9.578	3.063	-0.543	16.270	73	11	298.742	153 11 35 50.765	153 11 40 49.504
9.580	3.019	-10.150	16.440	71	11	299.566	138 15 9 10.495	138 15 14 10.059
9.839	3.394	-20.920	18.630	71	12	299.574	138 15 14 10.641	138 15 19 10.211
9.946	3.350	0.185	17.960	72	31	299.134	138 16 59 .620	138 17 3 59.754
10.370	4.005	-0.283	18.170	72	21	298.848	138 16 46 22.829	138 16 51 21.675
10.740	2.550	2.868	17.050	66	11	299.144	135 13 48 40.309	135 13 53 39.449
10.760	3.379	1.005	19.190	72	12	299.134	138 16 38 30.898	138 16 43 30.029
11.230	2.587	2.583	16.900	68	11	299.142	135 17 7 28.499	135 17 12 27.639
11.450	3.325	-5.436	17.900	68	12	295.419	135 17 12 33.107	135 17 17 28.524
11.510	2.553	1.608	16.980	68	21	359.543	135 17 19 16.509	135 17 25 16.050
12.150	2.329	3.617	18.780	68	22	358.295	135 17 25 22.344	135 17 31 20.639
12.420	3.074	-26.520	20.350	71	31	298.721	138 15 31 10.531	138 15 36 9.250
12.970	2.201	3.802	19.900	71	41	299.130	138 15 42 40.824	138 15 47 39.951
13.250	2.834	1.005	18.990	72	22	299.146	138 16 51 22.508	138 16 56 21.652
13.330	2.679	2.105	20.850	71	32	299.142	138 15 36 10.629	138 15 41 9.770
13.380	3.420	0.017	21.010	71	21	299.176	138 15 20 15.474	138 15 25 14.650
14.950	1.825	5.734	20.500	71	42	287.025	138 15 47 46.530	138 15 52 33.554

714: Normal Force Coefficient, Cn, at 80% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.271	0.218	-0.873	0.521	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-0.183	0.186	-5.730	13.800	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-0.020	0.181	-4.496	3.949	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-0.007	0.067	-7.506	12.540	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-0.006	0.035	-5.611	1.676	73	31	298.996	153 11 57 20.314	153 12 2 19.306
-0.001	0.035	-3.394	3.843	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-0.001	0.033	-3.407	6.484	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.001	0.032	-3.731	4.135	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.002	0.034	-4.890	3.858	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.003	0.037	-9.800	2.623	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.005	0.132	-3.768	0.990	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.006	0.032	-6.057	2.274	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.020	0.227	-5.227	0.880	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.028	0.401	-8.290	11.720	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.031	0.203	-8.457	5.109	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.087	0.223	-8.892	5.578	75	21	299.812	206 10 28 .205	206 10 33 .013
0.105	0.224	-10.530	5.391	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.127	0.301	-9.096	10.870	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.140	0.168	-3.703	3.732	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.146	0.247	-8.958	9.929	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.178	0.137	-0.367	4.132	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.180	0.245	-10.280	11.930	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.196	0.195	-3.629	3.153	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.199	0.162	-3.746	6.149	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.255	0.339	-11.700	9.125	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.290	0.194	-3.630	4.238	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.308	0.206	-23.270	14.220	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.324	0.217	-9.943	12.120	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.343	0.295	-11.110	13.050	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.370	0.226	-7.351	7.474	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.412	0.236	-3.636	4.294	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.420	0.240	-13.130	7.611	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.423	0.193	-3.704	4.881	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.435	0.191	-4.390	4.103	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.463	0.302	-9.957	11.410	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.478	0.387	-8.492	7.271	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.494	0.195	-3.422	10.950	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.539	0.183	-8.926	10.910	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.562	0.242	-9.131	9.879	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.571	0.232	-4.434	3.714	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.615	0.350	-10.510	10.690	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.631	0.168	-6.226	6.268	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.653	0.388	-5.730	4.334	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.691	0.182	-3.299	4.310	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.779	0.247	-10.270	11.370	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.782	0.245	-10.850	8.860	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.797	0.254	-7.624	8.327	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.809	0.280	-7.443	7.380	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.831	0.226	-3.235	1.557	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.839	0.172	-3.502	3.439	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.852	0.188	-1.641	3.107	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.852	0.244	-9.865	9.573	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.871	0.221	-10.470	11.170	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.881	0.235	-7.405	8.669	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.894	0.188	-7.917	5.899	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.914	0.207	-7.342	6.957	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.921	0.256	-9.768	10.010	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.952	0.249	-5.917	6.340	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.972	0.206	-10.710	4.258	71	42	287.025	138 15 47 46.530	138 15 52 33.554

715: Normal Force Coefficient, Cn, at 63.3% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.125	0.275	-0.776	0.825	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.035	0.234	-7.246	8.634	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.039	0.336	-0.914	1.082	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.046	0.245	-0.904	4.449	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.173	0.268	-0.688	1.033	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.180	0.250	-0.392	5.154	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.187	0.234	-2.491	13.490	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.219	0.169	-0.471	0.939	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.226	0.274	-0.531	1.059	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.228	0.447	-2.056	5.301	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.351	0.277	-0.474	1.300	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.362	0.300	-0.861	1.111	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.366	0.266	-1.661	12.130	75	21	299.812	206 10 28 .205	206 10 33 .013
0.371	0.267	-6.256	6.795	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.387	0.263	-5.753	2.577	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.387	0.201	-9.448	1.052	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.410	0.161	-0.259	4.271	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.423	0.241	-0.362	1.230	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.431	0.160	-2.998	1.008	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.454	0.235	-0.389	1.780	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.463	0.182	-0.220	0.976	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.507	0.334	-2.966	1.270	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.536	0.268	-2.924	1.326	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.558	0.217	-0.436	3.295	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.584	0.215	-0.729	3.512	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.597	0.211	-0.258	1.435	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.598	0.291	-11.840	1.306	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.605	0.202	-6.318	7.910	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.619	0.210	-0.294	1.349	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.633	0.254	-3.763	1.500	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.642	0.201	-2.252	1.300	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.663	0.191	-0.052	1.271	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.694	0.252	-8.857	3.573	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.696	0.388	-1.183	2.446	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.702	0.194	-3.070	1.402	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.758	0.191	-7.833	1.845	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.771	0.191	-0.745	2.051	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.810	0.236	-0.624	1.662	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.841	0.399	-13.380	2.290	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.850	0.310	-2.834	2.209	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.887	0.215	-0.014	3.066	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.889	0.167	-5.407	2.309	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.904	0.175	0.111	2.689	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.978	0.217	-4.855	6.848	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.982	0.228	-0.955	2.210	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.991	0.261	-0.029	2.110	72	31	299.134	138 16 59 .620	138 17 3 59.754
1.003	0.197	-2.586	2.296	73	11	298.742	153 11 35 50.765	153 11 40 49.504
1.015	0.252	-0.324	2.040	68	12	295.419	135 17 12 33.107	135 17 17 28.524
1.019	0.280	-6.479	2.309	72	21	298.848	138 16 46 22.829	138 16 51 21.675
1.020	0.200	-7.818	6.760	68	11	299.142	135 17 7 28.499	135 17 12 27.639
1.030	0.218	-3.810	2.289	68	21	359.543	135 17 19 16.509	135 17 25 16.050
1.030	0.238	-2.785	2.621	72	12	299.134	138 16 38 30.898	138 16 43 30.029
1.048	0.209	-6.577	1.950	68	22	358.295	135 17 25 22.344	135 17 31 20.639
1.069	0.245	-5.900	3.685	71	31	298.721	138 15 31 10.531	138 15 36 9.250
1.090	0.223	-4.164	8.273	71	41	299.130	138 15 42 40.824	138 15 47 39.951
1.095	0.232	-3.391	5.846	71	32	299.142	138 15 36 10.629	138 15 41 9.770
1.096	0.263	-2.584	7.424	71	21	299.176	138 15 20 15.474	138 15 25 14.650
1.141	0.270	-0.121	2.434	72	22	299.146	138 16 51 22.508	138 16 56 21.652
1.158	0.232	-1.732	5.435	71	42	287.025	138 15 47 46.530	138 15 52 33.554

716: Normal Force Coefficient, Cn, at 46.6% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.082	0.352	-5.255	1.303	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.248	0.187	-4.030	4.637	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.272	0.398	-0.935	2.843	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.311	0.304	-1.654	7.114	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.327	0.285	-8.915	10.830	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.339	0.183	-3.560	4.077	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.410	0.324	-6.519	5.699	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.440	0.312	-4.136	1.455	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.468	0.480	-1.725	4.902	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.500	0.293	-0.714	4.268	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.530	0.308	-3.396	1.431	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.550	0.209	-2.758	1.306	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.632	0.312	-8.605	7.854	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.636	0.283	-11.270	7.392	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.652	0.270	-10.900	8.417	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.668	0.262	-7.538	9.289	75	21	299.812	206 10 28 .205	206 10 33 .013
0.697	0.280	-5.433	8.812	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.714	0.218	-0.593	1.490	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.737	0.242	-3.498	9.847	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.742	0.227	-0.673	1.442	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.752	0.324	-7.579	6.002	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.760	0.175	-5.552	1.735	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.781	0.177	-0.112	1.334	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.785	0.239	-0.116	1.665	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.826	0.285	-8.504	8.243	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.831	0.228	-6.762	10.640	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.838	0.237	-5.574	7.617	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.845	0.219	-0.387	2.011	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.851	0.188	-7.403	6.608	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.870	0.195	-4.139	8.993	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.877	0.217	-7.468	5.002	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.879	0.355	-1.250	2.482	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.884	0.239	-7.646	8.582	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.896	0.183	-0.842	1.778	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.927	0.197	-0.146	7.435	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.944	0.199	-6.073	7.698	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.949	0.346	-6.524	8.133	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.962	0.202	-2.726	2.019	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.962	0.256	-1.835	2.109	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.976	0.245	-6.870	9.923	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.984	0.229	-5.819	8.892	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.986	0.217	-6.967	7.700	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.998	0.219	-1.009	5.214	67	32	299.149	135 16 16 .526	135 16 20 59.672
1.007	0.201	-6.097	6.096	73	11	298.742	153 11 35 50.765	153 11 40 49.504
1.027	0.235	-7.025	6.884	71	11	299.566	138 15 9 10.495	138 15 14 10.059
1.028	0.191	-5.981	5.734	68	11	299.142	135 17 7 28.499	135 17 12 27.639
1.037	0.238	-6.504	9.532	71	12	299.574	138 15 14 10.641	138 15 19 10.211
1.053	0.237	-0.722	2.560	68	12	295.419	135 17 12 33.107	135 17 17 28.524
1.057	0.258	-5.587	3.830	72	31	299.134	138 16 59 .620	138 17 3 59.754
1.060	0.246	-7.820	7.253	72	21	298.848	138 16 46 22.829	138 16 51 21.675
1.061	0.222	0.146	2.434	72	12	299.134	138 16 38 30.898	138 16 43 30.029
1.069	0.211	-6.404	7.719	68	22	358.295	135 17 25 22.344	135 17 31 20.639
1.075	0.225	-2.281	3.701	68	21	359.543	135 17 19 16.509	135 17 25 16.050
1.080	0.226	-5.274	7.053	71	31	298.721	138 15 31 10.531	138 15 36 9.250
1.080	0.221	-7.243	7.521	71	41	299.130	138 15 42 40.824	138 15 47 39.951
1.101	0.224	-6.084	7.069	71	32	299.142	138 15 36 10.629	138 15 41 9.770
1.115	0.237	-4.302	6.724	71	21	299.176	138 15 20 15.474	138 15 25 14.650
1.149	0.250	-4.109	7.727	72	22	299.146	138 16 51 22.508	138 16 56 21.652
1.153	0.208	-5.289	7.021	71	42	287.025	138 15 47 46.530	138 15 52 33.554

717: Normal Force Coefficient, Cn, at 30% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
0.359	0.474	-1.983	1.978	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.456	0.446	-10.080	13.440	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.638	0.543	-1.032	3.998	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.658	0.415	-8.106	5.531	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.664	0.413	-3.884	31.370	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.677	0.449	-11.370	18.930	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.721	0.387	-19.900	14.160	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.849	0.677	-10.930	6.875	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.851	0.425	-3.816	2.616	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.892	0.388	-11.620	1.959	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.947	0.424	-6.557	2.531	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.988	0.284	-11.700	5.331	70	41	299.230	135 21 57 .617	135 22 1 59.845
1.080	0.441	-11.550	8.224	69	21	254.521	135 20 19 45.149	135 20 23 59.668
1.106	0.430	-25.220	18.400	73	31	298.996	153 11 57 20.314	153 12 2 19.306
1.112	0.420	-13.140	21.400	75	12	299.929	206 10 14 10.124	206 10 19 10.051
1.122	0.410	-1.392	14.220	69	31	299.679	135 20 30 40.268	135 20 35 39.945
1.125	0.375	-14.020	17.990	75	21	299.812	206 10 28 .205	206 10 33 .013
1.194	0.333	-0.447	2.603	70	42	293.820	135 22 2 5.718	135 22 6 59.537
1.242	0.381	-12.020	22.070	70	22	254.371	135 21 39 45.583	135 21 43 59.952
1.243	0.295	-0.937	6.137	70	21	298.798	135 21 34 40.919	135 21 39 39.715
1.256	0.349	-0.141	2.673	67	21	299.545	135 15 59 30.414	135 16 4 29.957
1.259	0.491	-16.530	8.314	69	22	224.076	135 20 24 45.731	135 20 28 29.803
1.272	0.301	0.010	12.270	70	11	299.601	135 21 23 35.137	135 21 28 34.737
1.300	0.399	-0.076	2.883	73	22	292.729	153 11 51 46.543	153 11 56 39.272
1.341	0.391	-12.440	15.040	66	21	299.234	135 14 2 40.759	135 14 7 39.991
1.371	0.466	-10.040	13.690	65	11	299.439	115 9 31 15.541	115 9 36 14.980
1.380	0.495	-19.180	11.280	72	42	289.056	138 17 14 30.447	138 17 19 19.501
1.384	0.360	-0.664	2.739	70	12	299.255	135 21 28 35.569	135 21 33 34.824
1.389	0.390	-8.703	14.190	65	21	299.355	115 9 42 40.316	115 9 47 39.669
1.398	0.333	-12.710	10.740	75	11	299.439	206 10 9 10.688	206 10 14 10.124
1.456	0.384	-16.880	13.290	66	22	299.046	135 14 7 40.831	135 14 12 39.875
1.491	0.661	-4.345	6.512	72	32	261.823	138 17 4 6.242	138 17 8 28.064
1.514	0.366	0.233	6.853	66	32	298.883	135 14 19 10.642	135 14 24 9.523
1.523	0.550	-18.910	14.940	65	12	299.224	115 9 36 15.822	115 9 41 15.046
1.571	0.368	0.110	3.499	67	12	298.846	135 15 52 40.559	135 15 57 39.403
1.666	0.364	-4.576	3.751	67	11	299.499	135 15 47 40.226	135 15 52 39.723
1.719	0.433	-7.246	13.160	66	31	299.449	135 14 14 10.599	135 14 19 10.046
1.719	0.477	-0.516	3.683	67	31	299.288	135 16 11 .492	135 16 15 59.778
1.727	0.576	-9.730	11.370	72	41	298.986	138 17 9 30.796	138 17 14 29.779
1.809	0.419	-15.000	18.680	73	21	299.151	153 11 46 40.756	153 11 51 39.904
1.829	0.696	-13.410	28.350	72	11	299.750	138 16 33 30.441	138 16 38 30.191
1.869	0.487	-6.571	19.130	73	12	300.008	153 11 40 50.094	153 11 45 50.101
1.875	0.550	-7.534	15.520	72	31	299.134	138 16 59 .620	138 17 3 59.754
1.927	0.436	0.353	3.983	67	32	299.149	135 16 16 .526	135 16 20 59.672
1.986	0.562	-6.926	10.380	72	21	298.848	138 16 46 22.829	138 16 51 21.675
2.005	0.519	-8.948	16.020	71	12	299.574	138 15 14 10.641	138 15 19 10.211
2.010	0.574	-2.951	8.673	72	22	299.146	138 16 51 22.508	138 16 56 21.652
2.031	0.560	-14.400	22.030	71	21	299.176	138 15 20 15.474	138 15 25 14.650
2.039	0.511	-5.426	4.347	72	12	299.134	138 16 38 30.898	138 16 43 30.029
2.039	0.537	-7.816	13.000	71	42	287.025	138 15 47 46.530	138 15 52 33.554
2.045	0.507	-13.480	17.090	71	11	299.566	138 15 9 10.495	138 15 14 10.059
2.046	0.514	-4.068	10.530	71	32	299.142	138 15 36 10.629	138 15 41 9.770
2.079	0.552	-8.741	9.170	71	31	298.721	138 15 31 10.531	138 15 36 9.250
2.082	0.458	-17.110	11.670	73	11	298.742	153 11 35 50.765	153 11 40 49.504
2.090	0.558	-4.829	4.318	68	12	295.419	135 17 12 33.107	135 17 17 28.524
2.107	0.519	-11.290	10.940	71	41	299.130	138 15 42 40.824	138 15 47 39.951
2.136	0.503	0.308	5.807	68	21	359.543	135 17 19 16.509	135 17 25 16.050
2.175	0.449	-3.022	4.355	68	11	299.142	135 17 7 28.499	135 17 12 27.639
2.181	0.485	-10.840	13.470	68	22	358.295	135 17 25 22.344	135 17 31 20.639

718: Tangent Force Coefficient, Ct, at 80% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.006	0.005	-0.205	0.538	70	41	299.230	135 21 57 .617	135 22 1 59.845
-0.005	0.013	-1.528	1.637	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-0.004	0.007	-0.249	0.371	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-0.004	0.010	-0.667	0.500	75	21	299.812	206 10 28 .205	206 10 33 .013
-0.003	0.008	-0.094	0.670	70	42	293.820	135 22 2 5.718	135 22 6 59.537
-0.003	0.008	-0.652	0.983	70	21	298.798	135 21 34 40.919	135 21 39 39.715
-0.002	0.022	-1.728	1.535	69	31	299.679	135 20 30 40.268	135 20 35 39.945
-0.002	0.010	-0.425	0.187	70	11	299.601	135 21 23 35.137	135 21 28 34.737
-0.002	0.011	-0.971	0.501	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-0.002	0.006	-0.218	1.497	73	21	299.151	153 11 46 40.756	153 11 51 39.904
-0.002	0.015	-1.011	3.286	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-0.002	0.004	-0.360	0.511	73	22	292.729	153 11 51 46.543	153 11 56 39.272
-0.001	0.011	-0.394	0.438	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.000	0.013	-0.856	1.474	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.000	0.005	-0.286	0.566	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.000	0.009	-1.190	1.001	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.000	0.004	-0.402	0.535	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.001	0.004	-0.800	0.070	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.002	0.023	-1.591	1.215	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.002	0.007	-0.963	1.488	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.002	0.029	-1.839	1.274	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.003	0.009	-1.911	2.193	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.007	0.017	-1.258	0.474	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.008	0.019	-0.991	1.650	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.008	0.030	-1.812	1.740	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.008	0.014	-0.026	0.073	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.009	0.023	-1.915	1.550	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.012	0.038	-2.635	1.982	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.015	0.028	-1.482	1.292	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.016	0.035	-1.526	1.866	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.018	0.026	-0.873	0.902	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.018	0.028	-1.093	0.523	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.021	0.033	-0.363	0.542	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.021	0.036	-1.886	1.245	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.026	0.030	-0.614	1.024	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.029	0.048	-2.100	1.767	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.030	0.047	-1.368	0.877	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.046	0.060	-1.650	1.417	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.046	0.039	-0.858	0.208	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.047	0.045	-1.305	1.540	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.056	0.069	-2.082	1.488	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.069	0.041	-0.544	0.385	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.073	0.092	-1.621	1.025	71	42	287.025	138 15 47 46.530	138 15 52 33.554
0.077	0.064	-1.184	0.667	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.078	0.052	-1.272	1.020	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.085	0.087	-2.131	1.911	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.086	0.078	-0.979	1.130	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.087	0.090	-0.973	1.631	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.087	0.073	-1.947	0.970	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.087	0.061	-1.476	1.341	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.091	0.070	-1.558	1.578	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.093	0.059	-1.646	1.998	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.094	0.063	-1.461	1.397	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.102	0.060	-0.094	0.363	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.103	0.071	-1.765	1.422	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.106	0.052	-0.823	0.918	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.109	0.067	-0.929	1.154	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.109	0.048	-0.608	0.841	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.114	0.056	-1.836	1.645	68	22	358.295	135 17 25 22.344	135 17 31 20.639

719: Tangent Force Coefficient, Ct, at 63.3% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.002	0.011	-0.591	0.079	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-0.001	0.013	-0.931	0.415	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.001	0.014	-0.063	0.737	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.001	0.019	-0.077	0.147	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.002	0.014	-0.514	0.678	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.003	0.016	-0.026	0.890	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.003	0.013	-0.026	0.087	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.004	0.022	-0.081	0.151	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.007	0.083	-0.275	1.515	71	42	287.025	138 15 47 46.530	138 15 52 33.554
0.009	0.029	-0.029	0.159	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.018	0.032	-0.651	1.035	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.018	0.036	-0.036	0.263	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.018	0.029	-0.699	1.849	75	21	299.812	206 10 28 .205	206 10 33 .013
0.019	0.024	-0.148	0.170	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.023	0.032	-1.496	0.427	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.023	0.023	-0.181	0.172	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.024	0.086	-1.235	0.904	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.025	0.038	-0.029	0.176	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.028	0.033	-0.028	1.356	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.029	0.040	-0.135	0.985	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.029	0.038	-0.025	0.207	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.030	0.029	-0.022	0.144	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.040	0.091	-0.830	1.153	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.041	0.046	-0.044	0.842	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.041	0.101	-1.530	0.478	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.046	0.037	-0.204	2.415	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.048	0.050	-0.361	0.726	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.049	0.037	-0.046	1.167	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.052	0.038	-0.032	0.220	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.052	0.033	-0.726	0.188	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.058	0.091	-1.244	0.674	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.058	0.089	-1.255	0.968	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.059	0.050	-0.269	0.248	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.059	0.084	-0.677	0.467	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.059	0.043	-0.494	0.212	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.061	0.051	-0.369	0.275	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.064	0.041	-0.016	0.213	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.068	0.068	-0.373	0.373	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.069	0.075	-0.658	0.464	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.071	0.087	-0.429	0.489	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.072	0.043	-0.471	0.269	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.073	0.085	-0.132	0.411	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.073	0.054	-0.985	0.265	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.074	0.086	-1.047	0.431	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.079	0.045	-0.245	0.243	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.082	0.080	-0.825	0.457	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.084	0.045	-0.678	0.395	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.084	0.080	-1.254	1.624	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.085	0.085	-0.129	0.455	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.086	0.080	-1.192	0.452	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.087	0.084	-0.122	0.424	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.087	0.077	-1.289	0.885	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.098	0.052	-0.073	0.302	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.104	0.076	-0.489	0.501	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.110	0.057	-0.121	0.364	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.116	0.043	-0.054	0.638	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.116	0.053	-0.090	0.318	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.125	0.045	-0.388	0.319	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.140	0.069	-0.165	0.354	66	11	299.144	135 13 48 40.309	135 13 53 39.449

720: Tangent Force Coefficient, Ct, at 46.6% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.034	0.047	-1.055	1.138	71	42	287.025	138 15 47 46.530	138 15 52 33.554
-0.025	0.065	-1.079	1.243	71	41	299.130	138 15 42 40.824	138 15 47 39.951
-0.024	0.065	-1.092	1.475	71	32	299.142	138 15 36 10.629	138 15 41 9.770
-0.022	0.064	-0.495	0.944	68	11	299.142	135 17 7 28.499	135 17 12 27.639
-0.022	0.069	-1.755	1.254	68	22	358.295	135 17 25 22.344	135 17 31 20.639
-0.020	0.067	-1.208	1.220	71	21	299.176	138 15 20 15.474	138 15 25 14.650
-0.018	0.071	-1.003	1.315	71	31	298.721	138 15 31 10.531	138 15 36 9.250
-0.016	0.073	-0.553	1.031	72	22	299.146	138 16 51 22.508	138 16 56 21.652
-0.012	0.076	-0.302	0.460	68	12	295.419	135 17 12 33.107	135 17 17 28.524
-0.011	0.081	-0.503	0.752	68	21	359.543	135 17 19 16.509	135 17 25 16.050
-0.007	0.078	-0.188	0.419	72	12	299.134	138 16 38 30.898	138 16 43 30.029
-0.006	0.077	-0.960	0.869	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.002	0.084	-1.849	1.146	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.005	0.085	-1.451	1.345	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.006	0.083	-0.986	0.908	66	12	297.491	135 13 53 45.999	135 13 58 43.489
0.006	0.089	-1.008	1.443	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.008	0.029	-0.126	0.388	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.013	0.082	-1.418	1.047	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.017	0.094	-0.826	0.908	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.017	0.037	-1.284	2.009	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.017	0.031	-0.700	0.916	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.024	0.068	-0.393	0.802	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.025	0.044	-0.754	0.272	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.029	0.042	-0.662	0.983	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.031	0.092	-0.155	0.999	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.034	0.045	-0.057	0.275	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.035	0.091	-0.889	1.041	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.041	0.038	-0.746	1.306	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.041	0.043	-0.027	0.254	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.041	0.088	-1.071	1.839	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.043	0.052	-0.084	0.300	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.050	0.067	-1.364	0.617	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.051	0.091	-0.955	0.904	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.055	0.086	-1.642	1.851	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.056	0.054	-1.022	0.765	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.057	0.055	-0.878	1.185	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.058	0.085	-0.168	0.465	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.060	0.055	-1.309	1.449	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.062	0.091	-0.767	0.466	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.066	0.069	-0.923	0.740	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.069	0.052	-1.206	1.695	75	21	299.812	206 10 28 .205	206 10 33 .013
0.071	0.054	-0.913	1.311	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.074	0.056	-1.235	0.730	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.077	0.047	-0.109	0.277	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.078	0.060	-0.106	0.323	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.078	0.071	-0.865	1.250	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.078	0.054	-0.100	0.282	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.080	0.074	-0.965	1.497	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.081	0.066	-0.475	0.655	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.084	0.070	-0.156	0.453	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.086	0.078	-1.813	0.905	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.086	0.043	-0.188	0.575	70	21	298.798	135 21 34 40.919	135 21 39 39.715
0.090	0.062	-1.898	1.389	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.090	0.047	-0.076	0.271	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.091	0.059	-0.551	1.691	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.092	0.066	-1.357	1.328	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.096	0.054	-1.839	1.384	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.097	0.070	-0.130	0.858	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.098	0.058	-0.215	0.287	70	12	299.255	135 21 28 35.569	135 21 33 34.824

721: Tangent Force Coefficient, Ct, at 30% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.308	0.108	-1.695	3.560	66	12	297.491	135 13 53 45.999	135 13 58 43.489
-0.109	0.114	-1.138	0.634	66	11	299.144	135 13 48 40.309	135 13 53 39.449
0.007	0.052	-0.116	2.203	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.023	0.090	-1.737	2.385	71	42	287.025	138 15 47 46.530	138 15 52 33.554
0.026	0.068	-1.156	1.101	69	32	293.222	135 20 35 46.647	135 20 40 39.869
0.027	0.063	-0.962	0.300	73	42	299.532	153 12 12 50.225	153 12 17 49.755
0.030	0.062	-4.056	3.808	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.032	0.095	-2.823	2.846	71	21	299.176	138 15 20 15.474	138 15 25 14.650
0.032	0.059	-2.005	2.966	73	32	273.569	153 12 2 26.020	153 12 6 59.586
0.037	0.071	-3.733	2.583	65	22	299.370	115 9 47 40.501	115 9 52 39.869
0.038	0.100	-2.100	1.662	72	22	299.146	138 16 51 22.508	138 16 56 21.652
0.041	0.097	-2.100	2.594	71	32	299.142	138 15 36 10.629	138 15 41 9.770
0.049	0.087	-2.273	1.302	67	11	299.499	135 15 47 40.226	135 15 52 39.723
0.049	0.065	-0.273	0.314	73	41	298.510	153 12 7 50.870	153 12 12 49.378
0.053	0.089	-3.317	3.663	72	21	298.848	138 16 46 22.829	138 16 51 21.675
0.054	0.094	-2.102	3.471	73	21	299.151	153 11 46 40.756	153 11 51 39.904
0.054	0.083	-0.159	0.381	67	12	298.846	135 15 52 40.559	135 15 57 39.403
0.056	0.096	-2.649	3.099	71	41	299.130	138 15 42 40.824	138 15 47 39.951
0.057	0.092	-1.676	1.267	68	12	295.419	135 17 12 33.107	135 17 17 28.524
0.057	0.096	-1.688	2.587	71	31	298.721	138 15 31 10.531	138 15 36 9.250
0.058	0.096	-2.407	3.592	71	12	299.574	138 15 14 10.641	138 15 19 10.211
0.058	0.078	-2.630	2.561	69	22	224.076	135 20 24 45.731	135 20 28 29.803
0.060	0.092	-0.793	0.502	67	31	299.288	135 16 11 .492	135 16 15 59.778
0.060	0.095	-2.873	4.165	66	31	299.449	135 14 14 10.599	135 14 19 10.046
0.062	0.070	-4.101	2.551	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.062	0.065	-1.175	0.384	70	32	299.151	135 21 50 20.307	135 21 55 19.458
0.062	0.102	-2.987	3.149	72	31	299.134	138 16 59 .620	138 17 3 59.754
0.063	0.094	-0.265	0.553	72	12	299.134	138 16 38 30.898	138 16 43 30.029
0.063	0.088	-0.194	0.798	67	32	299.149	135 16 16 .526	135 16 20 59.672
0.066	0.092	-2.009	3.689	68	22	358.295	135 17 25 22.344	135 17 31 20.639
0.068	0.096	-3.342	5.434	72	11	299.750	138 16 33 30.441	138 16 38 30.191
0.069	0.096	-1.579	4.633	71	11	299.566	138 15 9 10.495	138 15 14 10.059
0.069	0.073	-2.330	6.650	75	12	299.929	206 10 14 10.124	206 10 19 10.051
0.069	0.093	-0.550	0.715	68	21	359.543	135 17 19 16.509	135 17 25 16.050
0.069	0.088	-2.609	4.517	72	42	289.056	138 17 14 30.447	138 17 19 19.501
0.069	0.063	-0.295	1.306	70	31	299.305	135 21 45 20.167	135 21 50 19.470
0.071	0.082	-0.483	2.772	66	32	298.883	135 14 19 10.642	135 14 24 9.523
0.072	0.054	-4.576	2.074	70	41	299.230	135 21 57 .617	135 22 1 59.845
0.072	0.069	-1.816	5.930	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.073	0.081	-5.033	3.586	75	11	299.439	206 10 9 10.688	206 10 14 10.124
0.074	0.097	-4.409	3.529	73	12	300.008	153 11 40 50.094	153 11 45 50.101
0.074	0.077	-0.212	0.414	70	12	299.255	135 21 28 35.569	135 21 33 34.824
0.076	0.072	-0.179	0.432	73	22	292.729	153 11 51 46.543	153 11 56 39.272
0.076	0.088	-0.795	1.131	68	11	299.142	135 17 7 28.499	135 17 12 27.639
0.076	0.091	-1.627	0.893	72	32	261.823	138 17 4 6.242	138 17 8 28.064
0.077	0.091	-2.287	2.172	73	11	298.742	153 11 35 50.765	153 11 40 49.504
0.077	0.097	-3.862	3.691	72	41	298.986	138 17 9 30.796	138 17 14 29.779
0.078	0.072	-5.207	4.962	75	21	299.812	206 10 28 .205	206 10 33 .013
0.078	0.070	-2.748	3.200	70	22	254.371	135 21 39 45.583	135 21 43 59.952
0.078	0.066	-0.187	0.358	67	21	299.545	135 15 59 30.414	135 16 4 29.957
0.079	0.082	-3.018	3.653	66	22	299.046	135 14 7 40.831	135 14 12 39.875
0.079	0.075	-0.212	3.821	65	11	299.439	115 9 31 15.541	115 9 36 14.980
0.079	0.078	-2.576	2.868	65	21	299.355	115 9 42 40.316	115 9 47 39.669
0.080	0.071	-2.680	3.689	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.084	0.087	-4.487	1.057	65	12	299.224	115 9 36 15.822	115 9 41 15.046
0.086	0.062	-0.164	0.351	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.088	0.066	-0.433	1.499	70	11	299.601	135 21 23 35.137	135 21 28 34.737
0.089	0.087	-5.739	4.615	66	21	299.234	135 14 2 40.759	135 14 7 39.991
0.094	0.063	-0.182	2.098	70	21	298.798	135 21 34 40.919	135 21 39 39.715

935: Angle of Attack at 82% Span (deg)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-2.850	1.916	-10.630	7.934	67	22	298.959	135 16 4 30.791	135 16 9 29.750
-2.798	1.701	-17.080	21.650	75	22	289.576	206 10 33 5.072	206 10 37 54.646
-2.275	2.350	-11.560	12.270	73	42	299.532	153 12 12 50.225	153 12 17 49.755
-2.203	1.675	-12.500	7.676	73	32	273.569	153 12 2 26.020	153 12 6 59.586
-1.336	1.764	-10.850	6.973	73	41	298.510	153 12 7 50.870	153 12 12 49.378
-0.763	1.714	-10.330	30.860	70	31	299.305	135 21 45 20.167	135 21 50 19.470
-0.573	3.257	-13.480	16.100	69	32	293.222	135 20 35 46.647	135 20 40 39.869
-0.553	2.035	-17.540	13.080	75	21	299.812	206 10 28 .205	206 10 33 .013
-0.544	1.303	-9.837	11.590	70	41	299.230	135 21 57 .617	135 22 1 59.845
-0.427	1.931	-12.970	32.580	75	12	299.929	206 10 14 10.124	206 10 19 10.051
-0.409	2.101	-11.090	10.470	70	32	299.151	135 21 50 20.307	135 21 55 19.458
-0.258	1.591	-16.960	6.131	65	22	299.370	115 9 47 40.501	115 9 52 39.869
-0.240	2.090	-12.790	15.430	73	31	298.996	153 11 57 20.314	153 12 2 19.306
0.460	2.362	-12.230	13.010	69	21	254.521	135 20 19 45.149	135 20 23 59.668
0.469	1.946	-12.720	37.210	69	31	299.679	135 20 30 40.268	135 20 35 39.945
0.610	1.655	-12.560	16.980	70	42	293.820	135 22 2 5.718	135 22 6 59.537
0.939	1.788	-11.190	38.110	67	21	299.545	135 15 59 30.414	135 16 4 29.957
1.009	1.468	-10.410	38.280	70	21	298.798	135 21 34 40.919	135 21 39 39.715
1.055	2.060	-10.810	13.300	70	22	254.371	135 21 39 45.583	135 21 43 59.952
1.072	2.213	-9.950	30.680	73	22	292.729	153 11 51 46.543	153 11 56 39.272
1.256	1.594	-10.100	10.140	70	11	299.601	135 21 23 35.137	135 21 28 34.737
1.277	1.871	-12.360	37.990	75	11	299.439	206 10 9 10.688	206 10 14 10.124
1.539	2.715	-9.394	38.030	69	22	224.076	135 20 24 45.731	135 20 28 29.803
2.061	1.943	-8.617	11.390	70	12	299.255	135 21 28 35.569	135 21 33 34.824
2.187	1.980	-9.249	37.200	66	21	299.234	135 14 2 40.759	135 14 7 39.991
2.588	2.641	-8.548	39.040	72	42	289.056	138 17 14 30.447	138 17 19 19.501
2.676	2.201	-8.361	13.350	66	22	299.046	135 14 7 40.831	135 14 12 39.875
3.049	2.048	-7.448	17.320	67	12	298.846	135 15 52 40.559	135 15 57 39.403
3.089	1.981	-5.310	36.340	65	21	299.355	115 9 42 40.316	115 9 47 39.669
3.197	2.310	-7.946	15.500	66	32	298.883	135 14 19 10.642	135 14 24 9.523
3.251	2.510	-5.831	16.280	65	11	299.439	115 9 31 15.541	115 9 36 14.980
3.577	2.155	-7.623	36.300	67	11	299.499	135 15 47 40.226	135 15 52 39.723
3.784	2.695	-4.821	21.010	65	12	299.224	115 9 36 15.822	115 9 41 15.046
4.515	2.225	-12.770	25.020	73	21	299.151	153 11 46 40.756	153 11 51 39.904
4.557	4.824	-12.210	31.130	72	32	261.823	138 17 4 6.242	138 17 8 28.064
4.704	2.783	-8.412	23.520	67	31	299.288	135 16 11 .492	135 16 15 59.778
4.894	2.962	-9.322	31.610	66	31	299.449	135 14 14 10.599	135 14 19 10.046
5.109	2.977	-8.514	28.830	73	12	300.008	153 11 40 50.094	153 11 45 50.101
6.083	2.664	-5.243	27.570	66	12	297.491	135 13 53 45.999	135 13 58 43.489
6.215	2.741	-7.995	26.970	67	32	299.149	135 16 16 .526	135 16 20 59.672
6.998	5.847	-9.743	39.440	72	41	298.986	138 17 9 30.796	138 17 14 29.779
7.114	5.418	-8.514	32.710	72	11	299.750	138 16 33 30.441	138 16 38 30.191
7.914	3.768	-15.640	30.680	73	11	298.742	153 11 35 50.765	153 11 40 49.504
9.398	4.524	-4.373	36.080	71	11	299.566	138 15 9 10.495	138 15 14 10.059
9.439	4.900	-6.757	39.610	72	31	299.134	138 16 59 .620	138 17 3 59.754
9.485	3.506	-4.666	27.570	66	11	299.144	135 13 48 40.309	135 13 53 39.449
9.671	5.328	-16.710	37.190	71	12	299.574	138 15 14 10.641	138 15 19 10.211
10.150	3.826	-18.650	31.700	68	11	299.142	135 17 7 28.499	135 17 12 27.639
10.400	4.155	-4.756	36.150	68	21	359.543	135 17 19 16.509	135 17 25 16.050
10.770	4.867	-4.260	33.790	68	12	295.419	135 17 12 33.107	135 17 17 28.524
11.020	6.120	-6.712	38.420	72	21	298.848	138 16 46 22.829	138 16 51 21.675
11.190	4.014	-4.110	34.800	68	22	358.295	135 17 25 22.344	135 17 31 20.639
11.190	5.923	-8.617	36.310	72	12	299.134	138 16 38 30.898	138 16 43 30.029
12.480	5.362	-8.044	36.740	71	31	298.721	138 15 31 10.531	138 15 36 9.250
13.060	4.694	-2.775	35.580	71	41	299.130	138 15 42 40.824	138 15 47 39.951
14.320	5.848	-6.519	39.610	71	32	299.142	138 15 36 10.629	138 15 41 9.770
15.250	6.520	-5.324	39.650	71	21	299.176	138 15 20 15.474	138 15 25 14.650
15.440	6.874	-8.110	39.660	72	22	299.146	138 16 51 22.508	138 16 56 21.652
17.480	5.442	-7.995	39.470	71	42	287.025	138 15 47 46.530	138 15 52 33.554

936: Angle of Attack at 63.3% Span

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-1.737	2.462	-12.000	10.130	67	22	298.959	135 16 4 30.791	135 16 9 29.750
0.266	2.133	-20.400	15.390	75	22	289.576	206 10 33 5.072	206 10 37 54.646
0.769	2.221	-6.211	12.470	65	22	299.370	115 9 47 40.501	115 9 52 39.869
1.118	2.260	-10.170	16.680	70	31	299.305	135 21 45 20.167	135 21 50 19.470
1.390	1.801	-13.480	33.260	70	41	299.230	135 21 57 .617	135 22 1 59.845
1.466	4.454	-16.130	27.760	69	32	293.222	135 20 35 46.647	135 20 40 39.869
1.624	2.923	-9.262	18.220	70	32	299.151	135 21 50 20.307	135 21 55 19.458
2.781	2.779	-9.392	30.750	69	31	299.679	135 20 30 40.268	135 20 35 39.945
2.843	3.318	-26.980	20.840	69	21	254.521	135 20 19 45.149	135 20 23 59.668
3.026	2.191	-8.616	21.150	70	42	293.820	135 22 2 5.718	135 22 6 59.537
3.128	2.740	-71.990	19.190	75	12	299.929	206 10 14 10.124	206 10 19 10.051
3.166	2.654	-43.790	19.890	75	21	299.812	206 10 28 .205	206 10 33 .013
3.436	2.635	-21.230	31.790	67	21	299.545	135 15 59 30.414	135 16 4 29.957
3.503	1.968	-11.700	19.590	70	21	298.798	135 21 34 40.919	135 21 39 39.715
3.649	2.931	-10.430	16.530	70	22	254.371	135 21 39 45.583	135 21 43 59.952
3.899	2.190	-10.270	18.310	70	11	299.601	135 21 23 35.137	135 21 28 34.737
4.306	3.897	-9.825	22.210	69	22	224.076	135 20 24 45.731	135 20 28 29.803
5.073	2.790	-9.025	29.070	70	12	299.255	135 21 28 35.569	135 21 33 34.824
5.150	2.982	-8.372	30.170	66	21	299.234	135 14 2 40.759	135 14 7 39.991
5.570	2.720	-51.700	25.960	75	11	299.439	206 10 9 10.688	206 10 14 10.124
5.583	3.859	-6.463	39.640	72	42	289.056	138 17 14 30.447	138 17 19 19.501
5.596	2.915	-9.244	23.430	65	21	299.355	115 9 42 40.316	115 9 47 39.669
5.701	3.627	-27.410	25.370	65	11	299.439	115 9 31 15.541	115 9 36 14.980
5.826	3.233	-8.116	26.040	66	22	299.046	135 14 7 40.831	135 14 12 39.875
6.473	3.092	-13.270	31.110	67	12	298.846	135 15 52 40.559	135 15 57 39.403
6.553	3.400	-8.338	35.560	66	32	298.883	135 14 19 10.642	135 14 24 9.523
6.622	3.973	-4.541	33.140	65	12	299.224	115 9 36 15.822	115 9 41 15.046
7.305	3.221	-13.650	28.730	67	11	299.499	135 15 47 40.226	135 15 52 39.723
8.522	7.108	-25.720	39.660	72	32	261.823	138 17 4 6.242	138 17 8 28.064
8.944	4.192	-10.110	34.700	67	31	299.288	135 16 11 .492	135 16 15 59.778
9.035	4.318	-5.835	34.450	66	31	299.449	135 14 14 10.599	135 14 19 10.046
10.500	3.059	-0.552	28.450	73	32	273.569	153 12 2 26.020	153 12 6 59.586
10.610	4.337	-0.751	30.230	73	42	299.532	153 12 12 50.225	153 12 17 49.755
10.720	4.107	-7.716	35.790	66	12	297.491	135 13 53 45.999	135 13 58 43.489
11.100	4.218	-15.070	38.950	67	32	299.149	135 16 16 .526	135 16 20 59.672
11.910	8.388	-5.594	39.660	72	41	298.986	138 17 9 30.796	138 17 14 29.779
12.010	7.816	-7.569	39.660	72	11	299.750	138 16 33 30.441	138 16 38 30.191
12.150	3.632	1.432	31.660	73	41	298.510	153 12 7 50.870	153 12 12 49.378
14.550	4.473	0.728	36.310	73	31	298.996	153 11 57 20.314	153 12 2 19.306
15.230	6.497	-3.241	39.650	71	11	299.566	138 15 9 10.495	138 15 14 10.059
15.490	7.249	-4.964	39.660	72	31	299.134	138 16 59 .620	138 17 3 59.754
15.510	7.290	-3.253	39.660	71	12	299.574	138 15 14 10.641	138 15 19 10.211
15.630	5.168	-6.796	39.400	66	11	299.144	135 13 48 40.309	135 13 53 39.449
16.450	5.558	-13.850	39.660	68	11	299.142	135 17 7 28.499	135 17 12 27.639
16.880	6.183	-8.235	39.660	68	21	359.543	135 17 19 16.509	135 17 25 16.050
17.290	4.775	3.465	39.020	73	22	292.729	153 11 51 46.543	153 11 56 39.272
17.310	7.011	-6.568	39.660	68	12	295.419	135 17 12 33.107	135 17 17 28.524
17.390	8.490	-7.766	39.660	72	21	298.848	138 16 46 22.829	138 16 51 21.675
17.580	7.983	-4.398	39.660	72	12	299.134	138 16 38 30.898	138 16 43 30.029
17.980	5.969	-7.075	39.660	68	22	358.295	135 17 25 22.344	135 17 31 20.639
19.320	7.427	-15.580	39.660	71	31	298.721	138 15 31 10.531	138 15 36 9.250
20.240	6.573	-11.420	39.660	71	41	299.130	138 15 42 40.824	138 15 47 39.951
21.700	7.725	-16.070	39.660	71	32	299.142	138 15 36 10.629	138 15 41 9.770
22.840	8.437	-22.230	39.660	71	21	299.176	138 15 20 15.474	138 15 25 14.650
23.050	8.822	-12.520	39.660	72	22	299.146	138 16 51 22.508	138 16 56 21.652
24.480	4.240	0.463	39.660	73	21	299.151	153 11 46 40.756	153 11 51 39.904
25.360	5.565	1.432	39.660	73	12	300.008	153 11 40 50.094	153 11 45 50.101
25.640	6.889	-28.070	39.660	71	42	287.025	138 15 47 46.530	138 15 52 33.554
29.720	5.602	2.892	39.660	73	11	298.742	153 11 35 50.765	153 11 40 49.504

937: Angle of Attack at 46.6% Span (deg)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
-0.816	2.183	-8.908	9.758	67	22	298.959	135 16 4 30.791	135 16 9 29.750
1.707	2.040	-7.481	38.140	70	31	299.305	135 21 45 20.167	135 21 50 19.470
1.948	1.610	-9.248	15.490	70	41	299.230	135 21 57 .617	135 22 1 59.845
2.072	4.014	-11.210	30.650	69	32	293.222	135 20 35 46.647	135 20 40 39.869
2.171	2.655	-6.235	15.460	70	32	299.151	135 21 50 20.307	135 21 55 19.458
2.308	2.905	-20.360	21.490	75	22	289.576	206 10 33 5.072	206 10 37 54.646
3.239	2.516	-18.930	22.230	69	31	299.679	135 20 30 40.268	135 20 35 39.945
3.263	3.313	-5.736	20.020	65	22	299.370	115 9 47 40.501	115 9 52 39.869
3.301	3.005	-14.860	18.410	69	21	254.521	135 20 19 45.149	135 20 23 59.668
3.455	1.981	-5.666	24.570	70	42	293.820	135 22 2 5.718	135 22 6 59.537
3.879	2.380	-6.935	25.350	67	21	299.545	135 15 59 30.414	135 16 4 29.957
3.897	1.780	-7.354	26.950	70	21	298.798	135 21 34 40.919	135 21 39 39.715
4.027	2.663	-6.249	14.670	70	22	254.371	135 21 39 45.583	135 21 43 59.952
4.264	1.982	-6.337	16.470	70	11	299.601	135 21 23 35.137	135 21 28 34.737
4.644	3.544	-6.514	26.020	69	22	224.076	135 20 24 45.731	135 20 28 29.803
5.346	2.523	-8.890	22.090	70	12	299.255	135 21 28 35.569	135 21 33 34.824
5.427	2.698	-5.107	24.000	66	21	299.234	135 14 2 40.759	135 14 7 39.991
5.897	3.520	-4.582	32.790	72	42	289.056	138 17 14 30.447	138 17 19 19.501
6.012	3.797	-16.810	29.710	75	21	299.812	206 10 28 .205	206 10 33 .013
6.047	2.924	-5.391	22.090	66	22	299.046	135 14 7 40.831	135 14 12 39.875
6.680	2.785	-8.979	28.050	67	12	298.846	135 15 52 40.559	135 15 57 39.403
6.716	3.070	-4.113	30.970	66	32	298.883	135 14 19 10.642	135 14 24 9.523
7.442	2.886	-17.470	23.840	67	11	299.499	135 15 47 40.226	135 15 52 39.723
7.455	3.742	-25.020	39.370	75	12	299.929	206 10 14 10.124	206 10 19 10.051
8.576	6.404	-14.000	39.550	72	32	261.823	138 17 4 6.242	138 17 8 28.064
8.933	3.744	-6.603	31.050	67	31	299.288	135 16 11 .492	135 16 15 59.778
8.979	3.870	-23.100	30.710	66	31	299.449	135 14 14 10.599	135 14 19 10.046
10.160	2.701	0.508	24.820	73	32	273.569	153 12 2 26.020	153 12 6 59.586
10.260	3.864	0.661	27.610	73	42	299.532	153 12 12 50.225	153 12 17 49.755
10.430	5.500	-11.520	38.830	65	11	299.439	115 9 31 15.541	115 9 36 14.980
10.510	4.719	-9.303	37.200	65	21	299.355	115 9 42 40.316	115 9 47 39.669
10.520	3.647	-4.175	31.620	66	12	297.491	135 13 53 45.999	135 13 58 43.489
10.890	3.722	-4.075	35.450	67	32	299.149	135 16 16 .526	135 16 20 59.672
11.120	3.873	-6.220	38.220	75	11	299.439	206 10 9 10.688	206 10 14 10.124
11.630	3.216	2.480	30.000	73	41	298.510	153 12 7 50.870	153 12 12 49.378
11.660	7.604	-5.081	39.550	72	41	298.986	138 17 9 30.796	138 17 14 29.779
11.750	7.042	-5.269	39.460	72	11	299.750	138 16 33 30.441	138 16 38 30.191
12.040	6.155	-23.360	39.550	65	12	299.224	115 9 36 15.822	115 9 41 15.046
13.790	3.971	2.172	33.500	73	31	298.996	153 11 57 20.314	153 12 2 19.306
14.670	5.756	-10.150	39.550	71	11	299.566	138 15 9 10.495	138 15 14 10.059
14.900	6.474	-1.746	39.550	72	31	299.134	138 16 59 .620	138 17 3 59.754
14.930	4.525	-3.815	37.930	66	11	299.144	135 13 48 40.309	135 13 53 39.449
14.960	6.537	-22.010	39.550	71	12	299.574	138 15 14 10.641	138 15 19 10.211
15.680	4.859	-19.500	38.780	68	11	299.142	135 17 7 28.499	135 17 12 27.639
16.070	5.447	-12.680	39.550	68	21	359.543	135 17 19 16.509	135 17 25 16.050
16.240	4.255	4.435	36.820	73	22	292.729	153 11 51 46.543	153 11 56 39.272
16.450	6.201	-3.803	39.550	68	12	295.419	135 17 12 33.107	135 17 17 28.524
16.620	7.661	-36.800	39.550	72	21	298.848	138 16 46 22.829	138 16 51 21.675
16.820	7.225	-2.229	39.550	72	12	299.134	138 16 38 30.898	138 16 43 30.029
17.040	5.244	-2.930	39.550	68	22	358.295	135 17 25 22.344	135 17 31 20.639
18.390	6.655	-20.090	39.550	71	31	298.721	138 15 31 10.531	138 15 36 9.250
19.200	5.846	-16.690	39.550	71	41	299.130	138 15 42 40.824	138 15 47 39.951
20.550	6.945	-10.960	39.550	71	32	299.142	138 15 36 10.629	138 15 41 9.770
21.610	7.683	-36.800	39.550	71	21	299.176	138 15 20 15.474	138 15 25 14.650
21.880	8.120	-3.308	39.550	72	22	299.146	138 16 51 22.508	138 16 56 21.652
22.770	3.801	3.486	39.550	73	21	299.151	153 11 46 40.756	153 11 51 39.904
23.650	5.108	4.774	39.550	73	12	300.008	153 11 40 50.094	153 11 45 50.101
24.170	6.284	-32.500	39.550	71	42	287.025	138 15 47 46.530	138 15 52 33.554
27.860	5.402	6.617	39.550	73	11	298.742	153 11 35 50.765	153 11 40 49.504

938: Angle of Attack at 30% Span (deg)

Mean	Std	Min	Max	Tape	Event	Time Length	Start Time	End Time
3.643	5.966	-20.350	39.120	67	22	298.959	135 16 4 30.791	135 16 9 29.750
6.611	5.776	-25.920	33.900	73	32	273.569	153 12 2 26.020	153 12 6 59.586
6.643	7.721	-25.920	39.660	73	42	299.532	153 12 12 50.225	153 12 17 49.755
6.877	5.799	-11.750	34.810	65	22	299.370	115 9 47 40.501	115 9 52 39.869
7.295	6.166	-25.920	38.070	75	22	289.576	206 10 33 5.072	206 10 37 54.646
9.199	6.663	-18.920	39.170	73	41	298.510	153 12 7 50.870	153 12 12 49.378
10.370	6.201	-23.850	39.280	70	31	299.305	135 21 45 20.167	135 21 50 19.470
11.150	5.011	-16.040	35.900	70	41	299.230	135 21 57 .617	135 22 1 59.845
11.600	7.873	-25.920	39.660	70	32	299.151	135 21 50 20.307	135 21 55 19.458
11.740	11.920	-25.920	39.660	69	32	293.222	135 20 35 46.647	135 20 40 39.869
12.750	8.147	-25.920	39.660	75	12	299.929	206 10 14 10.124	206 10 19 10.051
13.370	8.047	-25.920	39.660	73	31	298.996	153 11 57 20.314	153 12 2 19.306
13.790	7.795	-25.920	39.660	75	21	299.812	206 10 28 .205	206 10 33 .013
14.800	7.967	-25.920	39.660	69	31	299.679	135 20 30 40.268	135 20 35 39.945
14.810	6.761	-25.920	39.660	70	42	293.820	135 22 2 5.718	135 22 6 59.537
15.510	9.545	-25.920	39.660	69	21	254.521	135 20 19 45.149	135 20 23 59.668
16.030	6.775	-22.830	39.660	70	21	298.798	135 21 34 40.919	135 21 39 39.715
16.380	7.025	-16.410	39.640	67	21	299.545	135 15 59 30.414	135 16 4 29.957
16.440	7.733	-21.490	39.660	70	22	254.371	135 21 39 45.583	135 21 43 59.952
16.860	6.801	-18.050	39.660	70	11	299.601	135 21 23 35.137	135 21 28 34.737
16.910	8.372	-22.350	39.660	73	22	292.729	153 11 51 46.543	153 11 56 39.272
17.900	9.878	-21.000	39.660	69	22	224.076	135 20 24 45.731	135 20 28 29.803
18.040	8.807	-12.830	39.660	65	11	299.439	115 9 31 15.541	115 9 36 14.980
18.510	8.072	-25.760	39.660	65	21	299.355	115 9 42 40.316	115 9 47 39.669
18.520	8.338	-25.920	39.660	75	11	299.439	206 10 9 10.688	206 10 14 10.124
19.360	7.498	-19.110	39.660	70	12	299.255	135 21 28 35.569	135 21 33 34.824
19.770	9.742	-25.680	39.660	72	42	289.056	138 17 14 30.447	138 17 19 19.501
19.910	8.494	-25.920	39.660	66	21	299.234	135 14 2 40.759	135 14 7 39.991
20.240	9.718	-19.010	39.660	65	12	299.224	115 9 36 15.822	115 9 41 15.046
21.060	8.053	-15.630	39.660	66	22	299.046	135 14 7 40.831	135 14 12 39.875
22.260	8.050	-25.920	39.660	66	32	298.883	135 14 19 10.642	135 14 24 9.523
22.680	7.293	-25.920	39.660	67	12	298.846	135 15 52 40.559	135 15 57 39.403
23.020	11.880	-25.920	39.660	72	32	261.823	138 17 4 6.242	138 17 8 28.064
24.760	6.742	-10.870	39.660	67	11	299.499	135 15 47 40.226	135 15 52 39.723
26.700	8.031	-6.609	39.660	66	31	299.449	135 14 14 10.599	135 14 19 10.046
26.910	8.659	-21.530	39.660	67	31	299.288	135 16 11 .492	135 16 15 59.778
27.200	7.796	-12.330	39.660	73	21	299.151	153 11 46 40.756	153 11 51 39.904
27.200	10.360	-25.920	39.660	72	41	298.986	138 17 9 30.796	138 17 14 29.779
27.830	8.932	-25.920	39.660	73	12	300.008	153 11 40 50.094	153 11 45 50.101
28.210	11.410	-20.020	39.660	72	11	299.750	138 16 33 30.441	138 16 38 30.191
29.560	7.475	-7.480	39.660	66	12	297.491	135 13 53 45.999	135 13 58 43.489
30.490	7.103	-4.492	39.660	67	32	299.149	135 16 16 .526	135 16 20 59.672
32.050	8.772	-25.920	39.660	72	31	299.134	138 16 59 .620	138 17 3 59.754
32.450	7.365	-6.237	39.660	73	11	298.742	153 11 35 50.765	153 11 40 49.504
33.160	7.192	-3.946	39.660	71	12	299.574	138 15 14 10.641	138 15 19 10.211
33.300	8.709	-25.920	39.660	72	21	298.848	138 16 46 22.829	138 16 51 21.675
33.450	7.032	-16.300	39.660	71	11	299.566	138 15 9 10.495	138 15 14 10.059
33.920	7.026	-9.598	39.660	72	12	299.134	138 16 38 30.898	138 16 43 30.029
34.630	5.939	-6.745	39.660	66	11	299.144	135 13 48 40.309	135 13 53 39.449
34.680	6.423	-21.920	39.660	68	21	359.543	135 17 19 16.509	135 17 25 16.050
34.680	7.314	-25.920	39.660	68	12	295.419	135 17 12 33.107	135 17 17 28.524
34.760	5.880	-25.920	39.660	68	11	299.142	135 17 7 28.499	135 17 12 27.639
35.250	6.885	-25.920	39.660	71	31	298.721	138 15 31 10.531	138 15 36 9.250
35.800	5.378	0.961	39.660	68	22	358.295	135 17 25 22.344	135 17 31 20.639
35.930	6.007	-23.050	39.660	72	22	299.146	138 16 51 22.508	138 16 56 21.652
36.250	6.090	-23.130	39.660	71	21	299.176	138 15 20 15.474	138 15 25 14.650
36.310	5.787	-25.920	39.660	71	32	299.142	138 15 36 10.629	138 15 41 9.770
36.370	5.415	-9.730	39.660	71	41	299.130	138 15 42 40.824	138 15 47 39.951
37.680	3.882	-25.920	39.660	71	42	287.025	138 15 47 46.530	138 15 52 33.554

Appendix B:

CYCCAT.C
BEN_IO.C
READ.C
MAIN.H
MAKEFILE
LAST_BIN.DAT

```
/* cyccat.c   020492  Cycle Categorization program

        This program reads 400-byte, 101-datum records containing 1 pressure profile. When the new data with 5 profiles is available, a
new version will have to be written. The changes will be in the areas:
        - expand 'datain[]'
        - check read_in_chanls to see if it will handle new file (See pcbins.c for further detail).

        This version reads BRE format files - binary 4-byte reals, engineering units. For each cycle of data it computes the mean and
standard deviation of specified channels (one cycle consists of one set of data with azimuthal angle from 0 to 360 degrees). The data from
specified channels written to an ASCII file, hdr_file.out. This file may be used as-is to plot out data from each cycle. The data is also written
to a user file, hdr_file.sum. The following channels are specified:
        Wind Velocity (disc av wind speed channel)          #223
        Yaw (yaw angle - vpa prop vane direct hub height)   #60-#47
        Horizontal Shear  (3:00 - 9:00)/ 10.1               (#40-#44)/10.1
        Vertical Shear (12:00 - 6:00)/10.1                  (#38-#42)/10.1

WRITTEN:        M. Querijero, T. Young                      SERI/WIND                   */
/*      Wind Direction Modifications by Steve Huyer on 10/30/91     */

/* DECLARATION OF GLOBAL VARIABLES */

#define DEFALT_AZ_CHANL 57      /* default azimuth angle channel */
#define NDPMAX          40      /* max number of dependent variable */
#define NCHMAX          300     /* max number of channels */
#define NPTMAX          445     /* max number of pts/cycle */
#define NCYCM           440     /* max # cycles, 72rpm * 5m */
#define TRUE            1
#define FALSE           0

#include "main.h"
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <math.h>

extern  char    *get_response();    /* ben_io.c */
extern  int     read_in_chanls();  /* read.c  */

char            hdr_file    [80];   /* header file name & path */
char            out_file    [8];    /* Output file name */

int     ndep = 0;  /* number of dependent variables chosen */
int     arc_file;  /* file descriptor */

float   des_mean;/* desired mean to search for */
float   des_dev;  /* desired standard deviation to search for */
float   angle[NPTMAX];    /* angle data */

struct  channel_info        chan_des[NCHMAX]; /* desired channel array */

/* ********** MAIN PROGRAM ********** */

void    main (argc, argv)

int     argc;
char    *argv[];

/* MAIN can be run with or without arguments.  argv and argc specify whether or not to read a file containing desired channels. */

{       /* begin main */

/* VARIABLE DECLARATION -- MAIN */

 int    start;                  /* start cycle logical */
 int    npts;                   /* number of points in cycle, temporary */
```

```
int     i,j;
int     num_chnls;              /* number of channels to be read */
int     nread;                  /* size of read */
int     ncycle;                 /* number of cycles read */
int     sample_index;           /* index of sorting variable */
int     count;                  /* location counter for record information */
int     num_scans;              /* number of scans to make */
int     pass[NCYCM];            /* pass/fail logical for cycles */

char            start_string    [80];           /* start_time string */
char            end_string [80];                /* end_time string */
char            of_name         [80];   /* summary file name */

float   az_angle = 0;           /* temp azimuth angle variable */
float last_az;                  /* Previous azimuth angle value */
float   mydata[300];            /* data storage array */
float   sample_sum;             /* summation variable for sorting data */
float d_sum;                    /* x - mean, for stdev calculation */
float d_sqr;                    /* d_tot squared */
float d_tot;                    /* sigma squared */
float sample_avg;               /* average of each cycle */
float   sample[NPTMAX];         /* sorting data */
float   sample_dev;             /* stdev of each cycle */

float wd_sum;                   /*wind direction summary criteria*/
float wd[NPTMAX];
float wd_avg;
float wdsd_sum;
float wdsd_sqr;
float wdsd_tot;
float wd_dev;

float wsh[NPTMAX];          /*wind shear horizontal */
float wsh_avg;
float wsh_sum;
float wsh_sqr;
float wsh_tot;
float wsh_dev;
float wsv[NPTMAX];          /*wind shear vertical  */
float wsv_avg;
float wsv_sum;
float wsv_sqr;
float wsv_tot;
float wsv_dev;

long    loc;                    /* location of record information */

FILE    *of_unit;  /* user summary output file unit */
FILE *fout;                     /* numeric data output file */

/* BEGIN PROGRAM */

        printf ("\n*****************************************\n");
        printf ( "**             Cycle Categorization  Program  **\n");
        printf ( "*****************************************\n");

        arc_file = -1;
        while((
        arc_file = read_in_chanls(chan_des,&num_chnls,
                        start_string,end_string,&num_scans)) == -1);

/* get index to sort cycle on;  NOTE:  assumes a 360 degree cycle */

        sample_index = 223;   /* Disc Av Wind Speed for wind velocity*/

/* open up summary file */
```

```
        if (strrchr(hdr_file,'/') != NULL)
                strcpy(of_name,strrchr(hdr_file,'/')+1);
        else strcpy(of_name,hdr_file);

        of_name[strlen(of_name)-4] = '\0';
        strcpy (out_file, of_name);
        strcat(of_name,".sum");      /* this is user summary file */
/* output filename */
        strcat(out_file, ".out");            /* this is the numeric data file */
        printf("\nOutput cycle data will be written to files %s\n",out_file);
        unlink(of_name);
        of_unit = fopen(of_name,"w");

        if(of_unit == NULL) /* if file exists */
        {
                printf("Could not open %s for writing.\n",of_name);
                exit(-1);
        }
/* print user file header */
        (void)fprintf(of_unit,"Categorization of Data  %s\n\n",of_name);

        fout = fopen(out_file, "w");
        if(fout == NULL)     /* if file exists */
        {
                printf("Could not open %s for writing.\n",out_file);
                exit (-1);
        }

        printf("\nReading data file ... \n");

/* set up first read */

        count = 0; /*count indicates what record in file to get*/
        nread = REC_IN_SIZE;          /* bytes of data read */
        loc = count * REC_IN_SIZE;
        lseek (arc_file,loc,0);
        start = FALSE;       /* logical for cycle indicator */
        ncycle = 0;

/* do the following for all records in the file, binary read of data */

        while((nread = read(arc_file,mydata,REC_IN_SIZE) > 0 ))
        {       /* begin while */

                last_az = az_angle;
                az_angle = mydata[56];

/*  this series of if statements tests azimuthal angle to see what cycle it is in, whether to start or finish a cycle, etc. */

                if (!start)                     /* FALSE = 0 */
        /* check to see if we should start the cycle count */
                {       /* begin if */
                        start = TRUE;        /* start cycle */
                        sample_sum = 0.0;
                        sample_avg = 0.0;
                        sample_dev = 0.0;
                        wd_sum = 0.0;
                        wd_avg = 0.0;
                        npts = 0;
                        d_tot = 0.0;
                        wdsd_tot = 0.0;

                        wsh_sum = 0.0;
                        wsv_sum = 0.0;
                        wsh_avg = 0.0;
                        wsv_avg = 0.0;
                        wsh_dev = 0.0;
```

```
                wsv_dev = 0.0;
                wsh_tot = 0.0;
                wsv_tot = 0.0;

                ncycle++;
                pass[ncycle] = FALSE;                   /* set init value false */
                printf("\nStart cycle %d\n",ncycle);
        }       /* endif */

        if((az_angle < last_az) || (npts >= NPTMAX))
/* check to see if we should end the cycle count */
        {       /* begin if */
                start = FALSE;
                sample_avg =
                        (sample_sum/(float)npts);
                wd_avg = (wd_sum/(float)npts);
                wsh_avg = (wsh_sum/(float)npts);
                wsv_avg = (wsv_sum/(float)npts);

/* calculate standard deviation */
                for (i = 1;i<=npts;i++)
                {       /* begin for */
                        d_sum = sample[i] -
                                sample_avg;
                        d_sqr = d_sum * d_sum;
                        d_tot = d_tot + d_sqr;

                        wdsd_sum = wd[i]-wd_avg;
                        wdsd_sqr = wdsd_sum*wdsd_sum;
                        wdsd_tot = wdsd_tot + wdsd_sqr;

                        wsh_sum = wsh[i]-wsh_avg;
                        wsh_sqr = wsh_sum * wsh_sum;
                        wsh_tot = wsh_tot + wsh_sqr;

                        wsv_sum = wsv[i]-wsv_avg;
                        wsv_sqr = wsv_sum * wsv_sum;
                        wsv_tot = wsv_tot + wsv_sqr;

                }       /* end for */
                sample_dev = sqrt(d_tot/(float)npts);
                wd_dev = sqrt(wdsd_tot/(float)npts);
                wsv_dev = sqrt(wsv_tot/(float)npts);
                wsh_dev = sqrt(wsh_tot/(float)npts);

                /*  print cycle data */
                fprintf(of_unit,"   Cyc %3d  Vel:%6.3f+-%5.3f  Yaw:%6.3f+- %6.3f  HzShr:%6.3f+-%5.3f
                        Vshr:%6.3f+-%5.3f\n\n", ncycle,sample_avg,sample_dev,wd_avg,wd_dev,wsh_avg,
                        wsh_dev,wsv_avg,wsv_dev);  /* user file */

                if (ncycle != 1)    /* don't print 1st cycle-NaN */
                        fprintf(fout,"%3d%8.3f%8.3f%8.3f%8.3f%8.3f%8.3f%8.3f%8.3f\n",ncycle,sample_avg,
                                sample_dev,wd_avg,wd_dev,wsh_avg,wsh_dev,wsv_avg,wsv_dev);
                                /* numeric data file */
        }       /* endif (a complete cycle) */

        if (start)                              /* TRUE = 1 */
        {       /* begin if */
                npts++;
                angle[npts] = az_angle;
                sample[npts] = mydata[sample_index];
                sample_sum += sample[npts];
                wd[npts] = mydata[60]-mydata[47];
                wsh[npts] = (mydata[40]-mydata[44])/10.1;
                wsv[npts] = (mydata[38]-mydata[42])/10.1;
                wd_sum += wd[npts];
                wsv_sum += wsv[npts];
```

```
                    wsh_sum += wsh[npts];
                }           /* endif */

                count++;
                loc = count * REC_IN_SIZE;    /* set up next read */
                lseek(arc_file,loc,0);

        }           /* end while */

/* end loop for reading in information */

        fclose(of_unit);
        fclose(fout);
        (void)printf("\7\7\7\n");
}       /* end MAIN */
```

```
/* ben_io.c
/* This file contains IO functions*/

/* Functions include
        1:get_response
*/

#include   "main.h"
/* #include          </usr/ben/pressure/src/main.h> */
/*----------*/
/* function to read in an input string and return the string if the first value is a return, a default value is returned */

char *get_response(fin,defalt)
        FILE    *fin;
        char    *defalt;
{
        int     i;
        char    c;
        char    string  [80];
fflush(fin);
i = 0;
c = getc(fin);
while (c != '\n')
        {
        string[i] = c;
        c = getc(fin);
        ++i;
        }
string[i] = '\0';

if (i > 0)  return (string);

else        return (defalt);
}

/* function to read in an input string and return the string if the first value is a return, a default value is returned */

char *read_string(fin)
        FILE    *fin;
{
        int     i,j;
        char    c;
        char    string  [80];

i = 0;
c = getc(fin);

while (c != '\n' && c != '\0')
        {
        string[i] = c;
        c = getc(fin);
        ++i;
        }
string[i] = '\0';

if (i > 0)  return (string);

else        return (" \n");
}

/*----------------------*/
/* Function to test if you are closing the file pointer stdin '0' */

FCLOSE(fin)
        FILE    *fin;
{
if (fileno(fin) != 0)
```

```
        fclose(fin);

return;
}
```

```
/* read.c
   890317 - warning about different record lengths added
   890614 - different default hdr file added */
#include   "main.h"

extern     long        BYTES;
extern     long        REC_LEN;

static     int         i,j,k;
static     int         arc_file;

extern     char        hdr_file    [80];
extern     char        afile_name              [80];
extern     char        *get_response();

char pre_path[80];
char pre_file[80];
char cur_path[80];
char cur_file[80];

/*-----------------------------*/

read_in_chanls(header_name,chan,tot_num_chnls,start_time,end_time,num_scans)
           char        *header_name;
           struct      channel_info            chan[];
           int         *tot_num_chnls;
           char        *start_time;
           char        *end_time;
           int         *num_scans;
{
int        header_flag;
int        data_flag;

/* check if header file exists */

header_flag = check_header_file();

if (header_flag == -1)
           return (header_flag);

/* if the header file exists, check if the data file exists */

data_flag = check_data_file();

if (data_flag == -1)
           return (data_flag);

strcpy(header_name,hdr_file);

/* read in the header file */

read_header_file(chan,tot_num_chnls,start_time,end_time,num_scans);

return (arc_file);
}

/*#################################################################*/
prevpar()
{
static char fnam[] = "last_bin.dat";
FILE *fp;

           /* Read in parameters from last run */
           fp = fopen(fnam,"r");
           if(fp != NULL)                  /* If file exists */
           {
                      fscanf(fp,"%s",pre_path);                  /* Read previous path and */
```

```
                fscanf(fp,"%s",pre_file);                       /*      filename */
                fclose(fp);
        }
}
/*#################################################################*/
savepar()
{
static char fnam[] = "last_bin.dat";
FILE *fp;

        /* Write out current parameters for next run */
unlink(fnam);
fp = fopen(fnam,"w");
if(fp != NULL)                  /* If file exists */
{
        fprintf(fp,"%s\n",cur_path);
        fprintf(fp,"%s\n",cur_file);
        fclose(fp);
}
}
/*------------------------------------ */
/* This function will test if a header file exists. If it does then it will read the first line for a data format to confirm that its a  header file. If the
header file doesn't exist the user will be prompted for a new header file */

check_header_file()
{
FILE    *fin;

int     header_flag;

char    data_format     [5];
char    stuff           [80];
/*---------*/

prevpar();              /* Get prevous parameters */
printf("\nEnter Data DIRECTORY [%s] : ",pre_path);
strcpy(cur_path,get_response(stdin,pre_path));
strcpy(hdr_file,cur_path);
if( hdr_file[strlen(hdr_file)] != '/') strcat(hdr_file,"/");
printf("\nEnter HEADER file name [%s] : ",pre_file);
fflush(stdout);
strcpy(cur_file,get_response(stdin,pre_file));
strcat(hdr_file,cur_file);

if ((fin = fopen(hdr_file,"r")) == NULL)
        {
        printf("\n!!ERROR!!  %s does not exist\n ",hdr_file);
        printf("\nHit <RETURN> to continue : ");
        fflush(stdout);
        getc(stdin);
        header_flag = -1;
        } /* if */

/* if file exists check if its a header file by checking the first line for the data format */

else if (fscanf(fin,"%s\n",data_format) != EOF
        && strcmp("BRR",data_format) != 0 && strcmp("BRE",data_format) != 0
        && strcmp("BIR",data_format) != 0 && strcmp("BIE",data_format) != 0)
                {
                printf("\n!!ERROR!!  %s not a header file\n",hdr_file);
                printf("\nHit <RETURN> to continue : ");
                fflush(stdout);
                getc(stdin);
                header_flag = -1;
                } /* else if */

/* if the file exists and the first line is a data format, then set the header_flag to 1 for a good file */
```

```
else        header_flag = 1;

FCLOSE(fin);
savepar();

return (header_flag);

}
/*-------------------------------------------*/
/* This function will add a .dat extenter to a known header file and then check if the data file exists. If not the user will be prompted for a new
header file */

check_data_file()
{
int         data_flag;
int         count;
/*-----------*/

/* put a dat extender on the file name */

strcpy(afile_name,hdr_file);

count = strlen(afile_name);
afile_name[count-3] = 'd';
afile_name[count-2] = 'a';
afile_name[count-1] = 't';

/* check if data file exists  */

if ((arc_file = open(afile_name,0)) == -1)
            {
            printf("\n!!ERROR!!  %s does not exist",afile_name);
            printf("\n Hit <RETURN> to continue ");
            fflush(stdout);
            getc(stdin);
            }

return      (arc_file);

}

/*-------------------------------------------*/

read_header_file(chan,tot_num_chnls,start_string,end_string,num_scans)
            struct      channel_info        chan[];
            int         *tot_num_chnls;
            char        *start_string;
            char        *end_string;
            int         *num_scans;
{
char        stuff               [80];           /* junk character string */
char        cslope              [80];
char        coffset             [80];
char        data_format         [5];
char        name                [80];
char        unit                [80];

int         index;
int         num_ch_des;                         /* number of channel descriptions */
int         num_sets;                           /* number of channel sets */
int         num_chanls;                         /* number of channels */
int         raw_data;

double      delta_time;
FILE        *fin;                               /* parameter input file */
```

```
/*----------*/

/*  open header, read in & create parameter menu */

fin = fopen(hdr_file,"r");

/* read in the data format */

fscanf(fin,"%s\n",data_format);

/* if binary real BR, 4 bytes, if binary int BI, 2 bytes */

if (strcmp("BRR",data_format) == 0)
        {
        BYTES = 4;
        raw_data = 0;
        }

else if (strcmp("BRE",data_format) == 0)
        {
        BYTES = 4;
        raw_data = 1;
        }

else if (strcmp("BIR",data_format) == 0)
        {
        BYTES = 2;
        raw_data = 0;
        }

else if (strcmp("BIE",data_format) == 0)
        {
        BYTES = 2;
        raw_data = 1;
        };

/* read in daves comment string */

strcpy(stuff,read_string(fin));

/* read in the file start and end times */

strcpy(start_string,read_string(fin));
strcpy(end_string,read_string(fin));

/* read in the number of scans */

fscanf(fin,"%d %lf\n",num_scans,&delta_time);

/*read in the number of channel descriptions & the descriptions */

fscanf(fin,"%d\n",&num_ch_des);
for (i=1;i<=num_ch_des;++i)
        strcpy(stuff,read_string(fin));

/*read in the number of sets */

fscanf(fin,"%d\n",&num_sets);

index = 0;          /* index for channel names */

for(i=0;i<num_sets;++i)
        {

        /* read in the set number, description, version, & date taken */

        for (k=1;k<=4;++k)
```

```
                strcpy(stuff,read_string(fin));

        /* read in the number of channels */

        fscanf(fin,"%d\n",&num_chanls);

        for (j = 0;j < num_chanls;++j)
                {
                /* calculate the random access data offset location */

                chan[index+j].loc = BYTES * (index + j);

                /* skip 3 lines */

                for (k=1;k<=3;++k)
                        strcpy(stuff,read_string(fin));

                /* read in the channel offset and slope as strings */

                strcpy(cslope,read_string(fin));
                strcpy(coffset,read_string(fin));

                /* read the slope and offset from the above strings */
                /* if error reading then set slope 1, and offset 0 */

                if (sscanf(cslope,"%lf",&chan[index+j].slope) != 1)
                        chan[index+j].slope = 1.0;
                if (sscanf(coffset,"%lf",&chan[index+j].offset) != 1)
                        chan[index+j].offset = 0.0;

                /* skip 2 lines */

                for (k=1;k<=2;++k)
                        strcpy(stuff,read_string(fin));

                /* read in the channel max & min values */

                strcpy(stuff,read_string(fin));
                sscanf(stuff,"%f",&chan[index+j].max);

                strcpy(stuff,read_string(fin));
                sscanf(stuff,"%f",&chan[index+j].min);

                /* skip 1 line */

                strcpy(stuff,read_string(fin));

                /* read in the channel name, & units */

                strcpy(chan[index+j].name,read_string(fin));
                strcpy(chan[index+j].unit,read_string(fin));

                /* skip to the end of the channel des */

                for (k=13;k<=num_ch_des;++k)
                        strcpy(stuff,read_string(fin));
                }
        index = index + j;
        }

*tot_num_chnls = index;

/* set the record length        */

REC_LEN = *tot_num_chnls * BYTES;
if (REC_LEN != REC_IN_SIZE)
  {
```

```
 printf ("\nTHIS IS NOT A %d-BYTE RECORD FILE!!\n",REC_IN_SIZE);
 printf ("REC_LEN = %d\n", REC_LEN);
 exit (-1);
 }

/* close the header file              */

FCLOSE(fin);

} /* read_header_file */
```

```
/* main.h
 * General include files needed for the window program
 *
 */
#include <stdio.h>
#include <signal.h>
/* #include <errors.h> */
#include <sys/ioctl.h>
#include <math.h>

#define FULL 0                          /* full screen handle */
#define UNDEF -9999
#define REC_IN_SIZE 956 /* 239 channels * 4 bytes each */

struct      channel_info  {
                            char        name        [80];
                            char        unit        [80];
                            double      slope;
                            double      offset;
                            float       max;
                            float       min;
                            long        loc;
                            float       percent;
                            struct      channel_info        *next;
                            };

long        BYTES;
long        REC_LEN;
```

The following two files are also necessary to properly run CYCCAT.C:

1. makefile:

```
# MAKEFILE for cycle count software

 CFLAGS = -g

FILES  = cyccat.o read.o ben_io.c

MODULES = $(FILES)

# compile & link files

cycle: $(MODULES)
        cc $(CFLAGS) $(MODULES) -o cyccat -lm

$(MODULES) : main.h
```

2. last_bin.dat:

Appendix C

BESTCYC.F

```
      program bestcyc

************************************************************************************************************************
***********
*****   THIS PROGRAM IS USED TO IDENTIFY GROUPS OF CYCLES HAVING CERTAIN INFLOW CONDITIONS.  THE
      *****
*****   INFLOW CONDITIONS ANALYZED ARE THE WIND VELOCITY AND YAW.  THE PROGRAM OPERATES ON  *****
*****   THE DATABASE ESTABLISHED BY RUNNING THE PROGRAM cyccat ON EACH OF THE 59 EPISODES.  A LIST
      *****
*****   OF THE FILE NAMES GENERATED BY cyccat MUST BE LOCATED IN A FILE CALLED bestcyc.nam.  THE      *****
*****   PROGRAM GETS ITS NAME FROM THE ABILITY TO IDENTIFY GROUPS OF CYCLES HAVING LOW
      *****
*****   STANDARD DEVIATIONS IN YAW AND/OR VELOCITY.  THREE OPTIONS ARE AVAILABLE TO DO SO.  THE*****
*****   FIRST SIMPLY FINDS THE MEAN AND STANDARD DEVIATION IN VELOCITY AND YAW FOR A GIVEN      *****
*****   RANGE OF CYCLES ON A GIVEN TAPE.  THE SECOND OPTION LOCATES THE n CONSECUTIVE CYCLES ON
      *****
*****   EACH TAPE HAVING THE LOWEST STANDARD DEVIATION IN YAW OR VELOCITY (WHERE n IS A USER   *****
*****   DEFINED NUMBER OF CYCLES).  THE THIRD OPTION FINDS ALL GROUPS OF n CYCLES THAT HAVE A   *****
*****   MEAN VELOCITY AND YAW CORRESPONDING TO A SET OF USER DEFINED LIMITS.  EITHER OR BOTH    *****
*****   PARAMETERS MAY BE LIMITED.  THIS OPTION HAS TYPICALLY BEEN THE MOST USEFUL.  IT ALLOWS  *****
*****   FOR THE USER TO, FOR INSTANCE, FIND ALL GROUPS OF 50 CONSECUTIVE CYCLES HAVING A MEAN   *****
*****   VELOCITY BETWEEN 9.5 m/s AND 10.5 m/s AND A MEAN YAW BETWEEN -2.5 deg AND 2.5 deg.  THE USER *****
*****   CAN THEN VISUALLY INSPECT THE LIST TO DETERMINE WHICH SET SHOULD BE ANALYZED.  THIS IS  *****
*****   NECESSARY BECAUSE A GROUP OF CYCLES MAY HAVE A VERY LOW STANDARD DEVIATION IN
      *****
*****   VELOCITY BUT A LARGE YAW STANDARD DEVIATION.  VISUAL INSPECTION ALLOWS THE USER TO      *****
*****   OPTIMIZE THE CONSTANCY OF BOTH PARAMETERS.                                              *****
*****                                                                                           *****
*****   WRITTEN BY DEREK SHIPLEY                                                                *****
************************************************************************************************************************
***********

*********************************************************************************
*****   VARIABLE DEFINITIONS:                                                   *****
*****    opt      OPTION NUMBER FOR PROCESSING DATA                             *****
*****    tape()   ARRAY HOLDING NAMES OF INFLOW DATABASE FILES                  *****
*********************************************************************************

*****   VARIABLES  *****
      integer opt
      character tape(59)*11

*****   SET THE FILE NAMES CONTAINING INFLOW DATABASE FOR EACH TAPE  *****
      data (tape(j),j=1,28)/
     2  'd065011.out','d065012.out','d065021.out','d065022.out','d066011.out','d066012.out','d066021.out','d066022.out','d066031.out',
     3  'd066032.out','d067011.out','d067012.out','d067021.out','d067022.out','d067031.out','d067032.out','d068011.out','d068012.out',
     4  'd068021.out','d068022.out','d069021.out','d069022.out','d069031.out','d069032.out','d070011.out','d070012.out','d070021.out',
     5  'd070022.out'/
      data (tape(j),j=29,59)/
     2  'd070031.out','d070032.out','d070041.out','d070042.out','d071011.out','d071012.out','d071021.out','d071031.out','d071032.out',
     3  'd071041.out','d071042.out','d072011.out','d072012.out','d072021.out','d072022.out','d072031.out','d072032.out','d072041.out',
     4  'd072042.out','d073011.out','d073012.out','d073021.out','d073022.out','d073031.out',d073032.out','d073041.out','d073042.out',
     5  'd075011.out','d075012.out','d075021.out','d075022.out'/

*****   PRINT A CHEERFUL GREETING TO THE USER  *****
      print*
      print*, '************************************************************'
      print*, '*****   Welcome to BESTCYC - the event locator program   *****'
      print*, '************************************************************'

*****   DISPLAY MAIN MENU CONTAINING THREE PROCESSING OPTIONS  *****
100   print*
      print*, 'Which option do you wish to execute?'
      print*, ' 1.  Find the average for a given cycle range.'
      print*, ' 2.  Find the best cycles for either velocity or yaw angle.'
      print*, ' 3.  Find consecutive cycles corresponding to input criteria.'
      print*, ' 4.  Quit'
```

```
      write(6,1)
1     format('Enter the desired option number (1-4): ',$)
      read*, opt
***** BRANCH TO PROPER OPTION *****
      if (opt.eq.1) then
       call avegroup(tape)
      elseif (opt.eq.2) then
       call bestset(tape)
      elseif (opt.eq.3) then
       call fitcrit(tape)
      elseif (opt .eq. 4) then
       goto 200
      else
       print*,'Invalid option number.  Please try again.'
       goto 100
      endif
      goto 100

200   continue
      stop
      end

*******************************************************************************
*******************************************************************************

      subroutine avegroup(tape)

*******************************************************************************
*****   OPTION 1 - DETERMINE THE AVERAGE YAW AND VELOCITY AND          *****
*****   THEIR STANDARD DEVIATIONS FOR A GIVEN GROUP OF CYCLES  *****
*******************************************************************************

*****************************************************************************************************
*****   VARIABLE AND PARAMETER DEFINITIONS:                                                     *****
*****    cyc()          CYCLE NUMBER OF EACH DATABASE ENTRY                                     *****
*****    cychigh        UPPER BOUNDARY OF PRESENT CYCLE GROUPING                                *****
*****    cyclow         LOWER BOUNDARY OF PRESENT CYCLE GROUPING                                *****
*****    data(,)        DATA READ FROM INFLOW DATABASE FILES                                    *****
*****    infil          INPUT FILE NAME                                                         *****
*****    MAXCYC         MAXIMUM NUMBER OF CYCLES PER TAPE                                       *****
*****    NCOL           NUMBER OF DATA COLUMNS IN DATABASE FILES                                *****
*****    NUMTAPE        NUMBER OF DATA TAPES                                                    *****
*****    tape()         ARRAY HOLDING NAMES OF INFLOW DATABASE FILES                            *****
*****    vel()          INFLOW VELOCITY OF EACH DATABASE ENTRY                                  *****
*****    velsig()       VELOCITY STANDARD DEVIATION OF EACH ENTRY                               *****
*****    velmean        MEAN VELOCITY OF PRESENT CYCLE GROUPING                                 *****
*****    velsd          VELOCITY STANDARD DEVIATION OF PRESENT CYCLES                           *****
*****    velsdsum       SUM OF VELOCITY RESIDUALS SQUARED                                       *****
*****    velsum         SUM OF VELOCITIES IN PRESENT CYCLE GROUPING                             *****
*****    yaw()          YAW OF EACH DATABASE ENTRY                                              *****
*****    yawsig()       YAW STANDARD DEVIATION OF EACH DATABASE ENTRY                           *****
*****    yawmean        MEAN YAW OF PRESENT CYCLE GROUPING                                      *****
*****    yawsd          YAW STANDARD DEVIATION OF PRESENT CYCLES                                *****
*****    yawsdsum       SUM OF YAW RESIDUALS SQUARED IN                                         *****
*****    yawsum         SUM OF YAW IN PRESENT CYCLE GROUPING                                    *****
*****************************************************************************************************

***** DECLARE VARIABLES AND CONSTANTS *****
      parameter (MAXCYC=450)
      parameter (NCOL=9)
      parameter (NUMTAPE=59)
      real data(NCOL,MAXCYC),vel(MAXCYC),yaw(MAXCYC),cyc(MAXCYC)
      real velsig(MAXCYC),yawsig(MAXCYC)
      real yawsum,yawmean,yawsdsum,yawsd
      real velsum,velmean,velsdsum,velsd
      integer cyclow,cychigh
      character infil*11,tape(NUMTAPE)*11
```

```
*****   PROMPT THE USER FOR THE INPUT FILE AND OPEN IT  *****
100     print*
        write(6,1)
1       format('Enter the name of the input file (ext = .out): ',$)
        read*, infil
        open (unit=11,file=infil,iostat=inerr,status='old')
        if (inerr .ne. 0) then
          print*, 'File does not exist.  Please try again.'
          goto 100
        endif

*****   PROMPT THE USER FOR THE LOWEST CYCLE TO BE AVERAGED  *****
        print*
        write(6,2)
2       format('Enter the lowest cycle to average: ',$)
        read*, cyclow

*****   PROMPT THE USER FOR THE HIGHEST CYCLE TO BE AVERAGED  *****
        write(6,3)
3       format('Enter the highest cycle to average: ',$)
        read*, cychigh

*****   READ IN ALL THE DATA FROM THE GIVEN FILE  *****
        n = 2
15      read(11,*,end=16) (data(j,n),j=1,NCOL)
        cyc(n) = data(1,n)
        vel(n) = data(2,n)
        velsig(n) = data(3,n)
        yaw(n) = data(4,n)
        yawsig(n) = data(5,n)
        n = n+1
        goto 15
16      continue
        close (11)

*****   CALCULATE THE MEAN YAW AND VELOCITY FOR THE SPECIFIED CYCLES  *****
        velsum = 0
        yawsum = 0
        do 18,i=cyclow,cychigh
          velsum = velsum+vel(i)
          yawsum = yawsum+yaw(i)
18      continue
        velmean = velsum/(cychigh-cyclow+1)
        yawmean = yawsum/(cychigh-cyclow+1)

*****   CALCULATE THE VELOCITY AND YAW STANDARD DEVIATIONS  *****
        velsdsum = 0
        yawsdsum = 0
        do 19,j=cyclow,cychigh
          velsdsum = velsdsum+velsig(j)**2+(vel(j)-velmean)**2
          yawsdsum = yawsdsum+yawsig(j)**2+(yaw(j)-yawmean)**2
19      continue
        velsd = sqrt(velsdsum/(cychigh-cyclow+1))
        yawsd = sqrt(yawsdsum/(cychigh-cyclow+1))

*****   PRINT THE RESULTS TO THE SCREEN  *****
        print*
        print*, 'The results for cycles',cyclow,' -',cychigh,' is:'
        print9000
        print9100,infil,cyclow,cychigh,yawmean,yawsd,velmean,velsd
        print*
        print*

9000    format (/t12,'Tape',t22,'Cycles',t33,'Avg Yaw',t43,'Yaw S.D.',t53,'Avg Vel',t63,'Vel S.D.')
9100    format (10x,a7,3x,i3,2x,i3,4x,f6.2,4x,f6.2,4x,f6.2,4x,f6.2)
```

```
      return
      end

*************************************************************************
*************************************************************************

      subroutine bestset(tape)

*****************************************************************************************************************
***********
*****   OPTION 2 - FIND THE MOST CONSTANT CYCLES ON EACH TAPE WITH RESPECT TO YAW AND VELOCITY,
      *****
*****   AND PRINT THE MEANS AND STANDARD DEVIATIONS FOR THAT CYCLE RANGE TO A FILE                    *****
*****************************************************************************************************************
***********

*****************************************************************************************************
*****   VARIABLE AND PARAMETER DEFINITIONS:                                                   *****
*****    avgnum          NUMBER OF CONSECUTIVE CYCLES TO BE EXAMINED                          *****
*****    choice          FLAG FOR MINIMIZATION OF VELOCITY OR YAW                             *****
*****    cyc()           CYCLE NUMBER OF EACH DATABASE ENTRY                                  *****
*****    cychigh         UPPER BOUNDARY OF PRESENT CYCLE GROUPING                             *****
*****    cyclow          LOWER BOUNDARY OF PRESENT CYCLE GROUPING                             *****
*****    data(,)         DATA READ FROM INFLOW DATABASE FILES                                 *****
*****    infil           INPUT FILE NAME                                                      *****
*****    MAXCYC          MAXIMUM NUMBER OF CYCLES PER TAPE                                    *****
*****    NCOL            NUMBER OF DATA COLUMNS IN DATABASE FILES                             *****
*****    ncyc            NUMBER OF CYCLES ON A DATA TAPE                                      *****
*****    NUMTAPE         NUMBER OF DATA TAPES                                                 *****
*****    tape()          ARRAY HOLDING NAMES OF INFLOW DATABASE FILES                         *****
*****    vel()           INFLOW VELOCITY OF EACH DATABASE ENTRY                               *****
*****    velsig()        VELOCITY STANDARD DEVIATION OF EACH ENTRY                            *****
*****    velmean         MEAN VELOCITY OF PRESENT CYCLE GROUPING                              *****
*****    velsd           VELOCITY STANDARD DEVIATION OF PRESENT CYCLES                        *****
*****    velsdbst        LOWEST VELOCITY STANDARD DEVIATION                                   *****
*****    velsdsum        SUM OF VELOCITY RESIDUALS SQUARED                                    *****
*****    velsum          SUM OF VELOCITIES IN PRESENT CYCLE GROUPING                          *****
*****    yaw()           YAW OF EACH DATABASE ENTRY                                           *****
*****    yawsig()        YAW STANDARD DEVIATION OF EACH DATABASE ENTRY                        *****
*****    yawmean         MEAN YAW OF PRESENT CYCLE GROUPING                                   *****
*****    yawsd           YAW STANDARD DEVIATION OF PRESENT CYCLES                             *****
*****    yawsdbst        LOWEST YAW STANDARD DEVIATION                                              *****
*****    yawsdsum        SUM OF YAW RESIDUALS SQUARED IN                                      *****
*****    yawsum          SUM OF YAW IN PRESENT CYCLE GROUPING                                 *****
*****************************************************************************************************

*****   DECLARE VARIABLES AND CONSTANTS  *****
      parameter (MAXCYC=450)
      parameter (NCOL=9)
      parameter (NUMTAPE=59)
      real data(NCOL,MAXCYC),vel(MAXCYC),yaw(MAXCYC),cyc(MAXCYC)
      real velsig(MAXCYC),yawsig(MAXCYC)
      real yawsum,yawmean,yawsdsum,yawsd,yawsdbst
      real velsum,velmean,velsdsum,velsd,velsdbst
      integer avgnum,ncyc,cyclow,cychigh
      character infil*11,outfil*30,tape(NUMTAPE)*11
      character choice

*****   DETERMINE WHETHER YAW OR VELOCITY S.D. IS TO BE MINIMIZED  *****
50    print*
      write(6,1)
1     format('Do you want to minimize Velocity or Yaw angle S.D.? (v/y): ',$)
      read*, choice
      if (choice.ne.'v'.and.choice.ne.'V'.and.choice.ne.'y'.and.choice.ne.'Y') then
       print*, 'Invalid response'
       goto 50
      endif
```

```
*****   PROMPT THE USER FOR THE DESIRED SIZE OF CYCLE GROUP  *****
        print*
        write(6,2)
2       format('Enter the number of consecutive cycles to evaluate: ',$)
        read*, avgnum

*****   PROMPT THE USER FOR THE NAME OF THE OUTPUT FILE AND OPEN IT  *****
        print*
        write(6,3)
3       format('Enter the name of the output file: ',$)
        read*, outfil
        open (unit=16,file=outfil,status='unknown')

*****   WRITE A HEADER TO THE TOP OF THE OUTPUT FILE  *****
        if (choice.eq.'y'.or.choice.eq.'Y') then
         write (16,*),'      The most constant',avgnum,' cycles in terms of the yaw angle for each tape:'
         write (16,9000)
        else
         write (16,*),'      The most constant',avgnum,' cycles in terms of velocity for each tape:'
         write (16,9000)
        endif

*****   OPEN EACH INFLOW DATABASE FILE  *****
        do 700, n=1,NUMTAPE
         infil = tape(n)
         k = 2
         open (unit=11,file=infil,status='old')
*****   READ IN ALL DATA FROM EACH TAPE AND PLACE IT IN ARRAYS  *****
100      read (11,*,end=200)(data(j,k),j=1,NCOL)
          cyc(k) = data(1,k)
          vel(k) = data(2,k)
          velsig(k) = data(3,k)
          yaw(k) = data(4,k)
          yawsig(k) = data(5,k)
          k = k+1
          goto 100
200      continue
         close (11)
         ncyc = k-1
        print*,n,ncyc

*****   CALCULATE THE MEAN VELOCITYU AND YAW FOR EACH POSSIBLE BLOCK OF avgnum CYCLES          *****
         velsdbst = 200
         yawsdbst = 200
         do 300,i=2,ncyc-avgnum
          velsum = 0
          yawsum = 0
          do 400,j=1,avgnum
           velsum = velsum+vel(i+j-1)
           yawsum = yawsum+yaw(i+j-1)
400       continue
          yawmean = yawsum/avgnum
          velmean = velsum/avgnum

*****   CALCULATE THE VELOCITY AND YAW S.D. FOR EACH BLOCK  *****
          velsdsum=0
          yawsdsum=0
          do 500,j=1,avgnum
           velsdsum=velsdsum+velsig(i+j-1)**2+(vel(i+j-1)-velmean)**2
           yawsdsum=yawsdsum+yawsig(i+j-1)**2+(yaw(i+j-1)-yawmean)**2
500       continue
          velsd=sqrt(velsdsum/(avgnum-1))
          yawsd=sqrt(yawsdsum/(avgnum-1))

****    DETERMINE WHETHER THIS COMBINATION IS THE MOST CONSTANT  *****
          if (choice.eq.'y'.or.choice.eq.'Y') then
```

```
            if (yawsd .lt. yawsdbest) then
              yawbest=yawmean
              yawsdbst=yawsd
              velbest=velmean
              velsdbst=velsd
              cyclow=i
              cychigh=i+avgnum-1
            endif
           elseif (choice.eq.'v'.or.choice.eq.'V') then
            if (velsd .lt. velsdbst) then
              yawbest=yawmean
              yawsdbst=yawsd
              velbest=velmean
              velsdbst=velsd
              cyclow=i
              cychigh=i+avgnum-1
            endif
           endif
300        continue
           write(16,9100)infil,cyclow,cychigh,velbest,velsdbst,yawbest,yawsdbst
700        continue

9000       format (/t12,'Tape',t22,'Cycles',t33,'Avg Vel',t43,'Vel S.D.',t53,'Avg Yaw',t63,'Yaw S.D.')
9100       format (10x,a7,3x,i3,2x,i3,4x,f6.2,4x,f6.2,4x,f6.2,4x,f6.2)

           return
           end

*************************************************************************
*************************************************************************

           subroutine fitcrit(tape)

*****************************************************************************************************************
*
*****      OPTION 3 - TAKE A SET OF INFLOW CRITERIA PROVIDED BY THE USER AND THE NUMBER OF  *****
*****      CONSECUTIVE CYCLES TO MAKE UP A BLOCK AND FIND ALL BLOCKS THAT FIT THE CRITERIA          *****
*****************************************************************************************************************
*

*****************************************************************************************************************
*****      VARIABLE AND PARAMETER DEFINITIONS:                                                          *****
*****       avgnum          NUMBER OF CONSECUTIVE CYCLES TO BE EXAMINED                                 *****
*****       choice          FLAG FOR MINIMIZATION OF VELOCITY OR YAW                                    *****
*****       cyc()           CYCLE NUMBER OF EACH DATABASE ENTRY                                         *****
*****       cychigh         UPPER BOUNDARY OF PRESENT CYCLE GROUPING                                    *****
*****       cyclow          LOWER BOUNDARY OF PRESENT CYCLE GROUPING                                    *****
*****       data(,)         DATA READ FROM INFLOW DATABASE FILES                                        *****
*****       infil           INPUT FILE NAME                                                             *****
*****       MAXCYC          MAXIMUM NUMBER OF CYCLES PER TAPE                                           *****
*****       NCOL            NUMBER OF DATA COLUMNS IN DATABASE FILES                                    *****
*****       ncyc            NUMBER OF CYCLES ON A DATA TAPE                                             *****
*****       NUMTAPE         NUMBER OF DATA TAPES                                                        *****
*****       tape()          ARRAY HOLDING NAMES OF INFLOW DATABASE FILES                                *****
*****       vel()           INFLOW VELOCITY OF EACH DATABASE ENTRY                                      *****
*****       velsig()        VELOCITY STANDARD DEVIATION OF EACH ENTRY                                   *****
*****       velhigh         VELOCITY UPPER LIMIT FOR RETAINING CYCLES                                   *****
*****       vellow          VELOCITY LOWER LIMIT FOR RETAINING CYCLES                                   *****
*****       velmean         MEAN VELOCITY OF PRESENT CYCLE GROUPING                                     *****
*****       velsd           VELOCITY STANDARD DEVIATION OF PRESENT CYCLES                               *****
*****       velsdsum        SUM OF VELOCITY RESIDUALS SQUARED                                           *****
*****       velsum          SUM OF VELOCITIES IN PRESENT CYCLE GROUPING                                 *****
*****       yaw()           YAW OF EACH DATABASE ENTRY                                                  *****
*****       yawsig()        YAW STANDARD DEVIATION OF EACH DATABASE ENTRY                               *****
*****       yawhigh         YAW UPPER LIMIT FOR RETAINING CYCLES                                        *****
*****       yawlow          YAW LOWER LIMIT FOR RETAINING CYCLES                                        *****
*****       yawmean         MEAN YAW OF PRESENT CYCLE GROUPING                                          *****
```

```
*****     yawsd          YAW STANDARD DEVIATION OF PRESENT CYCLES                    *****
*****     yawsdsum       SUM OF YAW RESIDUALS SQUARED IN                             *****
*****     yawsum         SUM OF YAW IN PRESENT CYCLE GROUPING                        *****
*************************************************************************************************

*****     DECLARE VARIABLES AND CONSTANTS  *****
          parameter (MAXCYC=450)
          parameter (NCOL=9)
          parameter (NUMTAPE=59)
          real data(NCOL,MAXCYC),vel(MAXCYC),yaw(MAXCYC),cyc(MAXCYC)
          real velsig(MAXCYC),yawsig(MAXCYC)
          real yawsum,yawmean,yawsdsum,yawsd,yawhigh,yawlow
          real velsum,velmean,velsdsum,velsd,velhigh,vellow
          integer avgnum,ncyc,cyclow,cychigh
          character infil*11,outfil*30,tape(NUMTAPE)*11
          character choice

*****     PROMPT THE USER FOR THE DESIRED NUMBER OF CONSECUTIVE CYCLES  *****
          print*
          write(6,1)
1         format('Enter the number of consecutive cycles to examine: ',$)
          read*, avgnum

*****     PROMPT THE USER FOR THE NAME OF THE OUTPUT FILE AND OPEN IT  *****
          print*
          write(6,2)
2         format('Enter the name of the output file: ',$)
          read*, outfil
          open (unit=12, file=outfil, status='unknown')

*****     DETERMINE IF CYCLES HAVE TO MEET VELOCITY AND/OR YAW CRITERIA  *****
100       print*
          write(6,3)
          write(6,4)
3         format('Do you wish to specify Yaw or Velocity input criteria or Both?')
4         format('Please enter a V, Y, or B: ',$)
          read*, choice
          if (choice.ne.'v'.and.choice.ne.'V'.and.choice.ne.'y'.and.choice.ne.'Y'.and.choice.ne.'b'.and.choice.ne.'B') then
           print*, 'Invalid Option.  Please try again.'
           print*
           goto 100
          endif

*****     SPECIFY YAW CRITERIA IF SO DESIRED  *****
          if (choice.ne.'v'.and.choice.ne.'V') then
           print*
           write(6,5)
5          format('Enter the lowest yaw angle you desire: ',$)
           read*, yawlow
           write(6,6)
6          format('Enter the highest yaw angle you desire: ',$)
           read*, yawhigh
          endif

*****     SPECIFY VELOCITY CRITERIA IF SO DESIRED  *****
          if (choice.ne.'y'.and.choice.ne.'Y') then
           print*
           write(6,7)
7          format('Enter the lowest velocity you desire: ',$)
           read*, vellow
           write(6,8)
8          format('Enter the highest velocity you desire: ',$)
           read*, velhigh
          endif

*****     WRITE HEADER TO OUTPUT FILE  *****
          if (avgnum .eq. 1) then
```

```
          write(12,*), 'All cycles that meet the following criteria:'
         else
          write(12,*), 'All combinations of',avgnum,' consecutive cycles that meet the following critera:'
         endif
         write(12,*)
         if (choice.ne.'y'.and.choice.ne.'Y') write(12,9200),vellow,velhigh
         if (choice.ne.'v'.and.choice.ne.'V') write(12,9300),yawlow,yawhigh
         if (avgnum .eq. 1) then
          write(12,9050)
         else
          write(12,9000)
         endif

*****    OPEN EACH INFLOW DATABASE FILE  *****
         do 150,n=1,NUMTAPE
          infil = tape(n)
          open (unit=13,file=infil,status='old')
*****    READ IN ALL DATA FROM EACH TAPE AND PLACE IT IN ARRAYS  *****
          k = 2
200       read (13,*,end=300) (data(j,k),j=1,NCOL)
           cyc(k) = data(1,k)
           vel(k) = data(2,k)
           velsig(k) = data(3,k)
           yaw(k) = data(4,k)
           yawsig(k) = data(5,k)
           k = k+1
           goto 200
300       continue
          close(13)
          ncyc = k-1

*****    FOR EACH BLOCK CALCULATE THE MEAN VELOCITY AND YAW  *****
          do 400,i=2,ncyc-avgnum+1
           if (avgnum .eq. 1) then
            velmean = vel(i)
            yawmean = yaw(i)
           else
            velsum = 0
            yawsum = 0
            do 500,j=1,avgnum
             velsum = velsum+vel(i+j-1)
             yawsum = yawsum+yaw(i+j-1)
500         continue
            yawmean = yawsum/avgnum
            velmean = velsum/avgnum
           endif

*****       CALCULATE THE VELOCITY AND YAW STANDARD DEVIATIONS  *****
           if (avgnum .eq. 1) then
            velsd = velsig(i)
            yawsd = yawsig(i)
           else
            velsdsum = 0
            yawsdsum = 0
            do 600,j=1,avgnum
             velsdsum = velsdsum+velsig(i+j-1)**2+(vel(i+j-1)-velmean)**2
             yawsdsum = yawsdsum+yawsig(i+j-1)**2+(yaw(i+j-1)-yawmean)**2
600         continue
            velsd = sqrt(velsdsum/(avgnum-1))
            yawsd = sqrt(yawsdsum/(avgnum-1))
           endif

*****    SET THE BOUNDARY CYCLES FOR THE PRESENT BLOCK  *****
           cyclow = i
           cychigh = i+avgnum-1

*****    DETERMINE IF THIS BLOCK MEETS THE INFLOW CRITERIA  *****
```

```
*****   IF IT DOES, THEN WRITE THE DATA TO THE OUTPUT FILE  *****
650       if (choice .eq. 'y' .or. choice .eq. 'Y') then
           if (yawmean .le. yawhigh .and. yawmean .ge. yawlow) then
                 if (avgnum .eq. 1) then
                  write (12,9150) infil,cyclow,velmean,velsd,yawmean,yawsd
                 else
                  write (12,9100) infil,cyclow,cychigh,velmean,velsd,yawmean,yawsd
                 endif
           endif
          elseif (choice.eq.'v'.or.choice.eq.'V') then
           if (velmean .le. velhigh .and. velmean .ge. vellow) then
                 if (avgnum .eq. 1) then
                  write (12,9150) infil,cyclow,velmean,velsd,yawmean,yawsd
                 else
                  write (12,9100) infil,cyclow,cychigh,velmean,velsd,yawmean,yawsd
                 endif
           endif
          else
           if (yawmean .le. yawhigh .and. yawmean .ge. yawlow .and.velmean .le. velhigh .and. velmean .ge. vellow) then
                 if (avgnum .eq. 1) then
                  write (12,9150) infil,cyclow,velmean,velsd,yawmean,yawsd
                 else
                  write (12,9100) infil,cyclow,cychigh,velmean,velsd,yawmean,yawsd
                 endif
           endif
          endif
400        continue
150       continue

9000      format (/t12,'Tape',t22,'Cycles',t33,'Avg Vel',t43,'Vel S.D.',t53,'Avg Yaw',t63,'Yaw S.D.')
9050      format (/t12,'Tape',t22,'Cycle',t32,'Avg Vel',t42,'Vel S.D.',t52,'Avg Yaw',t62,'Yaw S.D.')
9100      format (10x,a7,3x,i3,2x,i3,4x,f6.2,4x,f6.2,4x,f6.2,4x,f6.2)
9150      format (10x,a7,5x,i3,6x,f6.2,4x,f6.2,4x,f6.2,4x,f6.2)
9200      format ('Average velocity between ',f5.2,' and ',f5.2,' m/s')
9300      format ('Average yaw angle between ',f6.2,' and ',f6.2,' degrees')

          return
          end
```

Appendix D

AVCNCT.C
DIRECTORY.DAT
MAKEFILE
LAST_BIN.DAT

```
/* avcnct.c --> avcnct*                041193          Cycle Averaging/Minimization Program

        This program reads 400-byte, 101-datum records containing 1 pressure profile.

        This version reads BRE format files - binary 4-byte reals, engineering units. For each cycle of data it computes the mean and
standard deviation of specified channels,ct & cn (one cycle consists of one set of data with azimuthal angle from 0 to 360 degrees). The data
from specified channels written to 3 ASCII files, hdr_file.out, one for cn & one for ct means & std dev, and one for cn & ct maximums. This
file may be used as-is to plot out data from each cycle. The data is also written to user files, hdr_file.sum. The user files contain headings.

The following channels are specified:

        #203    Normal Force Cn(80%)
        #205    Normal Force Cn(63%)
        #204    Normal Force Cn(47%)
        #206    Normal Force Cn(30%)

        #207    Normal Force Ct(80%)
        #208    Normal Force Ct(63%)
        #209    Normal Force Ct(47%)
        #210    Normal Force Ct(30%)

NOTE:   If using COMBINED EXPERIMENT PHASE II - REDUCED DATA FILE SUMMARY for the channels (when modifying
program), the correct mydata index number is the listed channel # - 1. For example, Cn(80%) is listed as channel #204, so 203 is used here.

 This version calls cln.c --> cln(float datarray, int numpts, int found) to remove anomalous points more than X standard deviations from the
mean. (At this writing, cln removes points which change more than 2(or 3)*standard deviations from the previous point.) If anomalous points
are found, the cycle number is recorded in a hdr_file.err file.

 INPUT: Raw binary data from Combined Experiment optical tapes
 OUTPUT:

 hdr_filecn.out & hdr_filecn.sum
        cycle# 30%span_cn+-sd 47%span_cn+-sd 63%span_cn+-sd 80%span_cn+-sd
 hdr_filect.out & hdr_filect.sum
        cycle# 30%span_ct+-sd 47%span_ct+-sd 63%span_ct+-sd 80%span_ct+-sd
 hdr_filemax.out & hdr_filemax.sum
        cycle# cn values for: 30%spanmax 47%spanmax 63%spanmax 80%spanmax
        ct values for: 30%spanmax 47%spanmax 63%spanmax 80%spanmax
 hdr_file.err
        anomalous point found in cycle#

WRITTEN:        M. Querijero, T. Young          SERI/WIND               */

/* DECLARATION OF GLOBAL VARIABLES */

#define DEFALT_AZ_CHANL 57      /* default azimuth angle channel */
#define NCHMAX          300     /* max number of channels */
#define NPTMAX          445     /* max number of pts/cycle */
#define NCYCMAX         440     /* max # cycles, 72rpm * 5m */
#define TRUE            1
#define FALSE           0
#define span            4       /* number of span locations of Cn & Ct data*/
#define cn30            206     /* mydata index number for 30% span cn */
#define ct30            210     /* mydata index number for 30% span ct */
#define cnmagnitude 2           /* for cln 2* standard deviation for cn */
#define ctmagnitude 3           /* for cln 3* stnadard deviation for ct */

#include "../old/main.h"
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <math.h>

extern  char    *get_response();    /* ben_io.c */
extern  int     read_in_chanls();  /* read.c  */
void    cln();
```

```
char            hdr_file    [80];   /* header file name & path */
char            cnsum_name  [80];   /* Output file name */
char            ctsum_name  [80];   /* Output file name */
char            maxsum_name [80];   /* Output file name */
char            cnout_name  [80];   /* Output file name */
char            ctout_name  [80];   /* Output file name */
char            maxout_name [80];   /* Output file name */
char            in_name     [80];   /* Input file name */
char            err_name    [80];   /* Anomalous listing file name */

FILE    *fpcnsum;                   /* user summary output files */
FILE    *fpctsum;                   /* user summary output files */
FILE    *fpmaxsum;                  /* user summary output files */

FILE *fpcnout;                      /* numeric data output files */
FILE *fpctout;                      /* numeric data output files */
FILE *fpmaxout;                     /* numeric data output files */
FILE    *fperr;                     /* listing of anomalous corrections
                                    ** returned by cln */

int     arc_file;   /* file descriptor */

float       angle[NPTMAX];          /* angle data */
struct channel_info chan_des[NCHMAX];       /*desired channel array*/


/* ********** MAIN PROGRAM ********** */

void    main (argc, argv)

int     argc;
char    *argv[];

/*  MAIN can be run with or without arguments.  argv and argc specify whether or not to read a file containing desired channels.*/

{       /* begin main */

/* VARIABLE DECLARATION -- MAIN */

 int    start;                      /* start cycle logical */
 int    npts;                       /* number of points in cycle, temporary */
 int    i,j;
 int    num_chnls;                  /* number of channels to be read */
 int    nread;                      /* size of read */
 int    ncycle = 0;                         /* number of cycles read */
 int    vel_index;                  /* index of sorting variable */
 int    ch;                         /* user specified data index */
 int    count;                      /* location counter for record information */
 int    num_scans;                  /* number of scans to make */
 int    pass[NCYCMAX];              /* pass/fail logical for cycles */
 int    first;                      /* first desired cycle or 0 for all cycles */
 int    last = NCYCMAX;             /* last desired cycle or NCYCMAX for all cycles */
 int    found;                      /* returned by cln to indicate if anomalous pts were found, 0 for none, 1 for anomalies */
 int    position;                   /* used in anomalous pts listing for span location*/

 char       start_string    [80];   /* start_time string */
 char       end_string [80];        /* end_time string */

 float maxc = 0.0;                  /* temp maxima */
 float  az_angle = 0;               /* temp azimuth angle variable */
 float last_az;                     /* Previous azimuth angle value */
 float  mydata[300];                /* data storage array for one sampling of data*/

 int    totalpts;                   /* for error file percentage changed */
 int    changed80cn,
        changed63cn,
```

```
        changed47cn,
        changed30cn;                    /* for error file percentage changed */
int     changed80ct,
        changed63ct,
        changed47ct,
        changed30ct;
int     changed;


float ct_sum[span];
float ct[span][NPTMAX];                 /* to keep one cycle's data at a time */
float ct_avg[span];
float ctsd_sum[span];
float ctsd_sqr[span];
float ctsd_tot[span];
float ct_dev[span];
float ct_max[span];

float cn_sum[span];
float cn[span][NPTMAX];                 /* to keep one cycle's data at a time */
float cn_avg[span];
float cnsd_sum[span];
float cnsd_sqr[span];
float cnsd_tot[span];
float cn_dev[span];
float cn_max[span];

long    loc;                            /* location of record information */

/* BEGIN PROGRAM */

        printf ("\n*****************************************\n");
        printf ( "**  Cycle Averaging-Maxima Program  **\n");
        printf ( "*****************************************\n");

        arc_file = -1;
        /*while*/((
        arc_file = read_in_chanls(chan_des,&num_chnls,
                start_string,end_string,&num_scans)) == -1);
        if (arc_file == -1) exit(-1);

/* get desired cycle numbers */

        (void)printf("You may select cycles or have all cycles included.\n");
        (void)printf("Please enter the first desired cycle [0 for all]:  ");
        (void)scanf("%d",&first);
        if (first > 0)
          {
           (void)printf("Please enter the last cycle:  ");
           (void)scanf("%d", &last);
          }

/* open up summary file */

        if (strrchr(hdr_file,'/') != NULL)
                strcpy(in_name,strrchr(hdr_file,'/')+1);
        else strcpy(in_name,hdr_file);
        /* remove filename extension */
        in_name[strlen(in_name)-4] = '\0';
        /* output filenames */
        strcpy (cnsum_name,in_name);
        strcpy (cnout_name, in_name);

                                        /* these are user summary files */
        strcpy (ctsum_name, cnsum_name);
        strcat (ctsum_name, "ct.sum");
        strcpy (maxsum_name, cnsum_name);
```

```
strcat (maxsum_name, "max.sum");
strcat (cnsum_name, "cn.sum");

strcpy (err_name, in_name);      /* error listing for anomalous pts*/
strcat (err_name, ".err");

                                 /*  these are the numeric data files */
strcpy (ctout_name, cnout_name);
strcat (ctout_name, "ct.out");
strcpy (maxout_name, cnout_name);
strcat (maxout_name, "max.out");
strcat (cnout_name, "cn.out");

fperr = fopen(err_name, "w");

fpcnsum = fopen(cnsum_name,"w");
fpctsum = fopen(ctsum_name,"w");
fpmaxsum = fopen(maxsum_name,"w");

if (fperr == NULL)
{
        printf("Could not open %s for writing.\n",err_name);
        exit (-1);
}

if(fpcnsum == NULL)                    /* if file exists */
{
        printf("Could not open %s for writing.\n",cnsum_name);
        exit(-1);
}
if(fpctsum == NULL)            /* if file exists */
{
        printf("Could not open %s for writing.\n",ctsum_name);
        exit(-1);
}
if(fpmaxsum == NULL)           /* if file exists */
{
        printf("Could not open %s for writing.\n",maxsum_name);
        exit(-1);
}
/* print user file headers */

(void)fprintf(fperr,"Anomalous points listing from %s\n\n",in_name);
(void)fprintf(fpcnsum,"Categorization of Cn Data %s\n\n",in_name);
(void)fprintf(fpctsum,"Categorization of Ct Data %s\n\n",in_name);
(void)fprintf(fpmaxsum,"Categorization of Cn & Ct Maxima %s\n\n",in_name);

fpcnout = fopen(cnout_name,"w");
fpctout = fopen(ctout_name,"w");
fpmaxout = fopen(maxout_name,"w");

if(fpcnout == NULL) /* if file exists */
{
        printf("Could not open %s for writing.\n",cnout_name);
        exit (-1);
}
if(fpctout == NULL)  /* if file exists */
{
        printf("Could not open %s for writing.\n",ctout_name);
        exit (-1);
}
if(fpmaxout == NULL)           /* if file exists */
{
        printf("Could not open %s for writing.\n",maxout_name);
        exit (-1);
}
```

```
        printf("\nReading data file ... \n");

        /* set up first read */

        count = 0; /*count indicates what record in file to get*/
        nread = REC_IN_SIZE;        /* bytes of data read */
        loc = count * REC_IN_SIZE;
        lseek (arc_file,loc,0);

        start = FALSE;      /* logical for cycle indicator */
        ncycle = 0;

        changed = 0;
        totalpts = 0;
        changed80cn = changed63cn = changed47cn = changed30cn = 0;
        changed80ct = changed63ct = changed47ct = changed30ct = 0;

/* do the following for all records in the file, binary read of data */
/* reads each channel of data from one time instance into mydata[]  */

        while((nread = read(arc_file,mydata,REC_IN_SIZE) > 0 )&&((ncycle-last) <= 0))
        {       /* begin while */
                last_az = az_angle;
                az_angle = mydata[56];

/*  this series of if statements tests azimuthal angle to see what cycle
it is in, whether to start or finish a cycle, etc. */
        /* check to see if we should start the cycle count */

                if (!start)                     /* FALSE = 0 */
                {       /* begin if */
                  start = TRUE;     /* start cycle */
                  for (i = 0; i < span; i++)
                  {
                        cn_sum[i] = 0.0;
                        cn_avg[i] = 0.0;
                        cn_dev[i] = 0.0;
                        cnsd_tot[i] = 0.0;
                        cn_max[i] = maxc;
                        npts = 0;

                        ct_sum[i] = 0.0;
                        ct_avg[i] = 0.0;
                        ct_dev[i] = 0.0;
                        ctsd_tot[i] = 0.0;
                        ct_max[i] = maxc;
                  } /*end for i*/

              ncycle++;
                   pass[ncycle] = FALSE;        /* set init value false */
                }       /* endif */

        /* check to see if we should end the cycle count
        ** end the cycle if the azimuth angle has passed 360 degrees
        ** or if there are too many data points in the cycle, indicating
        ** an error in the azimuth angle
        */

                if((az_angle < last_az) || (npts >= NPTMAX))
                {
                  if (npts >= NPTMAX)
                        (void)fprintf(fperr,"\n**Azimuth error in cycles %d & %d**\n\n",ncycle, ncycle + 1);
                  start = FALSE;
                  totalpts += npts;


                  /* for each span location */
```

```
for (j = 0; j < span; j++)
{
        /* clean the data before averaging
        ** if anomalies in are in the first array, both
        ** arrays are changed
        ** if cln returns 1, anomalies were found
        */
        found = 0;
        cln(cn[j], ct[j], npts, &found, &changed, cnmagnitude);
        if (found)
        {
                switch(j)
                {
                        case 0:  position = 30;
                                changed30cn += changed;
                                changed30ct += changed;
                                break;
                        case 1:  position = 47;
                                changed47cn += changed;
                                changed47ct += changed;
                                break;
                        case 2:  position = 63;
                                changed63cn += changed;
                                changed63ct += changed;
                                break;
                        case 3:  position = 80;
                                changed80cn += changed;
                                changed80ct += changed;
                                break;
                        default: printf ("Error in switch and span\n"); exit (-1); break;
                }
                fprintf(fperr,"\t\tin cyc %3d, at %2d%% span\n\n", ncycle, position);
        } /* end if */

        /* get max values and
        ** add points for mean
        */
        for (i = 1; i <= npts; i++)
        {
        if (cn_max[j] < cn[j][i])
                cn_max[j] = cn[j][i];
        if (ct_max[j] < ct[j][i])
                ct_max[j] = ct[j][i];
        cn_sum[j] += cn[j][i];
        ct_sum[j] += ct[j][i];
        } /* end for i */

        ct_avg[j] = (ct_sum[j]/(float)npts);
        cn_avg[j] = (cn_sum[j]/(float)npts);

        /* calculate standard deviation */
        /* for all points in a cycle */
        for (i = 1;i<=npts;i++)
        {
                ctsd_sum[j] = ct[j][i]-ct_avg[j];
                ctsd_sqr[j] = ctsd_sum[j]*ctsd_sum[j];
                ctsd_tot[j] = ctsd_tot[j] + ctsd_sqr[j];

                cnsd_sum[j] = cn[j][i]-cn_avg[j];
                cnsd_sqr[j] = cnsd_sum[j] * cnsd_sum[j];
                cnsd_tot[j] = cnsd_tot[j] + cnsd_sqr[j];
        }       /* end for i*/

        cn_dev[j] = sqrt(cnsd_tot[j]/(float)npts);
        ct_dev[j] = sqrt(ctsd_tot[j]/(float)npts);
} /*end for */
```

```
        /*  print cycle data */
                if (ncycle >= first)
                {
                fprintf(fpcnsum,"cyc:%3d 30%%:%6.3f+-%6.3f 47%%:%6.3f+-%6.3f 63%%:%6.3f+-%6.3f80%%:
                        %6.3f+-%6.3f\n", ncycle,cn_avg[0],cn_dev[0],cn_avg[1],cn_dev[1],cn_avg[2],cn_dev[2],
                        cn_avg[3],cn_dev[3]);  /* user file */

                fprintf(fpctsum,"cyc:%3d 30%%:%6.3f+-%6.3f 47%%:%6.3f+-%6.3f 63%%:%6.3f+-%6.3f 80%%:
                        %6.3f+-%6.3f\n", ncycle,ct_avg[0],ct_dev[0],ct_avg[1],ct_dev[1],ct_avg[2],ct_dev[2],
                        ct_avg[3],ct_dev[3]);  /* user file */

                fprintf(fpmaxsum,"cyc:%3d cn30%%:%6.3f cn47%%:%6.3f cn63%%:%6.3f cn80%%:%6.3f ct30%%:
                        %6.3f ct47%%:%6.3f ct63%%:%6.3f ct80%%:%6.3f \n", ncycle,cn_max[0],cn_max[1],
                        cn_max[2],cn_max[3],ct_max[0],ct_max[1],ct_max[2],ct_max[3]);  /* user file */

                /* don't print 1st cycle since it isn't complete*/
                if (ncycle != 1)
                {
                fprintf(fpctout,"%3d %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f\n", ncycle,ct_avg[0],
                        ct_dev[0],ct_avg[1],ct_dev[1],ct_avg[2],ct_dev[2],ct_avg[3],ct_dev[3]);  /* data file */
                fprintf(fpcnout,"%3d %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f\n", ncycle,cn_avg[0],
                        cn_dev[0],cn_avg[1],cn_dev[1],cn_avg[2],cn_dev[2],cn_avg[3],cn_dev[3]);  /* data file */
                fprintf(fpmaxout,"%3d %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f %6.3f \n", ncycle,cn_max[0],
                        cn_max[1],cn_max[2],cn_max[3],ct_max[0],ct_max[1],ct_max[2],ct_max[3]);  /* data file */
                } /*end if (ncycle) */
                } /*end if (print cycle data)  */
        }       /* endif (a complete cycle) */

        if (start)                                      /* a new cycle */
                                                        /* add data to arrays */
        {       /* begin if */
                npts++;
                angle[npts] = az_angle;
                for (j=0; j < span; j++)
                {
                        /* cn[0][npts] for 30% span
                           cn[1][npts] for 47% span
                           cn[2][npts] for 63% span
                           cn[3][npts] for 80% span
                        */
                        cn[j][npts] = mydata[cn30-j];
                        ct[j][npts] = mydata[ct30-j];
                } /* end for j */
        }       /* endif */
        count++;
        loc = count * REC_IN_SIZE;     /* set up next read */
        lseek(arc_file,loc,0);
}       /* end while */

/* print summary of changes made by cln */

        printf("************************************************\n");
        printf("Summary of changes made to data by cln procedure\n");
        printf("************************************************\n");
        printf("\n\n\n\t%d total points in %s\n", totalpts, in_name);
        printf("\nCn was changed at %d * standard deviation\n", cnmagnitude);
        printf("\tCn at 30%% span:\t%4d pts changed;\t%f%% of total\n",changed30cn, 100*(float)changed30cn/totalpts);
        printf("\tCn at 47%% span:\t%4d pts changed;\t%f%% of total\n",changed47cn, 100*(float)changed47cn/totalpts);
        printf("\tCn at 63%% span:\t%4d pts changed;\t%f%% of total\n",changed63cn, 100*(float)changed63cn/totalpts);
        printf("\tCn at 80%% span:\t%4d pts changed;\t%f%% of total\n",changed80cn, 100*(float)changed80cn/totalpts);
        printf("\n\nCt was changed when corresponding Cn was changed \n", ctmagnitude);
        printf("\n\tCt at 30%% span:\t%4d pts changed;\t%f%% of total\n",changed30ct, 100*(float)changed30ct/totalpts);
        printf("\tCt at 47%% span:\t%4d pts changed;\t%f%% of total\n",changed47ct, 100*(float)changed47ct/totalpts);
        printf("\tCt at 63%% span:\t%4d pts changed;\t%f%% of total\n",changed63ct, 100*(float)changed63ct/totalpts);
        printf("\tCt at 80%% span:\t%4d pts changed;\t%f%% of total\n",changed80ct, 100*(float)changed80ct/totalpts);
        fprintf(fperr,"************************************************\n");
        fprintf(fperr,"Summary of changes made to data by cln procedure\n");
```

```
        fprintf(fperr,"*************************************************\n");
        fprintf(fperr,"\n\n\n\t%d total points in %s\n", totalpts, in_name);
        fprintf(fperr,"\nCn was changed at %d * standard deviation\n", cnmagnitude);
        fprintf(fperr,"\tCn at 30%% span:\t%4d pts changed;\t%f%% of total\n",changed30cn, 100*(float)changed30cn/totalpts);
        fprintf(fperr,"\tCn at 47%% span:\t%4d pts changed;\t%f%% of total\n",changed47cn, 100*(float)changed47cn/totalpts);
        fprintf(fperr,"\tCn at 63%% span:\t%4d pts changed;\t%f%% of total\n",changed63cn, 100*(float)changed63cn/totalpts);
        fprintf(fperr,"\tCn at 80%% span:\t%4d pts changed;\t%f%% of total\n",changed80cn, 100*(float)changed80cn/totalpts);
        fprintf(fperr,"\n\nCt was changed at %4d * standard deviation\n", ctmagnitude);
        fprintf(fperr,"\n\tCt at 30%% span:\t%4d pts changed;\t%f%% of total\n",changed30ct, 100*(float)changed30ct/totalpts);
        fprintf(fperr,"\tCt at 47%% span:\t%4d pts changed;\t%f%% of total\n",changed47ct, 100*(float)changed47ct/totalpts);
        fprintf(fperr,"\tCt at 63%% span:\t%4d pts changed;\t%f%% of total\n",changed63ct, 100*(float)changed63ct/totalpts);
        fprintf(fperr,"\tCt at 80%% span:\t%4d pts changed;\t%f%% of total\n",changed80ct, 100*(float)changed80ct/totalpts);

        fclose(fpctsum);
        fclose(fpcnsum);
        fclose(fpmaxsum);
        fclose(fpctout);
        fclose(fpcnout);
        fclose(fpmaxout);
        fclose(fperr);
        /* beep at user to signal program end */
        (void)printf("\n\7\7");

}       /* end MAIN */

/*********************************cln*************************************/
/* cln removes the points which have changed more than 2 standard deviation from the previous point and replaces it with the average of
** the two preceding points
** NOTE:  the first point in datarray[0] is assumed invalid and is not used; numpts is the number of actual data points
**
** if and only if cln has found anomalies in cn, ct will be changed
*/
void    cln (indepdata, depdata, numpts, status, changed,magnitude)

float   indepdata[NPTMAX];
float   depdata[NPTMAX];
int     numpts;
int     *status;
int     *changed;
int     magnitude;

{
double  indepsubtot = 0.0;
float   indepmean = 0.0;
float   indepsd;
int     i;
float   limit;

*changed = 0;

/* compute the mean */
for (i=1; i<=numpts; i++)
{
        indepsubtot += indepdata[i];
}
indepmean = (float)(indepsubtot/(float)numpts);

/* compute the standard deviation */
indepsubtot = 0.0;
for (i=1; i<=numpts; i++)
{
        indepsubtot += (indepdata[i] - indepmean)* (indepdata[i] - indepmean);
}
indepsd = sqrt((float)(indepsubtot /(float)numpts));

/* compare each data point for anomalies */
/* NOTE: the second point is replaced with mean of the 2 following points
```

```
**      and the first point is not corrected
*/
limit = magnitude * indepsd;
i = 2;
        if (fabs(indepdata[i-1] - indepdata[i]) > limit)
        {
                fprintf(fperr,"change cn %f ", indepdata[i]);
                indepdata[i] = (indepdata[i+1] + indepdata[i+2])/2.0;
                *status = 1;
                fprintf(fperr," to %f\n", indepdata[i]);
                *changed += 1;
                fprintf(fperr,"change ct %f ", depdata[i]);
                depdata[i] = (depdata[i+1] + depdata[i+2])/2.0;
                fprintf(fperr," to %f\n", depdata[i]);
        }

for (i=3; i<=numpts; i++)
        if (fabs(indepdata[i-1] - indepdata[i]) > limit)
        {
                fprintf(fperr,"change cn %f ", indepdata[i]);
                indepdata[i] = (indepdata[i-1] + indepdata[i-2])/2.0;
                *status = 1;
                fprintf(fperr," to %f\n", indepdata[i]);
                *changed += 1;

                fprintf(fperr,"change ct %f ", depdata[i]);
                depdata[i] = (depdata[i-1] + depdata[i-2])/2.0;
                fprintf(fperr," to %f\n", depdata[i]);
        }
} /* end cln */
```

The program AVCNCT.C also needs BEN_IO.C, READ.C, and MAIN.H to run properly. These files can be found in Appendix B.

The following three files are also necessary to properly run AVCNCT.C:

1. directory.dat:

```
/usr/data5
d071021
/home/windslc/tyoung/working/Avcnct
d7121
```

2. makefile:

```
# Makefile for NREL Wind Programs  2/7/92
#
HFILES = main.h
PROG  = avcnct
SRC   = avcnct.c read.c ben_io.c
OBJ   = avcnct.o read.o ben_io.o

CFLAGS = -g

#
#  the default is to make PROG
#
all: ${PROG}

#
#  program depends on the .o files  (included is -lm for math.h)
#
${PROG}: ${OBJ}
        cc -c ${OBJ}
        cc ${OBJ} -o $@ -lm

#
#  the .o files depend on the .c file (implicit) and the .h file
#
${OBJ}:   ${HFILES}

clean:
        rm -f ${OBJ} ${PROG} core

lint:
        lint ${SRC}
```

3. last_bin.dat

```
/usr/data5
d068011.hdr
```

Appendix E

CNCTMAX.F

```
      program cnctmax

************************************************************************************************************
***********
*****   THIS PROGRAM USES A NUMBER OF DIFFERENT DATABASES TO RANK ALL CYCLES IN THE COMBINED *****
*****   EXPERIMENT ARCHIVES BY MEAN CN OR CT FROM HIGHEST TO LOWEST FOR A GIVEN SPAN LOCATION.
      *****
*****   THE USER CHOOSES THE NUMBER OF VALUES THAT WILL BE WRITTEN TO THE OUTPUT. THE          *****
*****   PROGRAM WRITES THE RANK, TAPE NAME, CYCLE NUMBER, MEAN CN, CN STANDARD DEVIATION,      *****
*****   MAXIMUM CN IN THAT CYCLE, MEAN ANGLE OF ATTACK FOR THE CYCLE, VELOCITY MEAN AND        *****
*****   STANDARD DEVIATION, AND YAW MEAN AND STANDARD DEVIATION. A NUMBER OF FILES ARE         *****
*****   REQUIRED FOR PROGRAM EXECUTION. THE FILES cnval.nam AND ctval.nam SHOULD CONTAIN THE NAMES
      *****
*****   OF THE FILES IN WHICH EACH CYCLE IS CATEGORIZED BY THE MEAN AND STANDARD DEVIATION CN *****
*****   AND CT, RESPECTIVELY, THAT WERE CREATED BY avcnct.c. THE DATABASE FILES THAT ARE REQUIRED
      *****
*****   FOR EACH ARCHIVE TAPE ARE ???????.out, ???????cn.out, ???????ct.out, and ???????max.out, WHERE THE  *****
*****   QUESTION MARKS ARE REPLACED BY THE TAPE NAME. FOR INSTANCE, FOR TAPE d068011, THE
      *****
*****   REQUIRED DATA FILES WOULD BE d068011.out, d068011cn.out, d068011ct.out, and d068011max.out. FOR PHASE
      *****
*****   II OF THE COMBINED EXPERIMENT THERE WERE 59 DATA TAPES, SO THERE SHOULD BE A TOTAL OF 59 *****
*****   OF EACH FILE TYPE.                                                                       *****
*****                                                                                            *****
*****   WRITTEN BY DEREK SHIPLEY                                                                 *****
************************************************************************************************************
***********

*****************************************************************************************************
*****   VARIABLE AND CONSTANT DEFINITIONS                                    *****
*****    cnctfil        NAME OF THE MEAN CN OR CT DATABASE FILE                      *****
*****    coeff          INDICATES IF CN OR CT IS TO BE EXAMINED              *****
*****    cyc()          ARRAY HOLDING CYCLE NUMBERS                          *****
*****    dat()          DATA FOR EACH CYCLE FROM MAX CN/CT DATABASE          *****
*****    data()         DATA FOR EACH CYCLE FROM MEAN CN OR CT DATABASE      *****
*****    dots           COUNTER FOR CREATING A "RUNNING .." MESSAGE          *****
*****    intdots        INTERVAL BETWEEN DOTS IN RUNNING MESSAGE             *****
*****    MAXCYC         MAXIMUM NUMBER OF CYCLES IN DATA ARCHIVES            *****
*****    maxfil         NAME OF THE PEAK CN AND CT DATABASE FILE             *****
*****    mean()         ARRAY HOLDING MEAN CN OR CT VALUES                   *****
*****    ncmean         COLUMN IN DATABASE HOLDING MEAN CN OR CT VALUES      *****
*****    ncpeak         COLUMN IN DATABASE HOLDING MAX CN OR CT VALUES       *****
*****    ncoeff         INTEGER INDICATING WHETHER CN OR CT WAS CHOSEN               *****
*****    nval           NUMBER OF VALUES TO BE WRITTEN TO THE OUTPUT         *****
*****    outfil         NAME OF THE OUTPUT FILE                              *****
*****    over           SHOULD EXISTING FILE BE OVERWRITTEN BY OUTPUT        *****
*****    peak           PEAK CN OR CT VALUE FOR EACH CYCLE                   *****
*****    sd()           ARRAY HOLDING CN OR CT STANDARD DEVIATIONS           *****
*****    span           SPAN LOCATION TO BE EXAMINED                         *****
*****    tape()         ARRAY HOLDING ARCHIVE DATA TAPE NAMES                        *****
*****    vyfil          NAME OF VELOCITY AND YAW DATABASE FILE                       *****
*****************************************************************************************************

*****   DECLARE VARIABLES AND CONSTANTS  *****
      parameter (MAXCYC=22000)
      character*30 cnctfil,maxfil,vyfil,outfil
      character coeff*1,tape(MAXCYC)*7,over*1
      real data(9),dat(9),mean(MAXCYC),peak,sd(MAXCYC)
      integer cyc(MAXCYC),span

*****   DISPLAY A CHEERFUL GREETING MESSAGE ON THE SCREEN  *****
      print*
      print*,'*************************************************************'
      print*,'*****  Welcome to CNCTMAX - the maximum coefficient locator  *****'
      print*,'*************************************************************'

*****   PROMPT THE USER FOR WHETHER CN OR CT IS TO BE EXAMINED  *****
```

```
50      print*
        write (6,1)
1       format ('Do you wish to examine Cn"s <N> or Ct"s <T>: ',$)
        read*,coeff
        if (coeff .eq. 'n' .or. coeff .eq. 'N') then
         ncoeff = 1
        elseif (coeff .eq. 't' .or. coeff .eq. 'T') then
         ncoeff = 5
        else
         print*,'Enter a <N> or a <T>.  Please try again.'
         goto 50
        endif

*****   OPEN THE PROPER FILE CONTAINING LISTS OF DATABASE FILES          *****
*****   DEPENDING UPON WHETHER CN OR CT IS TO BE EXAMINED        *****
        if (coeff .eq. 'n' .or. coeff .eq. 'N') then
         open (unit=11,file='cnval.nam',status='old')
        else
         open (unit=11,file='ctval.nam',status='old')
        endif

*****   PROMPT THE USER FOR THE SPAN LOCATION TO BE EXAMINED AND         *****
*****   SET THE CORRECT COLUMNS TO SEARCH DEPENDING UPON THE ANSWER              *****
60      print*
        write (6,2)
2       format ('Enter the span location <30>, <47>, <63>, or <80>: ',$)
        read*,span
        if (span .eq. 30) then
         ncpeak = ncoeff+1
         ncmean = 2
        elseif (span .eq. 47) then
         ncpeak = ncoeff+2
         ncmean = 4
        elseif (span .eq. 63) then
         ncpeak = ncoeff+3
         ncmean = 6
        elseif (span .eq. 80) then
         ncpeak = ncoeff+4
         ncmean = 8
        else
         print*,'Invalid span location.  Please try again.'
         goto 60
        endif

*****   PROMPT THE USER FOR THE NUMBER OF VALUES TO BE WRITTEN TO THE OUTPUT FILE  *****
65      print*
        write (6,3)
3       format ('Enter the number of values to be written to a file: ',$)
        read*,nval
        if (nval .le. 0) then
         print*,'Number of values must be a positive.  Please try again.'
         goto 65
        endif

*****   PROMPT THE USER FOR THE NAME OF THE OUTPUT FILE AND OPEN IT  *****
75      print*
        write (6,4)
4       format ('Enter the name of the output file: ',$)
        read*,outfil
        open (unit=13,file=outfil,iostat=inerr,status='new')
        if (inerr .ne. 0) then
         write(6,5)
5        format('File already exists.  Overwrite? (y/n): ',$)
         read*,over
         if (over .eq. 'y' .or. over .eq. 'Y') then
          open (unit=13,file=outfil,status='unknown')
         else
```

```
          goto 75
         endif
        endif

*****   WRITE HEADER INFO TO OUTPUT FILE IF CN IS BEING EXAMINED  *****
        if (coeff .eq. 'n' .or. coeff .eq. 'N') then
         if (nval .lt. 10) then
          write (13,900)nval,'Cn',span
         elseif (nval .lt. 100) then
          write (13,901)nval,'Cn',span
         else
          write (13,902)nval,'Cn',span
         endif
         write (13,*)
         write (13,905)
*****   WRITE HEADER INFO TO OUTPUT FILE IF CT IS BEING EXAMINED  *****
        else
         if (nval .lt. 10) then
          write (13,900)nval,'Ct',span
         elseif (nval .lt. 100) then
          write (13,901)nval,'Ct',span
         else
          write (13,902)nval,'Ct',span
         endif
         write (13,*)
         write (13,906)
        endif
        write (13,925)

*****   DISPLAY STATUS MESSAGE TO SCREEN  *****
        print*
        write (6,6)
6       format ('Reading Data ',$)
        call flush(6)
        n = 1
        dots = 1
*****   READ MEAN CN OR CT DATABASE FILE NAME FROM LIST AND OPEN FILE  *****
100     read (11,*,end=300)cnctfil
        open (unit=12,file=cnctfil,status='old')
*****   ADD DOTS TO STATUS MESSAGE TO INDICATE PROGRAM OPERATION  *****
        if (int(dots/2.) .eq. dots/2.) then
         write(6,7)
7        format('.',$)
         call flush(6)
        endif
        dots = dots+1
*****   READ DATA FROM A DATABASE FILE AND RENAME DESIRED VARIABLES  *****
200     read (12,*,end=400) (data(j),j=1,9)
        if (data(1) .gt. 1) then
         tape(n) = cnctfil(1:7)
         cyc(n) = data(1)
         mean(n) = data(ncmean)
         sd(n) = data(ncmean+1)
         n = n+1
        endif
        goto 200
400     continue
        goto 100
300     continue
        nmax = n-1

*****   CALL SUBROUTINE TO SORT VALUES IN DESCENDING ORDER  *****
        call sort (nmax,mean,tape,cyc,sd)
        print*

*****   BEGIN LOOP TO FIND THE PEAK CN OR CT, VELOCITY MEAN AND STANDARD DEVAIATION,      *****
*****   AND YAW MEAN AND STANDARD DEVIATION FOR EACH CYCLE TO BE WRITTEN TO THE OUTPUT *****
```

```
      write(6,9)
*****  DISPLAY STATUS MESSAGE TO THE SCREEN  *****
9     format ('Getting Velocity and Yaw Values ',$)
      call flush(6)
      intdots = nint(nval/20.)
      dots = 1
*****  BEGIN LOOP TO SEARCH FOR DESIRED CYCLES  *****
      do 500,n=nmax,nmax-nval+1,-1
*****  OPEN MAXIMUM CN/CT DATABASE FILE  *****
       maxfil = tape(n)//'max.out'
       open (unit=15,file=maxfil,status='old')
*****  ADD DOTS TO THE STATUS MESSAGE TO INDICATE PROGRAM ACTIVITY  *****
       if (int(dots/intdots) .eq. dots/intdots) then
        write(6,7)
        call flush(6)
       endif
       dots = dots+1
*****  READ DATA UNTIL DESIRED CYCLE IS LOCATED  *****
550    read (15,*) (dat(j),j=1,9)
        if (dat(1) .ne. cyc(n)) then
         goto 550
        else
*****  GET PEAK CN OR CT VALUE FOR THIS CYCLE  *****
         peak = dat(ncpeak)
         close (15)
        endif

*****  OPEN CORRECT VELOCITY AND YAW DATABASE FILE  *****
       vyfil = tape(n)//'.out'
       open (unit=14,file=vyfil,status='old')
*****  READ DATA UNTIL DESIRED CYCLE IS LOCATED  *****
600    read (14,*,end=700) (dat(j),j=1,9)
        if (dat(1) .eq. cyc(n)) then
*****  CALL SUBOUTINE TO CALCULATE MEAN ANGLE OF ATTACK  *****
         call v2asub(dat(2),dat(4),span,aoa)
*****  WRITE APPROPRIATE DATA TO THE OUTPUT FILE  ****
         write (13,930)nmax-n+1,tape(n),cyc(n),mean(n),sd(n),peak,aoa,(dat(k),k=2,5)
         close (14)
         goto 700
        endif
        goto 600
700    continue
500    continue

*****  DISPLAY STATUS MESSAGE REMINDING USER OF THE OUTPUT FILE NAME  *****
      print*
      print*
      write (6,10),outfil
10    format ('Output is Contained in the File - ',a30)
      print*

*****  FORMAT STATEMENTS FOR OUTPUT FILE  *****
900   format ('The ',i1,' highest mean values over a cycle for ',a2,' at ',i2,'% span are:')
901   format ('The ',i2,' highest mean values over a cycle for ',a2,' at ',i2,'% span are:')
902   format ('The ',i3,' highest mean values over a cycle for ',a2,' at ',i2,'% span are:')
905   format ('Rank',t8,'Tape',t15,'Cyc',t20,'Mean Cn',t28,'Cn sd',t35,'Max Cn',t43,'AOA',t49,'Mean Vel',t59,'Vsd',t65,'Mean
Yaw',t76,'Ysd')
906   format ('Rank',t8,'Tape',t15,'Cyc',t20,'Mean Ct',t28,'Ct sd',t35,'Max Ct',t43,'AOA',t49,'Mean Vel',t59,'Vsd',t65,'Mean Yaw',t76,'Ysd')
925   format ('-------------------------------------------------------------------')
930   format (i3,t6,a7,t15,i3,t20,f6.3,t28,f5.3,t34,f6.2,t42,f6.3,t49,f7.3,t56,f7.3,t65,f7.3,t73,f7.3)

      stop
      end

*************************************************************************
*************************************************************************
```

```
      subroutine sort(n,ra,rb,rc,rd)

**********************************************************************************************************************
***********
*****     THIS SUBROUTINE SORTS THE ENTRIES IN INCREASING ORDER BY THE MEAN CN OR CT.  THE TAPE      *****
*****     NAME, CYCLE NUMBER, AND CN/CT STANDARD DEVIATION ARE ALSO SORTED AND THE ORIGINAL          *****
*****     DATA IS OVERWRITTEN BY THE SORTED VERSION.  SORTING IS ACCOMPLISHED USING A HEAPSORT.      *****
*****                                                                                                *****
*****     SORT ROUTINE WRITTEN BY W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY, AND W.T. VETTERLING.    *****
*****     NUMERICAL RECIPES: THE ART OF SCIENTIFIC COMPUTING.  CAMBRIDGE UNIVERSITY PRESS,
          *****
*****     CAMBRIDGE, 1989.  MODIFIED SLIGHTLY BY DEREK SHIPLEY.                                      *****
**********************************************************************************************************************
***********

*****     DECLARE VARIABLES  *****
      dimension ra(n),rd(n)
      character rb(n)*7,rrb*7
      integer,rc(n),rrc

*****     DISPLAY STATUS MESSAGE INDICATING THAT VALUES ARE BEING SORTED  *****
      print*
      write(6,1)
1     format('Sorting Values ',$)
      call flush(6)
      intdots = nint(n/20.)
      dots = 1

      l=n/2+1
      ir=n
10    continue
*****  ADD DOTS TO THE STATUS MESSAGE INDICATING PROGRAM ACTIVITY  *****
      if (int(dots/intdots) .eq. dots/intdots) then
       write(6,2)
2      format('.',$)
       call flush(6)
      endif
      dots = dots+1
*****     BEGIN LOOP TO SORT DATA IN PLACE (HEAPSORT METHOD)  *****
      if (l .gt. 1) then
       l=l-1
       rra=ra(l)
       rrb=rb(l)
       rrc=rc(l)
       rrd=rd(l)
      else
       rra=ra(ir)
       rrb=rb(ir)
       rrc=rc(ir)
       rrd=rd(ir)
       ra(ir)=ra(1)
       rb(ir)=rb(1)
       rc(ir)=rc(1)
       rd(ir)=rd(1)
       ir=ir-1
       if (ir .eq. 1) then
        ra(1)=rra
        rb(1)=rrb
        rc(1)=rrc
        rd(1)=rrd
        return
       endif
      endif
      i=l
      j=l+l
20    if (j .le. ir) then
       if (j .lt. ir) then
```

```
          if (ra(j) .lt. ra(j+1)) j=j+1
         endif
         if (rra .lt. ra(j)) then
          ra(i)=ra(j)
          rb(i)=rb(j)
          rc(i)=rc(j)
          rd(i)=rd(j)
          i=j
          j=j+j
         else
          j=ir+1
         endif
         goto 20
        endif
        ra(i)=rra
        rb(i)=rrb
        rc(i)=rrc
        rd(i)=rrd
        goto 10

        end

*************************************************************************
*************************************************************************

        subroutine v2asub(vinf,gamma,span,alpha)

*****************************************************************************************************************
***********
*****   THIS SUBROUTINE COMPUTES A VALUE FOR THE LOCAL ANGLE OF ATTACK BASED UPON THE
        *****
*****   GEOMETRY OF THE TURBINE RELATIVE TO THE INFLOW.          THE MODEL INCOPRORATES INDUCED
        *****
*****   VELOCITIES PREDICTED FOR THE COMBINED EXPERIMENT ROTOR BY THE PROP CODE, A SKEWED     *****
*****   WAKE EFECT, AND A TOWER SHADOW MODEL.  THE MEAN ANGLE OF ATTACK OVER A CYCLE IS       *****
*****   COMPUTED BASED UPON THE MEAN VELOCITY AND YAW CONDITIONS FOR THE CYCLE.               *****
*****                                                                                         *****
*****   WRITTEN BY DEREK SHIPLEY (AS MODIFIED FROM STEVE HUYER)
        *****
*****************************************************************************************************************
***********

*************************************************************************************************
*****   VARIABLE AND CONSTANT DEFINITIONS:                                    *****
*****    alpha          MEAN ANGLE OF ATTACK FOR THE CYCLE                    *****
*****    aoa()          ANGLE OF ATTACK AT 360 AZIMUTHAL ANGLES               *****
*****    atot           SUM OF ANGLE OF ATTACK AT ALL AZIMUTHAL ANGLES              *****
*****    az             AZIMUTH ANGLE (DEG)                                   *****
*****    a??            AXIAL INDUCED VEL. WITH SKEWED WAKE                   *****
*****    ao??           AXIAL INDUCED VELOCITY                                *****
*****    azrad          AZIMUTH ANGLE (RAD)                                   *****
*****    betarad        BLADE PITCH ANGLE (RAD)                               *****
*****    dia            TOWER DIAMETER (M)                                    *****
*****    gamma          YAW (DEG)                                             *****
*****    gamrad         YAW (RAD)                                             *****
*****    omega          ROTATIONAL FREQUENCY                                  *****
*****    r              RADIUS OF TURBINE ROTOR                                     *****
*****    sw??           SKEWED WAKE EFFECT                                    *****
*****    tmn??          AZIMUTH ANGLE BLADE ENTERS TOWER SHADOW               *****
*****    tmx??          AZIMUTH ANGLE BLADE LEAVES TOWER SHADOW               *****
*****    towdef         MAX TOWER SHADOW VELOCITY DEFICIT                     *****
*****    tsw            TOWER SHADOW WIDTH                                    *****
*****    vinf           INFLOW VELOCITY                                       *****
*****    v??c           VELOCITY CROSSFLOW COMPONENT                          *****
*****    v??n           VELOCITY COMPONENT NORMAL TO ROTOR                    *****
*****    v??t           VELOCITY COMPONENT TANGENT TO BLADE                   *****
*****    v??tow         INST. TOWER SHADOW VELOCITY DEFICIT                   *****
```

```
*****************************************************************************************************

*****   VARIABLE DECLARATIONS  *****
      real aoa(360)
      integer span

*****   SET CONSTANTS  *****
      pi = 4.0*atan(1.0)
      dia = 0.406
      r = 5.05
      omega = 2.0*pi*1.2
      betarad = 12.0*pi/180.
      towdef = 0.30

*****   CONVERT YAW MEASUREMENT FROM DEGREES TO RADIANS  *****
      gamrad = gamma*pi/180.

*************************************************************************
*****   set up parameters to calculate tower shadow velocity defecit  *****
*************************************************************************

*****   DEFINE CONSTANTS  *****
      tsw = (2.5*dia)/cos(gamrad)
      x = 0.5*(tsw + 0.457)
      twd = 0.9*tan(gamrad)

*****             30% SPAN          *****
      th30 = asin(x/(0.3*r))
      thd30 = asin(twd/(0.3*r))
      tmn30 = pi-th30-thd30
      tmx30 = pi+th30-thd30

*****             47% SPAN          *****
      th47 = asin(x/(0.466*r))
      thd47 = asin(twd/(0.466*r))
      tmn47 = pi-th47-thd47
      tmx47 = pi+th47-thd47

*****             63% SPAN          *****
      th63 = asin(x/(0.633*r))
      thd63 = asin(twd/(0.633*r))
      tmn63 = pi-th63-thd63
      tmx63 = pi+th63-thd63

*****             80% SPAN          *****
      th80 = asin(x/(0.82*r))
      thd80 = asin(twd/(0.82*r))
      tmn80 = pi-th80-thd80
      tmx80 = pi+th80-thd80

      do 100 i=1,360
        azrad = i*pi/180.

        if (span .eq. 30) then
*****************************************************************
*****   CALCULATE ANGLE OF ATTACK AT 30% SPAN  *****
*****************************************************************

*****   CALCULATE 30% TOWER SHADOW DEFICIT (ZERO IF NOT WITHIN SHADOW)  *****
          if (azrad.ge.tmn30 .and. azrad.le.tmx30) then
            v30tow = .5*towdef*vinf*(1.-cos(2.*pi*(azrad-tmn30)/(2.*th30)))
          else
            v30tow = 0.
          endif
          vtot30 = vinf-v30tow

*****   INCORPORATE PROP INDUCED VELOCITY AT 30% SPAN  *****
```

```
          ao30 = 1/(.000360046*vtot30**4-.0206024*vtot30**3+.415013*vtot30**2-1.32587*vtot30+3.16774)

*****     CALCULATE SKEWED WAKE EFFECT AT 30% SPAN *****
          sw30 = (1+15*pi/32*sqrt((1-cos(gamrad))/(1+cos(gamrad)))*.30*sin(azrad))
          a30 = ao30*sw30

*****     CALCULATE 30% SPAN VELOCITY COMPONENTS AND ANGLE OF ATTACK *****
          v30n = vtot30*(1-a30)*cos(gamrad)
          v30c = -vtot30*sin(gamrad)
          v30t = (0.30*r*omega)+v30c*cos(azrad)
          aoa(i) = (atan(v30n/v30t)-betarad)*180.0/pi

         elseif (span .eq. 47) then
*****************************************************************
*****  CALCULATE ANGLE OF ATTACK AT 47% SPAN  *****
*****************************************************************

*****     CALCULATE 47% TOWER SHADOW DEFICIT (ZERO IF NOT WITHIN SHADOW) *****
          if (azrad.ge.tmn47 .and. azrad.le.tmx47) then
           v47tow = .5*towdef*vinf*(1.-cos(2.*pi*(azrad-tmn47)/(2.*th47)))
          else
           v47tow = 0.0
          endif
          vtot47 = vinf-v47tow

*****     INCORPORATE PROP INDUCED VELOCITY AT 47% SPAN *****
          ao47 = 1/(.000653402*vtot47**4-.0426646*vtot47**3+1.02377*vtot47**2-8.33547*vtot47+28.3554)

*****     CALCULATE SKEWED WAKE EFFECT AT 47% SPAN *****
          sw47 = (1+15*pi/32*sqrt((1-cos(gamrad))/(1+cos(gamrad)))*.466*sin(azrad))
          a47 = ao47*sw47

*****     CALCULATE 47% SPAN VELOCITY COMPONENTS AND ANGLE OF ATTACK *****
          v47n = vtot47*(1-a47)*cos(gamrad)
          v47c = -vtot47*sin(gamrad)
          v47t = (0.466*r*omega)+v47c*cos(azrad)
          aoa(i) = (atan(v47n/v47t)-betarad)*180.0/pi

         elseif (span .eq. 63) then
*****************************************************************
*****  CALCULATE ANGLE OF ATTACK AT 63% SPAN  *****
*****************************************************************

*****     CALCULATE 63% TOWER SHADOW DEFICIT (ZERO IF NOT WITHIN SHADOW) *****
          if (azrad.ge.tmn63 .and. azrad.le.tmx63) then
           v63tow = .5*towdef*vinf*(1.-cos(2.*pi*(azrad-tmn63)/(2.*th63)))
          else
           v63tow = 0.0
          endif
          vtot63 = vinf-v63tow

*****     INCORPORATE PROP INDUCED VELOCITY AT 63% SPAN *****
          ao63 = 1/(.000857403*vtot63**4-.0580386*vtot63**3+1.45897*vtot63**2-14.2769*vtot63+55.6623)

*****     CALCULATE SKEWED WAKE EFFECT AT 63% SPAN *****
          sw63 = (1+15*pi/32*sqrt((1-cos(gamrad))/(1+cos(gamrad)))*.633*sin(azrad))
          a63 = ao63*sw63

*****     CALCULATE 63% SPAN VELOCITY COMPONENTS AND ANGLE OF ATTACK *****
          v63n = vtot63*(1-a63)*cos(gamrad)
          v63c = -vtot63*sin(gamrad)
          v63t = (0.633*r*omega)+v63c*cos(azrad)
          aoa(i) = (atan(v63n/v63t)-betarad)*180./pi

         elseif (span .eq. 80) then
*****************************************************************
*****  CALCULATE ANGLE OF ATTACK AT 80% SPAN  *****
```

```
*****************************************************************

*****   CALCULATE 80% TOWER SHADOW DEFICIT (ZERO IF NOT WITHIN SHADOW) *****
        if (azrad.ge.tmn80 .and. azrad.le.tmx80) then
         v80tow = .5*towdef*vinf*(1.-cos(2.*pi*(azrad-tmn80)/(2.*th80)))
        else
         v80tow = 0.0
        endif
        vtot80 = vinf-v80tow

*****   INCORPORATE PROP INDUCED VELOCITY AT 80% SPAN *****
        ao80 = 1/(.001142*vtot80**4-.0769829*vtot80**3+1.95194*vtot80**2-20.8757*vtot80+88.372)

*****   CALCULATE SKEWED WAKE EFFECT AT 80% SPAN *****
        sw80 = (1+15*pi/32*sqrt((1-cos(gamrad))/(1+cos(gamrad)))*.80*sin(azrad))
        a80 = ao80*sw80

*****   CALCULATE 80% SPAN VELOCITY COMPONENTS AND ANGLE OF ATTACK *****
        v80n = vtot80*(1-a80)*cos(gamrad)
        v80c = -vtot80*sin(gamrad)
        v80t = (0.82*r*omega)+v80c*cos(azrad)
        aoa(i) = (atan(v80n/v80t)-betarad)*180.0/pi
       endif
100    continue

*****   CALCULATE MEAN ANGLE OF ATTACK OVER THE CYCLE *****
       atot=0.
       do 200,m=1,360
        atot=atot+aoa(m)
200    continue
       alpha=atot/360.

       return
       end
```

Appendix F

DSTALLTAPE.F

```
      program dstalltape

************************************************************************************************************
***********
*****   THIS PROGRAM COMPILES THE MAXIMUM LEADING EDGE SUCTION DATABASE USED BY THE                    *****
*****   PROGRAM dstall.f TO IDENTIFY DYNAMIC STALL EVENTS.  THE MAXIMUM LEADING EDGE SUCTION           *****
*****   DURING EACH CYCLE WILL BE USED AS THE INDICATOR FOR DYNAMIC STALL.  THEREFORE, THIS            *****
*****   PROGRAM FINDS THAT VALUE, AS WELL AS THE INSTANTANEOUS VELOCITY, YAW, AND AZIMUTH              *****
*****   AT WHICH IT OCCURRED.  THE PROGRAM DIRECTLY ANALYZES A COMBINED EXPERIMENT ARCHIVE             *****
*****   DATA TAPE.  THE USER MAY ALSO CHOOSE TO RE-NORMALIZE THE Cp'S BY A DYNAMIC PRESSURE            *****
*****   THAT IS A FUNCTION OF YAW, RATHER THAN THE Q USED ON THE COMBINED EXPERIMENT DATA              *****
*****   TAPES.  THE ORIGINAL Q IS ONLY A FUNCTION OF FREESTREAM VELOCITY AND ROTATIONAL
      *****
*****   VELOCITY.  THE OUTPUT FILE BEGINS WITH A HEADER LISTING THE SELECTED OPTIONS FOLLOWED          *****
*****   BY THE PERTINENT DATA FOR EACH INSTANCE OF DYNAMIC STALL ENCOUNTERED.
      *****
*****                                                                                                  *****
*****   WRITTEN BY DEREK SHIPLEY                                                                       *****
************************************************************************************************************
***********

*****************************************************************************
*****   OUTPUT FILE FORMAT:                                          *****
*****    COLUMN 1 - TAPE NUMBER                                            *****
*****    COLUMN 2 - CYCLE NUMBER                                      *****
*****    COLUMN 3 - SPAN LOCATION OF DYNAMIC STALL EVENT              *****
*****    COLUMN 4 - WIND VELOCITY AT CYCLE MAXIMUM Cp                 *****
*****    COLUMN 5 - YAW AT CYCLE MAXIMUM Cp                           *****
*****    COLUMN 6 - MAXIMUM Cp VALUE OF CYCLE                         *****
*****    COLUMN 7 - AZIMUTH ANGLE AT CYCLE MAXIMUM Cp                 *****
*****************************************************************************

*************************************************************************************************************
*****   VARIABLE DEFINITIONS:                                                                    *****
*****    az              CURRENT AZIMUTH ANGLE (deg)                                            *****
*****    azlast          AZIMUTH ANGLE OF LAST DATA POINT (deg)                                 *****
*****    azmax()         AZIMUTH ANGLE AT MAXIMUM Cp FOR EACH SPAN (deg)                              *****
*****    cp()            ORIGINAL Cp FOR ALL FOUR SPANS                                         *****
*****    cpmax()         MAXIMUM Cp VALUE IN CYCLE FOR ALL SPANS                                      *****
*****    cpnew           RE-NORMALIZED Cp (=cp IF NO RE-NORMALIZATION)                          *****
*****    cwrite()        INDICATES IF DYN. STALL EVENT WAS FOUND                                *****
*****    cyc             CYCLE NUMBER                                                           *****
*****    hh?             THREE MEASURES OF WIND DIRECTION (deg)                                 *****
*****    infil           FILE CONTAINING PRESSURE FILE NAMES                                    *****
*****    infl            INDICATES IF SPECIFIC INFLOW COND. ARE EXAMINED                        *****
*****    norm            INDICATES IF RE-NORMALIZATION IS DESIRED                               *****
*****    outfil          OUTPUT FILE NAME                                                       *****
*****    sp              TEMPORARY VALUE OF SPAN LOCATION FOR RENORM (%)                        *****
*****    span()          SPAN LOCATION FOR EACH SPAN (%)                                        *****
*****    vel             INSTANTANEOUS WIND VELOCITY (m/s)                                      *****
*****    velmax()        WIND VELOCITY AT MAXIMUM Cp FOR EACH SPAN (m/s)                              *****
*****    ya              TURBINE YAW ANGLE (deg)                                                *****
*****    yaw             INSTANTANEOUS YAW (deg)                                                *****
*****    yawmax()                YAW AT MAXIMUM Cp FOR EACH SPAN (deg)                                *****
*************************************************************************************************************

      dimension bin(239)

*****   VARIABLE DECLARATIONS  *****
      character infil*30,outfil*30,norm*1
      real cp(4),cpmax(4),velmax(4),yawmax(4),azmax(4)
      integer cyc,cwrite(4),span(4),sp

*****   SET CONSTANTS  *****
      MAXPTS = 445
      span(1) = 30
      span(2) = 47
```

```
      span(3) = 63
      span(4) = 80

***** GREET USER WITH A CHEERFUL WELCOME MESSAGE *****
      print*
      print*,'*********************************************************************************************'
      print*,'***** WELCOME TO DSTALLTAPE - THE MAX LEADING EDGE SUCTION LOCATOR *****'
      print*,'*********************************************************************************************'

***** PROMPT THE USER FOR THE NAME OF THE INPUT FILE AND OPEN IT *****
100   print*
      write (6,1)
1     format ('Enter the name of the input file (including path): ',$)
      read'(a30)', infil
      open (unit=11,recl=956,form='unformatted',file=infil,iostat=inerr, access='direct',status='old')
      if (inerr .ne. 0) then
       print*, 'File does not exist.  Please try again.'
       goto 100
      endif

      do 125,i=1,30
       if (infil(i:i+1) .eq. 'd0') then
        infil=infil(i:i+6)
        goto 135
       endif
125   continue
135   continue

***** ASK THE USER IF THEY WANT TO RE-NORMALIZE THE Cp'S *****
      print*
      write (6,4)
4     format ('Do you wish to re-normalize the Cp"s with a yaw dependent q? (y/n): ',$)
      read*, norm

***** PROMPT THE USER FOR THE OUTPUT FILE NAME AND OPEN FILE *****
200   print*
      write (6,3)
3     format ('Enter the name of the output file: ',$)
      read'(a30)', outfil
      open (unit=13,file=outfil,status='unknown')

***** CREATE OUTPUT FILE HEADER *****
      write (13,9000)
      if (norm .eq. 'y' .or. norm .eq. 'Y') then
       write (13,9100)
      else
       write (13,9200)
      endif
      write (13,9300)
      write (13,9400)
      write (13,9500)
      write (13,*)
      write (13,9600)
      write (13,9700)
      write (13,*)

      print*
      azlast = 0.
      numpts = 0
      cyc = 1
      nfound = 0
      numrec=1
      print*,'Examining tape ',infil
***** READ DATA FROM INPUT FILE ONE POINT AT A TIME *****
500   read(11,rec=numrec,end=600)(bin(i),i=1,239)
       numpts = numpts+1
       az = bin(57)
```

```
          vel = bin(224)
          ya = bin(61)
          hh1 = bin(48)
          hh2 = bin(50)
          hh3 = bin(53)
          cp(1) = bin(135)
          cp(2) = bin(167)
          cp(3) = bin(20)
          cp(4) = bin(104)
          yaw = ya-(hh1+hh2+hh3)/3.
*****     DETERMINE IF A NEW CYCLE HAS BEEN REACHED.  IF SO, WRITE              *****
*****     THE DATA TO THE OUTPUT FILE FOR ALL FOUR SPAN LOCATIONS*****
          if (az .lt. azlast .or. numpts .gt. MAXPTS) then
            print*,'Cycle',cyc,' contains',numpts,' points'
            do 550, k=1,4
              if (cwrite(k) .eq. 1) then
                if (cyc .ne. 1 ) write(13,9800),infil,cyc,span(k),velmax(k),yawmax(k),cpmax(k),azmax(k)
              endif
              cwrite(k) = 0
              cpmax(k) = 999
550         continue
            cyc = cyc+1
            numpts = 1
          endif
*****     CHECK ALL FOUR SPAN LOCATIONS FOR A DYNAMIC STALL EVENT  *****
          do 700,k=1,4
            c=cp(k)
            cpnew=cp(k)
            sp=span(k)
*****     CALL ROUTINE TO RE-NORMALIZE Cp'S IF DESIRED  *****
            if (norm .eq. 'y' .or. norm .eq. 'Y') call renorm(az,vel,yaw,sp,cpnew)
*****     CHECK TO SEE IF CURRENT VALUE OF Cp IS THE MAXIMUM Cp VALUE ENCOUNTERED IN CURRENT CYCLE
*****
            if (cpnew .lt. cpmax(k)) then
              cpmax(k) = cpnew
              velmax(k) = vel
              yawmax(k) = yaw
              azmax(k) = az
              cwrite(k) = 1
            endif
700         continue
          azlast = az
          numrec=numrec+1
        goto 500
600     continue
        close(11)

*****   FORMAT STATEMENTS FOR THE OUTPUT FILE HEADER AND DATA  *****
9000    format('The maximum leading edge suction in each cycle is',
   2    ' is listed below')
9100    format('The Cp''s were re-normalized using a yaw dependent q')
9200    format('The Cp''s were not re-normalized')
9300    format('The entire cycle was evaluated for max leading edge suction')
9400    format('All wind velocities were evaluated for max leading edge suction')
9500    format('All yaws were evaluated for max leading edge suction')
9600    format('  Tape',t10,'Cycle',t18,'Span',t25,'Velocity',t39,'Yaw',t50,'Max Cp',t63,'Azimuth')
9700    format('-------------------------------------------------------------')
9800    format(a7,t11,i3,t19,i2,t26,f6.3,t37,f7.3,t49,f8.4,t62,f8.4)

        stop
        end

*************************************************************************
*************************************************************************

        subroutine renorm(az,vel,yaw,nspan,cp)
```

```
pi = 4.0*atan(1.0)
azrad = az*pi/180.
gamrad = yaw*pi/180.
r = 5.05
rho = 0.0019
omega = 2.0*pi*1.2
span = nspan/100.

qold = 0.07475*0.5*rho*(vel**2+(omega*span*r)**2)

if (nspan .eq. 30) then
  a = 1/(.000360046*vel**4-.0206024*vel**3+.415013*vel**2-1.32587*vel+3.16774)
elseif (nspan .eq. 47) then
  a = 1/(.000653402*vel**4-.0426646*vel**3+1.02377*vel**2-8.33547*vel+28.3554)
elseif (nspan .eq. 63) then
  a = 1/(.000857403*vel**4-.0580386*vel**3+1.45897*vel**2-14.2769*vel+55.6623)
else
  a = 1/(.001142*vel**4-.0769829*vel**3+1.95194*vel**2-20.8757*vel+88.372)
endif

vn = vel*(1-a)*cos(gamrad)
vc = -vel*sin(gamrad)
vt = (omega*span*r)+vc*cos(azrad)
vs = vc*sin(azrad)
qnew = 0.07475*0.5*rho*(vn**2+vt**2+vs**2)
cp = cp*qold/qnew

return
end
```

Appendix G

DSTALL.F

```
      program dstall

************************************************************************************************************
***********
*****   THIS PROGRAM LOCATES AND PROVIDES STATISTICS ABOUT INSTANCES OF DYNAMIC STALL BASED  *****
*****   UPON THE VALUE OF THE LEADING EDGE SURFACE PRESSURE.  THE PROGRAM DSTALLFILE MUST BE  *****
*****   RUN BEFORE THIS PROGRAM TO GENERATE THE PROPER INPUT FILE.                            *****
*****                                                                                         *****
*****   WRITTEN BY DEREK SHIPLEY                                                              *****
************************************************************************************************************
***********

*****************************************************************************
*****   INPUT FILE FORMAT:                                          *****
*****    COLUMN 1 - TAPE NUMBER                                           *****
*****    COLUMN 2 - CYCLE NUMBER                                     *****
*****    COLUMN 3 - SPAN LOCATION OF DYNAMIC STALL EVENT             *****
*****    COLUMN 4 - WIND VELOCITY AT CYCLE MAXIMUM Cp                *****
*****    COLUMN 5 - YAW AT CYCLE MAXIMUM Cp                          *****
*****    COLUMN 6 - MAXIMUM Cp VALUE OF CYCLE                        *****
*****    COLUMN 7 - AZIMUTH ANGLE AT CYCLE MAXIMUM Cp                *****
*****************************************************************************

*****************************************************************************
*****   VARIABLE DEFINITIONS:                                       *****
*****************************************************************************

*****   VARIABLE DECLARATIONS  *****
      character tape(90000)*7,otape(90000)*7,continue*1
      real data(4,90000),odata(4,90000)
      integer cyc(90000),span(90000),ocyc(90000),ospan(90000),onmax

*****   GREET USER WITH A CHEERFUL WELCOME MESSAGE  *****
      print*
      print*,'*************************************************************************'
      print*,'*****  WELCOME TO DSTALL - THE DYNAMIC STALL LOCATOR  *****'
      print*,'*************************************************************************'

*****   READ IN PEAK Cp VALUES CREATED BY dstallfil  *****
      call readin (tape,cyc,span,data,nmax)

*****   COPY DATA TO AN ARRAY FOR LATER RESOTRATION TO ORIGINAL FORM  *****
      call reset (tape,cyc,span,data,nmax,otape,ocyc,ospan,odata,onmax)

*****   DISPLAY MAIN MENU OF PROCESSING OPTIONS  *****
100   print*
      print*,'*****************************************************'
      print*,'*****          DSTALL MAIN MENU          *****'
      print*,'*****************************************************'
      print*
      print*,' 0 )Quit'
      print*,' 1 )Reset to original data set'
      print*,' 2 )Limit data to above a given Cp'
      print*,' 3 )Limit data to a single span location'
      print*,' 4 )Limit data by inflow velocity'
      print*,' 5 )Limit data by yaw'
      print*,' 6 )Exclude an azimuthal region'
      print*,' 7 )Display the present number of occurrences'
      print*,' 8 )Find cycles with dynamic stall at more than one span'
      print*,' 9 )Sort data by a given parameter'
      print*,' 10 )          Bin data by velocity and yaw'
      print*,' 11 )          Write data to a file'
      print*,' 12 )          Average Cp values'
      print*
      write(6,1)
1     format('Enter the desired option number (0-12): ',$)
      read*,nopt
```

```
*****   BRANCH TO THE PROPER SUBROUTINE BASED UPON CHOSEN OPTION  *****
        if (nopt .eq. 0) then
          goto 900
        elseif (nopt .eq. 1) then
          call reset (otape,ocyc,ospan,odata,onmax,tape,cyc,span,data,nmax)
          print*
          print 9000
          write (12,9000)
        elseif (nopt .eq. 2) then
          call limcp (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 3) then
          call limspan (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 4) then
          call limvel (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 5) then
          call limyaw (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 6) then
          call limaz (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 7) then
          print*
          print 9100, nmax
          write (12,9100), nmax
        elseif (nopt .eq. 8) then
          call multspan(tape,cyc,span,data,nmax)
        elseif (nopt .eq. 9) then
          call sort(tape,cyc,span,data,nmax)
        elseif (nopt .eq. 10) then
          call bindat(data,nmax,odata,onmax)
        elseif (nopt .eq. 11) then
          call writdat (tape,cyc,span,data,nmax)
        elseif (nopt .eq. 12) then
          call avecp (data,nmax)
        else
          print*,'You entered an invalid option.  Please try again.'
          goto 100
        endif
        print*
        write(6,2)
2       format('Hit ENTER to continue: ',$)
        read'(a1)',continue
        goto 100

900     continue

9000    format ('Data reset to original conditions')
9100    format ('The current number of occurrences is ',i5)

        stop
        end

*************************************************************************
*************************************************************************

        subroutine readin(tape,cyc,span,data,nmax)

*************************************************************************************************************************
***********
*****   THIS SUBROUTINE READS DATA FROM AN INPUT FILE CREATED BY dstallfile.  THIS FILE CONTAINS THE *****
*****   TAPE NAME, CYCLE NUMBER, SPAN LOCATION, VELOCITY AT PEAK Cp, YAW AT PEAK Cp, PEAK Cp, AND
        *****
*****   AZIMUTH ANGLE AT PEAK Cp.  IT ALSO PROMPTS THE USER FOR THE NAME OF A LOG FILE TO RECORD *****
*****   THE OPERATIONS PERFORMED ON THE DATA.                                                   *****
*****                                                                                           *****
*****   WRITTEN BY DEREK SHIPLEY                                                                *****
*************************************************************************************************************************
***********
```

```
*****************************************************************************************
*****   VARIABLE DEFINITIONS:                                                       *****
*****    cyc()    CYCLE NUMBER OF EACH ENTRY                                       *****
*****    data(,)  VELOCITY,YAW,PEAK Cp, AND AZIMUTH OF EACH ENTRY                  *****
*****    header   VARIABLE USED TO STRIP OFF DATA FILE HEADER                      *****
*****    infil    INPUT FILE NAME                                                  *****
*****    logfil   LOG FILE NAME                                                    *****
*****    nmax     TOTAL NUMBER OF ENTRIES IN INPUT DATA FILE                       *****
*****    span()   SPAN LOCATION OF EACH ENTRY                                      *****
*****    tape()   DATA TAPE NAME OF EACH ENTRY                                     *****
*****************************************************************************************

*****   DECLARE VARIABLES  *****
      character infil*30,logfil*30,tape(*)*7,header*20,over*1
      real data(4,*)
      integer cyc(*),span(*)

*****   PROMPT THE USER FOR THE INPUT FILE NAME AND OPEN FILE     *****
100   print*
      write (6,1)
1     format ('Enter the name of the input file: ',$)
      read'(a30)',infil
      open (unit=11,file=infil,iostat=inerr,status='old')
      if (inerr .ne. 0) then
       print*, 'File does not exist.  Please try again.'
       goto 100
      endif

*****   PROMPT THE USER FOR THE NAME OF THE LOG FILE AND OPEN FILE  *****
400   print*
      write (6,2)
2     format ('Enter the file name to save the operations log under: ',$)
      read'(a30)',logfil
      open (unit=12,file=logfil,iostat=inerr,status='new')
      if (inerr .ne. 0) then
       write(6,3)
3      format('Log file already exists.  Overwrite? (y/n): ',$)
       read'(a1)',over
       if (over .eq. 'y' .or. over .eq. 'Y') then
        open (unit=12,file=logfil,status='unknown')
       else
        goto 400
       endif
      endif
      write (12,9000)infil
      write (12,*)

*****   READ ALL DATA INTO AN ARRAY AND COUNT THE NUMBER OF ENTRIES  *****
      print*
      print*, 'Reading data ...'
      do 150,j=1,7
       read(11,*)header
150   continue
      n=1
200   read (11,*,end=300)tape(n),cyc(n),span(n),(data(j,n),j=1,4)
      n = n+1
      goto 200
300   continue
      close(11)
      nmax = n-1

9000  format('OPERATIONS PERFORMED TO THE FILE ',a30)

      return
      end
```

```
*************************************************************************
*************************************************************************

      subroutine reset(oldtape,oldcyc,oldspan,olddata,oldnmax,newtape,newcyc,newspan,newdata,newnmax)

*****************************************************************************************************************
***********
*****   THIS SUBROUTINE COPIES TAPE NAME, CYCLE NUMBER, SPAN LOCATION, VELOCITY, YAW, PEAK Cp,   *****
*****   AND AZIMUTH ANGLE ARRAYS INTO IDENTICAL ARRAYS WITH DIFFERENT NAMES.  THIS CAN BE USED
      *****
*****   N TWO WAYS: TO COPY THE INFORMATION TO A BLANK ARRAY OR TO OVERWRITE THE CONTENTS OF*****
*****   ONE ARRAY WITH THOSE OF ANOTHER.  THE ARRAYS DO NOT HAVE TO BE THE SAME LENGTH.  BY     *****
*****   STORING THE INITIAL STATE OF THE ARRAYS USING THIS ROUTINE, THE STATE CAN BE RESTORED     *****
*****   AFTER THE DATA HAS BEEN MANIPULATED OR LIMITED.
      *****
*****                                                                                             *****
*****   WRITTEN BY DEREK SHIPLEY                                                                  *****
*****************************************************************************************************************
***********

*************************************************************************************************
*****   VARIABLE DEFINITIONS:                                                   *****
*****    newcyc()        ARRAY HOLDING "NEW" CYCLE NUMBERS                      *****
*****    newdata(,)      ARRAY HOLDING "NEW" VEL, YAW, PEAK Cp, AND AZ          *****
*****    newnmax         TOTAL NUMBER OF "NEW" ARRAY ENTRIES                    *****
*****    newspan()       ARRAY HOLDING "NEW" SPAN LOCATION VALUES               *****
*****    newtape()               ARRAY HOLDING "NEW" DATA TAPE NAMES                  *****
*****    oldcyc()        ARRAY HOLDING "OLD" CYCLE NUMBERS                      *****
*****    olddata(,)      ARRAY HOLDING "OLD" VEL, YAW, PEAK Cp, AND AZ          *****
*****    oldmax          TOTAL NUMBER OF "OLD" ARRAY ENTRIES                    *****
*****    oldspan()       ARRAY HOLDING "OLD" SPAN LOCATION VALUES               *****
*****    oldtape()       ARRAY HOLDING "OLD" DATA TAPE NAMES                    *****
*************************************************************************************************

*****   DECLARE VARIABLES  *****
      character oldtape(*)*7,newtape(*)*7
      real olddata(4,*),newdata(4,*)
      integer oldcyc(*),oldspan(*),newcyc(*),newspan(*),oldnmax,newnmax

*****   COPY THE "OLD" ARRAYS TO THE "NEW" ONES  *****
      do 100,n=1,oldnmax
       newtape(n) = oldtape(n)
       newcyc(n) = oldcyc(n)
       newspan(n) = oldspan(n)
       do 200,j=1,4
        newdata(j,n) = olddata(j,n)
200    continue
100   continue
      newnmax = oldnmax

      return
      end

*************************************************************************
*************************************************************************

      subroutine limcp(tape,cyc,span,data,nmax)

*****************************************************************************************************
*****   THIS SUBROUTINE LIMITS THE DATA TO ONLY CYCLES HAVING A PEAK                *****
*****   Cp ABOVE A USER DEFINED CUTOFF POINT (ABOVE IS DEFINED AS MORE NEGATIVE).          *****
*****                                                                               *****
*****   WRITTEN BY DEREK SHIPLEY                                                    *****
*****************************************************************************************************

*************************************************************************************************
```

```
*****    VARIABLE DEFINITIONS:                                                  *****
*****     cutoff    Cp CUTOFF VALUE FOR RETAINING ENTRIES                       *****
*****     cyc()     CYCLE NUMBER OF EACH ENTRY                                  *****
*****     data(,)   VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY                *****
*****     new       INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA                    *****
*****     nmax      NUMBER OF ENTRIES IN ARRAYS                                 *****
*****     span()    SPAN LOCATION OF EACH ENTRY                                 *****
*****     tape()    DATA TAPE NAME OF EACH ENTRY                                *****
*******************************************************************************************

*****    DECLARE VARIABLES  *****
       character tape(*)*7
       real data(4,*)
       integer cyc(*),span(*)

*****    PROMPT THE USER FOR THE Cp CUTOFF VALUE  *****
       print*
       write(6,1)
1      format('Enter the desired Cp cutoff value: ',$)
       read*,cutoff

*****    CYCLE THROUGH ALL CURRENT ENTRIES  *****
       new=0
       print*
       do 100,n=1,nmax
*****    IF Cp IS ABOVE THE CUTOFF, "KEEP" THE ENTRY BY OVERWRITING THE CURRENT DATA  *****
        if (data(3,n) .lt. cutoff) then
          new = new+1
          tape(new) = tape(n)
          cyc(new) = cyc(n)
          span(new) = span(n)
          do 200, j=1,4
            data(j,new) = data(j,n)
200       continue
        endif
100    continue
       nmax = new

*****    DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
       print 9000,cutoff
       write (12,9000) cutoff
9000   format('Data limited to Cp < ',f5.1)

       return
       end

*************************************************************************
*************************************************************************

       subroutine limspan(tape,cyc,span,data,nmax)

*******************************************************************************************
*****    THIS SUBROUTINE LIMITS THE DATA ENTRIES TO INCLUDE ONLY THOSE  *****
*****    WITH A USER SPECIFIED SPAN LOCATION.                           *****
*****                                                                   *****
*****    WRITTEN BY DEREK SHIPLEY                                       *****
*******************************************************************************************

*******************************************************************************************
*****    VARIABLE DEFINITIONS:                                                  *****
*****     cyc()     CYCLE NUMBER OF EACH ENTRY                                  *****
*****     desspan   SPAN LOCATION OF DATA TO RETAIN                                   *****
*****     data(,)   VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY                *****
*****     new       INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA                    *****
*****     nmax      NUMBER OF ENTRIES IN ARRAYS                                 *****
*****     span()    SPAN LOCATION OF EACH ENTRY                                 *****
*****     tape()    DATA TAPE NAME OF EACH ENTRY                                *****
```

```
*****************************************************************************************

*****   DECLARE VARIABLES  *****
      character tape(*)*7
      real data(4,*)
      integer cyc(*),span(*),desspan

*****   PROMPT THE USER FOR THE SPAN LOCATION OF THE DATA TO RETAIN  *****
50    print*
      write(6,1)
1     format('Enter the desired span location (30,47,63,80): ',$)
      read*,desspan
      if (desspan .ne. 30 .and. desspan .ne. 47 .and. desspan .ne. 63.and. desspan .ne.80) then
       print*,'Invalide span location.  Please try again.'
       goto 50
      endif

*****   CYCLE THROUGH ALL CURRENT ENTRIES  *****
      new=0
      do 100,n=1,nmax
*****   IF THE SPAN LOCATION EQUALS THE DESIRED SPAN, "KEEP" THE  *****
*****   ENTRY BY OVERWRITING THE CURRENT DATA                          *****
       if (span(n) .eq. desspan) then
        new = new+1
        tape(new) = tape(n)
        cyc(new) = cyc(n)
        span(new) = span(n)
        do 200, j=1,4
         data(j,new) = data(j,n)
200     continue
       endif
100   continue
      nmax = new

*****   DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
      print*
      print 9000,desspan
      write (12,9000) desspan
9000  format('Data limited to ',i2,'% span')

      return
      end

*****************************************************************************
*****************************************************************************

      subroutine limvel(tape,cyc,span,data,nmax)

*****************************************************************************************
*****   THIS SUBROUTINE LIMITS THE DATA ENTRIES TO INCLUDE ONLY THOSE   *****
*****   WITH A VELOCITY WITHIN A USER SPECIFIED RANGE.                  *****
*****                                                                   *****
*****   WRITTEN BY DEREK SHIPLEY                                        *****
*****************************************************************************************

*****************************************************************************************
*****   VARIABLE DEFINITIONS:                                           *****
*****    cyc()     CYCLE NUMBER OF EACH ENTRY                           *****
*****    data(,)   VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY         *****
*****    new       INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA                *****
*****    nmax      NUMBER OF ENTRIES IN ARRAYS                          *****
*****    span()    SPAN LOCATION OF EACH ENTRY                          *****
*****    tape()    DATA TAPE NAME OF EACH ENTRY                         *****
*****    vellow    LOWER VELOCITY LIMIT FOR ENTRIES TO BE RETAINED      *****
*****    velhigh   UPPER VELOCITY LIMIT FOR ENTRIES TO BE RETAINED      *****
*****************************************************************************************
```

```
*****   DECLARE VARIABLES  *****
        character tape(*)*7
        real data(4,*)
        integer cyc(*),span(*)

*****   PROMPT THE USER FOR THE VEL REGION FOR ENTRIES TO BE RETAINED *****
50      print*
        write(6,1)
1       format('Enter the lowest wind velocity to be retained: ',$)
        read*,vellow
        write(6,2)
2       format('Enter the highest wind velocity to be retained: ',$)
        read*,velhigh

*****   CYCLE THROUGH ALL CURRENT ENTRIES  *****
        new=0
        do 100,n=1,nmax
*****   IF THE ENTRY VELOCITY FALLS WITHIN SPECIFIED REGION,  *****
*****   "KEEP" THE CYCLE BY OVERWRITING ORIGINAL DATA        *****
         if (data(1,n) .ge. vellow .and. data(1,n) .le. velhigh) then
           new = new+1
           tape(new) = tape(n)
           cyc(new) = cyc(n)
           span(new) = span(n)
           do 200, j=1,4
             data(j,new) = data(j,n)
200        continue
         endif
100     continue
        nmax = new

*****   DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
        print*
        print 9000,vellow,velhigh
        write (12,9000) vellow,velhigh
9000    format('Velcoity limited to between ',f4.1,' and ',f4.1,' m/s')

        return
        end

*************************************************************************
*************************************************************************

        subroutine limyaw(tape,cyc,span,data,nmax)

*****************************************************************************************
*****   THIS SUBROUTINE LIMITS THE DATA ENTRIES TO INCLUDE ONLY THOSE   *****
*****   WITH A YAW WITHIN A USER SPECIFIED RANGE.                       *****
*****                                                                   *****
*****   WRITTEN BY DEREK SHIPLEY                                        *****
*****************************************************************************************

*****************************************************************************************
*****   VARIABLE DEFINITIONS:                                           *****
*****    cyc()    CYCLE NUMBER OF EACH ENTRY                            *****
*****    data(,)  VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY          *****
*****    new      INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA                *****
*****    nmax     NUMBER OF ENTRIES IN ARRAYS                           *****
*****    span()   SPAN LOCATION OF EACH ENTRY                           *****
*****    tape()   DATA TAPE NAME OF EACH ENTRY                          *****
*****    yawlow   LOWER YAW LIMIT FOR ENTRIES TO BE RETAINED            *****
*****    yawhigh  UPPER YAW LIMIT FOR ENTRIES TO BE RETAINED            *****
*****************************************************************************************

*****   DECLARE VARIABLES  *****
        character tape(*)*7
        real data(4,*)
```

```
      integer cyc(*),span(*)

*****  PROMPT THE USER FOR THE YAW REGION FOR ENTRIES TO BE RETAINED *****
50    print*
      write(6,1)
1     format('Enter the lowest yaw to be retained: ',$)
      read*,yawlow
      write(6,2)
2     format('Enter the highest yaw to be retained: ',$)
      read*,yawhigh

*****  CYCLE THROUGH ALL CURRENT ENTRIES *****
      new=0
      do 100,n=1,nmax
*****  IF THE ENTRY YAW FALLS WITHIN SPECIFIED REGION, *****
*****  "KEEP" THE CYCLE BY OVERWRITING ORIGINAL DATA *****
       if (data(2,n) .ge. yawlow .and. data(2,n) .le. yawhigh) then
         new = new+1
         tape(new) = tape(n)
         cyc(new) = cyc(n)
         span(new) = span(n)
         do 200, j=1,4
           data(j,new) = data(j,n)
200      continue
       endif
100   continue
      nmax = new

*****  DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA *****
      print*
      print 9000,yawlow,yawhigh
      write (12,9000) yawlow,yawhigh
9000  format('Yaw limited to between ',f5.1,' and ',f5.1,' degrees')

      return
      end

*************************************************************************
*************************************************************************

      subroutine limaz(tape,cyc,span,data,nmax)

*****************************************************************************************
*****  THIS SUBROUTINE LIMITS THE DATA ENTRIES TO EXCLUDE THOSE WITH  *****
*****  AN AZIMUTH ANGLE THAT FALLS WITHIN A USER SPECIFIED RANGE.     *****
*****                                                                 *****
*****  WRITTEN BY DEREK SHIPLEY                                       *****
*****************************************************************************************

*****************************************************************************************
*****  VARIABLE DEFINITIONS:                                          *****
*****    azlow   LOWER AZIMUTH LIMIT FOR ENTRIES TO BE EXCLUDED       *****
*****    azhigh  UPPER AZIMUTH LIMIT FOR ENTRIES TO BE EXCLUDED       *****
*****    cyc()   CYCLE NUMBER OF EACH ENTRY                           *****
*****    data(,) VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY         *****
*****    new     INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA                *****
*****    nmax    NUMBER OF ENTRIES IN ARRAYS                          *****
*****    span()  SPAN LOCATION OF EACH ENTRY                          *****
*****    tape()  DATA TAPE NAME OF EACH ENTRY                         *****
*****************************************************************************************

*****  DECLARE VARIABLES *****
      character tape(*)*7
      real data(4,*)
      integer cyc(*),span(*)

*****  PROMPT THE USER FOR THE AZIMUTHAL REGION FOR ENTRIES TO BE EXCLUDED *****
```

```
50      print*
        write(6,1)
1       format('Enter the lower azimuthal boundary of the exclusion region: ',$)
        read*,azlow
        write(6,2)
2       format('Enter the upper azimuthal boundary of the exclusion region: ',$)
        read*,azhigh

*****   CYCLE THROUGH ALL CURRENT ENTRIES  *****
        new=0
        do 100,n=1,nmax
*****   IF THE ENTRY AZIMUTH ANGLE FALLS OUTSIDE OF A SPECIFIED  *****
*****   REGION, "KEEP" THE CYCLE BY OVERWRITING ORIGINAL DATA  *****
         if (data(4,n) .le. azlow .or. data(4,n) .ge. azhigh) then
           new = new+1
           tape(new) = tape(n)
           cyc(new) = cyc(n)
           span(new) = span(n)
           do 200, j=1,4
             data(j,new) = data(j,n)
200        continue
         endif
100     continue
        nmax = new

*****   DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
        print*
        print 9000,azlow,azhigh
        write (12,9000) azlow,azhigh
9000    format('Azimuth limited to between ',f5.1,' and ',f5.1,' degrees')

        return
        end

*************************************************************************
*************************************************************************

        subroutine multspan(tape,cyc,span,data,nmax)

*****************************************************************************************************************
***********
*****   THIS SUBROUTINE LIMITS THE DATA ENTRIES TO INCLUDE THOSE THAT HAVE ENTRIES AT MORE THAN
        *****
*****   ONE SPAN LOCATION.  TYPICALLY, THIS WOULD BE USED IN CONJUNCTION WITH LIMITING THE Cp TO *****
*****   LOCATE CYCLES IN WHICH DYNAMIC STALL OCCURS AT MORE THAN ONE SPAN LOCATION.          *****
*****                                                                                       *****
*****   WRITTEN BY DEREK SHIPLEY                                                            *****
*****************************************************************************************************************
***********

*************************************************************************************
*****   VARIABLE DEFINITIONS:                                          *****
*****    cyc()    CYCLE NUMBER OF EACH ENTRY                           *****
*****    data(,)  VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY         *****
*****    mult()   INDICATES IF THEIR IS > 1 ENTRY FOR A CYCLE                 *****
*****    new      INDEX FOR OVERWRITING ARRAYS WITH LIMITED DATA              *****
*****    nmax     NUMBER OF ENTRIES IN ARRAYS                          *****
*****    span()   SPAN LOCATION OF EACH ENTRY                          *****
*****    tape()   DATA TAPE NAME OF EACH ENTRY                         *****
*************************************************************************************

*****   DECLARE VARIABLES  *****
        character tape(*)*7
        real data(4,*)
        integer cyc(*),span(*),mult(90000)

*****   INITIALIZE INDICATOR ARRAY  *****
```

```
      do 50,n=1,nmax
       mult(n) = 0
50    continue

***** COMPARE EVERY PAIR OF ENTRIES TO DETERMINE IF THEY HAVE THE      *****
***** SAME TAPE NAME AND CYCLE NUMBER. IF SO, SET INDICATOR TO 1.      *****
      do 100,n=1,nmax-1
       do 200,k=n+1,nmax
        if (tape(n) .eq. tape(k) .and. cyc(n) .eq. cyc(k) .and. span(n) .ne. span(k)) then
         mult(n) = 1
         mult(k) = 1
        endif
200    continue
100   continue

***** CYCLE THROUGH ALL CURRENT ENTRIES  *****
      m = 1
      do 300,n=1,nmax
***** IF THE ENTRY INDICATOR EQUALS 1, "KEEP" THE ENTRY BY OVERWRITING THE OLD DATA    *****
       if (mult(n) .eq. 1) then
        tape(m) = tape(n)
        cyc(m) = cyc(n)
        span(m) = span(n)
        do 400,j=1,4
         data(j,m) = data(j,n)
400     continue
        m = m+1
       endif
300   continue
      nmax = m-1

***** DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
      print*
      print 9000
      write (12,9000)
9000  format ('Data reduced to include only cycles with dyn. stall at more than one span')

      return
      end

*************************************************************************
*************************************************************************

      subroutine sort(tape,cyc,span,data,n)

*************************************************************************************************************************
***********
***** THIS SUBROUTINE SORTS THE ENTRIES IN INCREASING ORDER BY YAW, VELOCITY, OR PEAK Cp. THE  *****
***** ENTIRE RECORD IS SORTED ALONG WITH THE DESIRED PARAMETER AND THE ORIGINAL DATA IS        *****
***** OVERWRITTEN BY THE SORTED VERSION. SORTING IS ACCOMPLISH USING A HEAPSORT.              *****
*****                                                                                        *****
***** SORT ROUTINE WRITTEN BY W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY, AND W.T. VETTERLING.  *****
***** NUMERICAL RECIPES: THE ART OF SCIENTIFIC COMPUTING. CAMBRIDGE UNIVERSITY PRESS,         *****
***** CAMBRIDGE, 1989.                                                                       *****
***** MODIFIED BY DEREK SHIPLEY                                                              *****
*************************************************************************************************************************
***********

***** DECLARE VARIABLES  *****
      character tape(*)*7,rtape*7,opt*1,sortcol*10
      real data(4,*),ra(90000),rdata(4)
      integer cyc(*),span(*),rcyc,rspan

***** DETERMINE IF USER WANTS TO SORT BY VELOCITY, YAW, OR PEAK CP  *****
5     print*
      write(6,1)
1     format('Do you want to sort by Velocity, Yaw, or Cp? (V,Y,C): ',$)
```

```
      read*,opt

*****  SET PROPER DATA COLUMN NUMBER AND OUTPUT MESSAGE  *****
      if (opt .eq. 'V' .or. opt .eq. 'v') then
       ncol = 1
       sortcol = 'velocity'
      elseif (opt .eq. 'Y' .or. opt .eq. 'y') then
       ncol = 2
       sortcol = 'yaw'
      elseif (opt .eq. 'C' .or. opt .eq. 'c') then
       ncol = 3
       sortcol = 'Cp'
      else
       print*,'Invalid option.  Please try again.'
       goto 5
      endif

*****  SET SORT VARIABLE EQUAL TO DESIRED DATA COLUMN  *****
      do 50,m=1,n
       ra(m)=data(ncol,m)
50    continue

*****  BEGIN LOOP TO SORT DATA IN PLACE (HEAPSORT METHOD)  *****
     l=n/2+1
     ir=n
10    continue
      if(l.gt.1)then
       l=l-1
       rra=ra(l)
       rtape=tape(l)
             rcyc=cyc(l)
             rspan=span(l)
             do 100,k=1,4
              rdata(k)=data(k,l)
100          continue
      else
       rra=ra(ir)
       rtape=tape(ir)
             rcyc=cyc(ir)
             rspan=span(ir)
             do 200,k=1,4
              rdata(k)=data(k,ir)
200          continue
       ra(ir)=ra(1)
       tape(ir)=tape(1)
             cyc(ir)=cyc(1)
             span(ir)=span(1)
             do 300,k=1,4
              data(k,ir)=data(k,1)
300          continue
       ir=ir-1
       if(ir.eq.1)then
        ra(1)=rra
              tape(1)=rtape
              cyc(1)=rcyc
              span(1)=rspan
              do 400,k=1,4
                    data(k,1)=rdata(k)
400           continue
              print*
              print 9000,sortcol
              write(12,9000) sortcol
        return
       endif
      endif
      i=l
      j=l+l
```

```
20      if(j.le.ir)then
        if(j.lt.ir)then
         if(ra(j).lt.ra(j+1))j=j+1
        endif
        if(rra.lt.ra(j))then
         ra(i)=ra(j)
              tape(i)=tape(j)
              cyc(i)=cyc(j)
              span(i)=span(j)
              do 500,k=1,4
                    data(k,i)=data(k,j)
500           continue
         i=j
         j=j+j
        else
         j=ir+1
        endif
       go to 20
       endif
       ra(i)=rra
            tape(i)=rtape
            cyc(i)=rcyc
            span(i)=rspan
            do 600,k=1,4
             data(k,i)=rdata(k)
600         continue
     go to 10

9000      format('Data sorted in increasing order by ',a10)

     end

*******************************************************************************
*******************************************************************************

          subroutine bindat(data,nmax,odata,onmax)

*****************************************************************************************************************************
***********
*****     THIS SUBROUTINE BINS THE CURRENT ENTRIES INTO TWO DIMENSIONAL BINS BY VELOCITY AND YAW.
          *****
*****     IT ALSO BINS THE DATA ORIGINALLY LOADED (BEFORE ANY MODIFICATION) INTO IDENTICAL BINS.  IT
          *****
*****     THEN CALCULATES THE PERCENTAGE OF THE ORIGINAL ENTRIES THAT ARE IN EACH OF THE CURRENT
          *****
*****     BINS.  ALL THREE VALUES ARE WRITTEN TO AN OUTPUT FILE IN MATRIX FORMAT.  THE USER CAN      *****
*****     SPECIFY THE UPPER AND LOWER VELOCITY AND YAW BIN LIMITS AND THE BIN WIDTH                   *****
*****                                                                                                *****
*****     WRITTEN BY DEREK SHIPLEY                                                                   *****
*****************************************************************************************************************************
***********

*****************************************************************************************
*****     VARIABLE DEFINITIONS:                                                 *****
*****      bin(,)          VELOCITY-YAW BINS FOR CURRENT ENTRIES                *****
*****      data(,)         VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY         *****
*****      MAXBIN                  MAXIMUM NUMBER OF IN VEL. AND YAW DIRECTIONS *****
*****      nmax            NUMBER OF ENTRIES IN CURRENT ARRAYS                  *****
*****      nvel            NUMBER OF VELOCITY BINS                                    *****
*****      nyaw            NUMBER OF YAW BINS                                   *****
*****      onmax           NUMBER OF ENTRIES IN ORIGINAL ARRAYS                 *****
*****      obin(,)         VELOCITY-YAW BINS FOR ORIGINAL ENTRIES               *****
*****      odata(,)        VEL, YAW, PEAK Cp, AND AZIMUTH OF ORIG. ENTRIES      *****
*****      outfil          OUTPUT FILE NAME                                     *****
*****      percent(,)      VELOCITY-YAW BINS FOR PERCENTAGE OF ORIG             *****
*****      vint            VELOCITY BIN WIDTH                                   *****
*****      vlow            LOWER LIMIT OF VELOCITY BINS                         *****
```

```
*****     vup             UPPER LIMIT OF VELOCITY BINS                    *****
*****     yint            YAW BIN WIDTH                                   *****
*****     ylow            LOWER LIMIT OF YAW BINS                         *****
*****     yup             UPPER LIMIT OF YAW BINS                         *****
*************************************************************************************************

*****     DECLARE CONSTANTS AND VARIABLES  *****
          parameter(MAXBIN=100)
          character outfil*30,over*1
          real data(4,*),odata(4,*),percent(MAXBIN,MAXBIN)
          integer bin(MAXBIN,MAXBIN),obin(MAXBIN,MAXBIN),onmax,nmax

*****     PROMPT THE USER FOR THE VELOCITY AND YAW BIN LIMITS AND BIN WIDTH  *****
          print*
          write(6,1)
1         format('Seperated by spaces enter the lower limit, upper limit, and bin width')
          write(6,2)
2         format('for Velocity: ',$)
          read*,vlow,vup,vint
          write(6,3)
3         format('for Yaw: ',$)
          read*,ylow,yup,yint

*****     PROMPT THE USER FOR THE OUTPUT FILE NAME AND OPEN FILE  *****
40        write(6,4)
4         format('Enter name of output file for results: ',$)
          read'(a30)',outfil
          open (unit=13,file=outfil,iostat=inerr,status='new')
          if (inerr .ne. 0) then
           write(6,5)
5          format('Log file already exists.  Overwrite? (y/n): ',$)
           read'(a1)',over
           if (over .eq. 'y' .or. over .eq. 'Y') then
            open (unit=13,file=outfil,status='unknown')
           else
            goto 40
           endif
          endif

*****     DETERMINE THE NUMBER OF VELOCITY AND YAW BINS  *****
          nvel = (vup-vlow)/vint
          nyaw = (yup-ylow)/yint

*****     INITIALIZE BIN VALUES TO ZERO  *****
          do 50,j=1,nvel
           do 60,k=1,nyaw
            bin(j,k) = 0
            obin(j,k) = 0
60         continue
50        continue

*****     FIND THE NUMBER OF ENTRIES THAT FALL INTO EACH VEL-YAW BIN  *****
          do 100,n=1,nmax
           do 200,j=1,nvel
            do 300,k=1,nyaw
             if (data(1,n).ge.(vlow+vint*(j-1)).and.data(1,n).lt.(vlow+j*vint).and.data(2,n).ge.(ylow+yint*(k-
1)).and.data(2,n).lt.(ylow+k*yint))
   3          then
                    bin(j,k)=bin(j,k)+1
            endif
300         continue
200        continue
100       continue

*****     FIND THE NUM OF ORIGINAL ENTRIES THAT FALL INTO EACH VEL-YAW BIN  *****
          do 400,n=1,onmax
           do 500,j=1,nvel
```

```
            do 600,k=1,nyaw
              if (odata(1,n).ge.(vlow+vint*(j-1)).and.odata(1,n).lt.(vlow+j*vint).and.odata(2,n).ge.(ylow+yint*(k-1)).and.
     3        odata(2,n).lt.(ylow+k*yint)) obin(j,k)=obin(j,k)+1
600         continue
500       continue
400     continue

*****   DETERMINE THE PERCENTAGE OF THE ORIGINAL ENTRIES THAT FALL INTO EACH VEL-YAW BIN  *****
        do 700,j=1,nvel
         do 800,k=1,nyaw
          if (obin(j,k).gt.0) then
           percent(j,k) = float(bin(j,k))/float(obin(j,k))*100.
          else
           percent(j,k) = 0
          endif
800       continue
700     continue

*****   WRITE THE PERCENTAGES OF ORIGINAL ENTRIES IN EACH VEL-YAW BIN TO THE OUTPUT FILE  *****
        write(13,9000)0,(ylow+(yint*(2.*k-1)/2.),k=1,nyaw)
        do 900,j=1,nvel
         write(13,9000)vlow+(vint*(2.*j-1)/2.),(percent(j,k),k=1,nyaw)
900     continue
9000    format(101f8.2)
9100    format(f8.2,100i8)

*****   WRITE THE NUMBER OF CURRENT ENTRIES IN EACH VEL-YAW BIN TO THE OUTPUT FILE  *****
        write(13,*)
        write(13,9000)0,(ylow+(yint*(2.*k-1)/2.),k=1,nyaw)
        do 1000,j=1,nvel
         write(13,9100)vlow+(vint*(2.*j-1)/2.),(bin(j,k),k=1,nyaw)
1000    continue

*****   WRITE THE NUMBER OF ORIGINAL ENTRIES IN EACH VEL-YAW BIN TO THE OUTPUT FILE  *****
        write(13,*)
        write(13,9000)0,(ylow+(yint*(2.*k-1)/2.),k=1,nyaw)
        do 1100,j=1,nvel
         write(13,9100)vlow+(vint*(2.*j-1)/2.),(obin(j,k),k=1,nyaw)
1100    continue

*****   DISPLAY MESSAGE DESCRIBING THE PROCEDURE PERFORMED ON THE DATA  *****
        print*
        print 9200,vlow,vup,ylow,yup
        write(12,9200)vlow,vup,ylow,yup
9200    format('Data binned from ',f4.1,' - ',f4.1,' m/s and ',f5.1,' - ',
     2  f5.1,' deg')
        close(13)

        return
        end

*************************************************************************
*************************************************************************

        subroutine writdat(tape,cyc,span,data,nmax)

*****************************************************************************************************
*****   THIS SUBROUTINE ALLOWS THE USER TO WRITE ALL OF THE CURRENT     *****
*****   ENTRIES TO A SPECIFIED OUTPUT FILE.                             *****
*****                                                                   *****
*****   WRITTEN BY DEREK SHIPLEY                                        *****
*****************************************************************************************************

*************************************************************************
*****   VARIABLE DEFINITIONS:                                   *****
*****    cyc()    CYCLE NUMBER OF EACH ENTRY                    *****
*****    data(,)  VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY  *****
```

```
*****     nmax    NUMBER OF ENTRIES IN ARRAYS                          *****
*****     span()  SPAN LOCATION OF EACH ENTRY                          *****
*****     tape()  DATA TAPE NAME OF EACH ENTRY                         *****
*******************************************************************************

*****   DECLARE VARIABLES  *****
      character tape(*)*7,outfil*30,over*1
      real data(4,*)
      integer cyc(*),span(*)

*****   PROMPT THE USER FOR THE OUTPUT FILE NAME AND OPEN FILE  *****
200   print*
      write(6,1)
1     format('Enter the name of the output file: ',$)
      read*,outfil
      open (unit=13,file=outfil,iostat=inerr,status='new')
      if (inerr .ne. 0) then
       write(6,2)
2      format('Log file already exists.  Overwrite? (y/n): ',$)
       read'(a1)',over
       if (over .eq. 'y' .or. over .eq. 'Y') then
        open (unit=13,file=outfil,status='unknown')
       else
        goto 200
       endif
      endif

*****   WRITE ALL CURRENT ENTRIES TO THE SPECIFIED OUTPUT FILE  *****
      do 100,n=1,nmax
       write(13,9000) tape(n),cyc(n),span(n),(data(j,n),j=1,4)
100   continue
      close(13)
9000  format(a7,t11,i3,t19,i2,t26,f6.3,t37,f7.3,t49,f8.4,t62,f8.4)

      return
      end

*******************************************************************************
*******************************************************************************

      subroutine avecp (data,nmax)

*************************************************************************************************
*****   THIS SUBROUTINE FINDS THE AVERAGE Cp VALUE OF ALL CURRENT          *****
*****   ENTRIES AND DISPLAYS ON THE SCREEEN.                               *****
*****                                                                      *****
*****   WRITTEN BY DEREK SHIPLEY                                           *****
*************************************************************************************************

*******************************************************************************
*****   VARIABLE DEFINITIONS:                                        *****
*****    ave      AVERAGE Cp VALUE OF ALL ENTRIES                          *****
*****    data(,)  VEL, YAW, PEAK Cp, AND AZIMUTH OF EACH ENTRY     *****
*****    nmax     NUMBER OF ENTRIES IN ARRAYS                      *****
*****    sum      SUM OF ALL Cp VALUES                             *****
*******************************************************************************

*****   DECLARE VARIABLES  *****
      real data(4,*)

*****   LOOP THROUGH ALL Cp VALUES AND FIND AVERAGE  *****
      if (nmax .gt. 0) then
       sum = 0.
       do 100,n=1,nmax
        sum = sum+data(3,n)
100    continue
       ave = sum/nmax
```

```
      print*
      if (nmax .lt. 10) then
        write(6,9010)nmax,ave
        write(12,9010)nmax,ave
      elseif (nmax .lt. 100) then
        write(6,9020)nmax,ave
        write(12,9020)nmax,ave
      elseif (nmax .lt. 1000) then
        write(6,9030)nmax,ave
        write(12,9030)nmax,ave
      elseif (nmax .lt. 10000) then
        write(6,9040)nmax,ave
        write(12,9040)nmax,ave
      else
        write(6,9050)nmax,ave
        write(12,9050)nmax,ave
      endif
    else
      write(6,9100)
      write(12,9100)
    endif

9010  format('The average of ',i1,' Cp values is ',f6.2)
9020  format('The average of ',i2,' Cp values is ',f6.2)
9030  format('The average of ',i3,' Cp values is ',f6.2)
9040  format('The average of ',i4,' Cp values is ',f6.2)
9050  format('The average of ',i5,' Cp values is ',f6.2)
9100  format('There are no values to average')

      return
      end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1.	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1995	Subcontract Report

4. TITLE AND SUBTITLE

Combined Experiment Phase II Data Characterization

5. FUNDING NUMBERS

C:

TA: WE618110

6. AUTHOR(S)

Mark S. Miller, Derek E. Shipley, Teresa S. Young, Michael C. Robinson, Marvin W. Luttges, David A. Simms

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Colorado at Boulder
Campus Box 19
Boulder, CO 80309

National Renewable Energy Laboratory
1617 Cole Boulevard
Golden, CO 80401-3393

8. PERFORMING ORGANIZATION REPORT NUMBER

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

National Renewable Energy Laboratory
1617 Cole Blvd.
Golden, CO 80401-3393

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

TP-442-6916

DE96000473

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

12b. DISTRIBUTION CODE

UC-1211

13. ABSTRACT *(Maximum 200 words)*

The National Renewable Energy Laboratory's "Combined Experiment" has yielded a large quantity of experimental data on the operation of a downwind horizontal axis wind turbine under field conditions. To fully utilize this valuable resource and identify particular episodes of interest, a number of databases were created that characterize individual data events and rotational cycles over a wide range of parameters. Each of the 59 five-minute data episodes collected during Phase II of the Combined Experiment have been characterized by the mean, minimum, maximum, and standard deviation of all data channels, except the blade surface pressures. Inflow condition, aerodynamic force coefficient, and minimum leading edge pressure coefficient databases have also been established, characterizing each of nearly 21,000 blade rotational cycles. In addition, a number of tools have been developed for searching these databases for particular episodes of interest. Due to their extensive size, only a portion of the episode characterization databases are included in an appendix, and examples of the cycle characterization databases are given. The search tools are discussed and the FORTRAN or C code for each is included in appendices.

14. SUBJECT TERMS

horizontal axis wind turbines; wind turbine aerodynamics

15. NUMBER OF PAGES

16. PRICE CODE

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