NASA TECHNICAL MEMORANDUM

NASA TM X-3323

NASA TM X-3323

CASE FILE COPY

A COMPUTER PROGRAM FOR THE DETERMINATION OF THE ACOUSTIC PRESSURE SIGNATURE OF HELICOPTER ROTORS DUE TO BLADE THICKNESS

G. H. Mall and F. Farassat

Langley Research Center
Hampton, Va. 23665

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JANUARY 1976

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-3323		

4. Title and Subtitle	5. Report Date
A COMPUTER PROGRAM FOR THE DETERMINATION OF THE ACOUSTIC PRESSURE SIGNATURE OF HELICOPTER ROTORS DUE TO BLADE THICKNESS	January 1976
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
G. H. Mall and F. Farassat	L-10437
	10. Work Unit No.
9. Performing Organization Name and Address	505-10-26-02
NASA Langley Research Center Hampton, Va. 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

G. H. Mall: Computer Sciences Corporation.

F. Farassat: The George Washington University, Joint Institute for Acoustics and Flight Sciences.

16. Abstract

This report presents a computer program for the determination of the thickness noise of helicopter rotors. The results are obtained in the form of an acoustic pressure time history. The parameters of the program are the rotor geometry and the helicopter motion descriptors. The formulation employed is valid in the near and far fields. The blade planform must be rectangular, but the helicopter motion is arbitrary. The observer position is fixed with respect to the ground with a maximum elevation of 45^{o} above or below the rotor plane. With these restrictions, the program can also be used for the calculation of thickness noise of propellers.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Aeroacoustics Aerodynamic noise Helicopter rotor noise	Unclassified – Unlimited Subject Category 71

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	79	$4.75

*For sale by the National Technical Information Service, Springfield, Virginia 22161

CONTENTS

Preceding Page Blank

A COMPUTER PROGRAM FOR THE DETERMINATION OF THE ACOUSTIC PRESSURE SIGNATURE OF HELICOPTER ROTORS DUE TO BLADE THICKNESS

G. H. Mall* and F. Farassat**
Langley Research Center

SUMMARY

This report presents a computer program for the determination of the thickness noise of helicopter rotors. The results are obtained in the form of an acoustic pressure time history. The parameters of the program are the rotor geometry and the helicopter motion descriptors. The formulation employed is valid in the near and far fields. The blade planform must be rectangular, but the helicopter motion is arbitrary. The observer position is fixed with respect to the ground with a maximum elevation of 45° above or below the rotor plane. With these restrictions, the program can also be used for the calculation of thickness noise of propellers.

INTRODUCTION

Thickness noise of helicopter rotors or propellers is the noise generated by the normal velocity distribution on the surface of these rotating bodies. In the past, thickness noise of helicopter rotors and propellers has not been studied thoroughly even after the introduction of some approximations. This noise was investigated by Deming (ref. 1) and Arnoldi (ref. 2); the latter based his analysis on the work of Billing (ref. 3). Deming's and Arnoldi's analyses contain some of the following restrictions which should be removed in applications. These restrictions are of three types:

(1) The rotor or the propeller is required to be stationary or to move in axial direction at uniform speed.

(2) The observer is located in the far field.

(3) The acoustic source distribution must be compact.

---

*Computer Sciences Corporation.

**The George Washington University, Joint Institute for Acoustics and Flight Sciences.

For example, Deming's analysis is developed with restrictions 1 and 2. The expression derived in references 4 and 5 removes restrictions 1, 2, and 3. This paper documents a computer program developed to evaluate this expression.

The program, as presented, is for blades with rectangular planform. A variety of uniform airfoil sections can be defined by the user in the program input data. The observer is fixed with respect to the ground; this condition is preferred when experimental and theoretical results are compared. Numerical examples showing favorable comparison with some experimental results are presented in reference 5.

THE FORMULATION

Let the surface of the rotor system be specified by $f(\vec{Y},\tau) = 0$, where $\vec{Y}$ is a Cartesian frame fixed to the undisturbed medium and τ is the time. If v_n is the local normal velocity of the blade surface and p is the acoustic pressure, then the governing equation for the determination of the thickness noise is (refs. 4 and 5)

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial \tau^2} - \nabla^2 p = \frac{\partial}{\partial \tau}\left[\rho_o v_n \left|\nabla f\right| \delta(f)\right] \tag{1}$$

where c is the speed of sound in the undisturbed medium, $\delta(f)$ is the Dirac delta function, and ρ_o is the density of the undisturbed medium. The solution of equation (1) is presented in reference 4 and in slightly more general form in reference 5.

In the present work, the $\vec{\eta}'$- and $\vec{\eta}$-frames in reference 5 are replaced by the $\vec{\eta}$- and $\overrightarrow{YH}$-frames, respectively. Consider one single blade and a nonrotating coordinate system: the $\overrightarrow{YH}$-frame is fixed to the center of rotation (fig. 1), and a rotating system is the $\vec{\eta}$-frame, fixed to the blade (fig. 2). Let $T(\eta_1,\eta_2)$ be the thickness function of the blade in this frame. Then the solution of equation (1) is given by equation (32) of reference 5 as follows:

$$p(\vec{X},t) = \frac{\rho_o c}{2\pi}\frac{\partial}{\partial t}\int_{\tau_1}^{\tau_2}\int_{\Gamma(D.P.)} \frac{T_1\tilde{V}_1 + T_2\tilde{V}_2}{r\left[1 - \hat{r}_3^{\,2} + T_1^{\,2}\left(1 - \hat{r}_1^{\,2}\right)\right]^{1/2}}\, d\Gamma\, d\tau \tag{2}$$

where

$p(\vec{X},t)$ acoustic pressure

ρ_o density of undisturbed medium

$\vec{X},t$ observer position and time

τ_1,τ_2 source times when the sphere $g = \tau - t + \left|\vec{X} - \vec{Y}\right|/c = 0$ enters and leaves the blade, respectively ($\vec{X}$ and t fixed), $\left|\vec{X} - \vec{Y}\right| = r$

$\vec{Y}$ source location $\vec{Y}$-frame

(VH_1,VH_2) vehicle velocity components along Y_1 and Y_2 axes of $\vec{Y}$-frame

Ω angular velocity of rotor

$(\hat{r}_1,\hat{r}_2,\hat{r}_3)$ unit vector along radiation direction in $\vec{\eta}$-frame

Γ curve of intersection of sphere $g = 0$ and mean surface of the blade

D.P. disk plane

T_1 $= \partial T/\partial\eta_1$, $T_2 = \partial T/\partial\eta_2$

$\tilde{V}_1$ $= -VH_1 + \eta_2\Omega$

$\tilde{V}_2$ $= -VH_2 - \eta_1\Omega$

For several blades, the contribution of each to the acoustic pressure is added linearly.

THE PROGRAM

General Discussion

For this program, the blades are assumed to have a rectangular planform with a uniform airfoil section along the span. The thickness distribution function $T(\eta_1,\eta_2)$ is denoted as $F(\eta_1)$ here since there is no dependence on η_2. The Rotor Thickness Noise program (RTN) evaluates the integral in equation (2) numerically. Evaluation of this double integral involves one integration over the curve of intersection of the sphere $g = 0$ and the rotor blade; the other integration is over source time. The first of these integrals is evaluated using a trapezoidal integration. The second integral is performed with Simpson's rule and a trapezoidal end-point correction where required. The Rotor Thickness Noise program postprocessor (RTNPP) completes the calculation of the acoustic pressure by differentiating the RTN output numerically with respect to observer time. The purpose of this section is to present detailed descriptions of the RTN and RTNPP programs, to describe the manner in which input data must be prepared, and to describe the program output. Flow charts and FORTRAN listings of the programs are included as appendixes.

The solution technique employed by the RTN program is presented in the section of this report entitled "Method of Solution." Within this section, several key internal program variables are identified parenthetically. The RTN program reads a number of parameters describing the rotor system structure, the amount of detail to be used in performing the integrations, and various program options. A detailed description of the input required by the program is given in the section entitled "Preparation of Input Data." Output from the RTN program includes an echo of the input data and a table of ordered pairs $\left[\Phi(\vec{X},t),t\right]$ where t is the observer time and $\Phi(\vec{X},t)$ is defined by equation (3).

$$\Phi(\vec{X},t) = \frac{\rho_o c}{2\pi} \int_{\tau_1}^{\tau_2} \int_{\Gamma(\mathrm{D.P.})} \frac{\tilde{V}_1 F'(\eta_1)}{r\left[1 - \hat{r}_3^{\,2} + F'^2(\eta_1)\left(1 - \hat{r}_1^{\,2}\right)\right]^{1/2}} \, d\Gamma \, d\tau \tag{3}$$

The section entitled "Postprocessor Description" discusses the numerical differentiation methodology used by the RTNPP program in computing the thickness noise from equation (3).

Method of Solution

Based upon the translational velocity of the rotor system (VH) and the specified initial value of source time (TAU), the location of the center of the rotor system relative to a fixed coordinate system is determined. This fixed coordinate system (Y) is described in figure 1. The origin of this system is the location of the center of the rotor system at time TAU = 0. The Y_1Y_2-plane contains the rotor disk and the Y_1-axis is parallel to the initial orientation of blade number one. The direction from the blade tip to the hub is the positive direction. A new reference frame (YH) is constructed with its origin coinciding with the center of the rotor system at time TAU. This nonrotating coordinate system remains fixed to the helicopter. The position of the observer at time TAU in the YH-frame (X) is computed from the specified location in the Y-frame (X0). The blade-fixed coordinate system is shown in figure 2.

At time TAU, a sphere is constructed with its center at the position of the observer with a radius (RM) equal to the maximum distance between the observer and any blade tip. The observer time (OTIME) is determined by the relation OTIME = TAU - RM/SNDSPD, where SNDSPD is the speed of sound. The radius of this sphere is subsequently permitted to collapse at the speed of sound. The points of intersection of the projection of this sphere on the rotor system plane and the circle defined by the outer radii of the blades are determined as shown in figure 3. At the position of the observer, the more clockwise of these points (YSTART) defines the initial sweep angle (PHI0). The other point of intersection (YEND) defines the final sweep angle.

The integral over the intersection of the sphere and the rotor blade is performed by sweeping the arc of the intersection of the projection of the sphere and the rotor system plane. The arc is swept in a counterclockwise sense. To execute the sweep of such an arc efficiently, the rotor system is divided into four regions as illustrated in figure 4. These regions are defined as follows:

IREGION = 0	The point lies outside the circle defined by the outer radius of the blades.
= 1	The point lies between the circles defined by the outer and inner radii of the blades, but is not on a blade.
= 3	The point lies inside the circle defined by the inner (hub) radius of the blades.
= 6	The point lies on a blade.

Three sweep increments are used by the program. In region 1, a rather coarse angular increment (DELPHI1) is used to space across the region. When a blade is encountered, the intersection of the arc with the edge of the blade is computed. A smaller angular increment (DELPHI2) is used to sweep across the blade. An even smaller angular increment is allowed for points near the leading edge for blunt leading edges. The values of the integrand and differential increment are calculated for each point on a blade. The line integral (trapezoidal) is computed dynamically by cumulative summing. The individual contributions of each blade are also summed. The hub region is bypassed by using the computed value of the intersection points of the arc with the inner circle. The line integral is completed when the sweep across the arc reaches the final sweep angle.

When a sweep has been completed, the value of the sweep time (SWPTAU) is increased by a small increment (DELTAU). The center position of the rotor system relative to the observer is recomputed. A new sphere is constructed and its line integral is calculated. This process is continued until the sphere no longer intersects the rotor system. As these line integrals are calculated, they are dynamically accumulated into a Simpson's rule sum. The integrands are multiplied by two or four if they are even or odd terms, respectively, and are then added to the sum. If the last term is even, a trapezoidal correction is made.

After the last line integral has been calculated, the value of the double integral and the corresponding observer time are output. The value of the source time is increased by some multiple of DELTAU and the entire process is repeated until one of three possible termination criteria is satisfied. These criteria are:

(1) TAU exceeds a specified maximum value.

(2) The radius of the contracting sphere becomes less than the outer radius of the blades.

(3) The angle of the observer relative to the center of the rotor system and the rotor system plane exceeds 45^o.

The hierarchy of program RTN and its 10 subprograms is illustrated in figure 5. Flow charts and FORTRAN listings of every RTN subprogram are presented in appendix A.

Preparation of Input Data

All input data to RTN are associated with the single NAMELIST name ROTOR. These variables are summarized in table I according to dimension, type, and units and are described in the following paragraphs.

CH — Blade chord.

COEFFS — The coefficients for describing the airfoil shape. The airfoil shape used in the subprogram FOFETA1 is given in nondimensional form by

$$F(E)/CH = T\left(C_1\sqrt{E} + C_2E + C_3E^2 + C_4E^3 + C_5E^4\right)$$

where $E = \eta_1/CH$, η_1 is the distance along the chord from the leading edge, CH is the blade chord, and T is the thickness ratio. The function F(E) is the airfoil thickness function which is the airfoil ordinate at η_1. This equation is suitable for many airfoil shapes, in particular for the NACA four digit airfoil series. The coefficients for this series (from ref. 6) are as follows:

$C_1 = 1.4845$

$C_2 = -0.6300$

$C_3 = -1.7580$

$C_4 = 1.4215$

$C_5 = -0.5075$

DELTET — The angular displacement of the rotor system between consecutive positions of the contracting sphere. This variable is used with the rotor system angular velocity to compute DELTAU, the time required for the rotor to rotate DELTET degrees. DELTAU is then used as the differential increment in the integration over time. The source time TAU is increased by some multiple of DELTAU between computations of $\varphi(\vec{X},t)$. A useful formula to select DELTET is

$$\text{DELTET} = \frac{(\text{CH})(\text{OMEGA}) \cos \psi}{2.5(345 - \text{VH} \cos \psi)}$$

where ψ is the angle that the line from observer to the center of rotation makes with the vertical and VH is the helicopter speed.

DTAUM — A factor to determine the time increment between data points. The time increment derived from the variable DELTET is multiplied by DTAUM to determine the time interval between data points. Consequently, the rotor system rotates DTAUM*DELTET degrees between data points. A useful rule for selecting this factor is

$$\text{DTAUM} = \frac{4 \text{ to } 6}{(\text{NBLADES})(\text{DELTET})}$$

ETAMAX — The maximum η_1 value for which increased resolution is desired near the leading edge for airfoils with blunt leading edges. Generally, a value between 10 to 15 percent of the chord is sufficient.

JFLG — Rotor system translational motion indicator:

JFLG = 0	Hovering helicopter
= 1	Helicopter in forward flight

KFLG — Airfoil section type indicator:

KFLG = 0	Sharp leading edge
= 1	Blunt leading edge

MORDATA — Additional data indicator:

MORDATA = .T.	Process another data set
= .F.	Terminate

N — A factor to control the spacing along the arc for points not on a blade (i.e., to skip from one blade to the next). The variable N should not be made too small since a large sweep increment near a blade tip might skip over the blade completely. Approximately N points are used to traverse a distance along the arc equal to the blade chord. A value between 3 and 8 is suggested.

NBLADES	The number of blades in the rotor system. The program allows a maximum of eight blades.
NS	A factor to control spacing along the arc for points on a blade. The variable NS is used to compute the sweep angular increment, DELPHI2. The DELPHI2 determines the line integral differential increment. Approximately NS points are used to traverse a distance along the arc equal to the blade chord. A value between 15 and 25 is suggested.
OMEGA	Angular velocity of the rotor system.
PERDM	The number of periods of the blade rotation to be processed. Because of the lengthy computation time, this number should not be given a large value. This variable is used to establish the maximum time termination criterion.
R	Radius of the blades.
RO	Hub radius.
SNDSPD	The speed of sound.
TAUINT	The initial value of TAU, the source time. Nonzero values of this variable provide a restart capability as well as a means of generating high-resolution calculations in TAU regions of interest.
THKRAT	Blade thickness ratio.
X0	The location of the observer relative to (0,0,0), the initial position of the center of the rotor system at time TAU = 0 (i.e., the Y-frame).

Postprocessor Description

The relationship between the RTN program and the postprocessor (RTNPP) is illustrated in figure 6. Sample RTN output which is input to RTNPP is shown in figure 7. No input, other than the information generated on TAPE12 by RTN, is required by the postprocessor.

The Rotor Thickness Noise program postprocessor in use at Langley Research Center (LRC) performs three functions. The RTN output ordered pairs (φ,t) are used to find the thickness noise by differentiating them numerically with respect to t, the observer time. The resulting thickness noise is interpolated to provide a specification at equally spaced observer time values. The results are then plotted.

Because φ is the result of numerical integration, error resulting from finite resolution should be eliminated before attempting to differentiate the RTN output. This elimination is accomplished with a subroutine which fits a smooth cubic spline curve to the set of data points. This routine requires a specification of the allowed variation of each functional value. For each data point, the difference between the raw data and a value obtained by trigonometric smoothing is used to satisfy this requirement.

The postprocessor output consists of four plots (illustrated in figs. 8 to 11) for each data set. The raw data as output by RTN are displayed in figure 8. The cubic spline curve is presented in figure 9. Figure 10 displays the smoothed data over the raw data to provide a measure of the quality of the cubic spline fit. Figure 11 shows the acoustic pressure time history.

Flow charts and FORTRAN listings of several RTNPP subprograms are presented in appendix B. Several routines required by RTNPP were obtained from the LRC math and graphics libraries. Descriptions of these subroutines are included in appendix B.

CONCLUDING REMARKS

This report presents a computer program for calculating the thickness noise of helicopter rotors with rectangular blades. Favorable comparisons with experimental results containing high-speed helicopter blade slap have been found and are reported in reference 5. The program is written in Control Data Corporation (CDC) FORTRAN and, although several postprocessor subroutines are not fully documented because of proprietary rights, sufficient information is supplied to adapt the program to any facility.

Langley Research Center
National Aeronautics and Space Administration
Hampton, Va. 23665
December 12, 1975

APPENDIX A

ROTOR THICKNESS NOISE PROGRAM

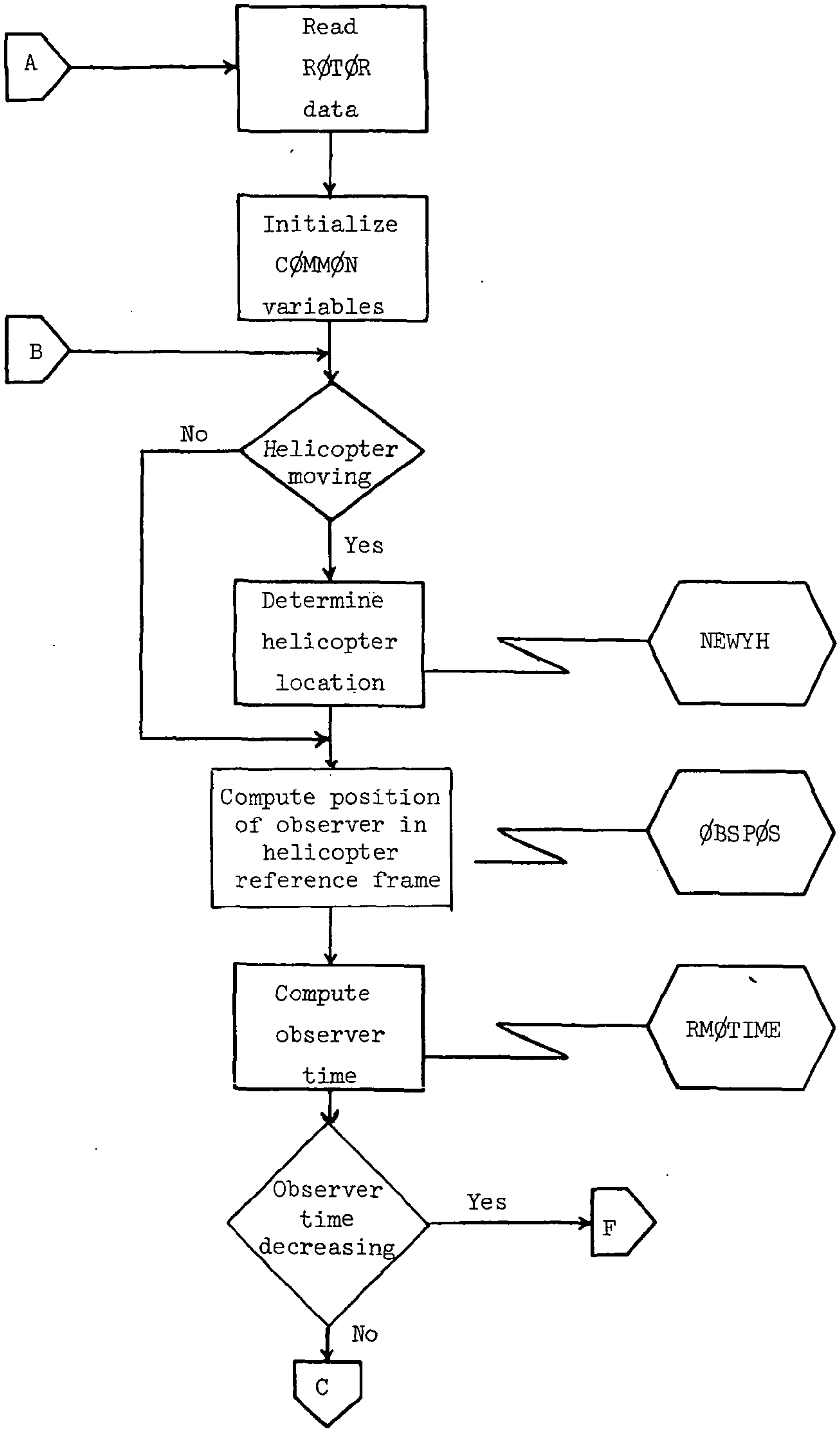
This appendix provides a brief description of the calculations performed by each RTN subprogram. The subprogram flow chart and FORTRAN listing follow the description.

Program RTN

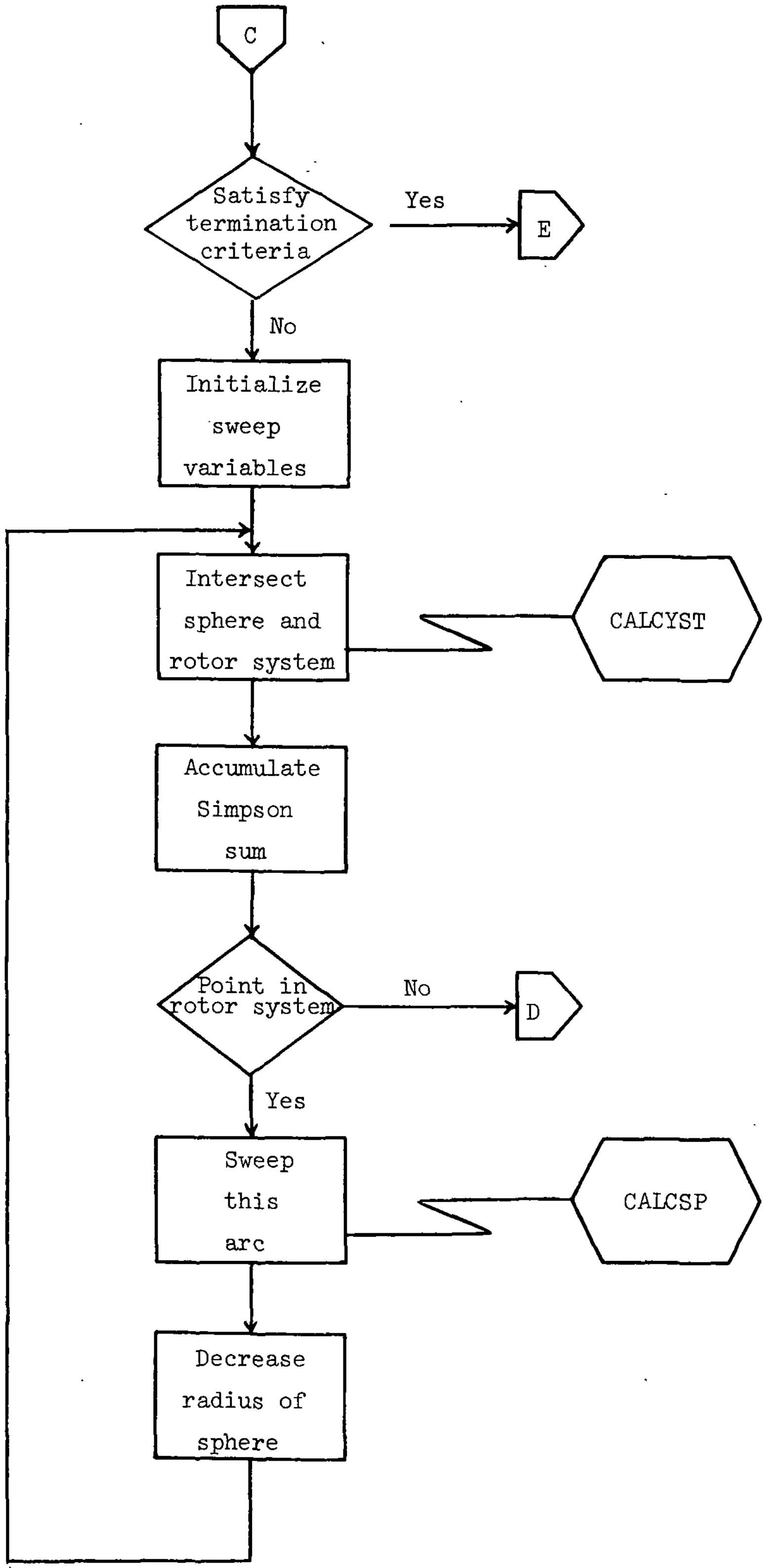
RTN reads and writes the NAMELIST input data and initializes other variables used elsewhere in the program. Following each contraction of the sphere, a check is made to see if any termination criteria have been satisfied or if the sphere has completed its pass through the rotor system. If the sphere is still intersecting the rotor system, an additional contribution is made to the Simpson's rule integral. If the sphere has passed out of the rotor system, the source time is incremented, the computed value of the double integral is output, and an additional sphere contraction is initiated. If one of the termination criteria has been satisfied, the program terminates unless an additional data set is specified.

APPENDIX A

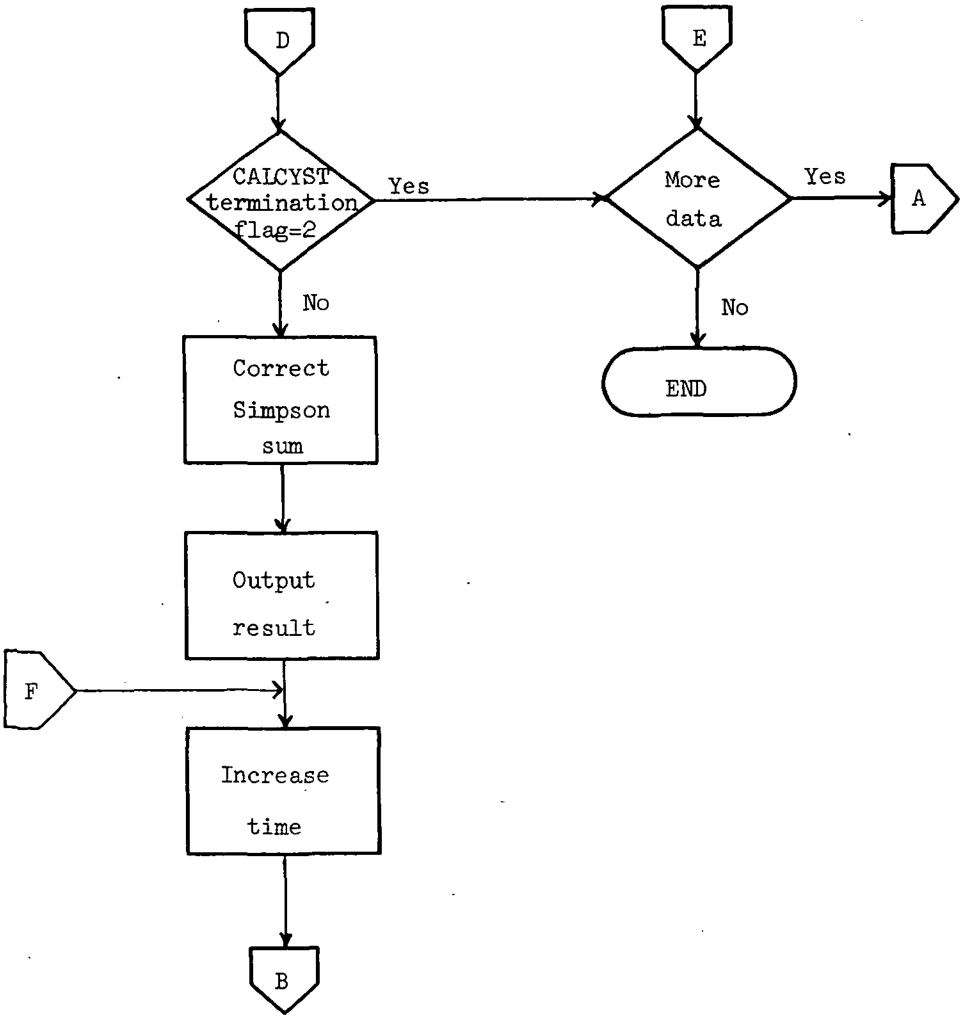
Flow Chart of Program RTN



APPENDIX A


C
Satisfy termination criteria
Yes
E
No
Initialize sweep variables
Intersect sphere and rotor system
CALCYST
Accumulate Simpson sum
Point in rotor system
No
D
Yes
Sweep this arc
CALCSP
Decrease radius of sphere

APPENDIX A


D
E
CALCYST
termination
flag=2
Yes
More
data
Yes
A
No
No
Correct
Simpson
sum
END
Output
result
F
Increase
time
B

APPENDIX A

Listing of Program RTN

```
      PROGRAM RTN(OUTPUT=101B,INPUT=101B,TAPE12=OUTPUT,TAPE5=INPUT)         RTN
C*********************************************************************** RTN
C*                                                                     * RTN
C*           R O T O R   T H I C K N E S S   N O I S E                 * RTN
C*                                                                     * RTN
C*********************************************************************** RTN
C*                                                                     * RTN
C*                                                                     * RTN
C*     PURPOSE:                                                        * RTN
C*                    PROGRAM RTN DETERMINES THE ACOUSTIC PRESSURE     * RTN
C*                    GENERATED BY A ROTATING THICK BODY, SUCH AS AN   * RTN
C*                    IDEALIZED RIGID ROTOR, WITH SOME SPECIFIED MOTION * RTN
C*                    RELATIVE TO AN OBSERVER AT SOME INITIAL DISTANCE * RTN
C*                    FROM THE BODY. THE PRESENT PROGRAM INCLUDES      * RTN
C*                    ROUTINES DEVELOPED SOLELY FOR ROTOR SYSTEMS WITH * RTN
C*                    RECTANGULAR BLADES AND UNIFORM AIRFOIL SECTION   * RTN
C*                    ACROSS THE SPAN OF THE BLADES.                   * RTN
C*                                                                     * RTN
C*                                                                     * RTN
C*      METHOD                                                         * RTN
C*         OF                                                          * RTN
C*    SOLUTION:                                                        * RTN
C*                    THE FORM OF THE SOLUTION UTILIZED BY THIS PROGRAM * RTN
C*                    INCLUDES A DOUBLE INTEGRAL WHERE ONE INTEGRATION * RTN
C*                    IS OVER THE INTERSECTION OF THE CONTRACTING      * RTN
C*                    SPHERE C*(TAU-T)+R=0 (CENTERED AT OBSERVER)      * RTN
C*                    AND THE BODY, AND THE OTHER INTEGRATION IS OVER  * RTN
C*                    SOURCE TIME, TAU. THE FIRST OF THESE INTEGRALS IS * RTN
C*                    EVALUATED USING A TRAPEZOIDAL INTEGRATION. THE   * RTN
C*                    SECOND INTEGRAL IS PERFORMED WITH SIMPSONS       * RTN
C*                    RULE AND A TRAPEZOIDAL END POINT CORRECTION      * RTN
C*                    WHERE REQUIRED.                                  * RTN
C*                                                                     * RTN
C*                    BEGINNING WITH AN INITIAL VALUE OF TIME          * RTN
C*                    (TAU), THE TRANSLATIONAL VELOCITY OF THE ROTOR   * RTN
C*                    SYSTEM IS USED TO LOCATE THE POSITION OF ITS     * RTN
C*                    CENTER. THE LOCATION OF THE OBSERVER RELATIVE    * RTN
C*                    TO THIS CENTER POSITION IS DETERMINED AND A      * RTN
C*                    SPHERE IS CONSTRUCTED (CENTERED AT THE OBSERVER) * RTN
C*                    WITH A RADIUS EQUAL TO THE MAXIMUM DISTANCE      * RTN
C*                    BETWEEN THE OBSERVER AND ANY BLADE TIP. THIS     * RTN
C*                    TIME REPRESENTS THE LOWER LIMIT ON THE TIME      * RTN
C*                    INTEGRAL AND THIS SPHERE REPRESENTS THE INITIAL  * RTN
C*                    POSITION OF THE CONTRACTING SPHERE. THE POINTS   * RTN
C*                    OF INTERSECTION OF THE PROJECTION OF THIS        * RTN
C*                    SPHERE ON THE ROTOR SYSTEM PLANE AND THE CIRCLE  * RTN
C*                    DEFINED BY THE BLADES OUTER RADIUS ARE DETERMINED. * RTN
C*                    AT THE OBSERVERS POSITION THE MORE CLOCKWISE OF  * RTN
C*                    THESE POINTS DEFINES THE INITIAL SWEEP ANGLE.    * RTN
C*                    THE ARC THROUGH THE ROTOR SYSTEM PLANE COR-      * RTN
C*                    RESPONDING TO THIS POSITION OF THE SPHERE IS     * RTN
C*                    SWEPT IN A COUNTERCLOCKWISE SENSE TO COMPUTE     * RTN
C*                    THE LINE INTEGRAL FOR THIS VALUE OF SOURCE TIME. * RTN
C*                                                                     * RTN
C*                    TO DECREASE EXECUTION TIME, TWO SWEEP INCREMENTS * RTN
C*                    ARE USED BY THE PROGRAM. FOR POINTS ON THE ARC   * RTN
C*                    WHICH ARE NOT ON A BLADE A LARGE ANGULAR INCREMENT * RTN
C*                    IS USED TO SPACE ACROSS THESE REGIONS. SMALLER   * RTN
C*                    ANGULAR INCREMENTS ARE AUTOMATICALLY USED BY THE * RTN
C*                    PROGRAM WHEN A BLADE IS ENCOUNTERED. FOR BLUNT   * RTN
C*                    EDGES A SMALLER INCREMENT MAY BE GENERATED FOR   * RTN
C*                    POINTS BETWEEN THE LEADING EDGE AND A MAXIMUM VALUE * RTN
C*                    OF THE DISTANCE FROM THE LEADING EDGE. FOR EACH  * RTN
C*                    POINT ON A BLADE THE VALUE OF THE INTEGRAND AND  * RTN
C*                    DIFFERENTIAL INCREMENT ARE COMPUTED. THE LINE    * RTN
```

```
C*                 INTEGRAL (TRAPEZOIDAL) IS COMPUTED DYNAMICALLY       *
C*                 BY CUMULATIVE SUMMING. WHEN A POINT ON THE ARC       *
C*                 IS REACHED WHICH IS OUTSIDE THE CIRCLE DEFINED       *
C*                 BY THE ROTOR SYSTEM, THE VALUE OF THE RESULTANT      *
C*                 INTEGRAL IS MULTIPLIED BY THE APPROPRIATE PHYSICAL   *
C*                 CONSTANT.                                            *
C*                                                                      *
C*                 THE VALUE OF SOURCE TIME IS INCREASED SLIGHTLY.      *
C*                 THE CENTER POSITION OF THE ROTOR SYSTEM RELATIVE     *
C*                 TO THE OBSERVER IS RECOMPUTED. A NEW SPHERE IS       *
C*                 CONSTRUCTED AND THIS LINE INTEGRAL IS CALCULATED.    *
C*                 THIS PROCESS IS CONTINUED UNTIL THE PROJECTION       *
C*                 OF THE SPHERE HAS COMPLETELY PASSED THROUGH THE      *
C*                 ROTOR SYSTEM. THE FINAL VALUE OF SOURCE TIME         *
C*                 REPRESENTS THE UPPER LIMIT ON THE TIME INTEGRAL.     *
C*                                                                      *
C*                 SINCE THE SOURCE TIME DIFFERENTIAL INCREMENT IS      *
C*                 CONSTANT, A SIMPSON RULE INTEGRATION IS POSSIBLE.    *
C*                 THIS INTEGRAL IS PERFORMED DYNAMICALLY AS A          *
C*                 CUMULATIVE SUM. AS LINE INTEGRALS ARE CALCULATED     *
C*                 THEY ARE MULTIPLIED BY TWO(FOUR) IF THEY ARE         *
C*                 EVEN(ODD) TERMS AND ADDED TO THE SUM. IF THE LAST    *
C*                 TERM IS EVEN, A TRAPEZOIDAL CORRECTION IS MADE.      *
C*                                                                      *
C*                 THE VALUE OF TAU IS INCREASED AND THE ENTIRE         *
C*                 PROCESS IS REPEATED UNTIL ONE OF THREE POSSIBLE      *
C*                 TERMINATION CRITERIA IS SATISFIED. THESE CRITERIA    *
C*                 ARE:                                                 *
C*                                                                      *
C*                   1-TAU EXCEEDS A SPECIFIED MAXIMUM VALUE            *
C*                                                                      *
C*                   2-THE RADIUS OF THE CONTRACTING SPHERE             *
C*                     BECOMES LESS THAN THE RADIUS OF THE BLADES       *
C*                     OUTER RADIUS                                     *
C*                                                                      *
C*                   3-THE ANGLE OF THE OBSERVER RELATIVE TO            *
C*                     THE CENTER OF THE ROTOR SYSTEM AND THE           *
C*                     ROTOR SYSTEM PLANE EXCEEDS 45 DEGREES            *
C*                                                                      *
C*                 ONCE ONE OF THE TERMINATION CRITERIA IS SATISFIED    *
C*                 THE PROGRAM TERMINATES UNLESS AN ADDITIONAL DATA     *
C*                 SET IS SPECIFIED. NUMERICAL DIFFERENTIATION OF THE   *
C*                 OUTPUT, AS WELL AS PLOTTING THE RESULTS AND          *
C*                 FORMATTING THE RESULTS FOR SUBSEQUENT ANALYSIS       *
C*                 IS PERFORMED IN A COMPANION POST-PROCESSOR.          *
C*                                                                      *
C*                                                                      *
C* PARAMETERS:                                                          *
C*                                                                      *
C*           X0    THE LOCATION OF THE OBSERVER RELATIVE TO (0,0,0),    *
C*                 THE INITIAL POSITION OF THE ROTOR SYSTEM AT TIME     *
C*                 TAU = 0. (METERS)                                    *
C*                                                                      *
C*       NBLADES   THE NUMBER OF BLADES IN THE ROTOR SYSTEM. THE        *
C*                 PROGRAM ALLOWS A MAXIMUM OF 8 BLADES.                *
C*                                                                      *
C*            R    RADIUS OF THE BLADES. (METERS)                       *
C*                                                                      *
C*           CH    BLADE CHORD. (METERS)                                *
C*                                                                      *
C*        OMEGA    ANGULAR VELOCITY OF THE ROTOR SYSTEM. (REVOLUTIONS   *
C*                 PER MINUTE)                                          *
C*                                                                      *
C*           RO    HUB RADIUS. (METERS)                                 *
C*                                                                      *
C*       THKRAT    BLADE THICKNESS RATIO.                               *
C*                                                                      *
```

APPENDIX A

```
C*     DELTET     ANGULAR DISPLACEMENT OF THE ROTOR SYSTEM BETWEEN       *
C*                CONSECUTIVE POSITIONS OF THE CONTRACTING SPHERE.       *
C*                THIS VARIABLE IS USED WITH THE ROTOR SYSTEM ANGULAR    *
C*                VELOCITY TO COMPUTE THE SOURCE TIME DIFFERENTIAL       *
C*                INCREMENT, DELTAU. (DEGREES)                           *
C*                                                                       *
C*          N     FACTOR TO CONTROL SPACING ALONG ARC BETWEEN BLADES     *
C*                (I.E., TO SKIP FROM ONE BLADE TO THE NEXT).            *
C*                APPROXIMATELY N POINTS ARE USED TO TRAVERSE A          *
C*                DISTANCE ALONG THE ARC EQUAL TO THE BLADE CHORD.       *
C*                                                                       *
C*         NS     FACTOR TO CONTROL SPACING ALONG ARC ON A BLADE.        *
C*                APPROXIMATELY NS POINTS ARE USED TO TRAVERSE A         *
C*                DISTANCE ALONG THE ARC EQUAL TO THE BLADE CHORD.       *
C*                THIS VARIABLE DETERMINES THE LINE INTEGRAL             *
C*                DIFFERENTIAL INCREMENTS.                               *
C*                                                                       *
C*       KFLG     AIRFOIL SECTION TYPE INDICATOR.                        *
C*                                                                       *
C*                  KFLG = 0:  SHARP LEADING EDGE                        *
C*                                                                       *
C*                       = 1:  BLUNT LEADING EDGE                        *
C*                                                                       *
C*     ETAMAX     MAXIMUM ETA VALUE FOR WHICH INCREASED RESOLUTION       *
C*                IS DESIRED NEAR THE LEADING EDGE FOR BLUNT LEADING     *
C*                EDGES.                                                 *
C*                                                                       *
C*      BLNTN     FACTOR TO GENERATE FINE SPACING NEAR THE LEADING       *
C*                EDGE FOR BLUNT LEADING EDGES. FOR ETA VALUES           *
C*                BETWEEN ZERO AND ETAMAX, THE SPACING FACTOR NS IS      *
C*                MULTIPLIED BY BLNTN.                                   *
C*                                                                       *
C*       JFLG     ROTOR SYSTEM TRANSLATIONAL MOTION INDICATOR.           *
C*                                                                       *
C*                  JFLG = 0:  HOVERING HELICOPTER.                      *
C*                                                                       *
C*                       = 1:  HELICOPTER IN FORWARD FLIGHT.             *
C*                                                                       *
C*     COEFFS     COEFFICIENTS IN THE FUNCTION DESCRIBING THE AIRFOIL    *
C*                SECTION. A MINUS SIGN MUST APPEAR EXPLICITLY FOR       *
C*                NEGATIVE TERMS.                                        *
C*                                                                       *
C*     TAUINT     THE INITIAL VALUE OF SOURCE TIME (TAU). (SECONDS)      *
C*                                                                       *
C*      PERDM     THE NUMBER OF PERIODS OF THE ROTOR SYSTEM TO BE        *
C*                PROCESSED. THIS VARIABLE IS USED TO ESTABLISH          *
C*                THE MAXIMUM TIME TERMINATION CRITERION.                *
C*                                                                       *
C*      DTAUM     TIME INCREMENT BETWEEN DATA POINTS. THE SOURCE         *
C*                TIME DIFFERENTIAL INCREMENT DERIVED FROM DELTET        *
C*                IS MULTIPLIED BY DTAUM TO DETERMINE THIS INCREMENT.    *
C*                                                                       *
C*     SNDSPD     SPEED OF SOUND. (METERS/SEC)                           *
C*                                                                       *
C*    MORDATA     A LOGICAL VARIABLE TO SPECIFY IF AN ADDITIONAL         *
C*                DATA SET IS TO BE PROCESSED.                           *
C*                                                                       *
C*                                                                       *
C*   REQUIRED                                                            *
C*   ROUTINES:                                                           *
C*                SUBROUTINE NEWYH                                       *
C*                SUBROUTINE CALCVH (SEE ADDITIONAL DETAILS)             *
C*                SUBROUTINE OBSPOS                                      *
C*                SUBROUTINE RMOTIME                                     *
C*                SUBROUTINE CALCYST                                     *
C*                SUBROUTINE CALCSP                                      *
C*                FUNCTION FOFETA1 (SEE ADDITIONAL DETAILS)              *
```

```
C*                 FUNCTION PFUNCN                                      *
C*                 SUBROUTINE TEST (SEE ADDITIONAL DETAILS)             *
C*                 SUBROUTINE LEETA                                     *
C*                                                                      *
C*                                                                      *
C* ADDITIONAL                                                           *
C*    DETAILS:                                                          *
C*                 FOR A MOVING ROTOR SYSTEM (JFLG = 1), A SUBROUTINE   *
C*                 MUST BE PROVIDED TO DESCRIBE THE MOTION.             *
C*                                                                      *
C*                   SUBROUTINE CALCVH(VH,TIME)                         *
C*                                                                      *
C*                     SUBROUTINE CALCVH DEFINES THE VECTOR VH(3)       *
C*                     WHICH MUST CONTAIN THE THREE COMPONENTS OF       *
C*                     THE ROTOR SYSTEM TRANSLATIONAL VELOCITY AT       *
C*                     T = TIME. THESE COMPONENTS ARE TO BE SPECIFIED   *
C*                     IN THE ROTOR SYSTEM REFERENCE FRAME. THE THREE   *
C*                     AXES IN THIS COORDINATE SYSTEM ARE DEFINED AS    *
C*                     FOLLOWS.                                         *
C*                                                                      *
C*                       THE Y1 AXIS IS PARALLEL TO THE INITIAL         *
C*                       ORIENTATION OF BLADE NUMBER ONE WITH THE       *
C*                       DIRECTION FROM THE BLADE TIP TO THE CENTER     *
C*                       OF THE ROTOR SYSTEM TAKEN AS POSITIVE.         *
C*                                                                      *
C*                       THE Y2 AXIS IS PERPENDICULAR TO Y1 PASSING     *
C*                       THROUGH THE CENTER OF THE ROTOR SYSTEM AND     *
C*                       LYING IN THE ROTOR SYSTEM PLANE. THE           *
C*                       DIRECTION TO THE LEFT OF AN OBSERVER FACING    *
C*                       TOWARD POSITIVE Y1 IS TAKEN AS POSITIVE.       *
C*                                                                      *
C*                       THE Y3 AXIS IS PERPENDICULAR TO THE PLANE      *
C*                       OF THE ROTOR SYSTEM WITH THE POSITIVE          *
C*                       DIRECTION DETERMINED BY THE USUAL CONVENTION   *
C*                       FOR A RIGHT-HANDED COORDINATE SYSTEM.          *
C*                                                                      *
C*                 THE PRESENT VERSION OF FUNCTION FOFETA1 IMPOSES      *
C*                 A LIMITATION TO UNIFORM AIRFOIL SECTION ACROSS THE   *
C*                 SPAN OF THE BLADES. FOR NONUNIFORM AIRFOIL SECTION   *
C*                 THIS FUNCTION MUST BE MODIFIED.                      *
C*                                                                      *
C*                 THE PRESENT VERSION OF SUBROUTINE TEST IMPOSES A     *
C*                 LIMITATION TO ROTOR SYSTEMS WITH RECTANGULAR         *
C*                 BLADES. FOR OTHER BLADE SHAPES THIS SUBROUTINE       *
C*                 MUST BE MODIFIED.                                    *
C*                                                                      *
C*                                                                      *
C*     SOURCE:                                                          *
C*                 RTN WAS PROGRAMMED AT LANGLEY RESEARCH CENTER BY     *
C*                 COMPUTER SCIENCES CORPORATION BASED UPON A DESIGN    *
C*                 BY F. FARASSAT.                                      *
C*                                                                      *
C*                                                                      *
C*   LANGUAGE:                                                          *
C*                 FORTRAN                                              *
C*                                                                      *
C*                                                                      *
C*       DATE                                                           *
C*   RELEASED:                                                          *
C*                 MAY 1, 1974                                          *
C*                                                                      *
C*                                                                      *
C*     LATEST                                                           *
C*   REVISION:                                                          *
C*                 DECEMBER 16, 1975                                    *
C*                                                                      *
C*                                                                      *
C***********************************************************************
```

```
      LOGICAL MORDATA
      DIMENSION    BETA(8),        COEFFS(5),      ETA(2,8),
     1             X(3),           X0(3),          Y(3),
     2             YEND(2),        YH(3),          YSTART(2),
     3             VH(3)
      COMMON        ALFA,     BETA,     BLADES,   BLNTN,    CBETA,
     1              CH,       COEFFS,   DELTAU,   ETA,      ETA2,
     2              ETAMAX,   FAC,      FN,       FNS,      FETA1,
     3              IFLAG,    IREGION,  J2,       JFLG,     KFLG,
     4              NBLADES,  OMEGAR,   OTIME,    PI,       R,
     5              RM,       RMIN,     S,        SBETA,    SNDSPD,
     6              SRCDST,   SWPCST,   SWPTAU,   TAU,      THKRAT,
     7              TIMTHRU,  VH,       X,        X0,       YEND,
     8              YH,       YSTART,   DELETA1,  DELF
      NAMELIST/ROTOR/X0,NBLADES,R,CH,OMEGA,RO,THKRAT,DELTET,N,NS,KFLG,
     1 ETAMAX,BLNTN,JFLG,COEFFS,TAUINT,PERDM,DTAUM,SNDSPD,MORDATA
C
C        READ NAMELIST DATA
C
   10 READ(5,ROTOR)
      WRITE(12,ROTOR)
C
C        INITALIZE VARIABLES
C
      PI = 3.14159265358979
      OMEGAR = 2.*OMEGA*PI/60.
      TESTIM = 0.
      YH(1) = 0.
      YH(2) = 0.
      YH(3) = 0.
      Y(1) = 0.
      Y(2) = 0.
      Y(3) = 0.
      VH(1) = 0.
      VH(2) = 0.
      VH(3) = 0.
      ALFA = ASIN(0.5*CH/R)
      RMIN = R
      RSQ=R*R
      ROSQ=RO*RO
      TAU = TAUINT
      IF(KFLG.EQ.0) BLNTN = 1.
      NPTS=0
      PKEP1 = 0.
      PKEP2 = 0.
      SCRIP1 = 0.
      SCRIPTP=0.0
      BLADES = FLOAT(NBLADES)
C
C        COMPUTE SOURCE TIME DIFFERENTIAL INCREMENT AND MAXIMUM TIME
C          TERMINATION CRITERION
C
      DELTAU = DELTET/(6.*OMEGA)
      PERIOD = 60./(OMEGA*BLADES)
      TAUMAX = TAUINT + PERDM*PERIOD
C
C        S. N. 20 BEGINS A NEW DATA POINT
C
   20 IF(JFLG.NE.0) CALL NEWYH(TAU)
C
C        DETERMINE RELATIVE LOCATIONS OF ROTOR SYSTEM AND OBSERVER
C
      CALL OBSPOS
C
C        CONSTRUCT INITIAL SPHERE
C
      CALL RMOTIME
      DTIM = OTIME - TESTIM
C
```

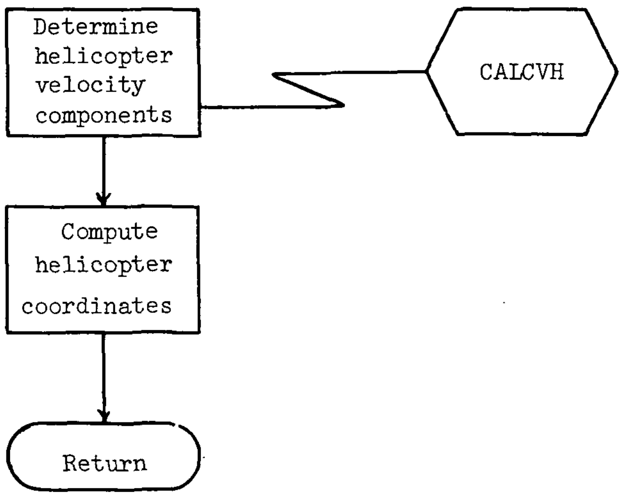
```
C        IF OBSERVER TIME IS DECREASING SKIP TO NEXT DATA POINT
C
      IF(DTIM.LE.0.) GO TO 40
      TESTIM = OTIME
      OBSANGL = ASIN(X(3)/SQRT(X(1)**2 + X(2)**2 + X(3)**2))
C
C        IF ANY TERMINATION CRITERION IS SATISFIED SKIP TO THE
C          CHECK FOR ADDITIONAL DATA
C
      IF(OBSANGL.GT.(PI/4.)) GO TO 60
      IF(RM.LT.RMIN) GO TO 60
      IF(TAU.GT.TAUMAX) GO TO 60
C
C        INITIALIZE SWEEP VARIABLES
C
      SWPTAU=TAU
      TIMTHRU=0.0
      IEVEN = -1
C
C        S. N. 30 BEGINS A NEW POSITION OF THE SPHERE
C
   30 TIMTHRU = TIMTHRU + 1.
      IEVEN = -IEVEN
      FN = N
      FNS = NS
      CALL CALCYST(RSQ,PHI0)
      PHI = PHI0
C
C        ADD TO CUMULATIVE SIMPSON SUM
C
      PKEP2 = PKEP1
      PKEP1 = SCRIP1
      PDELT = 2.*SCRIP1
      IF(IEVEN.GT.0) PDELT = 2.*PDELT
      IF(TIMTHRU.LT.2.5) PDELT = SCRIP1
      SCRIPTP = SCRIPTP + PDELT
C
C        IF IFLAG=0, WE ARE IN THE ROTOR SYSTEM
C
      IF(IFLAG.EQ.0) GO TO 50
C
C        IF IFLAG=2, A TERMINATION CRITERION WAS DETECTED IN CALCYST
C
      IF(IFLAG.EQ.2) GO TO 60
C
C        OTHERWISE, WE HAVE COMPLETED THIS DATA POINT. CORRECT THE
C          SIMPSON INTEGRATION AND OUTPUT THE RESULTS
C
      PDELT = -PKEP1
      IF(IEVEN.GT.0) PDELT = (PKEP2 - 5.*PKEP1)/2.
      SCRIPTP = SCRIPTP + PDELT
      SCRIPTP = SCRIPTP/3.
      NPTS=NPTS+1
      P = SCRIPTP*415./(2.*PI)
      WRITE(12,70) NPTS,TAU,OTIME,P
      SCRIP1 = 0.
      PKEP1 = 0.
      PKEP2 = 0.
      SCRIPTP=0.0
C
C        INCREASE THE TIME AND BEGIN A NEW DATA POINT
C
   40 TAU = TAU + DTAUM*DELTAU
      GO TO 20
C
C        WE ARE IN THE ROTOR SYSTEM. SWEEP THE ARC
```

```
C
   50 CALL CALCSP(RSQ,POSQ,PHI,Y,SCRIP1)
      GO TO 30
C
C        CHECK FOR ADDITIONAL DATA
C
   60 NPTS = 0
      WRITE(12,70) NPTS
      IF(MORDATA) GO TO 10
C
C        TERMINATION.
C
   70 FORMAT(I5,3E25.15)
      END
```

Subroutine NEWYH

This routine calculates YH, the three coordinates of the center of the rotor system relative to its initial location. This routine is called only for moving (translational) rotor systems.

Flow Chart of Subroutine NEWYH



APPENDIX A

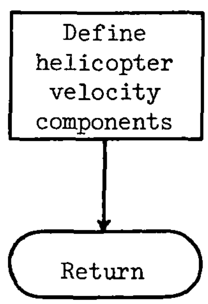
Listing of Subroutine NEWYH

```
      SUBROUTINE NEWYH(TIME)                                            NEWYH
C***********************************************************************NEWYH
C*                                                                     * NEWYH
C*      SUBROUTINE NEWYH IS CALLED BY PROGRAM RTN AND SUBROUTINE       * NEWYH
C*         CALCYST. THIS ROUTINE RETURNS YH(3), THE THREE              * NEWYH
C*         COORDINATES OF THE CENTER OF THE ROTOR SYSTEM RELATIVE      * NEWYH
C*         TO ITS INITIAL LOCATION (0,0,0). SUBROUTINE NEWYH IS        * NEWYH
C*         CALLED ONLY FOR MOVING (TRANSLATIONAL) ROTOR SYSTEMS TO     * NEWYH
C*         DETERMINE THE POSITION AT THE TIME SPECIFIED BY THE         * NEWYH
C*         VARIABLE TIME (M/SEC). THE ROTOR SYSTEM VELOCITY (POSSIBLY  * NEWYH
C*         TIME-DEPENDENT) IS OBTAINED FROM SUBROUTINE CALCVH.         * NEWYH
C*                                                                     * NEWYH
C***********************************************************************NEWYH
      DIMENSION      BETA(8),        COEFFS(5),       ETA(2,8),         NEWYH
     1               X(3),           X0(3),                             NEWYH
     2               YEND(2),        YH(3),           YSTART(2),        NEWYH
     3               VH(3)                                              NEWYH
      COMMON          ALFA,     BETA,     BLADES,   BLNTN,    CBETA,    NEWYH
     1                CH,       COEFFS,   DELTAU,   ETA,      ETA2,     NEWYH
     2                ETAMAX,   FAC,      FN,       FNS,      FETA1,    NEWYH
     3                IFLAG,    IREGION,  J2,       JFLG,     KFLG,     NEWYH
     4                NBLADES,  OMEGAR,   OTIME,    PI,       R,        NEWYH
     5                RM,       RMIN,     S,        SBETA,    SNDSPD,   NEWYH
     6                SRCDST,   SWPDST,   SWPTAU,   TAU,      THKRAT,   NEWYH
     7                TIMTHRU,  VH,       X,        X0,       YEND,     NEWYH
     8                YH,       YSTART,   DELETA1,  DELF                NEWYH
      CALL CALCVH(TIME)                                                 NEWYH
      DO 10 I=1,3                                                       NEWYH
      YH(I) = VH(I)*TIME                                                NEWYH
   10 CONTINUE                                                          NEWYH
      RETURN                                                            NEWYH
      END                                                               NEWYH
```

Subroutine CALCVH

CALCVH returns the three components of the translational velocity of the rotor system. In general, this subroutine is a user-supplied subroutine which must be provided for any case other than for that of a hovering helicopter. The routine included with the current version of RTN provides for a constant velocity of 61.73 m/sec (120 knots) directed along the Y_1-axis.

Flow Chart of Subroutine CALCVH



APPENDIX A

Listing of Subroutine CALCVH

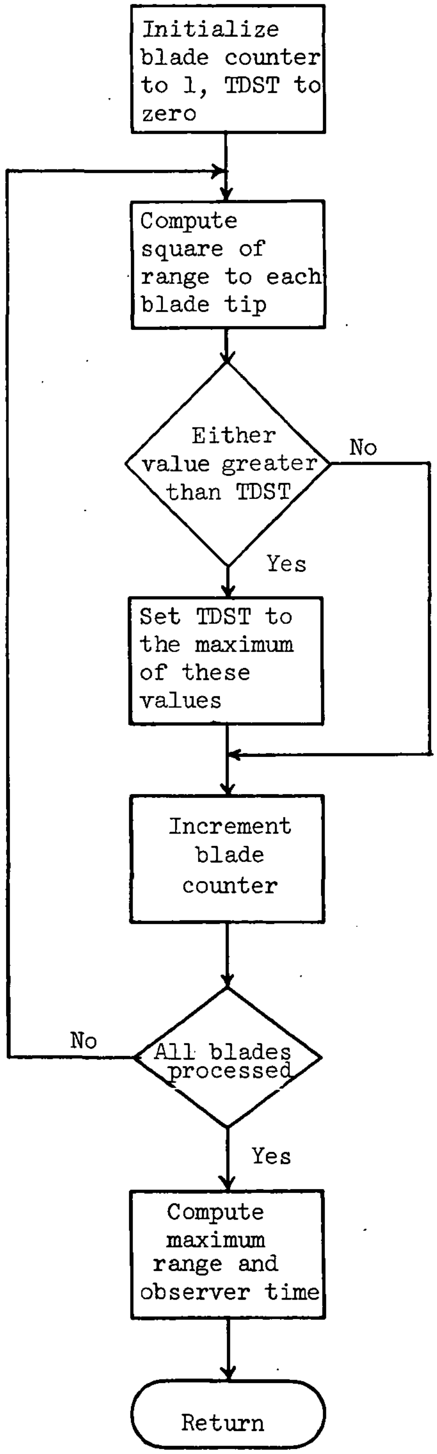
```
      SUBROUTINE CALCVH(TIME)                                           CALCVH
C*********************************************************************** CALCVH
C*                                                                    * CALCVH
C*      SUBROUTINE CALCVH IS CALLED BY SUBROUTINE NEWYH. CALCVH       * CALCVH
C*        RETURNS VH(3), THE THREE COMPONENTS OF THE VELOCITY         * CALCVH
C*        (TRANSLATIONAL) OF THE ROTOR SYSTEM. THE VARIABLE TIME      * CALCVH
C*        MAY BE USED TO SPECIFY A TIME-DEPENDENT VELOCITY.           * CALCVH
C*                                                                    * CALCVH
C*********************************************************************** CALCVH
      DIMENSION      BETA(8),         COEFFS(5),        ETA(2,8),       CALCVH
     1               X(3),            X0(3),                            CALCVH
     2               YEND(2),         YH(3),            YSTART(2),      CALCVH
     3               VH(3)                                              CALCVH
      COMMON          ALFA,     BETA,     BLADES,   BLNTN,    CBETA,    CALCVH
     1                CH,       COEFFS,   DELTAU,   ETA,      ETA2,     CALCVH
     2                ETAMAX,   FAC,      FN,       FNS,      FETA1,    CALCVH
     3                IFLAG,    IREGION,  J2,       JFLG,     KFLG,     CALCVH
     4                NBLADES,  OMEGAR,   OTIME,    PI,       R,        CALCVH
     5                RM,       RMIN,     S,        SBETA,    SNDSPD,   CALCVH
     6                SRCDST,   SWPOST,   SWPTAU,   TAU,      THKRAT,   CALCVH
     7                TIMTHRU,  VH,       X,        X0,       YEND,     CALCVH
     8                YH,       YSTART,   DELETA1,  DELF                CALCVH
      VH(1) = 61.73                                                     CALCVH
      VH(2)=0.0                                                         CALCVH
      VH(3)=0.0                                                         CALCVH
      RETURN                                                            CALCVH
      END                                                               CALCVH
```

Subroutine RMOTIME

This subroutine determines the range from the observer to the most distant blade tip in the rotor system. This range (RM) is the initial radius of the contracting sphere. Corresponding to this range and the current value of source time, the value of the observer time (OTIME) is also computed.

APPENDIX A

Flow Chart of Subroutine RMOTIME



APPENDIX A

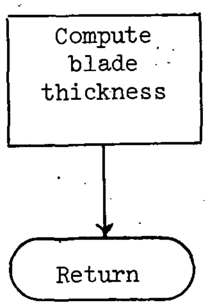
Listing of Subroutine RMOTIME

```
      SUBROUTINE RMOTIME                                                 RMOTIME
C*********************************************************************** RMOTIME
C*                                                                     * RMOTIME
C*       SUBROUTINE RMOTIME IS CALLED BY PROGRAM RTN. THIS ROUTINE     * RMOTIME
C*          DETERMINES THE RANGE TO THE MOST DISTANT TIP IN THE ROTOR  * RMOTIME
C*          SYSTEM, WHICH IS USED AS THE MAXIMUM SWEEPING DISTANCE     * RMOTIME
C*          FOR A GIVEN TAU. THE OBSERVERS TIME (CORRESPONDING TO TAU) * RMOTIME
C*          IS ALSO RETURNED.                                          * RMOTIME
C*                                                                     * RMOTIME
C*********************************************************************** RMOTIME
      DIMENSION     BETA(8),       COEFFS(5),      ETA(2,8),             RMOTIME
     1              X(3),          XO(3),                                RMOTIME
     2              YEND(2),       YH(3),          YSTART(2),            RMOTIME
     3              VH(3)                                                RMOTIME
      COMMON         ALFA,     BETA,     BLADES,   BLNTN,    CBETA,      RMOTIME
     1               CH,       COEFFS,   DELTAU,   ETA,      ETA2,       RMOTIME
     2               ETAMAX,   FAC,      FN,       FNS,      FETA1,      RMOTIME
     3               IFLAG,    IREGION,  J2,       JFLG,     KFLG,       RMOTIME
     4               NBLADES,  OMEGAR,   OTIME,    PI,       R,          RMOTIME
     5               RM,       RMIN,     S,        SBETA,    SNDSPD,     RMOTIME
     6               SRCDST,   SWPDST,   SWPTAU,   TAU,      THKRAT,     RMOTIME
     7               TIMTHRU,  VH,       X,        XO,       YEND,       RMOTIME
     8               YH,       YSTART,   DELETA1,  DELF                  RMOTIME
      DIST(A,B,C,D,E) = (A - D*COS(E))**2 + (B - D*SIN(E))**2 + C*C     RMOTIME
      TDST = 0.                                                          RMOTIME
      XBETA = OMEGAR*TAU                                                 RMOTIME
      DO 10 I=1,NBLADES                                                  RMOTIME
      TIP1 = XBETA + PI*(1. + 2.*FLOAT(I-1)/BLADES) - ALFA               RMOTIME
      TIP2 = TIP1 + 2.*ALFA                                              RMOTIME
      DI = DIST(X(1),X(2),X(3),R,TIP1)                                   RMOTIME
      IF(DI.GE.TDST) TDST = DI                                           RMOTIME
      DI = DIST(X(1),X(2),X(3),R,TIP2)                                   RMOTIME
      IF(DI.GE.TDST) TDST = DI                                           RMOTIME
   10 CONTINUE                                                           RMOTIME
      RM = SQRT(TDST)                                                    RMOTIME
      OTIME = TAU + RM/SNDSPD                                            RMOTIME
      RETURN                                                             RMOTIME
      END                                                                RMOTIME
```

Function FOFETA1

This subprogram computes the blade thickness for specified distance along the blade chord. The function included with the current version of RTN is limited to uniform airfoil section across the span.

Flow Chart of Function FOFETAL



APPENDIX A

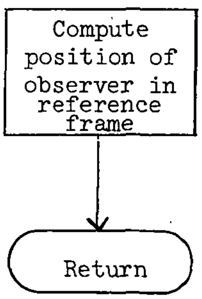
Listing of Function FOFETA1

```
      FUNCTION FOFETA1(XETA)                                                    FOFETA1
C************************************************************************* FOFETA1
C*                                                                       * FOFETA1
C*       FUNCTION FOFETA1 COMPUTES THE BLADE THICKNESS FOR SPECIFIED     * FOFETA1
C*          DISTANCE ALONG THE BLADE CHORD                               * FOFETA1
C*                                                                       * FOFETA1
C************************************************************************* FOFETA1
      DIMENSION     BETA(8),         COEFFS(5),        ETA(2,8),                FOFETA1
     1              X(3),            XO(3),                                     FOFETA1
     2              YEND(2),         YH(3),            YSTART(2),               FOFETA1
     3              VH(3)                                                       FOFETA1
      COMMON         ALFA,      BETA,      BLADES,    BLNTN,     CBETA,         FOFETA1
     1               CH,        COEFFS,    DELTAU,    ETA,       ETA2,          FOFETA1
     2               ETAMAX,    FAC,       FN,        FNS,       FETA1,         FOFETA1
     3               IFLAG,     IREGION,   J2,        JFLG,      KFLG,          FOFETA1
     4               NBLADES,   OMEGAR,    OTIME,     PI,        R,             FOFETA1
     5               RM,        RMIN,      S,         SBETA,     SNDSPD,        FOFETA1
     6               SRCOST,    SWPDST,    SWPTAU,    TAU,       THKRAT,        FOFETA1
     7               TIMTHRU,   VH,        X,         XO,        YEND,          FOFETA1
     8               YH,        YSTART,    DELETA1,   DELF                      FOFETA1
      E = XETA/CH                                                               FOFETA1
      A = COEFFS(1)*SQRT(E)                                                     FOFETA1
      A1 = COEFFS(5)*E + COEFFS(4)                                              FOFETA1
      A1 = A1*E + COEFFS(3)                                                     FOFETA1
      A1 = (A1*E + COEFFS(2))*E                                                 FOFETA1
      FOFETA1 = (A + A1)*THKRAT                                                 FOFETA1
   10 RETURN                                                                    FOFETA1
      END                                                                       FOFETA1
```

Subroutine OBSPOS

This subroutine returns X, the three coordinates of the observer relative to the YH-frame fixed to the helicopter.

Flow Chart of Subroutine OBSPOS



APPENDIX A

Listing of Subroutine OBSPOS

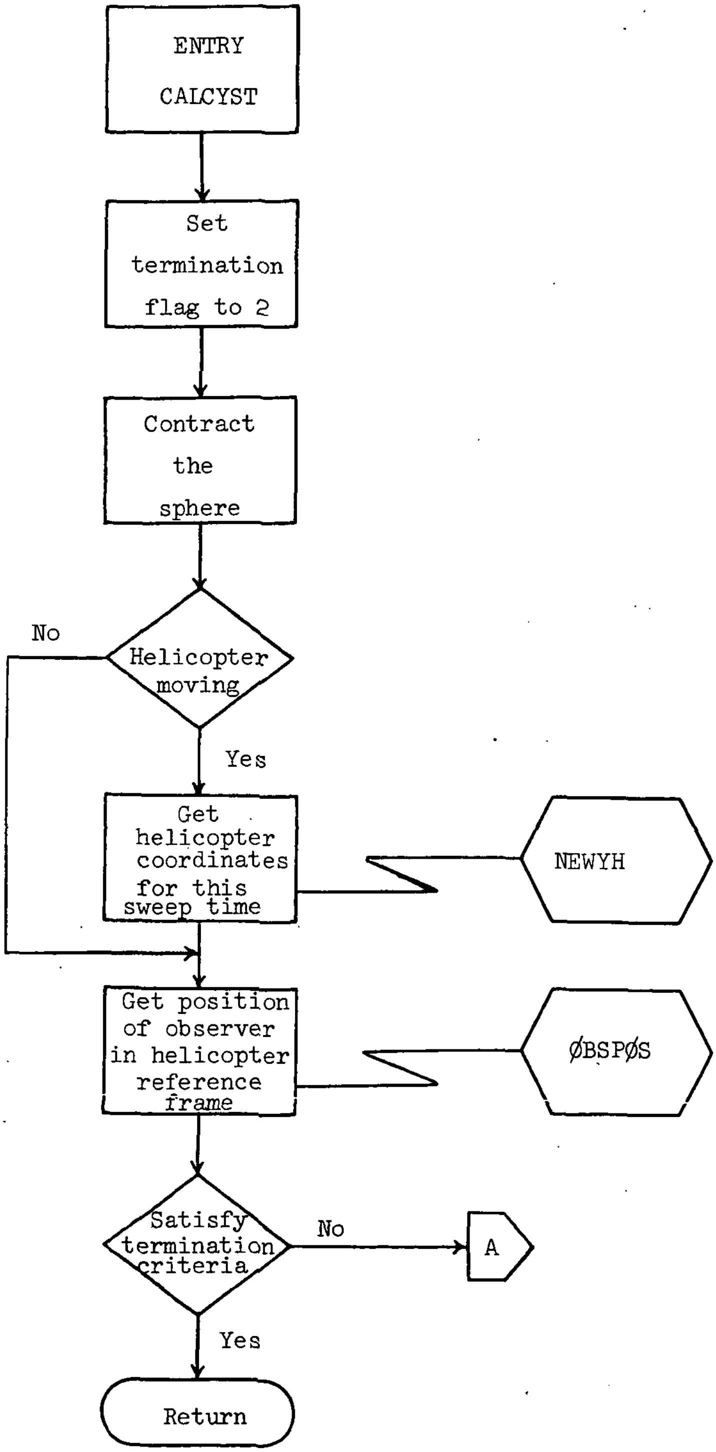
```
      SUBROUTINE OBSPOS                                                 OBSPOS
C*********************************************************************** OBSPOS
C*                                                                    * OBSPOS
C*      SUBROUTINE OBSPOS IS CALLED BY PROGRAM RTN AND BY SUBROUTINE  * OBSPOS
C*        CALCYST. THIS ROUTINE RETURNS X(3),  THE THREE              * OBSPOS
C*        COORDINATES OF THE OBSERVER IN THE CURRENT ROTOR SYSTEM     * OBSPOS
C*        REFERENCE FRAME.                                            * OBSPOS
C*                                                                    * OBSPOS
C*********************************************************************** OBSPOS
      DIMENSION    BETA(8),        COEFFS(5),      ETA(2,8),            OBSPOS
     1             X(3),           X0(3),                               OBSPOS
     2             YEND(2),        YH(3),          YSTART(2),           OBSPOS
     3             VH(3)                                                OBSPOS
      COMMON       ALFA,     BETA,      BLADES,    BLNTN,     CBETA,    OBSPOS
     1             CH,       COEFFS,    DELTAU,    ETA,       ETA2,     OBSPOS
     2             ETAMAX,   FAC,       FN,        FNS,       FETA1,    OBSPOS
     3             IFLAG,    IREGION,   J2,        JFLG,      KFLG,     OBSPOS
     4             NBLADES,  OMEGAR,    OTIME,     PI,        R,        OBSPOS
     5             RM,       RMIN,      S,         SBETA,     SNDSPD,   OBSPOS
     6             SRCOST,   SWPDST,    SWPTAU,    TAU,       THKRAT,   OBSPOS
     7             TIMTHRU,  VH,        X,         X0,        YEND,     OBSPOS
     8             YH,       YSTART,    DELETA1,   DELF                 OBSPOS
      X(1)=X0(1)-YH(1)                                                  OBSPOS
      X(2)=X0(2)-YH(2)                                                  OBSPOS
      X(3)=X0(3)-YH(3)                                                  OBSPOS
      RETURN                                                            OBSPOS
      END                                                               OBSPOS
```

Subroutine CALCYST

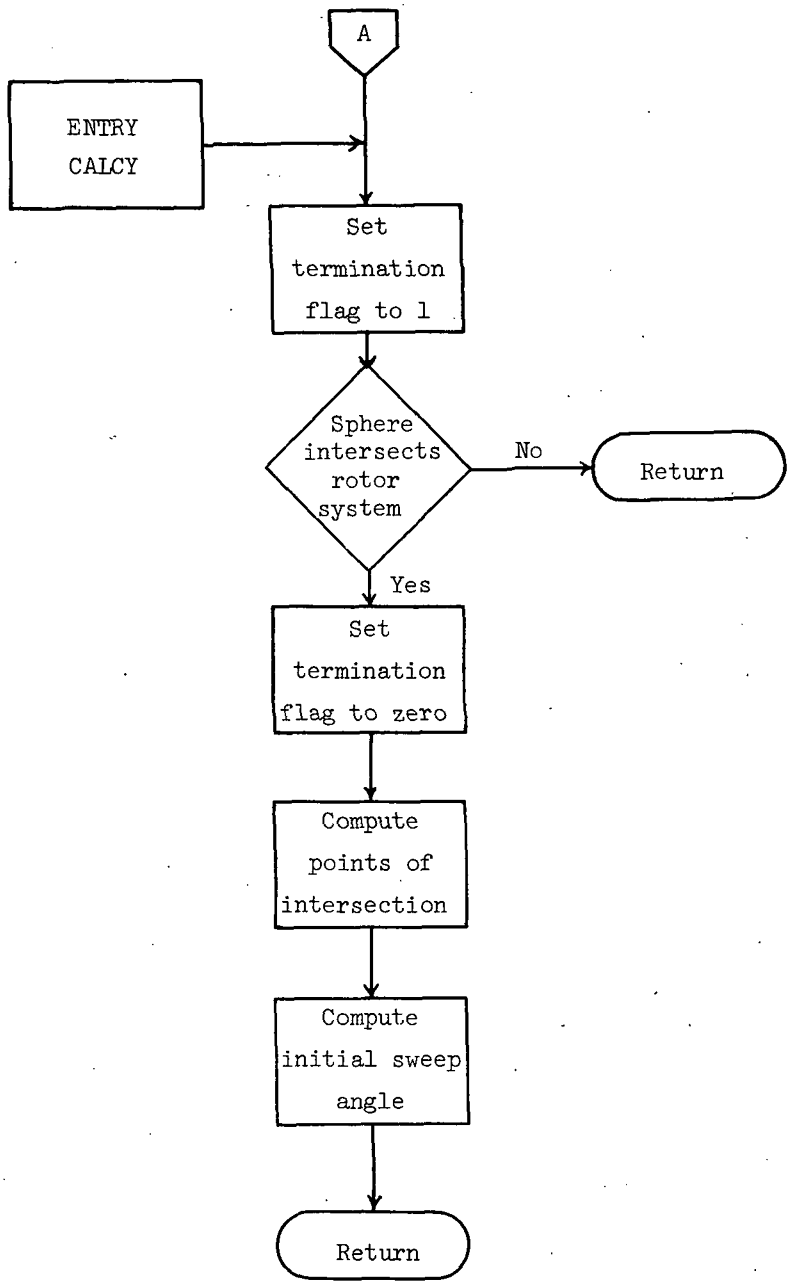
Subroutine CALCYST is called by RTN. A second entry point, CALCY, is used by subroutine CALCSP. This subprogram calculates the points of intersection of the projection of the sphere on the rotor system plane with the circles(s) defined by the rotor system. The calculation is made for either the circle defined by the outer blade radius or for that defined by the hub radius. The two components of the more clockwise point of intersection (relative to the observer) are returned in YSTART. The more counterclockwise point is returned in YEND. The angle (at the observer's position) defined by YSTART is returned in PHIO. The CALCYST entry contracts the sphere. The CALCY entry does not.

APPENDIX A

Flow Chart of Subroutine CALCYST



APPENDIX A


A
ENTRY CALCY
Set termination flag to 1
Sphere intersects rotor system
No
Return
Yes
Set termination flag to zero
Compute points of intersection
Compute initial sweep angle
Return

APPENDIX A

Listing of Subroutine CALCYST

```
      SUBROUTINE CALCYST(RSQ,PHIO)                                      CALCYST
C********************************************************************** CALCYST
C*                                                                    * CALCYST
C*      SUBROUTINE CALCYST IS CALLED BY PROGRAM RTN. A SECOND ENTRY   * CALCYST
C*        POINT, CALCY, IS USED BY SUBROUTINE CALCSP. THIS ROUTINE    * CALCYST
C*        CALCULATES THE POINTS OF INTERSECTION OF THE PROJECTION     * CALCYST
C*        OF THE CONTRACTING SPHERE ON THE ROTOR SYSTEM PLANE WITH    * CALCYST
C*        THE CIRCLE(S) DEFINED BY THE ROTOR SYSTEM. WHEN THE ROUTINE * CALCYST
C*        IS ENTERED WITH RSQ EQUAL TO THE OUTER BLADE RADIUS, THE    * CALCYST
C*        OUTER CIRCLE IS USED. WHEN ENTERED WITH RSQ EQUAL TO THE    * CALCYST
C*        INNER BLADE RADIUS, THE INNER CIRCLE IS USED. THE X AND Y   * CALCYST
C*        COMPONENTS OF THE POINT OF INTERSECTION MORE CLOCKWISE      * CALCYST
C*        (RELATIVE TO THE OBSERVER) ARE RETURNED IN YSTART(2). THE   * CALCYST
C*        MORE COUNTERCLOCKWISE POINT IS RETURNED IN YEND(2). THE     * CALCYST
C*        ANGLE (AT THE OBSERVERS POSITION) DEFINED BY YSTART IS      * CALCYST
C*        RETURNED IN PHIO. THE CALCYST ENTRY CONTRACTS THE SPHERE.   * CALCYST
C*        THE CALCY ENTRY DOES NOT.                                   * CALCYST
C*                                                                    * CALCYST
C********************************************************************** CALCYST
      DIMENSION     BETA(8),         COEFFS(5),       ETA(2,8),         CALCYST
     1              X(3),            XO(3),                             CALCYST
     2              YEND(2),         YH(3),           YSTART(2),        CALCYST
     3              VH(3)                                               CALCYST
      COMMON        ALFA,      BETA,      BLADES,    BLNTN,     CBETA,  CALCYST
     1              CH,        COEFFS,    DELTAU,    ETA,       ETA2,   CALCYST
     2              ETAMAX,    FAC,       FN,        FNS,       FETA1,  CALCYST
     3              IFLAG,     IREGION,   J2,        JFLG,      KFLG,   CALCYST
     4              NBLADES,   OMEGAR,    OTIME,     PI,        R,      CALCYST
     5              RM,        RMIN,      S,         SBETA,     SNDSPD, CALCYST
     6              SRCDST,    SWPDST,    SWPTAU,    TAU,       THKRAT, CALCYST
     7              TIMTHRU,   VH,        X,         XO,        YEND,   CALCYST
     8              YH,        YSTART,    DELETA1,   DELF               CALCYST
      IFLAG = 2                                                         CALCYST
      SWPTAU = SWPTAU + DELTAU                                          CALCYST
C                                                                       CALCYST
C        CONTRACT THE SPHERE                                            CALCYST
C                                                                       CALCYST
      SWPDST = RM - SNDSPD*TIMTHRU*DELTAU                               CALCYST
      IF(JFLG.NE.0) CALL NEWYH(SWPTAU)                                  CALCYST
      CALL OBSPOS                                                       CALCYST
      OBSANGL = ASIN(X(3)/SQRT(X(1)**2 + X(2)**2 + X(3)**2))            CALCYST
C                                                                       CALCYST
C        TEST TERMINATION CRITERIA                                      CALCYST
C                                                                       CALCYST
      IF(OBSANGL.GT.(PI/4.)) GO TO 10                                   CALCYST
      IF(RM.LT.RMIN) GO TO 10                                           CALCYST
      ENTRY CALCY                                                       CALCYST
      IFLAG = 1                                                         CALCYST
      SURDSQ = X(1)*X(1) + X(2)*X(2)                                    CALCYST
      OBSDSQ = SURDSQ + X(3)*X(3)                                       CALCYST
      DELTA = 0.5*(OBSDSQ + RSQ - SWPDST*SWPDST)                        CALCYST
      DISC = RSQ*SURDSQ - DELTA*DELTA                                   CALCYST
      IF(DISC.LE.0.) GO TO 10                                           CALCYST
      IFLAG = 0                                                         CALCYST
C                                                                       CALCYST
C        COMPUTE INTERSECTIONS AND INITIAL SWEEP ANGLE                  CALCYST
C                                                                       CALCYST
      FAC = 1.0                                                         CALCYST
      IF(X(1).LT.0.) FAC = -1.0                                         CALCYST
      IF(ABS(X(1)).LT.R) GO TO 5                                        CALCYST
      YSTART(2) = (X(2)*DELTA + X(1)*SQRT(DISC))/SURDSQ                 CALCYST
      YSTART(1) = (DELTA - X(2)*YSTART(2))/X(1)                         CALCYST
      YEND(2) = (X(2)*DELTA - X(1)*SQRT(DISC))/SURDSQ                   CALCYST
      YEND(1) = (DELTA - X(2)*YEND(2))/X(1)                             CALCYST
      GO TO 7                                                           CALCYST
    5 YSTART(1) = (X(1)*DELTA - FAC*X(2)*DISC)/SURDSQ                   CALCYST
      YSTART(2) = (DELTA - X(1)*YSTART(1))/X(2)                         CALCYST
      YEND(1) = (X(1)*DELTA + FAC*X(2)*DISC)/SURDSQ                     CALCYST
      YEND(2) = (DELTA - X(1)*YEND(1))/X(2)                             CALCYST
    7 SPCDST = SQRT(SWPDST**2 - X(3)**2)                                CALCYST
      PHIO = ACOS(0.9999999999*(YSTART(1) - X(1))/SPCDST)               CALCYST
      IF(YSTART(2).LT.X(2)) PHIO = 2.*PI - PHIO                         CALCYST
   10 RETURN                                                            CALCYST
      END                                                               CALCYST
```

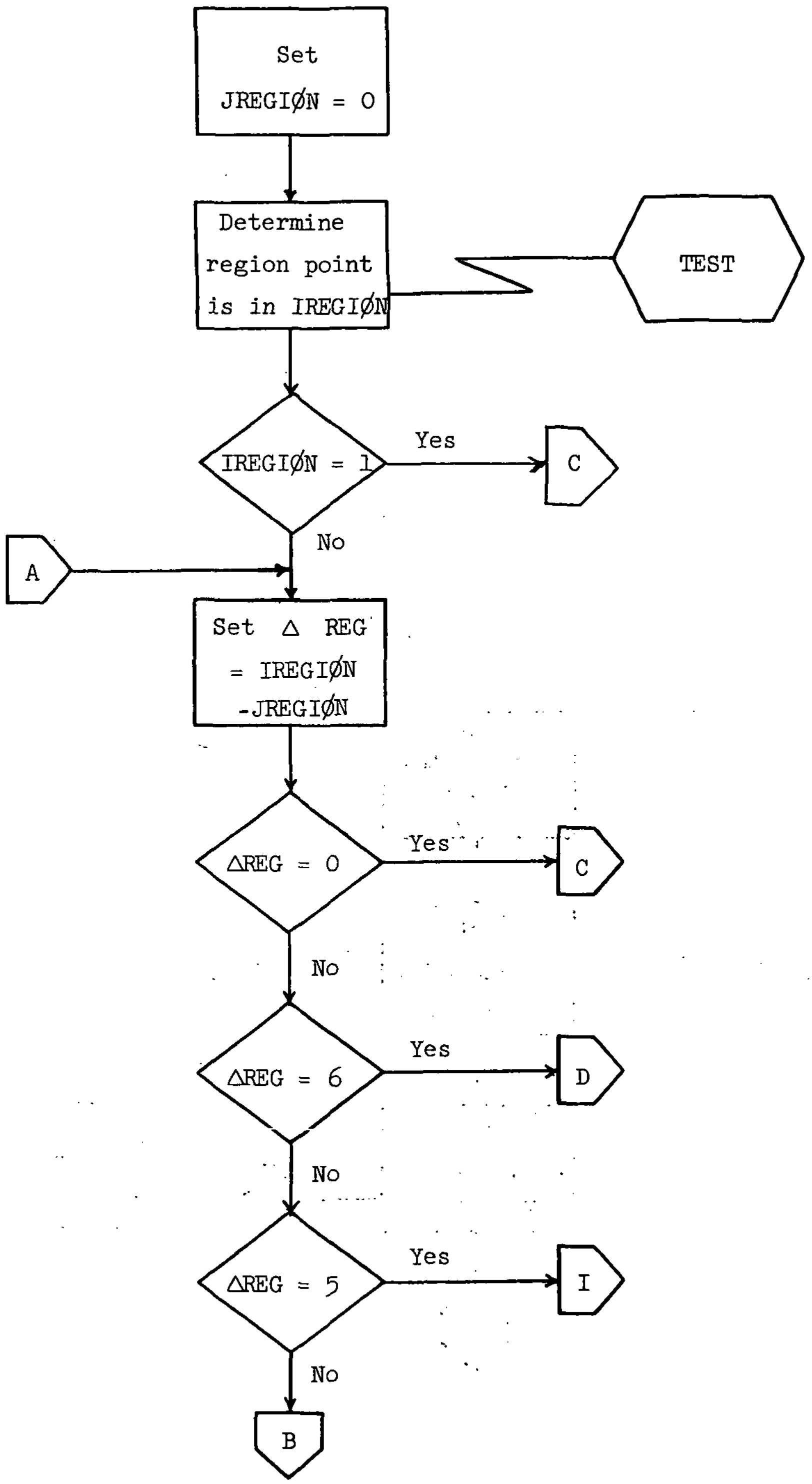
APPENDIX A

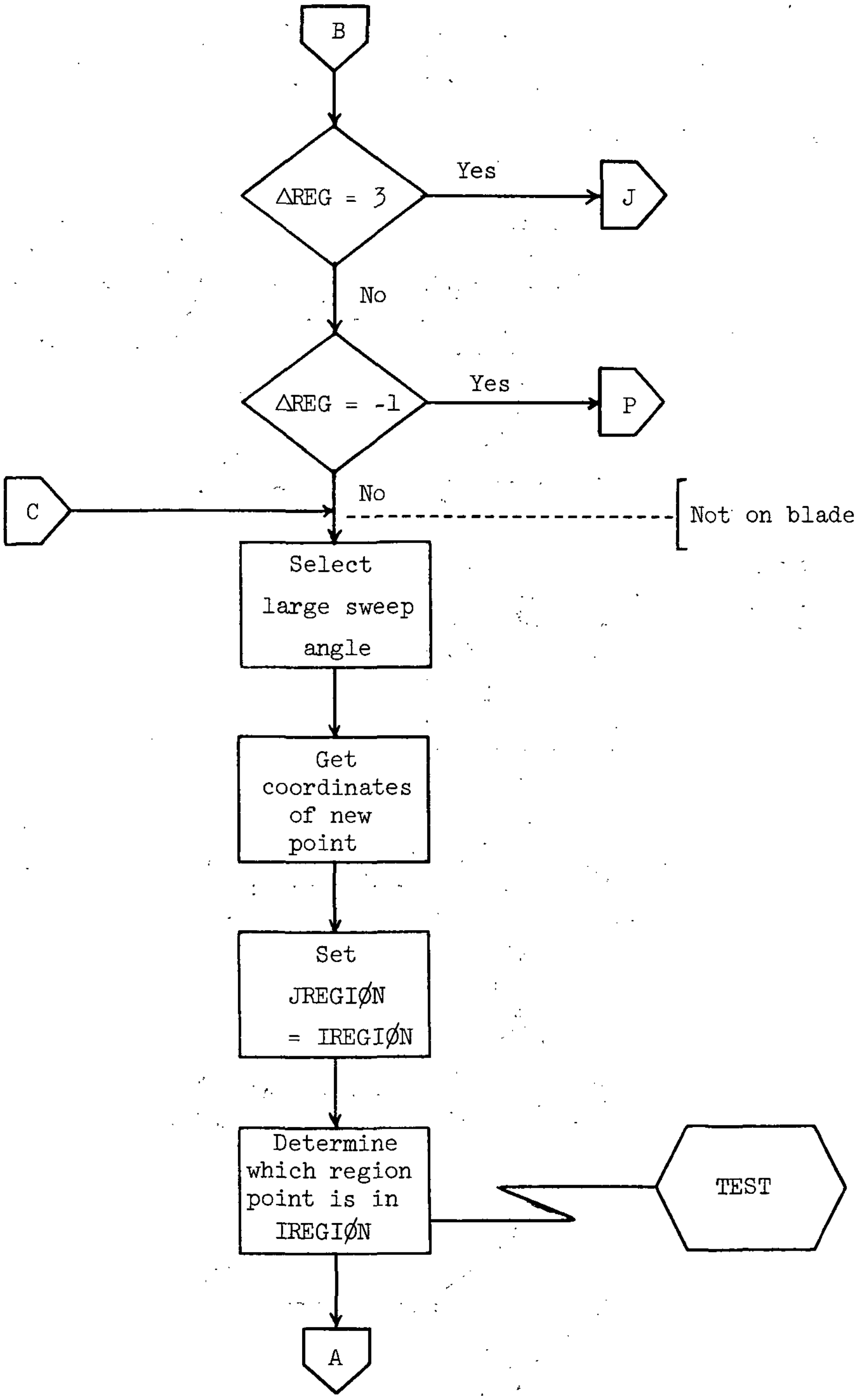
Subroutine CALCSP

This routine controls the sweeping across the arc defined by a fixed position of the contracting sphere. The sweep is in a counterclockwise direction beginning at PHI0 and terminating when a point is reached outside the circle defined by the rotor system. A trapezoidal integration over the intersection of the arc with each blade is performed dynamically. The coarseness of the sweep increment is determined as a function of whether or not a point is on a blade. Even finer sweep increments are used near the leading edge of blades with blunt leading edges.

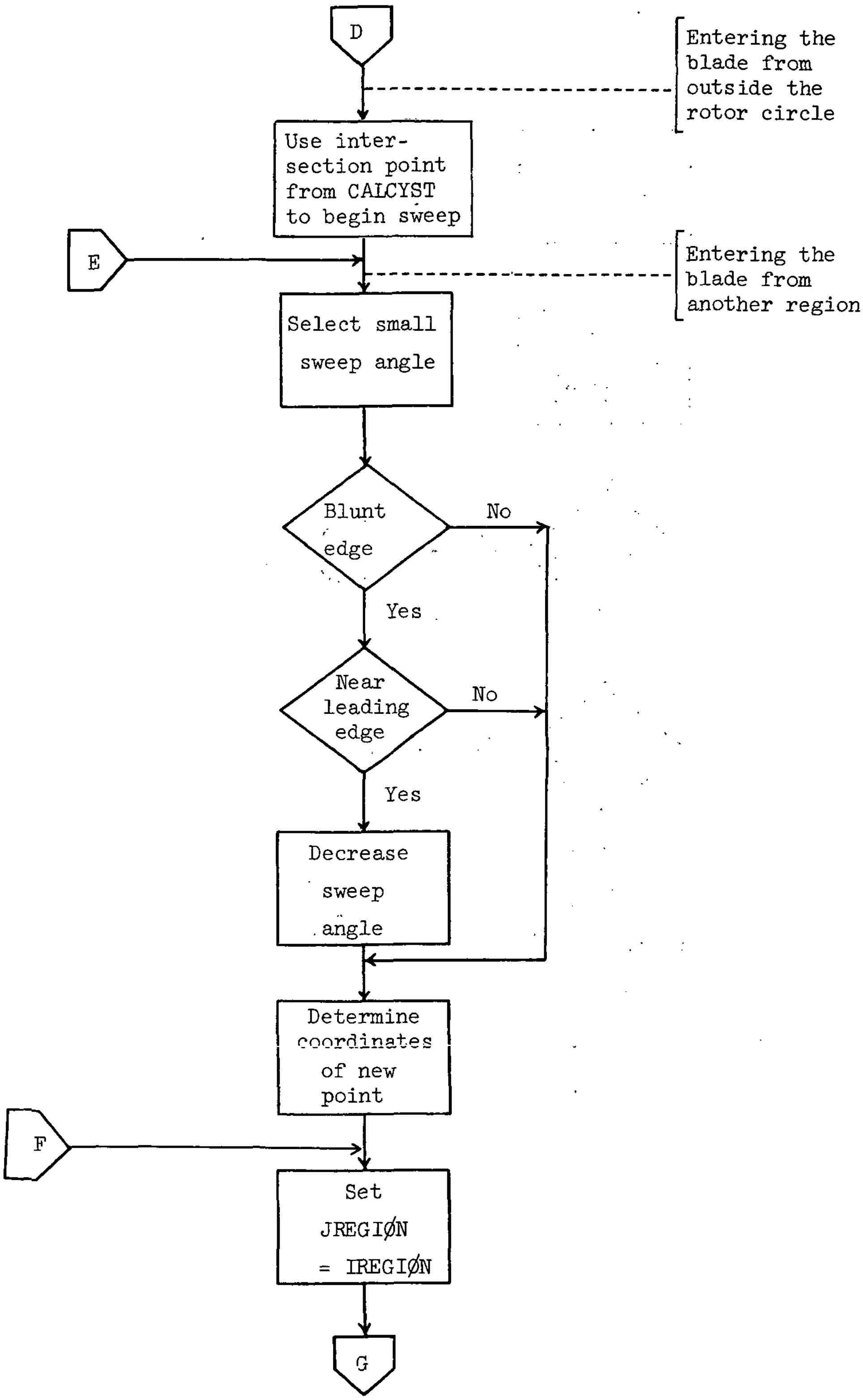
APPENDIX A

Flow Chart of Subroutine CALCSP

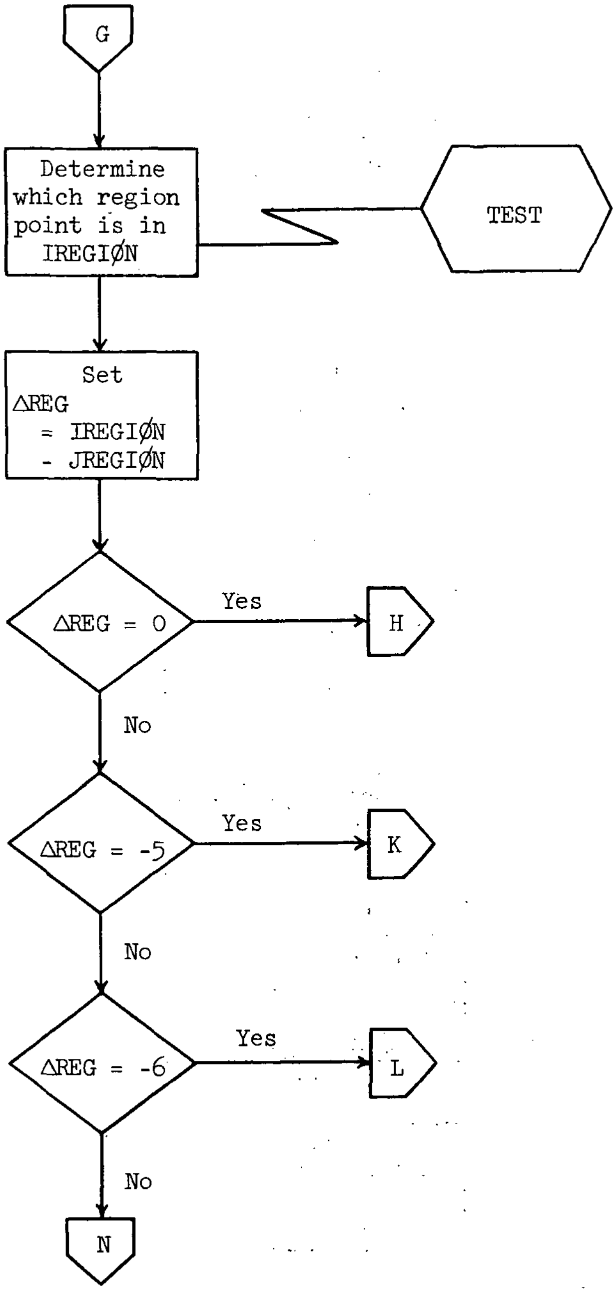



B
ΔREG = 3
Yes
J
No
ΔREG = -1
Yes
P
C
No
Not on blade
Select large sweep angle
Get coordinates of new point
Set JREGIØN = IREGIØN
Determine which region point is in IREGIØN
TEST
A

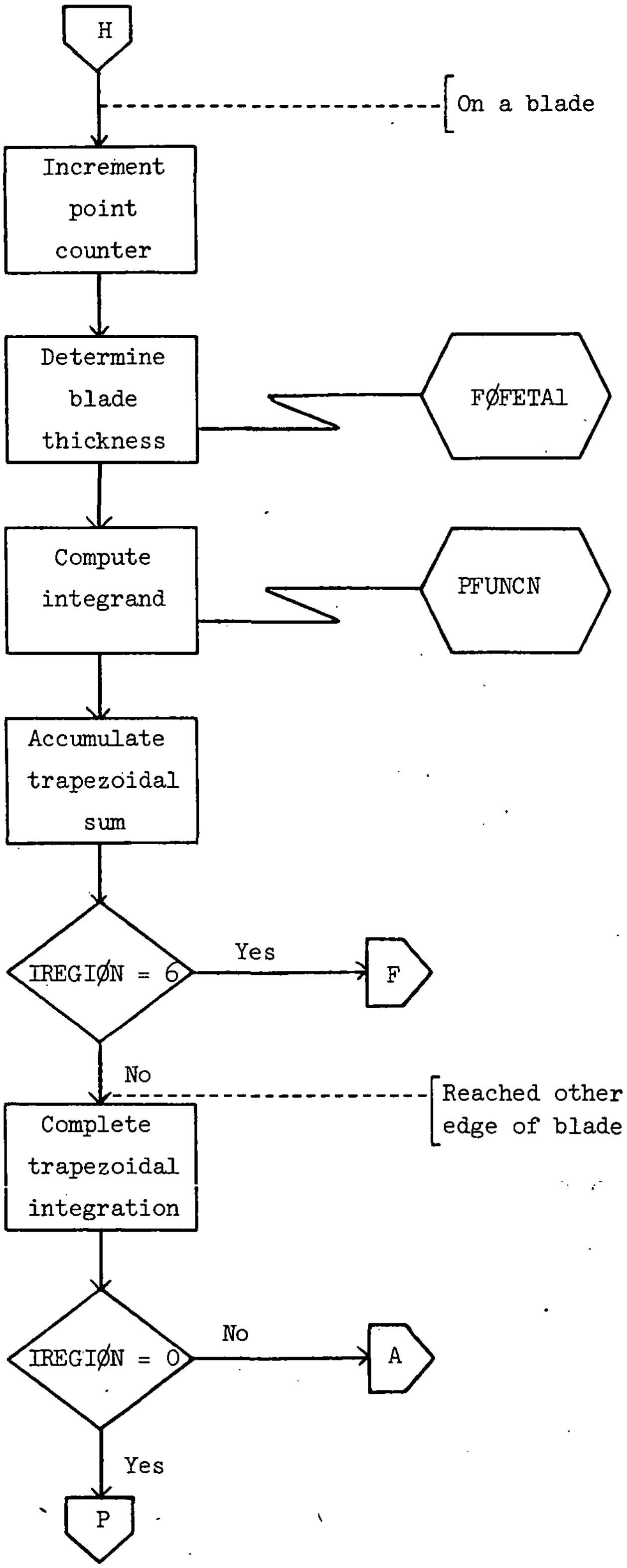
APPENDIX A


D
Entering the blade from outside the rotor circle
Use inter-section point from CALCYST to begin sweep
E
Entering the blade from another region
Select small sweep angle
Blunt edge
No
Yes
Near leading edge
No
Yes
Decrease sweep angle
Determine coordinates of new point
F
Set JREGIØN = IREGIØN
G

APPENDIX A

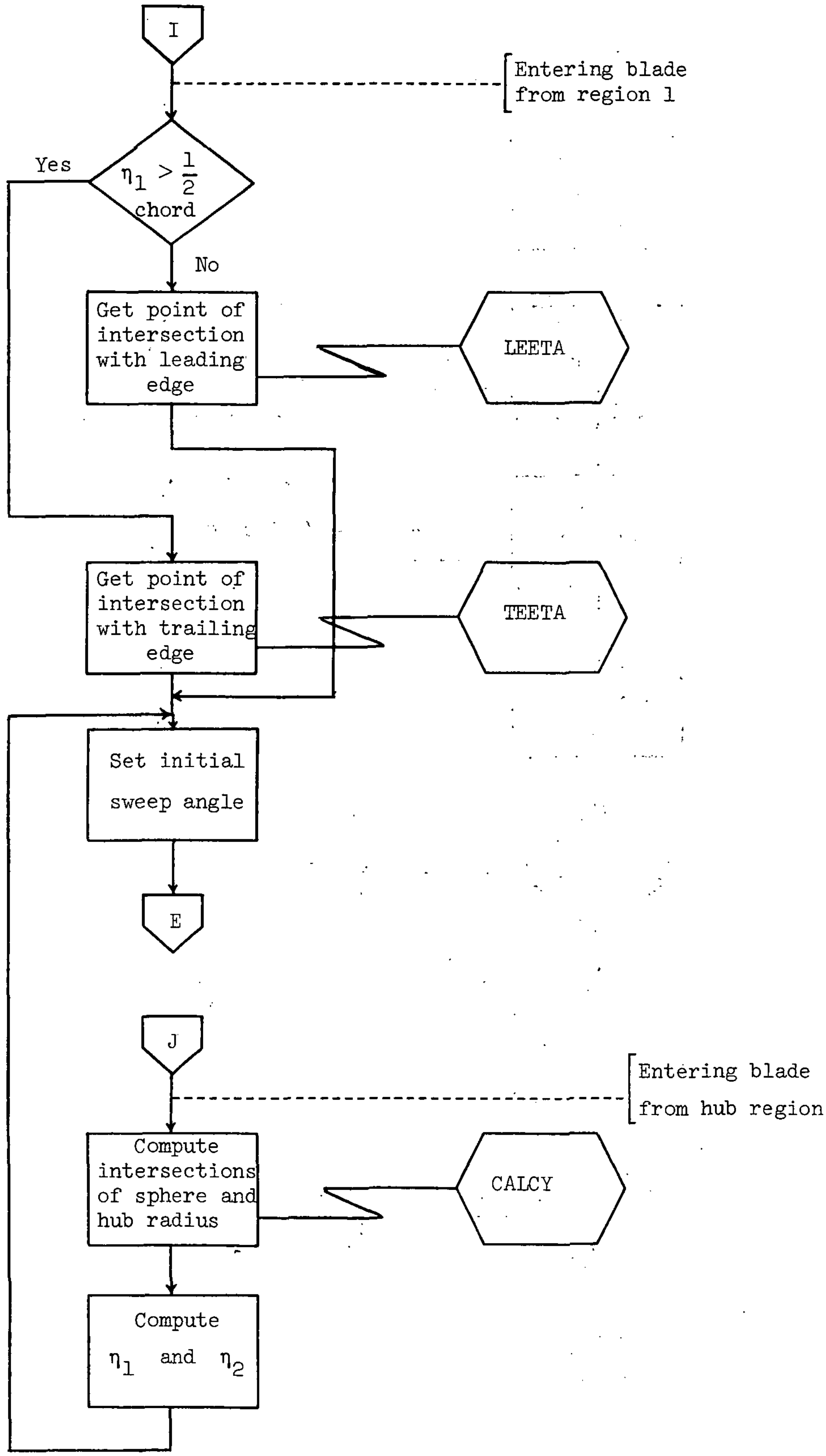

G
Determine which region point is in IREGIØN
TEST
Set ΔREG = IREGIØN - JREGIØN
ΔREG = 0
Yes
H
No
ΔREG = -5
Yes
K
No
ΔREG = -6
Yes
L
No
N

APPENDIX A

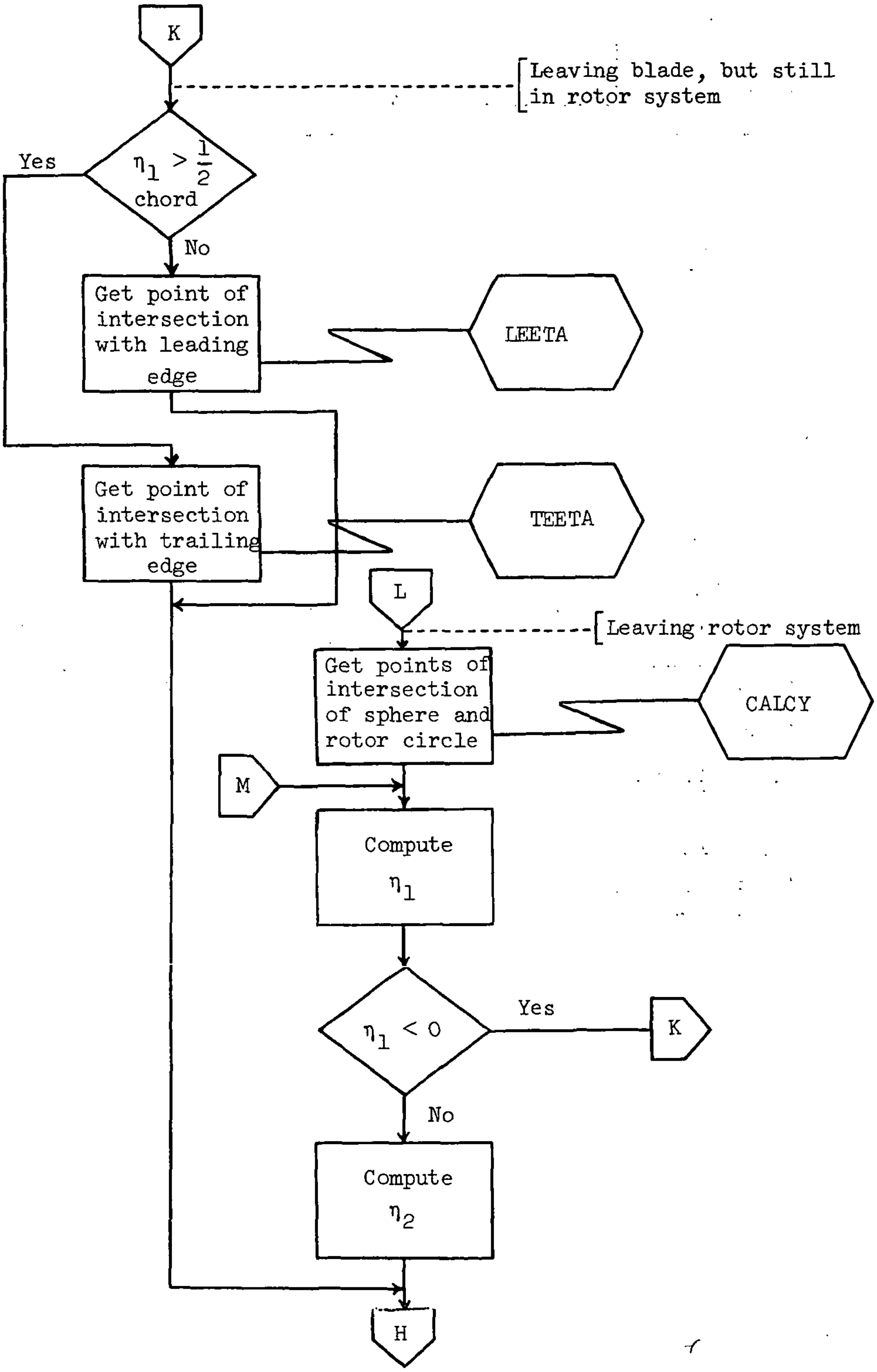


H
On a blade
Increment point counter
Determine blade thickness
FØFETA1
Compute integrand
PFUNCN
Accumulate trapezoidal sum
IREGIØN = 6
Yes
F
No
Reached other edge of blade
Complete trapezoidal integration
IREGIØN = 0
No
A
Yes
P

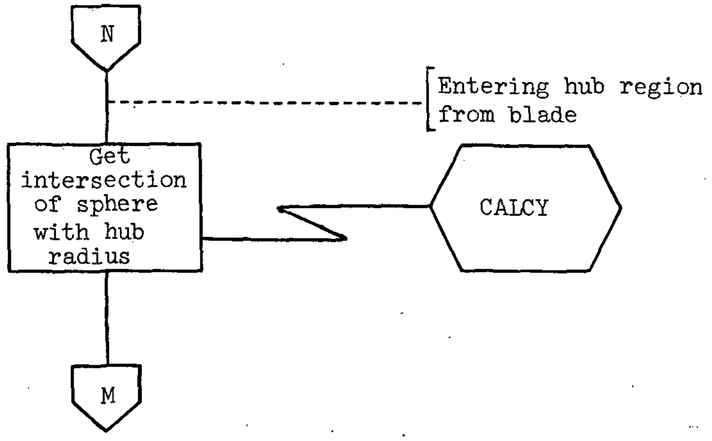
APPENDIX A

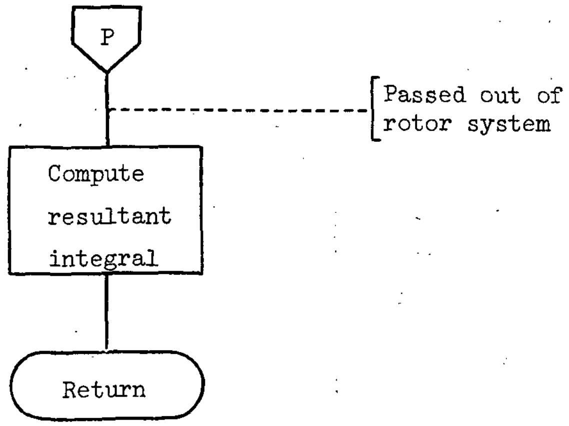

I
Entering blade from region 1
$\eta_1 > \frac{1}{2}$ chord
Yes
No
Get point of intersection with leading edge
LEETA
Get point of intersection with trailing edge
TEETA
Set initial sweep angle
E
J
Entering blade from hub region
Compute intersections of sphere and hub radius
CALCY
Compute η_1 and η_2

APPENDIX A


K
Leaving blade, but still in rotor system
$\eta_1 > \frac{1}{2}$ chord
Yes
No
Get point of intersection with leading edge
LEETA
Get point of intersection with trailing edge
TEETA
L
Leaving rotor system
Get points of intersection of sphere and rotor circle
CALCY
M
Compute η_1
$\eta_1 < 0$
Yes
K
No
Compute η_2
H

APPENDIX A


N
Entering hub region from blade
Get intersection of sphere with hub radius
CALCY
M


P
Passed out of rotor system
Compute resultant integral
Return

APPENDIX A

Listing of Subroutine CALCSP

```
      SUBROUTINE CALCSP(RSQ,ROSQ,PHI,Y,SCRIPTP)                         CALCSP
C*********************************************************************** CALCSP
C*                                                                     * CALCSP
C*      SUBROUTINE CALCSP IS CALLED BY PROGRAM RTN. THIS ROUTINE       * CALCSP
C*        CONTROLS THE SWEEPING (COUNTERCLOCKWISE) ACROSS THE ARC      * CALCSP
C*        DEFINED BY A FIXED POSITION OF THE CONTRACTING SPHERE.       * CALCSP
C*        TWO SWEEPING INCREMENTS ARE USED: DELPHI2 FOR SWEEPING       * CALCSP
C*        ACROSS A ROTOR BLADE AND DELPHI1 FOR THE SPACE BETWEEN       * CALCSP
C*        BLADES, AS DETERMINED BY SUBROUTINE TEST. THE LINE INTEGRAL  * CALCSP
C*        REPRESENTING THE CONTRIBUTION OF EACH BLADE IS A             * CALCSP
C*        TRAPEZOIDAL INTEGRATION. POINTS ON THE EDGE OF ANY BLADE     * CALCSP
C*        ARE OBTAINED FROM SUBROUTINES LEFTA AND TEETA.               * CALCSP
C*                                                                     * CALCSP
C*********************************************************************** CALCSP
      DIMENSION     BETA(8),        COEFFS(5),      ETA(2,8),           CALCSP
     1              X(3),           X0(3),          Y(3),               CALCSP
     2              YEND(2),        YH(3),          YLST(3),            CALCSP
     3              YSTART(2),      VH(3)                               CALCSP
      COMMON         ALFA,     BETA,     BLADES,   BLNTN,    CBETA,     CALCSP
     1               CH,       COEFFS,   DELTAU,   ETA,      ETA2,      CALCSP
     2               ETAMAX,   FAC,      FN,       FNS,      FETA1,     CALCSP
     3               IFLAG,    IREGION,  J2,       JFLG,     KFLG,      CALCSP
     4               NBLADES,  OMEGAR,   OTIME,    PI,       R,         CALCSP
     5               RM,       RMIN,     S,        SBETA,    SNDSPD,    CALCSP
     6               SRCOST,   SWPDST,   SWPTAU,   TAU,      THKRAT,    CALCSP
     7               TIMTHRU,  VH,       X,        X0,       YEND,      CALCSP
     8               YH,       YSTART,   DELETA1,  DELF                 CALCSP
C                                                                       CALCSP
C        INITIALIZE VARIABLES FOR THIS SWEEP                            CALCSP
C                                                                       CALCSP
      DELPHI1 = CH/(FN*SRCOST)                                          CALCSP
      DELPHI2 = CH/(FNS*SRCOST)                                         CALCSP
      DELETA1 = 0.                                                      CALCSP
      DELF = 0.                                                         CALCSP
      YLST(3) = 0.                                                      CALCSP
      JREGION = 0                                                       CALCSP
      SUMSP = 0.                                                        CALCSP
      DGLST = 0.                                                        CALCSP
      SCRIPTP = 0.                                                      CALCSP
      J = 1                                                             CALCSP
      J2 = 1                                                            CALCSP
C                                                                       CALCSP
C        DETERMINE WHICH REGION WE ARE IN                               CALCSP
C                                                                       CALCSP
      CALL TEST(YSTART,RSQ,ROSQ,SWPTAU)                                 CALCSP
      IF(IREGION.EQ.1) GO TO 20                                         CALCSP
   10 IDELREG = IREGION - JREGION                                       CALCSP
      IF(IDELREG.EQ.0) GO TO 20                                         CALCSP
      IF(IDELREG.EQ.6) GO TO 30                                         CALCSP
      IF(IDELREG.EQ.5) GO TO 90                                         CALCSP
      IF(IDELREG.EQ.3) GO TO 130                                        CALCSP
      IF(IDELREG.EQ.-1)GO TO 80                                         CALCSP
C                                                                       CALCSP
C        NOT ON A BLADE. SELECT LARGE ANGLE AND SEARCH THE ARC          CALCSP
C                                                                       CALCSP
   20 DELPHI = DELPHI1                                                  CALCSP
      PHI = PHI + FAC*DELPHI                                            CALCSP
      Y(1) = X(1) + SRCOST*COS(PHI)                                     CALCSP
      Y(2) = X(2) + SPCOST*SIN(PHI)                                     CALCSP
      JREGION = IREGION                                                 CALCSP
      CALL TEST(Y,RSQ,ROSQ,SWPTAU)                                      CALCSP
      GO TO 10                                                          CALCSP
C                                                                       CALCSP
C        ENTERING REGION 6 FROM REGION 0. USE THE INTERSECTION DATA     CALCSP
C                                                                       CALCSP
   30 Y(1) = YSTART(1)                                                  CALCSP
      Y(2) = YSTART(2)                                                  CALCSP
C                                                                       CALCSP
```

```
C        ENTERING REGION 6 FROM ANOTHER REGION. SELECT SMALL ANGLE
C          AND PROCESS THE BLADE
C
   40 DELPHI = DELPHI2
      IF(ETA(1,J2).LT.ETAMAX) DELPHI = DELPHI/BLNTN
      FETA1 = FOFETA1(ETA(1,J2))
   50 JREGION = IREGION
C
C        INCREASE THE SWEEP ANGLE AND TEST THE NEXT POINT
C
      PHI = PHI + FAC*DELPHI
      ETA11 = ETA(1,J2)
      ETA22 = ETA(2,J2)
      FETA11 = FETA1
      SAVETA = DELETA1
      SAVEF = DELF
      YLST(1) = Y(1)
      YLST(2) = Y(2)
      Y(1) = X(1) + SRCDST*COS(PHI)
      Y(2) = X(2) + SRCDST*SIN(PHI)
      CALL TEST(Y,RSQ,ROSQ,SWPTAU)
      IDELREG = IREGION - JREGION
      IF(IDELREG.EQ.0) GO TO 60
      IF(IDELREG.EQ.-5) GO TO 140
      IF(IDELREG.EQ.-6) GO TO 170
      GO TO 180
C
C        ON A BLADE
C
   60 J = J + 1
      DELETA1 = ETA(1,J2) - ETA11
      DELETA2 = ETA(2,J2) - ETA22
      FETA1 = FOFETA1(ETA(1,J2))
      DELF = FETA1 - FETA11
      DELGAM = SQRT(DELF**2 + DELETA1**2 + DELETA2**2)
      IF(DELETA1.LT.0.) DELGAM = - DELGAM
      SP = PFUNCN(ETA22,DELF,DELETA1,YLST)
      SUMSP = SUMSP + SP*DELGAM
      SP = PFUNCN(ETA22,SAVEF,SAVETA,YLST)
      SUMSP = SUMSP + SP*DGLST
      DGLST = DELGAM
      IF(IREGION.EQ.6) GO TO 50
C
C        JUST LEFT REGION 6. PERFORM THE TRAPEZOIDAL INTEGRATION
C
      SP = PFUNCN(ETA(2,J2),DELF,DELETA1,Y)
      SUMSP = SUMSP + SP*DELGAM
      IF(IREGION.EQ.0) GO TO 80
      DGLST = 0.
      DELETA1 = 0.
      DELF = 0.
      J = 1
      J2 = 1
      GO TO 20
C
C        PASSED OUT OF THE ROTOR SYSTEM FOR THIS ARC
C
   80 SCRIPTP = 0.5*DELTAU*SUMSP/SWPDST
      RETURN
C
C        ENTERING REGION 6 FROM REGION 1
C
   90 IF(ETA(1,J2).GT.CH/2.) GO TO 100
C
C        THROUGH THE LEADING EDGE
C
      CALL LEETA(ETA(2,J2),Y)
      ETA(1,J2) = 0.
      GO TO 110
C
C        THROUGH THE TRAILING EDGE
```

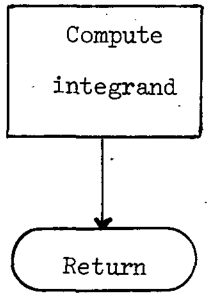
```
C
  100 CALL TEETA(ETA(2,J2),Y)
      ETA(1,J2) = CH
  110 ETA(2,J2) = ETA2
C
C         SET ANGLE AND RETURN TO BLADE LOGIC
C
  120 PHI = ACOS(0.9999999999*(Y(1) - X(1))/SRCDST)
      IF(Y(2).LT.X(2)) PHI = 2.*PI - PHI
      GO TO 40
C
C         ENTERING REGION 6 FROM REGION 3. START FROM INTERSECTION
C           WITH INNER RADIUS
C
  130 CALL CALCY(ROSQ,PHX)
      Y(1) = YEND(1)
      Y(2) = YEND(2)
      ETA(1,J2) = CH/2. - Y(1)*SBETA + Y(2)*CBETA
      IF(ETA(1,J2).LT.0.) GO TO 90
      ETA(2,J2) = -Y(1)*CBETA - Y(2)*SBETA
      GO TO 120
C
C         ENTERING REGION 1 FROM REGION 6
C
  140 IF(ETA(1,J2).GT.CH/2.) GO TO 150
C
C         THROUGH LEADING EDGE
C
      CALL LEETA(ETA(2,J2),Y)
      ETA(1,J2) = 0.
      GO TO 160
C
C         THROUGH TRAILING EDGE
C
  150 CALL TEETA(ETA(2,J2),Y)
      ETA(1,J2) = CH
  160 ETA(2,J2) = ETA2
      GO TO 60
C
C         LEAVING THE ROTOR SYSTEM FOR THIS ARC. GET INTERSECTION WITH
C           OUTER RADIUS
C
  170 CALL CALCY(RSQ,PHX)
      Y(1) = YEND(1)
      Y(2) = YEND(2)
      GO TO 190
C
C         ENTERING REGION 3 FROM REGION 6. GET INTERSECTION WITH
C           INNER RADIUS
C
  180 CALL CALCY(ROSQ,PHX)
      Y(1) = YSTART(1)
      Y(2) = YSTART(2)
  190 ETA(1,J2) = CH/2. - Y(1)*SBETA + Y(2)*CBETA
      IF(ETA(1,J2).LT.0.) GO TO 140
      ETA(2,J2) = -Y(1)*CBETA - Y(2)*SBETA
      GO TO 60
      END
```

APPENDIX A

Function PFUNCN

This function subprogram computes the integrand of the line integral at each sweep position.

Flow Chart of Function PFUNCN



Listing of Function PFUNCN

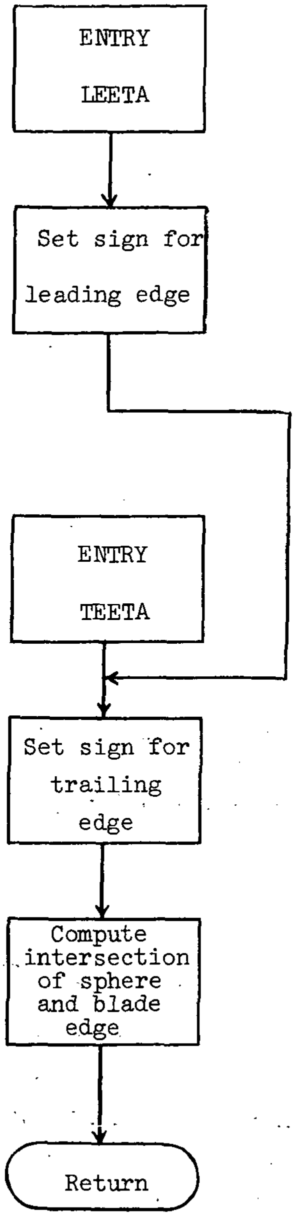
```
      FUNCTION PFUNCN(XETA,DELF,DELETA1,Y)                              PFUNCN
C*********************************************************************** PFUNCN
C*                                                                    * PFUNCN
C*       FUNCTION PFUNCN IS CALLED BY SUBROUTINE CALCSP. THIS FUNCTION * PFUNCN
C*          COMPUTES THE INTEGRAND OF THE LINE INTEGRAL AT EACH       * PFUNCN
C*          SWEEP POSITION.                                           * PFUNCN
C*                                                                    * PFUNCN
C*********************************************************************** PFUNCN
      DIMENSION     BETA(8),        COEFFS(5),      ETA(2,8),           PFUNCN
     1              X(3),           X0(3),                              PFUNCN
     2              YEND(2),        YH(3),          YSTART(2),          PFUNCN
     3              VH(3)                                               PFUNCN
      DIMENSION Y(3)                                                    PFUNCN
      COMMON         ALFA,     BETA,     BLADES,   BLNTN,    CBETA,     PFUNCN
     1               CH,       COEFFS,   DELTAU,   ETA,      ETA2,      PFUNCN
     2               ETAMAX,   FAC,      FN,       FNS,      FETA1,     PFUNCN
     3               IFLAG,    IREGION,  J2,       JFLG,     KFLG,      PFUNCN
     4               NBLADES,  OMEGAR,   OTIME,    PI,       R,         PFUNCN
     5               RM,       RMIN,     S,        SBETA,    SNDSPD,    PFUNCN
     6               SPCOST,   SWPOST,   SWPTAU,   TAU,      THKRAT,    PFUNCN
     7               TIMTHRU,  VH,       X,        XC,       YEND,      PFUNCN
     8 YH,YSTART,DELETAX,DELFX                                          PFUNCN
      PFUNCN = 0.                                                       PFUNCN
      S = (((X(1) - Y(1))*SBETA + (Y(2) - X(2))*CBETA)*DELF             PFUNCN
     1 + (X(3) - Y(3))*DELETA1)/SWPOST                                  PFUNCN
      Q = DELF**2 + DELETA1**2 - S**2                                   PFUNCN
      IF(Q.LT.1.E-10) RETURN                                            PFUNCN
      PFUNCN = DELF*(OMEGAR*XETA - VH(2)*CBETA + VH(1)*SBETA)/SQRT(Q)   PFUNCN
      RETURN                                                            PFUNCN
      END                                                               PFUNCN
```

Subroutine LEETA

Subroutine LEETA returns the coordinates (in the YH frame) of the point of intersection of a fixed arc and the leading edge of a blade. The second ETA component (see fig. 2) is also returned. A second entry point, TEETA, does the same thing for a trailing edge.

APPENDIX A

Flow Chart of Subroutine LEETA



APPENDIX A

Listing of Subroutine LEETA

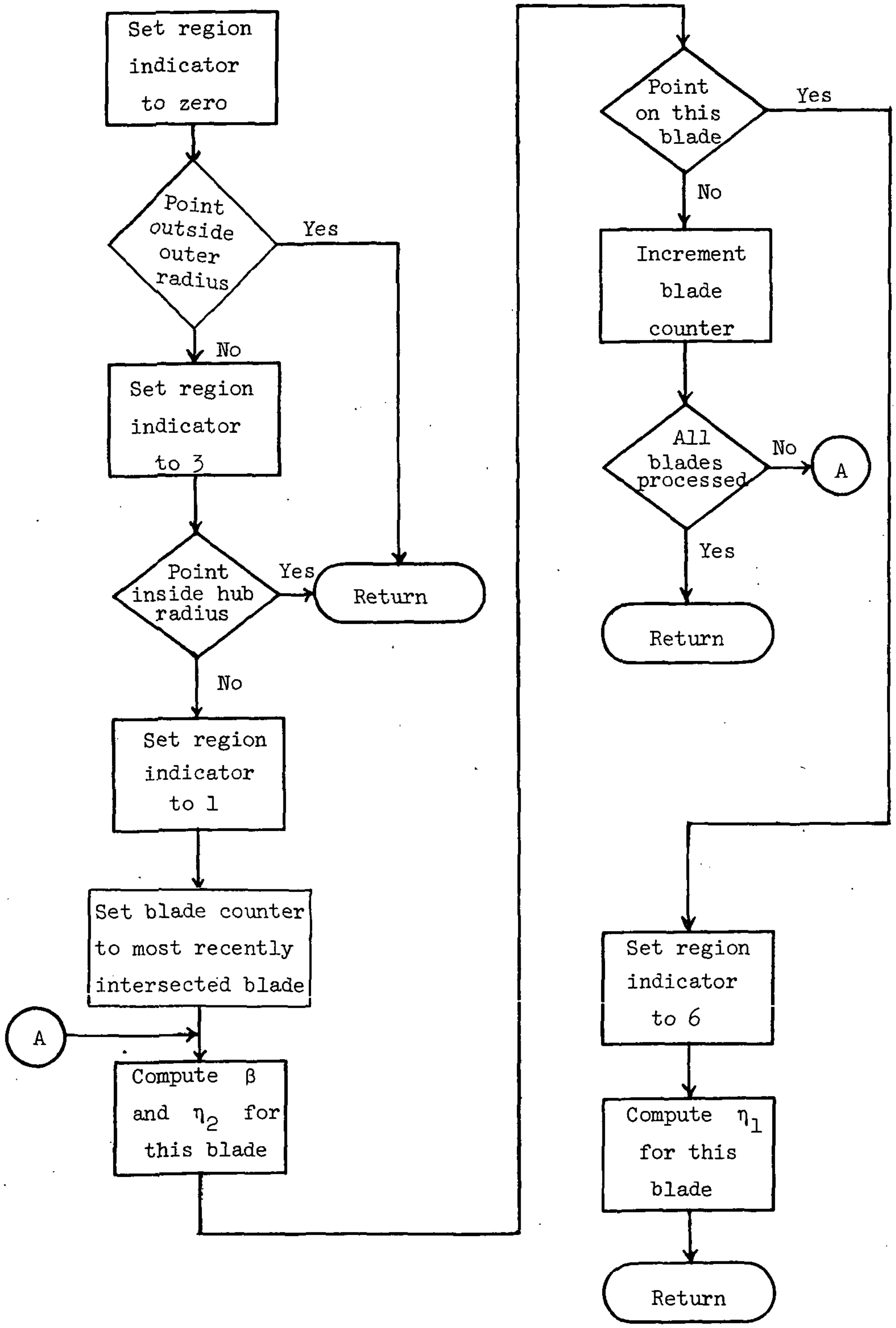
```
      SUBROUTINE LEETA(XETA,Y)
C***********************************************************************
C*                                                                     *
C*       SUBROUTINE LEETA IS CALLED BY SUBROUTINE CALCSP. THIS         *
C*         ROUTINE RETURNS THE COORDINATES (IN THE Y REFERENCE         *
C*         FRAME) OF THE INTERSECTION OF A FIXED SWEEPING ARC AND      *
C*         THE LEADING EDGE OF A BLADE. A SECOND ENTRY POINT, TEETA,   *
C*         DOES THE SAME FOR THE TRAILING EDGE. THE SECOND ETA         *
C*         COMPONENT IS ALSO RETURNED.                                 *
C*                                                                     *
C***********************************************************************
      DIMENSION     BETA(8),       COEFFS(5),     ETA(2,8),
     1              X(3),          XO(3),         Y(3),
     2              YEND(2),       YH(3),         YSTART(2),
     3              VH(3)
      COMMON         ALFA,     BETA,     BLADES,   BLNTN,    CBETA,
     1               CH,       COEFFS,   DELTAU,   ETA,      ETA2,
     2               ETAMAX,   FAC,      FN,       FNS,      FETA1,
     3               IFLAG,    IREGION,  J2,       JFLG,     KFLG,
     4               NBLADES,  OMEGAR,   OTIME,    PI,       R,
     5               RM,       RMIN,     S,        SBETA,    SNDSPD,
     6               SRCDST,   SWPDST,   SWPTAU,   TAU,      THKRAT,
     7               TIMTHRU,  VH,       X,        XC,       YEND,
     8               YH,       YSTART,   DELETA1,  DELF
      A = -0.5*CH
      GO TO 10
      ENTRY TEETA
      A = 0.5*CH
   10 B = -A
      XBAR1 = X(1) + A*SBETA
      XBAR2 = X(2) + B*CBETA
      XBAR = X(1)*CBETA + X(2)*SBETA
      XTILDE = XBAR1*XBAR1 + XBAR2*XBAR2 - SRCDST*SRCDST
      DISC1 = SQRT(XBAR*XBAR - XTILDE)
      ETA2 = -XBAR + DISC1
      IF(ABS(XETA - ETA2).GE.CH/FN) ETA2 = ETA2 - 2.*DISC1
      Y(1) = B*SBETA -ETA2*CBETA
      Y(2) = A*CBETA -ETA2*SBETA
      RETURN
      END
```

Subroutine TEST

TEST returns the region index (IREGION) for a given pair of coordinates in the YH reference frame at a given time. The region indices correspond to points outside the outer radius, inside the hub radius, within the rotor system but not a blade, and on a blade. When a point is found to be on a blade, the routine returns the blade number, the angle made by that blade, the sine and cosine of this angle, and the ETA coordinates of the point. The subroutine included with the current version of RTN is limited to rotor systems with rectangular blades.

APPENDIX A

Flow Chart of Subroutine TEST



APPENDIX A

Listing of Subroutine TEST

```
      SUBROUTINE TEST(Y,RSQ,ROSQ,TIME)                                  TEST
C*********************************************************************** TEST
C*                                                                    * TEST
C*     SUBROUTINE TEST IS CALLED BY SUBROUTINE CALCSP. THIS           * TEST
C*        ROUTINE RETURNS THE REGION INDEX FOR A GIVEN TIME AND       * TEST
C*        A GIVEN PAIR OF COORDINATES IN THE Y REFERENCE FRAME.       * TEST
C*        THE REGION INDEX HAS FOUR POSSIBLE VALUES.                  * TEST
C*                                                                    * TEST
C*           IREGION = 0:  THE POINT LIES OUTSIDE THE CIRCLE          * TEST
C*                         DEFINED BY THE BLADES OUTER RADIUS         * TEST
C*                                                                    * TEST
C*                   = 1:  THE POINT LIES BETWEEN THE CIRCLES         * TEST
C*                         DEFINED BY THE BLADES OUTER AND            * TEST
C*                         INNER RADII,BUT IS NOT ON A BLADE          * TEST
C*                                                                    * TEST
C*                   = 3:  THE POINT LIES INSIDE THE CIRCLE           * TEST
C*                         DEFINED BY THE BLADES INNER RADIUS         * TEST
C*                                                                    * TEST
C*                   = 6:  THE POINT LIES ON A BLADE.                 * TEST
C*                                                                    * TEST
C*     WHEN THE REGION INDEX IS 6, TEST RETURNS THE BLADE NUMBER,     * TEST
C*        J2, THE ANGLE MADE BY THAT BLADE, THE SIN AND COSINE OF     * TEST
C*        THIS ANGLE, SBETA AND CBETA, AND THE ETA COORDINATES OF     * TEST
C*        THE POINT.                                                  * TEST
C*                                                                    * TEST
C*********************************************************************** TEST
      DIMENSION     BETA(8),       COEFFS(5),      ETA(2,8),            TEST
     1              X(3),          X0(3),          Y(3),                TEST
     2              YEND(2),       YH(3),          YSTART(2),           TEST
     3              VH(3)                                               TEST
      COMMON         ALFA,      BETA,      BLADES,    BLNTN,     CBETA,  TEST
     1               CH,        COEFFS,    DELTAU,    ETA,       ETA2,   TEST
     2               ETAMAX,    FAC,       FN,        FNS,       FETA1,  TEST
     3               IFLAG,     IREGION,   J2,        JFLG,      KFLG,   TEST
     4               NBLADES,   OMEGAR,    OTIME,     PI,        R,      TEST
     5               RM,        RMIN,      S,         SBETA,     SNDSPD, TEST
     6               SRCDST,    SWPDST,    SWPTAU,    TAU,       THKRAT, TEST
     7               TIMTHRU,   VH,        X,         X0,        YEND,   TEST
     8               YH,        YSTART,    DELETA1,   DELF               TEST
      RYSQ = Y(1)**2 + Y(2)**2                                          TEST
      IREGION = 0                                                       TEST
      IF(RYSQ.GT.RSQ) GO TO 3                                           TEST
      IREGION = 3                                                       TEST
      IF(RYSQ.LT.ROSQ) GO TO 3                                          TEST
      IREGION = 1                                                       TEST
      LIMTOP = J2 + NBLADES - 1                                         TEST
      DO 1 J=J2,LIMTOP                                                  TEST
      I = J                                                             TEST
      IF(J.GT.NBLADES) I = J - NBLADES                                  TEST
      BETA(I) = OMEGAR*TIME + FLOAT(I-1)*6.283185308/FLOAT(NBLADES)     TEST
      CBETA = COS(BETA(I))                                              TEST
      SBETA = SIN(BETA(I))                                              TEST
      IF((Y(2)*CBETA - Y(1)*SBETA)**2.GT.CH*CH/4.) GO TO 1              TEST
      ETA(2,I) = -Y(1)*CBETA - Y(2)*SBETA                               TEST
      IF(ETA(2,I).GT.0.) GO TO 2                                        TEST
    1 CONTINUE                                                          TEST
      CBETA = COS(BETA(J2))                                             TEST
      SBETA = SIN(BETA(J2))                                             TEST
      GO TO 3                                                           TEST
    2 IREGION = 6                                                       TEST
      J2 = I                                                            TEST
      ETA(1,J2) = CH/2. - Y(1)*SBETA + Y(2)*CBETA                       TEST
    3 RETURN                                                            TEST
      END                                                               TEST
```

APPENDIX B

ROTOR THICKNESS NOISE POSTPROCESSOR PROGRAM

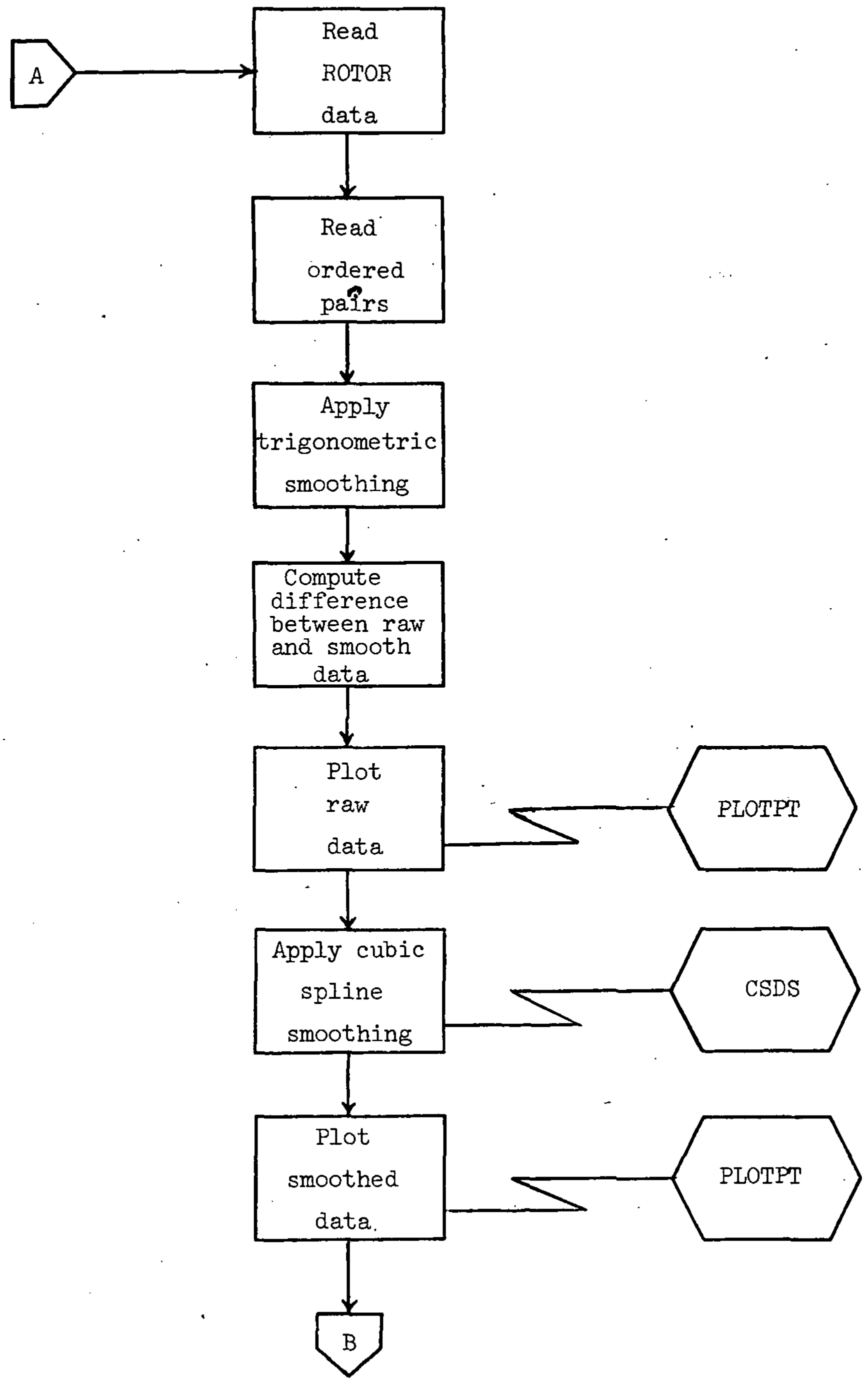
This appendix includes brief descriptions of each of the RTNPP subprograms. Where possible, each description is followed by a flow chart and FORTRAN listing of the subprogram. The CSDS subroutine is the property of International Mathematical and Statistical Library (IMSL). Four other subroutines, SPLDER, PSEUDO, INFOPLT, and CALPLT, are the property of Langley Research Center. For these five subroutines, FORTRAN listings are not included, but the more detailed descriptions (ref. 7) provided should permit their replacement by similar routines which might be available at other facilities.

Program RTNPP

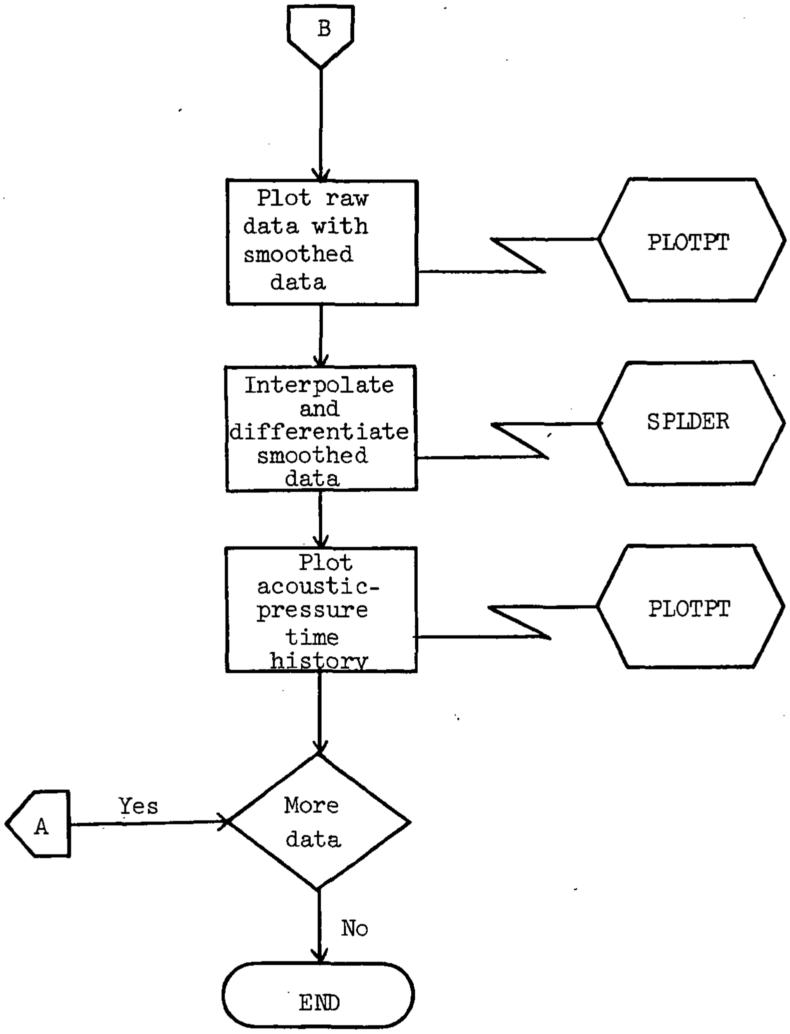
Program RTNPP reads the output from the RTN program. If the contents of TAPE12 were disposed to the card punch following execution of the RTN program, this deck should be placed in the RTNPP input stream. If TAPE12 is a disk file, it should be rewound prior to executing RTNPP. No other input is necessary for the postprocessor program. Once the data has been read and stored into arrays, trigonometric smoothing is used to obtain an estimate of the validity of each data point in the $\varphi(\vec{X},t)$ plotted against t spectrum. These estimates are then passed to a cubic spline smoothing routine. Interpolation is performed on the smoothed data to provide values at equally spaced values of the observer time. The smoothed data are numerically differentiated with respect to the observer time and the results are plotted.

APPENDIX B

Flow Chart of Program RTNPP



APPENDIX B


B
Plot raw data with smoothed data
PLOTPT
Interpolate and differentiate smoothed data
SPLDER
Plot acoustic-pressure time history
PLOTPT
More data
Yes
A
No
END

APPENDIX B

Listing of Program RTNPP

```
      PROGRAM RTNPP(INPUT=101B,OUTPUT=101B,TAPE5=INPUT,TAPE6=OUTPUT)
      DIMENSION SCRIPTP(350),OTIME(350),F(500),T(500),DER1(500),
     1 DER2(500),COEF(350,4),WK(3850),DF(350)
      DIMENSION X0(3),COEFFS(5)
      LOGICAL MORDATA
      NAMELIST/ROTOR/X0,NBLADES,R,CH,OMEGA,RO,THKRAT,DELTET,N,NS,KFLG,
     1 ETAMAX,BLNTN,JFLG,COEFFS,TAUINT,PERDM,DTAUM,SNDSPD,MORDATA
    1 FORMAT(1H0,*IERR = *,I5)
    2 FORMAT(I5,3E25.15)
      M = 0
      INIT = 0
   10 READ(5,ROTOR)
      WRITE(6,ROTOR)
   20 CONTINUE
      M = M + 1
   30 CONTINUE
      READ(5,2) I,Z,OTIME(M),SCRIPTP(M)
      OTIME(M) = 1000.*OTIME(M)
      IF(I.NE.0) GO TO 20
      M = M - 1
      MNPTS = 350
      NCVS = 1
      M = M/3
      M = 3*M
      DT = (OTIME(M) - OTIME(1))/499.
      DO 40 IJ=1,500
      T(IJ) = OTIME(1) + FLOAT(IJ-1)*DT
   40 CONTINUE
      MM = 500
      S = FLOAT(M) + SQRT(FLOAT(2*M))
      PI = 3.141592654
      NM1 = M - 1
      DO 50 I=1,M
      DER1(I) = 0.
      WK(I) = 0.
      DF(I) = SCRIPTP(I) - SCRIPTP(1) - FLOAT(I-1)*(SCRIPTP(M) - SCRIPTP
     1 (1))/FLOAT(NM1)
   50 CONTINUE
      NNMM= MIN0(85,M)
      DO 70 J=1,NNMM
      DO 60 I=2,NM1
      WK(J) = WK(J) + DF(I)*SIN(FLOAT(J*(I-1))*PI/FLOAT(NM1))
   60 CONTINUE
      WK(J) = 2.*WK(J)/FLOAT(NM1)
   70 CONTINUE
      DO 90 I=2,NM1
      DO 80 J=1,NNMM
      DER1(I) = DER1(I) + WK(J)*SIN(FLOAT(J*(I-1))*PI/FLOAT(NM1))
   80 CONTINUE
      COEF(I,1) = DER1(I) + SCRIPTP(1) + FLOAT(I-1)*(SCRIPTP(M)
     1 - SCRIPTP(1))/FLOAT(NM1)
   90 CONTINUE
      COEF(1,1) = SCRIPTP(1)
      COEF(M,1) = SCRIPTP(M)
      DO 100 I=1,M
      DF(I) =1.5*ABS(COEF(I,1) - SCRIPTP(I))
  100 CONTINUE
      ML = 0
      IF(INIT.GT.0) ML = 1
      INIT = INIT + 1
      CALL PLOTPT(OTIME,SCRIPTP,M,ML,1)
      ML = 1
      IW = -1
```

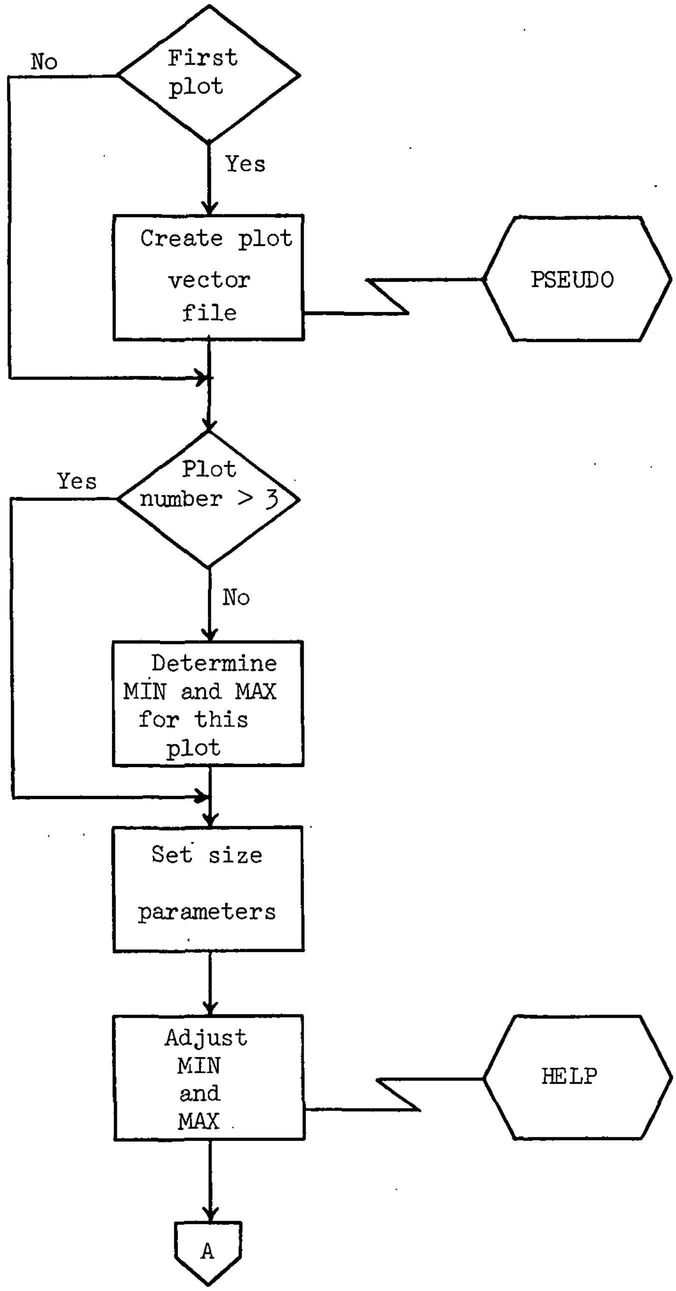
```
      CALL CSDS(MNPTS,M,OTIME,SCRIPTP,DF,S,IW,COEF,WK,IERR)
      WRITE(6,1) IERR
      CALL PLOTPT(OTIME,COEF,M,ML,2)
      XMN = OTIME(1)
      XMX = OTIME(M)
      YMN = 1.E10
      YMX = -1.E10
      DO 110 I=1,M
      IF(SCRIPTP(I).LT.YMN) YMN = SCRIPTP(I)
      IF(SCRIPTP(I).GT.YMX) YMX = SCRIPTP(I)
      IF(COEF(I,1).LT.YMN) YMN = COEF(I,1)
      IF(COEF(I,1).GT.YMX) YMX = COEF(I,1)
  110 CONTINUE
      OTIME(M+1) = XMN
      OTIME(M+2) = XMX
      SCRIPTP(M+1) = YMN
      SCRIPTP(M+2) = YMX
      COEF(M+1,1) = YMN
      COEF(M+2,1) = YMX
      CALL PLOTPT(OTIME,SCRIPTP,M,ML,4)
      CALL PLOTPT(OTIME,COEF,M,ML,5)
      IW = -1
      CALL SPLDER(MNPTS,M,NCVS,MM,MM,OTIME,COEF,T,F,DER1,DER2,IW,WK,
     1 IERR)
      WRITE(6,1) IERR
      DO 120 I=1,MM
      DER1(I) = 1000.*DER1(I)
  120 CONTINUE
      ML = 2
      IF(MORDATA) ML = 1
      CALL PLOTPT(T,DER1,MM,ML,3)
      M = 0
      IF(MORDATA) GO TO 10
      STOP
      END
```

Subroutine PLOTPT

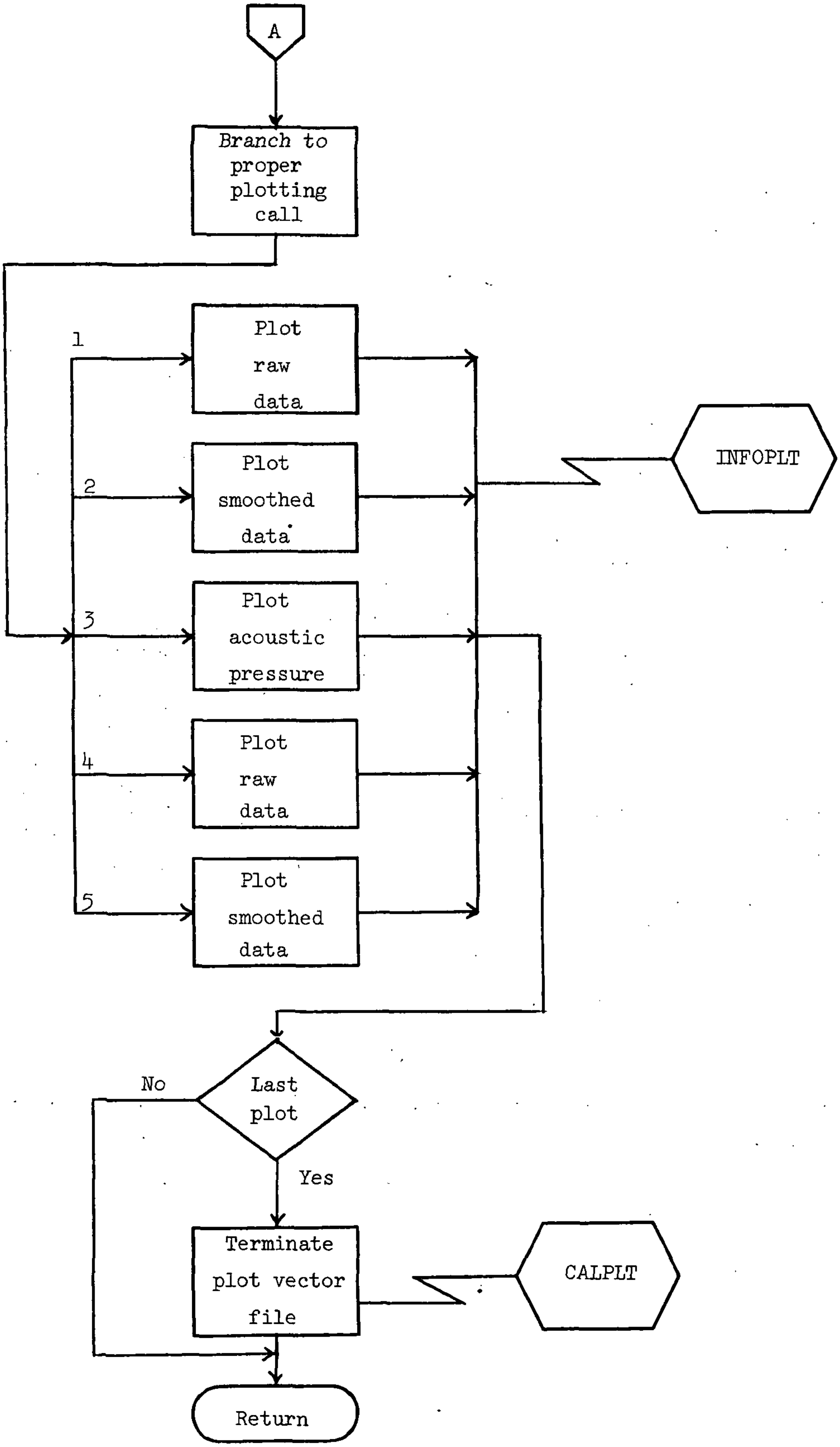
This subroutine serves as an interface between RTNPP and the graphics output library. Plot size and captions are provided for each plot.

APPENDIX B

Flow Chart of Subroutine PLOTPT



APPENDIX B


A
Branch to proper plotting call
1
2
3
4
5
Plot raw data
Plot smoothed data
Plot acoustic pressure
Plot raw data
Plot smoothed data
INFOPLT
Last plot
No
Yes
Terminate plot vector file
CALPLT
Return

APPENDIX B

Listing of Subroutine PLOTPT

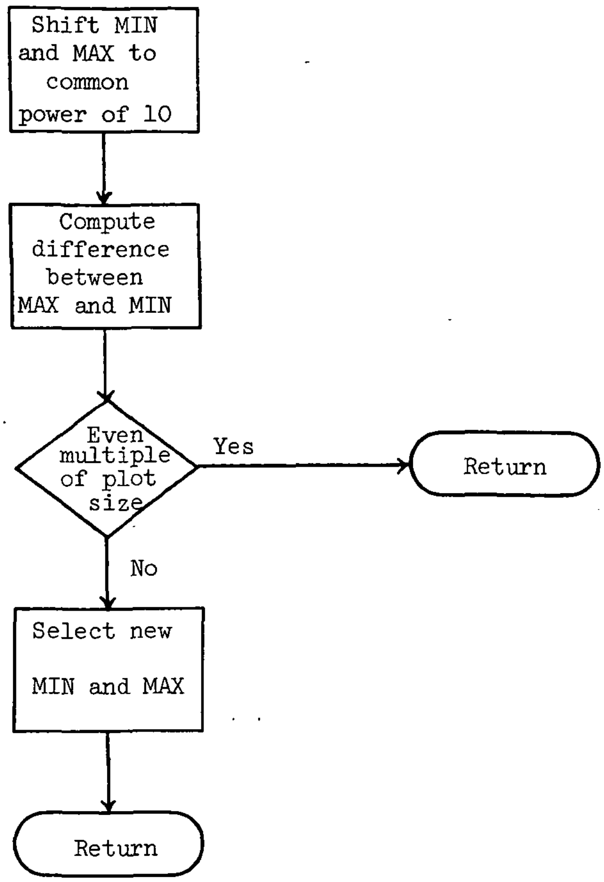
```
      SUBROUTINE PLOTPT(X,Y,N,ML,J)
      DIMENSION X(1),Y(1)
      IF(ML.EQ.0) CALL PSEUDO
      IF(J.GT.3) GO TO 15
      XMIN = 1.E10
      YMIN = 1.E10
      XMAX =-1.E10
      YMAX =-1.E10
      DO 10 I=1,N
      IF(X(I).LT.XMIN) XMIN = X(I)
      IF(Y(I).LT.YMIN) YMIN = Y(I)
      IF(X(I).GT.XMAX) XMAX = X(I)
      IF(Y(I).GT.YMAX) YMAX = Y(I)
   10 CONTINUE
      GO TO 16
   15 CONTINUE
      XMIN = X(N+1)
      XMAX = X(N+2)
      YMIN = Y(N+1)
      YMAX = Y(N+2)
   16 CONTINUE
      SH = 8.
      SV = 5.
      ISH = 8
      ISV = 5
      CALL HELP(XMIN,XMAX,ISH,XMIN,XMAX)
      CALL HELP(YMIN,YMAX,ISV,YMIN,YMAX)
      GO TO (20,30,40,60,70),J
   20 CALL INFOPLT(1,N,X,1,Y,1,XMIN,XMAX,YMIN,YMAX,1.,
     1 13,13HOBSERVER TIME,18,18HSCRIPTP (RAW DATA),0,SH,SV,1.,1.)
      GO TO 50
   30 CALL INFOPLT(1,N,X,1,Y,1,XMIN,XMAX,YMIN,YMAX,1.,
     1 13,13HOBSERVER TIME,16,16HSCRIPTP (SPLINE),0,SH,SV,1.,1.)
      GO TO 50
   40 CALL INFOPLT(1,N,X,1,Y,1,XMIN,XMAX,YMIN,YMAX,1.,
     1 28,28HOBSERVER TIME (MILLISECONDS),17,17HPRESSURE (N/M**2),0,
     2 SH,SV,1.,1.)
      GO TO 50
   60 CALL INFOPLT(0,N,X,1,Y,1,XMIN,XMAX,YMIN,YMAX,
     1 1.,13,13HOBSERVER TIME,7,7HSCRIPTP,0,SH,SV,1.,1.)
      GO TO 50
   70 CALL INFOPLT(1,N,X,1,Y,1,XMIN,XMAX,YMIN,YMAX,
     1 1.,13,13HOBSERVER TIME,7,7HSCRIPTP,0,SH,SV,1.,1.)
   50 IF(ML.EQ.2) CALL CALPLT(0.,0.,999)
      RETURN
      END
```

Subroutine HELP

This routine serves to eliminate the problem of rounding associated with the annotation of the values of the variable at the tic marks. If the specified values for the minimum and maximum result in rounding, a smaller minimum and larger maximum are substituted so that rounding does not occur.

APPENDIX B

Flow Chart of Subroutine HELP



Listing of Subroutine HELP

```
      SUBROUTINE HELP(X1,X2,D,XMIN,XMAX)
      INTEGER D
      P1 = ALOG10(ABS(X1))
      P2 = ALOG10(ABS(X2))
      P = AMAX1(P1,P2)
      IF(P.LT.0.) P = P - 1.
      IP = P - 2.
      X1P = X1/10.**IP
      X2P = X2/10.**IP
      IX1P = X1P - 1.
      IX2P = X2P + 1.
      IDIF = IX2P - IX1P
      N = IDIF/(D+1)
      JDIF = N*(D+1)
      IF(IDIF.EQ.JDIF) GO TO 10
      N = N + 1
      KDIF = N*(D+1)
      LDIF = (KDIF - IDIF)/2
      IX1P = IX1P - LDIF
      IX2P = IX1P + KDIF
   10 CONTINUE
      XMIN = FLOAT(IX1P)*10.**IP
      XMAX = FLOAT(IX2P)*10.**IP
      RETURN
      END
```

APPENDIX B

Subroutine CSDS

Language: FORTRAN

Purpose: To fit a smooth cubic spline curve to a set of data points representing a univariate function. The functional values are allowed to vary by specified amounts in smoothing the curve. Data points may be unequally spaced.

Use: CALL CSDS (MAX,IX,X,F,DF,S,IPT,COEF,WK,IERR)

MAX An input integer specifying the maximum number of data points for the independent variable, as given in the dimension statement of the calling program.

IX An input integer specifying the actual number of data points for the independent variable. IX ≦ MAX.

X A one-dimensional input array dimensioned at least IX in the calling program. Upon entry to CSDS, X(I) must contain the I-th value of the independent variable.

F A one-dimensional input array dimensioned at least IX in the calling program. Upon entry to CSDS, F(I) must contain the I-th value of the function.

DF A one-dimensional input array dimensioned at least IX in the calling program. Upon entry to CSDS, DF(I) must contain an estimate of the standard deviation of F(I). The value DF(I) determines the amount of variation permitted in the I-th functional value in smoothing the data. If the standard deviations of the functional values cannot be estimated, specifying DF = 1 is suggested.

S A nonnegative input parameter which controls the extent of smoothing. A value in the range (IX - (2*IX)**.5) ≦ S ≦ (IX + (2*IX)**.5) is suggested. Increasing values of S permit increasing amounts of smoothing of the functional values. If S = 0 is specified, an unsmoothed cubic spline curve passing through the data points is computed.

IPT An input initialization parameter. The user must specify IPT = -1 whenever a new X-array is input. The routine will then check to insure that the X-array is in strictly increasing order.

COEF A two-dimensional output array which must be dimensioned (MAX,4) in the calling program. Upon return, COEF(I,J) contains the J-th coefficient of the spline for the interval beginning at point X(I). The functional value of the spline at abscissa X1, where X(I) ≦ X1 ≦ X(I + 1) is given by:

F(X1) = ((COEF(I,4)*H + COEF(I,3))*H + COEF(I,2))*H + COEF(I,1)

where H = X1 - X(I)

WK A one-dimensional work area array dimensioned at least (7*IX + 9) in the calling program.

IERR An output integer error parameter.

= 0 Normal return.

= J The J-th element of the X-array is not in strictly increasing order. No calculation performed.

= -1 There are less than four values in the X-array. No calculation performed.

Upon return, this parameter should be tested in the calling program.

Method: A set of IX data points (X_i, F_i), i = 1,2,. . .IX are given, where F_i is the functional value at point X_i. A set of weights $DF_1, DF_2 . . . DF_{IX}$ and a nonnegative smoothing parameter S are also given.

The routine computes a cubic spline G(X) having the following properties:

$$\text{i)} \sum_{i=1}^{IX} \left\{ \left[G(X_i) - F_i \right] / DF_i \right\}^2 \leq S$$

$$\text{ii)} \int_{X_1}^{X_{IX}} \left[G''(X) \right]^2 dx = \text{Minimum of all splines satisfying property (i)}$$

Subroutine SPLDER

Language: FORTRAN

Purpose: To perform a cubic spline approximation, interpolation, and differentiation. SPLDER computes $F = f_i(X)$, $DER1 = f_i'(X)$, and $DER2 = f_i''(X)$ for any number of different dependent variable arrays associated with the independent variable array.

Use: CALL SPLDER (MNPTS,N,NCVS,MMAX,M,X,Y,T, F, DER1,DER2,IW,WK, IERR)

MNPTS An input integer specifying the maximum number of values in the independent variable array as stated in the dimension statement of the calling program.

N	An input integer specifying the number of values in the independent variable array. N $\leqq$ MNPTS.
NCVS	An input integer specifying the number of dependent variable tables associated with the independent variable.
NMAX	An input integer specifying the maximum number of values at which interpolation is desired as stated in the dimension statement of the calling program.
M	An input integer specifying the number of values at which interpolation is desired on this entry into SPLDER. M $\leqq$ MMAX.
X	A one-dimensional input array containing the independent variables. The array X should be dimensioned by at least N in the calling program, and the value must be monotonic.
Y	A two-dimensional input array containing the dependent variables. The array Y is dimensioned with variable dimension in the subroutine; therefore, Y must be dimensioned in the calling program with first dimension MNPTS and second dimension at least NCVS.
T	A one-dimensional input array containing the values of the independent variable for which values of the dependent variable and the first and second derivatives are desired. The array T must be dimensioned by at least M in the calling program.
F	A two-dimensional output array in which SPLDER stores the values of the function at the M values of the independent variable. The array F is dimensioned with variable dimension in the subroutine; therefore F must be dimensioned in the calling program with first dimension MMAX and second dimension at least NCVS.
DER1	A two-dimensional output array in which SPLDER stores the values of the derivative of the function at the M values of the independent variable. The array DER1 is dimensioned by variable dimension in the subroutine; therefore, DER1 must be dimensioned in the calling program with first dimension MMAX and second dimension at least NCVS.
DER2	A two-dimensional output array in which SPLDER stores the values of the second derivative of the function at the M values of the independent variable. The array DER2 is dimensioned by variable dimension in the subroutine; therefore, DER2 must be dimensioned in the calling program with first dimension MMAX and second dimension at least NCVS.
IW	An input-output integer.

INPUT: IW is the initialization integer. Whenever a new X- or Y-array is input, IW must be set to -1. This condition will cause the independent variable array to be tested to determine if it is increasing. Also, certain values pertaining to the X- and Y-arrays will be computed. These values will not change unless either the X- or Y-arrays are replaced.

OUTPUT: IW is an index pointer indicating that $X_{IW} \leqq X_O \leqq X_{IW+1}$. On the next call to SPLDER, the previous IW is used to begin the search for the interval containing the interpolation point.

WK An array used by SPLDER as a work area. WK must be dimensioned at least 3(NxNCVS) + 8N. This array should not be used elsewhere in the program.

IERR An output integer error code.

= 0 Normal return.

= 1 The independent variable array is not increasing. A message will be printed, "INDEPENDENT VARIABLE ARRAY NOT INCREASING IN SPLDER AT POSITION IIII X=XXXX,XXXX."

= 2 A value in the T-array is not within the limits of the X-array. There are no provisions for extrapolation; therefore, every T_j must satify $X_1 \leqq T_j \leqq X_n$.

Upon return to the calling program, the parameter IERR should be tested.

Method: The method used in SPLDER is that of the reference. The reference gives the derivative of a matrix equation relating the second derivative of a univariate spline function at the given values of the independent variable to the values of the function at these values of the independent variable. Values of the second derivative are assumed to be zero at the end points. The matrix equation is tridiagonal and is solved by the Thomas algorithm which is equivalent to Gaussian elimination without pivoting. Expressions are derived for the first and second derivatives of the spline function at any point in an interval in terms of the values of the spline function and its second derivative at the end points of the interval.

References: Greville, T. N. E., "Spline Functions, Interpolation and Numerical Quadrature," Mathematical Methods for Digital Computers, Vol. II, pp. 156-168, John Wiley and Sons, 1967.

APPENDIX B

Subroutine PSEUDO

Language: COMPASS

Purpose: To create and write an appropriately named Plot Vector File. Through linkages set up by an initial call to PSEUDO, all subsequent graphics data generated by the user will be routed through one of the PSEUDO entry points and written on the Plot Vector File.

Use: CALL PSEUDO

Subroutine INFOPLT

Language: FORTRAN

Purpose: To provide a one-call method of preparing plotting displays automatically.

Use: CALL INFOPLT(IEC,N,XDATA,KX,YDATA,KY,XMIN,XMAX,YMIN,YMAX, PCTPTS,NXMC,XM,NYMC,YM,ISYM,SX,SY,XOFF,YOFF), where

IEC The code for terminating the frame.

0 (1) Used to initialize a frame and plot first curve of multiple curves per frame. This value of IEC leaves frame incomplete and expects additional curves.

OR

(2) Used to plot second to nth curves on the frame. This value of IEC guarantees that all curves will be plotted in the same frame, although the scales for second to nth curve could be different depending on other options selected. If the scales are different from labeled scales, the rescaled values of the origin and the scale increment for the call are printed out in the right vertical margin of the frame.

1 (1) Used to plot one curve per frame. This value of IEC completes the frame.

OR

(2) Used to plot the last curve for the frame. This value of IEC guarantees that all curves will be plotted on the same frame, although the scales for the last curve could be different, depending on the other options selected. If the scales are different from label scales, the rescaled values of origin and scale increment for the call are printed out in the right vertical margin of the frame.

OR

(3) Used to complete frame only. Generates no plotting. Special call with two parameters only.

CALL INFOPLT(1,0)

-1 Used to plot a curve with scales from previous calls on the same frame. This value of IEC leaves frame incomplete. If the curve will not fit on the plot, a new frame is set up for the remaining curves.

Example:

Same Plot (Scales could change)

```
CALL INFOPLT(0,. . .
CALL INFOPLT(0,. . .
CALL INFOPLT(1,. . .
```

Same Plot if Possible (If scales change, create new frame)

```
CALL INFOPLT(0,. . .
CALL INFOPLT(-1,. . .
CALL INFOPLT(-1,. . .
CALL INFOPLT(1,0)
```

N The number of points to be plotted.

XDATA The name of the array containing the floating-point values of X to be plotted. If all data within the array are to be plotted, the parameter would be expressed as XDATA. If it is desired to plot only a portion of the array, the desired beginning location is specified; for example, XDATA(4) would begin the plotting at the fourth element of the array.

KX The interleave factor which specifies the sequence in which X-data are stored.

= 1 Indicates that values are stored sequentially.

= 2 Indicates that values are stored in every other location in the array, etc.

YDATA The name of the array containing the floating-point values of Y to be plotted.

KY The interleave factor which specifies the sequence in which Y-data are stored.

XMIN The minimum value for X.

XMAX The maximum value for X.

YMIN The minimum value for Y.

YMAX The maximum value for Y.

The routine checks for the first call only to determine if either (XMAX - XMIN) or (YMAX - YMIN) is equal to zero. When either is zero, the routine will scan the X- and/or Y-array to determine the limits. For multiple curves per display, the limits must be specified on the first call to include all curves since the limits from the first call will be used for all curves.

If any data falls outside the limits, it will be eliminated; but a count will be kept of all points dropped.

At the completion of a particular curve, if the percentage of points dropped exceeds the value of PCTPTS, the current curve data will be automatically rescaled and replotted. For multiple curves, the first curves would not be contained on the rescaled plot.

If values are given for XMIN, XMAX, YMIN, and YMAX different than on the first call and IEC = 0, the scales could be different on the second to nth call. If the scales are different, the new origin and scale increment will be printed out. The XMIN, XMAX, YMIN, and YMAX values are ignored on the second to nth call if IEC = -1.

PCTPTS The percentage of points in any one curve that may be off scale without automatic replotting. The value of PCTPTS is a floating-point number ranging from 0.0 to 1.

= 0.0 No points are allowed off scale. The data will be rescaled if any points are off scale.

= 0.5 50 percent of the points are allowed off scale before the curve will be rescaled. Any percentage may be expressed.

= 1.0 No rescaling will be done.

NXMC The number of characters for the X-label including embedded blanks. The number of characters is calculated in two ways depending on the option selected in XM.

(1) XM is written in form nHxxx. . ..

NXMC = n

(2) XM is the beginning storage location containing alphanumeric information.

NXMC = Number of words in array multiplied by 10. Each alphanumeric word contains 10 characters including blanks.

The sign of NXMC is used to control generation of the X axis.

> 0 An axis will be drawn and annotated.

= 0 No axis will be drawn; therefore, no annotation.

< 0 One-inch grid lines will be drawn in addition to the axis.

XM The label for horizontal annotation. This label may be expressed in two ways:

(1) The string of alphanumeric characters for the label may be written in the form:

nHxxx. . ..

(the same way an alpha message is written using FORTRAN format statements).

(2) The beginning storage location of an array containing alphanumeric information may be used.

NYMC The number of characters for the Y-label. (See explanation under NXMC.)

YM The label for vertical annotation. (See explanation under XM.)

ISYM An integer code specifying the symbol or mode to be used in plotting the data values.

0 Draws a line (no symbols)

SX The length of the X-axis in floating-point inches. The default SX is 10 inches.

SY The length of Y-axis in floating-point inches. The default SY is 10 inches.

XOFF The offset for Y-axis from frame origin. A nonzero XOFF allows room for the Y-axis annotation. The default XOFF is 0.75 inches. If preprinted grid paper is used, the user may specify XOFF = 1.0 or change offset with postprocessor option.

YOFF The offset for the X-axis from the frame origin. A nonzero YOFF allows room for the X-axis annotation. The default YOFF is 0.5 inches. If preprinted grid paper is used, the user may specify YOFF = 1.0 or change offset with postprocessor option.

Subroutine CALPLT

Language: FORTRAN

Purpose: To move the plotter pen to a new location with pen up or down.

Use: CALL CALPLT(X,Y,IPEN), where

X,Y Are the floating-point values for pen movement.

IPEN = 2 Pen down

= 3 Pen up

Negative IPEN will assign X = 0, Y = 0 as the location of the pen after moving the X,Y (create a new reference point).

= 999 Writes a terminating block address of 999 to terminate the Plot Vector File and all further processing is skipped.

CALL CALPLT(0.0,0.0,999)

REFERENCES

1. Deming, A. F.: Noise From Propellers With Symmetrical Sections at Zero Blade Angle. II. NACA TN 679, 1938.

2. Arnoldi, R. A : Propeller Noise Caused by Blade Thickness. Rep. R-0896-1, Res. Dep., United Aircraft Corp., Jan. 1956.

3. Billing, H., ed.: Modern Aeronautical Acoustics. Rep. & Transl. No. 960, British M. M.A.P. Volkenrode, July 1, 1947.

4. Farassat, F.: Some Research on Helicopter Rotor Noise Thickness and Rotational Noise. Second Interagency Symposium on University Research in Transportation Noise, Vol. I (Raleigh, N.C.), June 1974, pp. 363-370.

5. Farassat, F.: Theory of Noise Generation From Moving Bodies With an Application to Helicopter Rotors. NASA TR R-451, 1975.

6. Abbott, Ira H.; and Von Doenhoff, Albert E.: Theory of Wing Sections. McGraw-Hill Book Co., Inc., 1949.

7. Computer Programing Manual. Langley Research Center, NASA, 1975.
   Vol. I.- General Information.
   Vol. II.- Subprogram Library.
   Vol. III.- Manufacturers Manuals.
   Vol. IV.- Special Capabilities.

TABLE I.- ROTOR THICKNESS NOISE PROGRAM INPUT VARIABLES

Variable	Dimension	Type	Units
BLNTN	*	Real	**
CH	*	Real	meters
COEFFS	*	Real	**
DELTET	*	Real	degrees
DTAUM	*	Real	**
ETAMAX	*	Real	meters
JFLG	*	Integer	**
KFLG	*	Integer	**
MORDATA	*	Logical	**
N	*	Integer	**
NBLADES	*	Integer	**
NS	*	Integer	**
OMEGA	*	Real	rev/min
PERDM	*	Real	**
R	*	Real	meters
RO	*	Real	meters
SNDSPD	*	Real	meters/sec
TAUINT	*	Real	seconds
THKRAT	*	Real	**
X0	3	Real	meters

*Single variable, not an array.

**Not applicable, unitless.

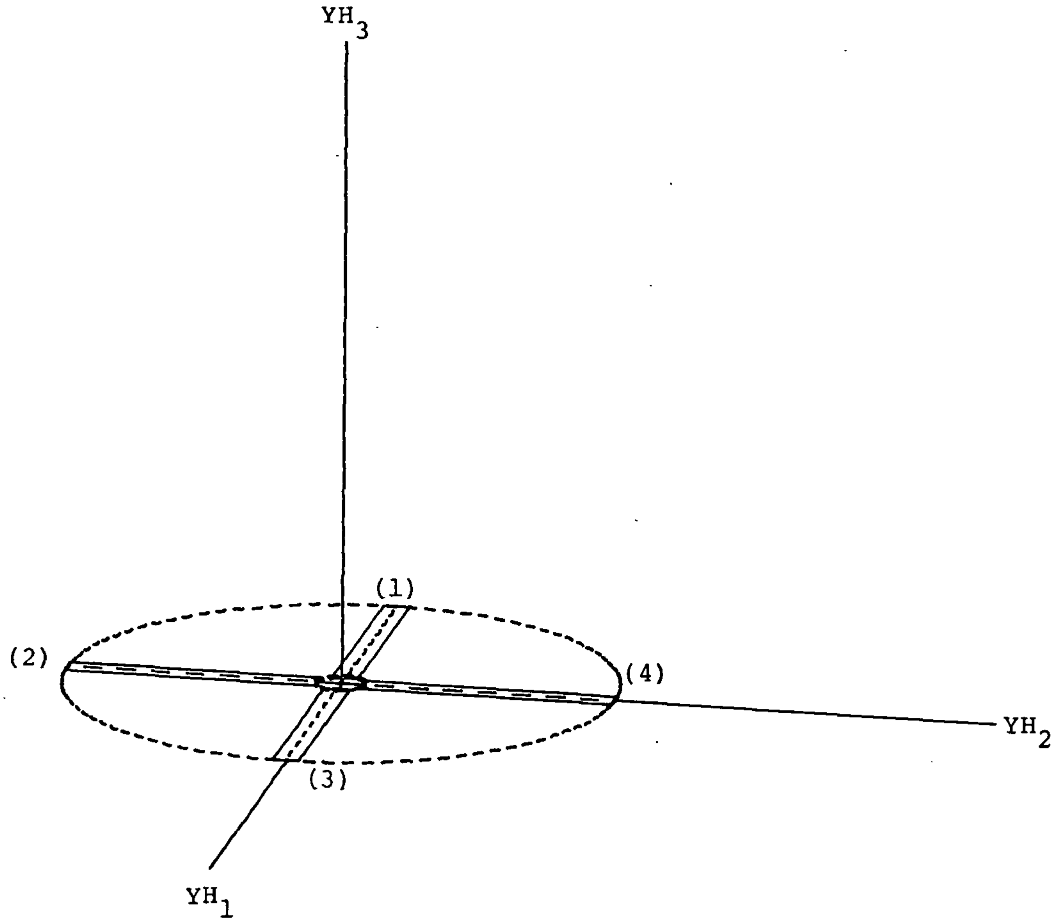


Figure 1.- Definition of YH reference frame based upon orientation of rotor system at time TAU = 0. (Numbers in parentheses are blade identification numbers.) Reference frames YH and Y are identical at TAU = 0.

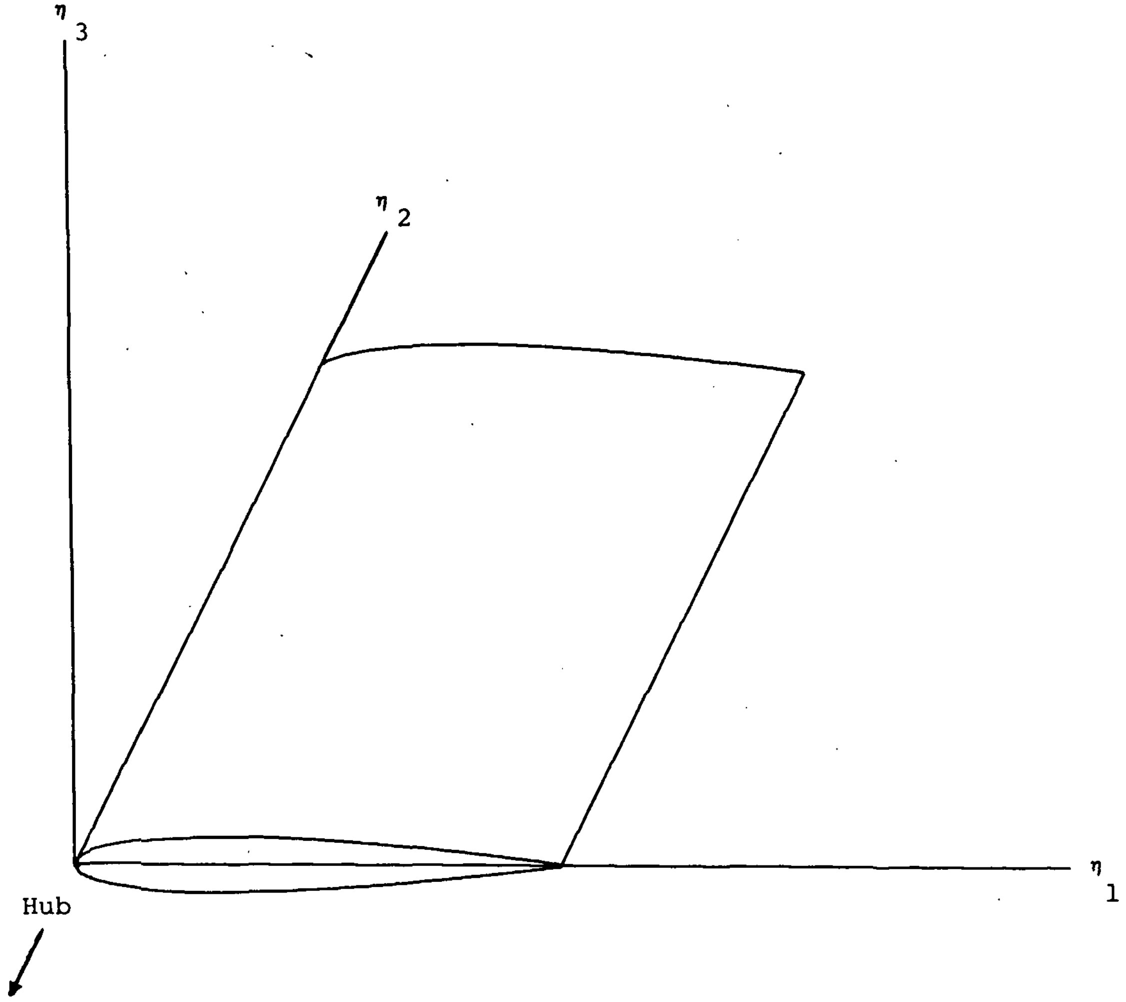


Figure 2.- Definition of blade coordinate system.

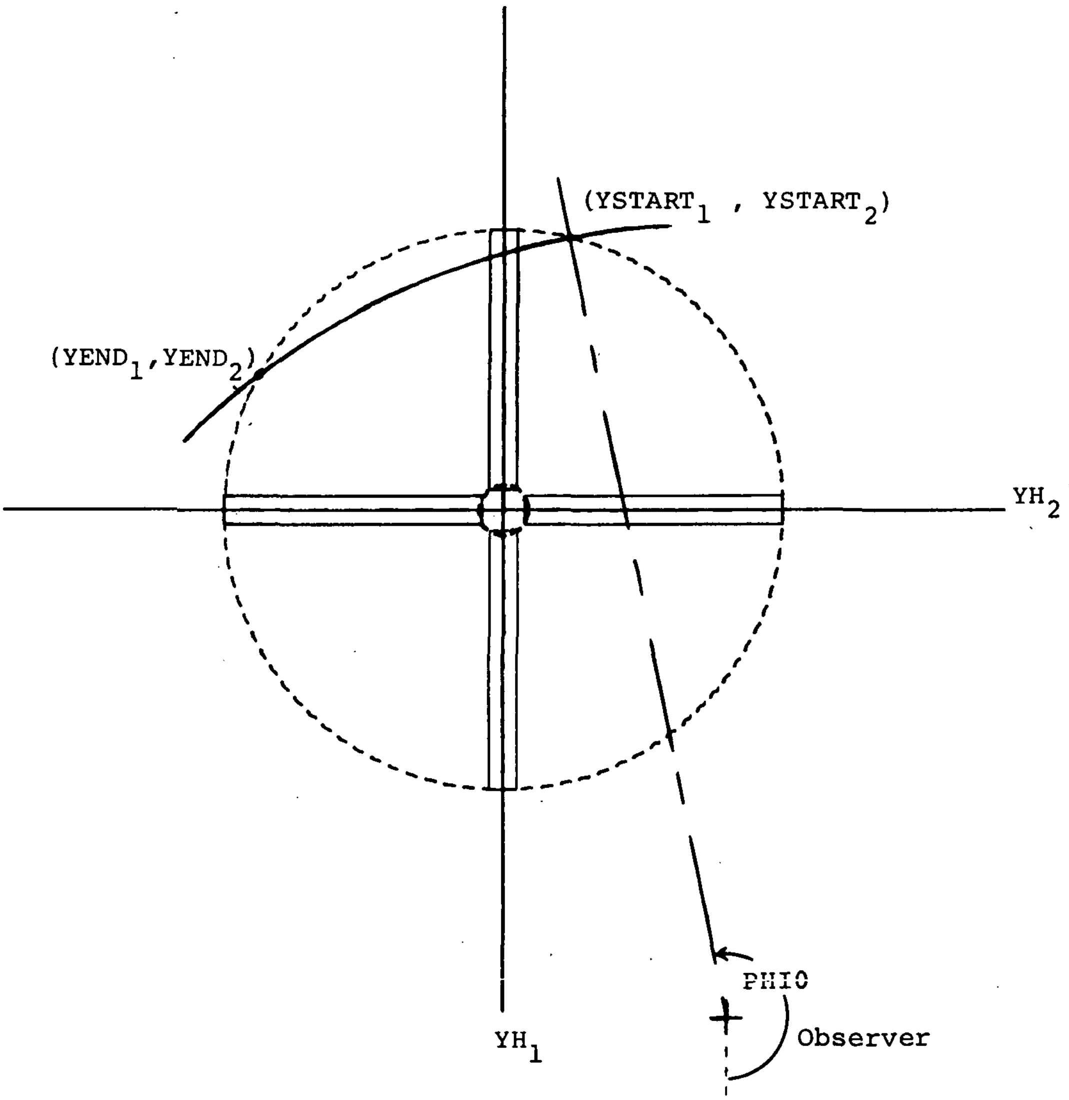


Figure 3.- Definition of arc used to sweep fixed position of rotor system.

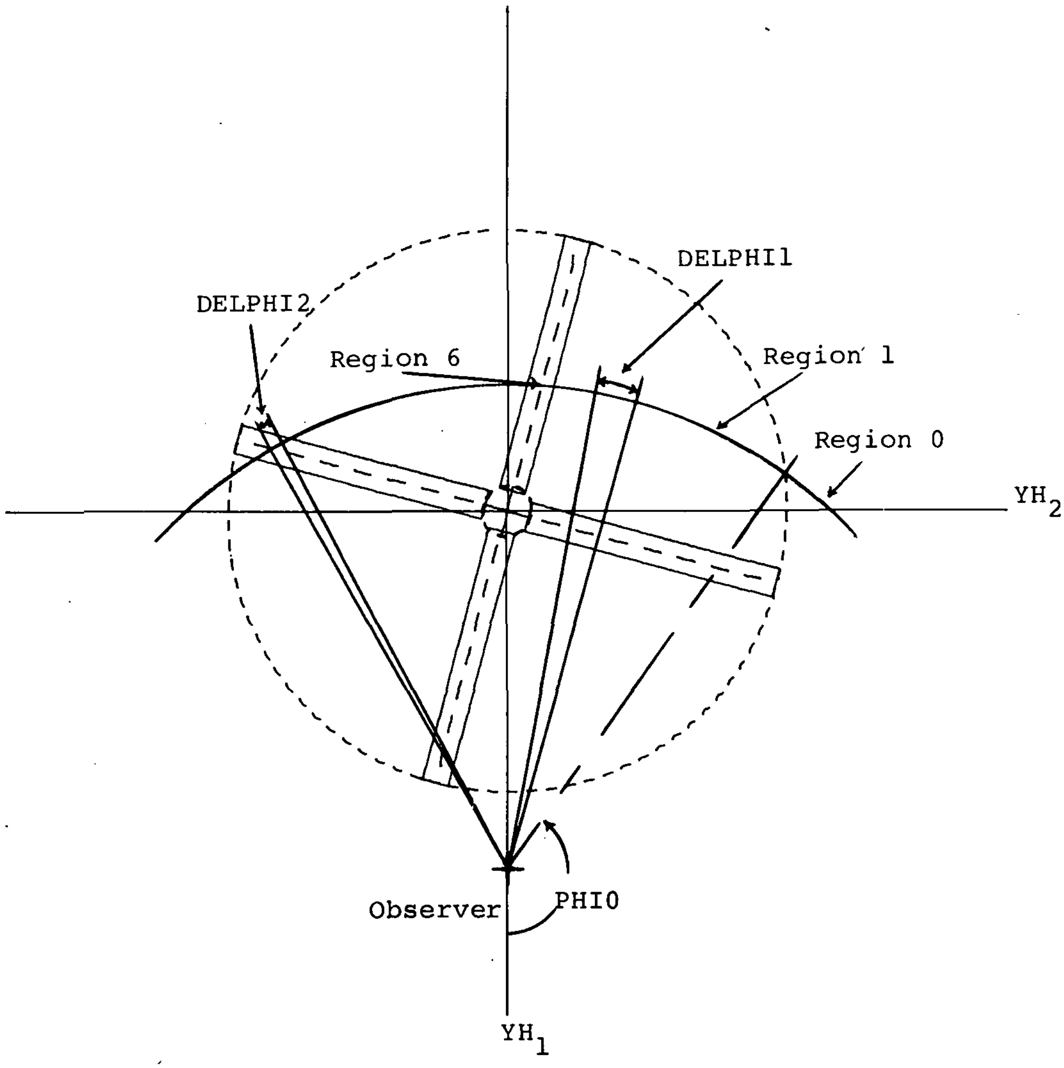


Figure 4.- Definition of sweep regions.

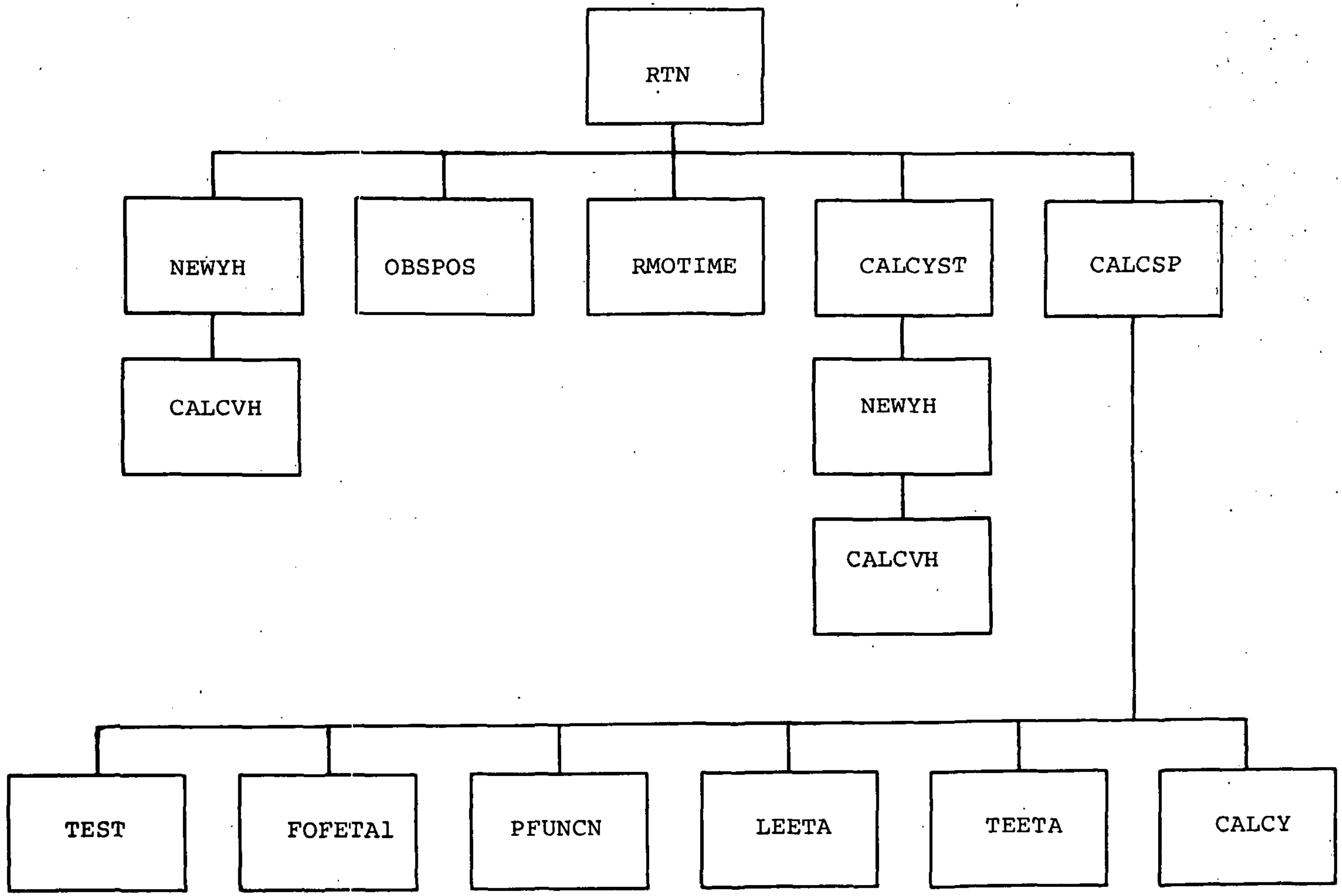


Figure 5.- Rotor Thickness Noise (RTN) subprogram hierarchy.

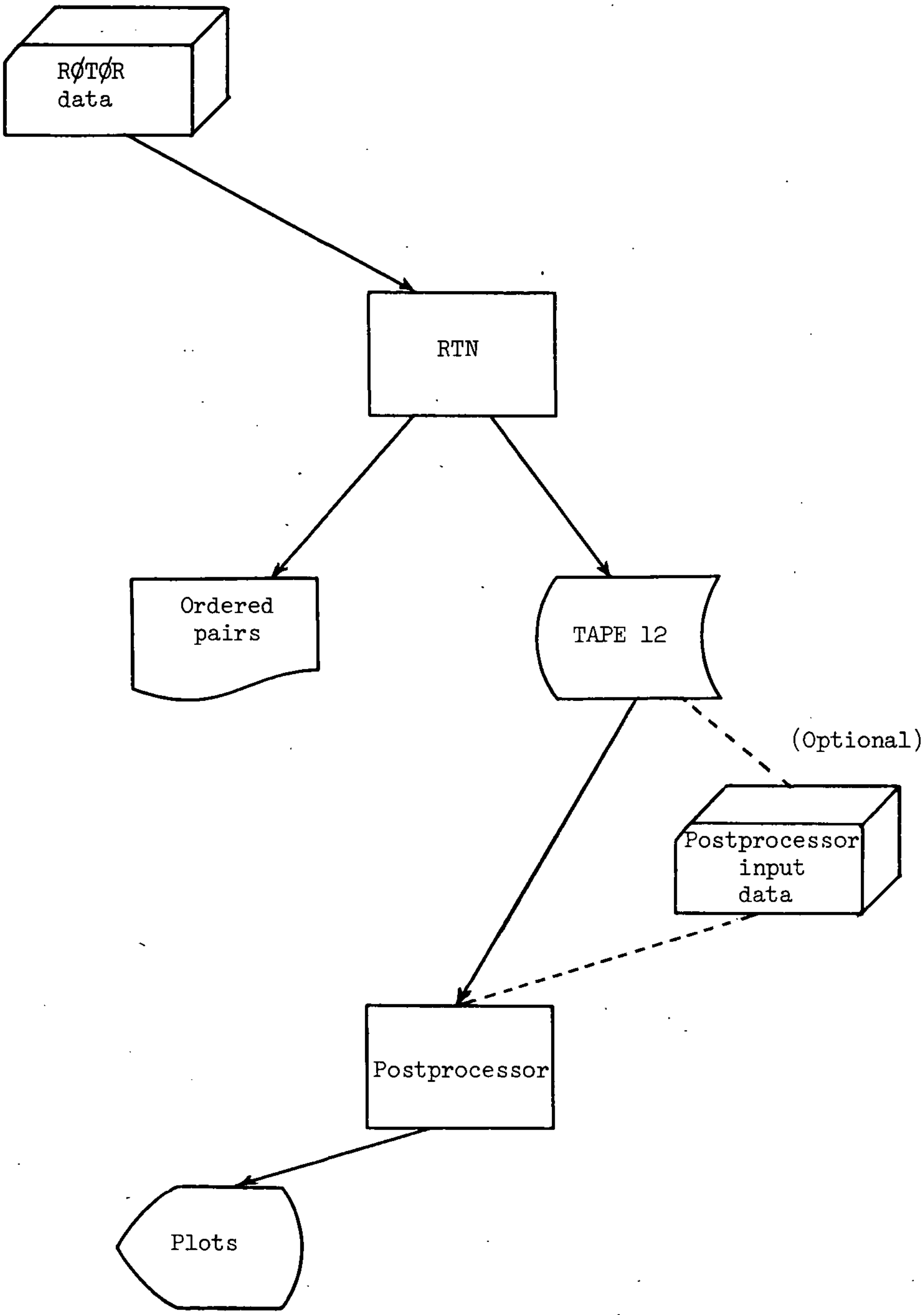


Figure 6.- RTN Postprocessor flow.

```
$ROTOR

X0      = .25E+03, 0.0, 0.0,

NBLADES = 2,

R       = .671E+01,

CH      = .7E+00,

OMEGA   = .48E+03,

RO      = .61E+00,

THKRAT  = .8E-01,

DELTET  = .4E+00,

N       = 6,

NS      = 20,

KFLG    = 0,

ETAMAX  = 0.0,

BLNTN   = .1E+01,

JFLG    = 0,

COEFFS  = 0.0, .2E+01, -.2E+01, 0.0, 0.0,

TAUINT  = 0.0,

PERDM   = .106E+01,

DTAUM   = .2E+01,

SNDSPD  = .34E+03,

MORDATA = F,

$END
   1    0.                         .755003247706419E+00     .368274186289126E-02
   2     .277777777777778E-03      .755293152072209E+00     .376488349357262E-02
   3     .555555555555556E-03      .755579312092344E+00     .412774739956806E-02
   4     .833333333333332E-03      .755861726257422E+00     .423309736144570E-02
   5     .111111111111110E-02      .756140393732565E+00     .462362007791356E-02
   6     .138888888888888E-02      .756415314357586E+00     .465687047375710E-02

   7     .166666666666665E-02      .756686488646952E+00     .500900726077658E-02
   8     .194444444444442E-02      .756953917789762E+00     .504101714574137E-02
   9     .222222222222219E-02      .757217603649526E+00     .544945062874086E-02
  10     .249999999999996E-02      .757477548764001E+00     .534413020731811E-02
  11     .277777777777773E-02      .757733756344756E+00     .565431454752224E-02
  12     .305555555555549E-02      .757986230276764E+00     .559454617899044E-02
  13     .333333333333326E-02      .758234975117880E+00     .600260255484447E-02
  14     .361111111111102E-02      .758479996098231E+00     .602330162628115E-02
  15     .388888888888879E-02      .758721299119451E+00     .652646018526923E-02
  16     .416666666666654E-02      .758958890753917E+00     .661387664236740E-02
  17     .444444444444431E-02      .759192778243825E+00     .678877483189216E-02
  18     .472222222222207E-02      .759422969500225E+00     .726628270774723E-02
  19     .499999999999984E-02      .759649473101881E+00     .712892068665186E-02
  20     .527777777777760E-02      .759872298294145E+00     .762862808628620E-02
  21     .555555555555537E-02      .760091454987670E+00     .785473094677686E-02
  22     .583333333333313E-02      .760306953757027E+00     .813629285486817E-02
  23     .611111111111090E-02      .760518805839279E+00     .874972874059439E-02
  24     .638888888888867E-02      .760727023132397E+00     .855715233570625E-02
  25     .666666666666643E-02      .760931618193645E+00     .935317605149660E-02
  26     .694444444444420E-02      .761132604237847E+00     .961866129864003E-02
  27     .722222222222196E-02      .761329995135512E+00     .975679084175196E-02
```

Figure 7.- Example of RTN output.

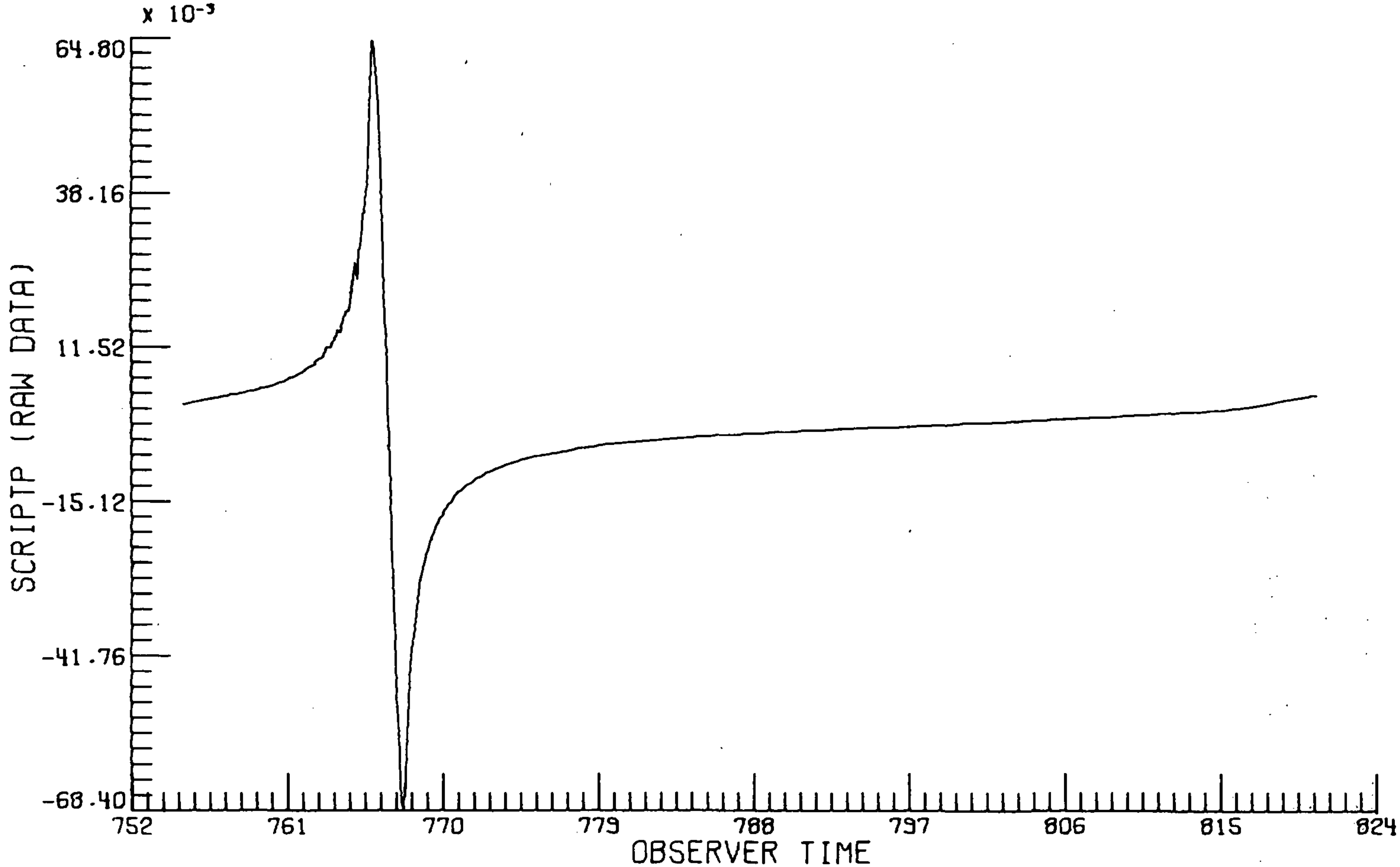


Figure 8.- Postprocessor graphic output of Rotor Thickness Noise output.

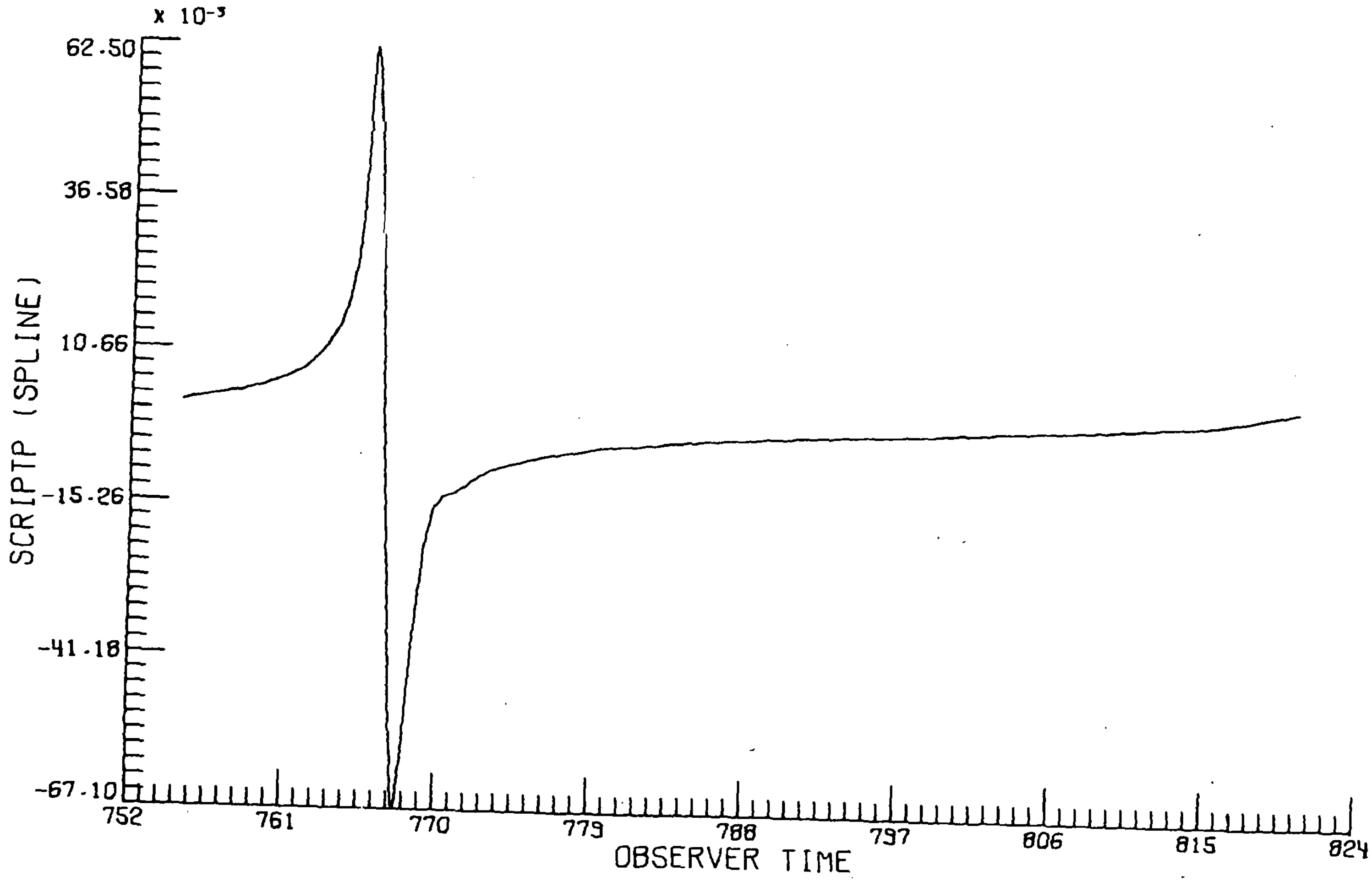


Figure 9.- Smoothed RTN output data using spline technique.

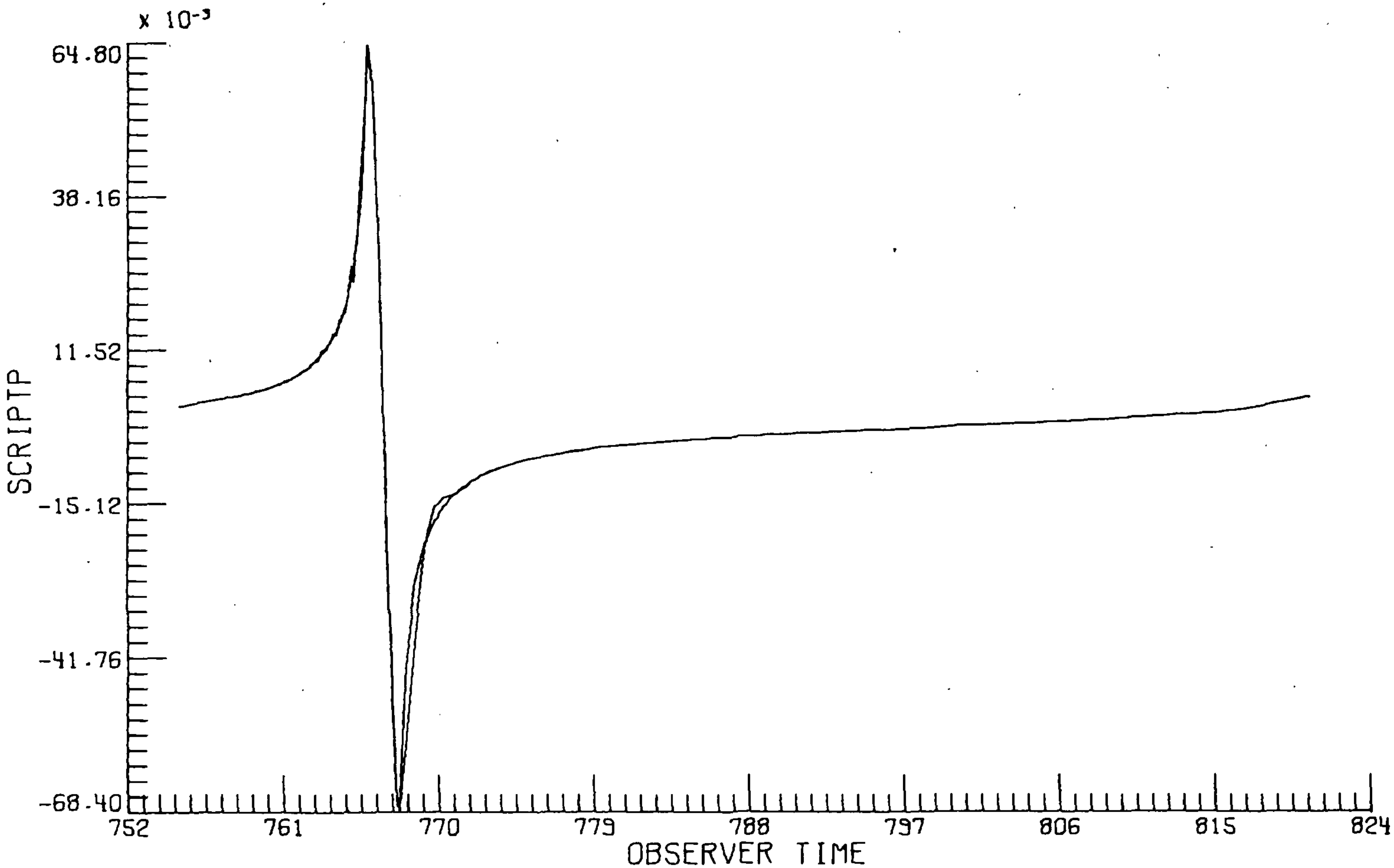


Figure 10.- Comparison of spline smoothed data with raw data.

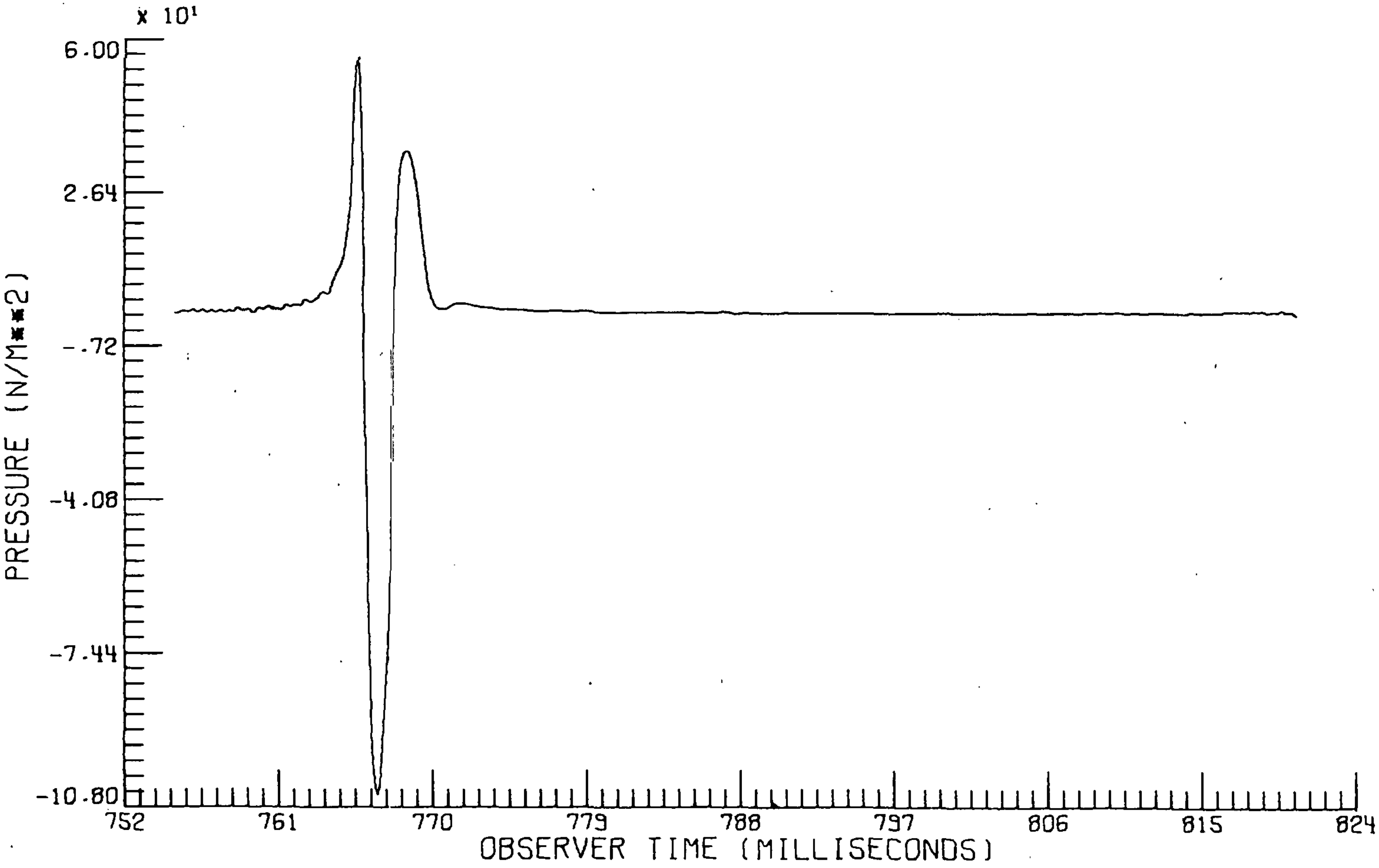


Figure 11.- Acoustic pressure caused by blade thickness for one period of blade system of example in figure 7.

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546