

**NASA CONTRACTOR
REPORT**



NASA CR-2415

NASA CR-2415

**SEPARATED FLOW OVER BODIES
OF REVOLUTION USING AN UNSTEADY
DISCRETE-VORTICITY CROSS WAKE**

Part II - Computer Program Description

by F. J. Marshall and F. D. Deffenbaugh

Prepared by

PURDUE UNIVERSITY

West Lafayette, Ind.

for Langley Research Center



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1974

1. Report No. NASA CR-2415		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle SEPARATED FLOW OVER BODIES OF REVOLUTION USING AN UNSTEADY DISCRETE-VORTICITY CROSS WAKE - PART II COMPUTER PROGRAM DESCRIPTION				5. Report Date June 1974	
				6. Performing Organization Code	
7. Author(s) F.J. Marshall and F.D. Deffenbaugh				8. Performing Organization Report No.	
9. Performing Organization Name and Address Purdue University School of Aeronautics and Astronautics W. Lafayette, Indiana				10. Work Unit No. 760-65-11-0200	
				11. Contract or Grant No. NGR 15-005-119 (181)	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546				13. Type of Report and Period Covered Contractor Report	
				14. Sponsoring Agency Code	
15. Supplementary Notes Topical report.					
16. Abstract A method is developed to determine the flow field of a body of revolution in separated flow. The technique employed is the use of the computer to integrate various solutions and solution properties of the sub-flow fields which made up the entire flow field without resorting to a finite difference solution to the complete Navier-Stokes equations. The technique entails the use of the unsteady cross flow analogy and a new solution to the required two-dimensional unsteady separated flow problem based upon an unsteady, discrete-vorticity wake. Data for the forces and moments on aerodynamic bodies at low speeds and high angle of attack (outside the range of linear inviscid theories) such that the flow is substantially separated are produced which compare well with experimental data. In addition, three dimensional steady separated regions and wake vortex patterns are determined. This report describes the computer program developed to perform the numerical calculations.					
17. Key Words (Suggested by Author(s)) Bodies of Revolution Separated flow Unsteady flow Crossflow Flow field			18. Distribution Statement Unclassified STAR Category 12		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 154	22. Price* \$5.00

TABLE OF CONTENTS

	Page
INTRODUCTION	1
ABBREVIATIONS AND ADDITIONAL LIST OF SYMBOLS	2
METHOD OF SOLUTION	3
VCF PROGRAM DESCRIPTION	3
VCF PROGRAM INPUT DATA	3
VCF PROGRAM OUTPUT DATA	6
VCF PROGRAM STRUCTURE	7
VCF AUXILIARY FILES	7
VCF OPERATING INSTRUCTIONS	7
ADYNF PROGRAM DESCRIPTION	9
ADYNF PROGRAM INPUT DATA	9
ADYNF PROGRAM OUTPUT DATA	11
ADYNF PROGRAM STRUCTURE	12
ADYNF OPERATING INSTRUCTIONS	12
APPENDIX I: PROGRAM VCF AND SUBPROGRAM DESCRIPTIONS	13
APPENDIX Ia: PROGRAM ADYNF AND SUBPROGRAM DESCRIPTIONS	47
APPENDIX Ib: SUBROUTINE IC	52
APPENDIX II: CANNED SUBPROGRAMS - CADRE - SECOND - SPLINE, ARITH1 - TRID	65
APPENDIX III: PROGRAM LISTINGS - VCF - ADYNF - SUBPROGRAMS	101
APPENDIX IV: SAMPLE CASE - VCF - ADYNF	137
REFERENCES	152

INTRODUCTION

In part 1 of this two part report the theory was developed to determine the forces acting on a body of revolution in a uniform stream. The body was inclined at an angle of attack sufficient for the flow to become separated and for vortices to form on the lee side. This part of the report describes the computer programs, VCF and ADYNF, which carry out the numerical calculations of part 1. VCF is the main body of the work which calculates the sectional normal force coefficient, c_n , at stations along the body axis. ADYNF using the results obtained from VCF numerically integrates $c_n(\hat{z})$ over the body length to obtain the normal force coefficient C_N , and the moment coefficient $C_{M,\lambda}$. Useful vortex patterns are available, and a line of separation along the body can be determined.

Abbreviations and Additional List of Symbols

Abbreviations

3DS	three dimensional steady
2DUS	two dimensional unsteady
b.l.	boundary layer
r.s.l.	rear shear layer
t.b.l.	top boundary layer
b.b.l.	bottom boundary layer
t.r.s.l.	top rear shear layer
b.r.s.l.	bottom rear shear layer

Sub- and Superscript

() ^v	point vortex contribution
() ^p	potential flow contribution

Numerical Integration Parameters

$(\theta_i, t_k, \bar{r}_j)$	set of points in \bar{r}, t, θ plane $i = 0,1,2 \dots, \text{IDIM}$ $k = 0,1,2 \dots, \text{KFINAL}$ $j = 0,1,2 \dots, \text{JDIM}$
$u(\theta_i, t_k, \bar{r}_j)$	$(u)^i$ evaluated at point $(\theta_i, t_k, \bar{r}_j)$
$v(\theta_i, t_k, \bar{r}_j)$	$(v)^i$ evaluated at point $(\theta_i, t_k, \bar{r}_j)$

METHOD OF SOLUTION

The forces acting on a body of revolution at high angle of attack are calculated by the two complementary programs VCF, and ADYNF. Program VCF calculates the normal force distribution on the body by first solving the two dimensional unsteady problem of a circular cylinder started impulsively from rest, and then by using the viscous cross flow analogy to relate the cylinder drag to the normal force sectional coefficient. ADYNF then integrates the normal force distribution supplied by VCF to obtain the normal force and moment coefficients. A detailed description of the method is given in part 1 of this report.

VCF PROGRAM DESCRIPTION

The computer program was written in FORTRAN IV language, and was run on a CDC 6500 computer at Purdue University under the MACE operating system. VCF requires approximately 110000 octal words central memory initially, and CP time for an average run is approximately 30 minutes. The program requires three peripheral disc files in addition to the input, output, and punch files.

VCF PROGRAM INPUT DATA

The input to VCF consists of the Reynolds number based on body length, the angle of attack, the body geometry, and program control data.

Description of Input Decks

The user must supply the body geometry in the form of a FUNCTION subprogram named RZERO. The value of the dimensional body radius r_0^* , should be returned as a function of dimensional axial distance along the body, \hat{z}^* . In addition ENTRY DRZERO must return $dr_0^*/d\hat{z}^*$. FUNCTION RZERO is a subprogram deck of VCF supplied by the user.

Deck Setup:

FUNCTION RZERO (ZSTAR)	CARD 1
RZERO = (body radius at ZSTAR)	CARD 2
RETURN	CARD 3

ENTRY DRZERO	CARD 4
RZERO = (rate of change of body radius with respect to axial distance evaluated at ZSTAR)	CARD 5
RETURN	CARD 6
END	CARD 7

Description of Input Cards

CARD 1 - Identification. - Card 1 contains any desired identifying information in columns 1-80.

CARD 2 - Flow parameters, body length. - Card 2 contains 3 real numbers, punched in a 12-column field. Columns 37-80 may be used in any desired manner. Card 2 contains the following.

Columns	Variable	Value	Description
1-12	AATACK		Angle of attack in degrees
13-24	RE		$Re_{3DS} = V\ell/\nu$
25-36	LENGTH		Dimensional body length, ℓ , same units as r_o^* .

CARD 3 - Remaining input data. - Card 3 contains three parameters particular to the finite difference and outer flow numerical computation schemes. They are punched in a 12-column field. Columns 37-80 may be used in any desired manner. Card 3 contains the following.

Columns	Variable	Value	Description
1-12	DELT	.125	The time increment used in the finite difference solution to the boundary layer equations, Δt_k
13-24	RC	.05	Potential vortex core cutoff radius, r_c
25-36	SIGMA	.05-1.0	Empirical vortex flux factor, σ

CARD 4 - Program control - The three parameters punched in Card 4, the first right justified in a three column field, and the rest in twelve column fields, determine when the program is to STOP. All data necessary for further execution of the program is written on an auxiliary disk file. Card 4 contains the following.

Columns	Variable	Value	Description
1-3	KFINAL	1-100	If KFINAL = K, where K is an integer count of the number of program time cycles completed, appropriate data is written on an auxiliary disk file and the program STOPS.
13-24	TFINAL		If TFINAL equals or exceeds the central processor time used, data is written on file and the program STOPS.
25-36	ZFINAL	0.0-1.0	If the nondimensional distance along the body axis \hat{z} , equals or exceeds ZFINAL data is written on file and the program STOPS.

CARD 5 - Input/Output Control. - Card 5 contains 4 numbers each punched right justified in a 2-column field. These parameters specify which auxiliary disk files are to be used in input/output operations, the type of punched output, and the type of printed output. Card 5 contains the following.

Columns	Variable	Value	Description
1-2	LR	0,3,4	The continue data is READ from TAPE (LR). If LR = 0 no data is read from the auxiliary disk file.
3-4	LW	3-4	The continue data is WRITTEN on TAPE (LW).
5-6	LEVEL	5	Printed Output Level.
7-8	KPUN		Cards are punched every KPUN cycles.

VCF PROGRAM OUTPUT DATA

Printed output is processed by a standard 132 characters-per-line printer, punched cards are in Hollerith (026) code, 80 characters per cards. Binary coded data is written on auxiliary files, TAPE 1, TAPE 3, and TAPE 4. The program printed output options are described below:

LEVEL = 3 The program prints the dimensional body geometry as the radius as a function of axial distance. The maximum body diameter, the characteristic length, the fineness ratio, and frontal area, which are all functions of input body geometry are printed. Remaining card input data is printed. Point vortex locations, velocities and strengths, are printed. The pressure distribution around the two dimensional cylinder corresponding to an axial station along the three dimensional body is printed for each program cycle, as well as values of shear and pressure drag. Data specifying the rear separation angle is printed for each cycle after the backflow velocity has exceeded .1 of the free stream velocity. Elapsed computer execution time is printed at the end of each program cycle.

LEVEL = 5 In addition to level 3 output, the boundary layer profile for the top portion of the cylinder is printed.

The program printed output is illustrated by the sample case presented in Appendix IV.

The program punched output consists of variables punched every program cycle, and of variables punched every KPUN cycles. Values of nondimensional axial distance, coefficient of drag, nondimensional time, and the number of time cycles completed are punched every cycle. Vortex positions, velocities, and strengths are punched every KPUN cycles.

The binary coded data output consists of initial boundary layer velocities, (reference 2), written on TAPE 1. Values of boundary layer velocities, vortex positions, vortex velocities, vortex strengths, program indices, and rear shear layer data are written on TAPE (LW) at the termination of each run.

VCF PROGRAM STRUCTURE

VCF consists of a MAIN program, 18 SUBROUTINE subprograms, 5 FUNCTION subprograms, and 1 FUNCTION subprogram supplied by the user. Detailed descriptions of the MAIN program and of the subprograms are given in Appendix I. An example of a user supplied geometry description deck is given in Appendix IV for the case of an ogive cylinder.

VCF AUXILIARY FILES

VCF designates TAPE 5 as the input file, TAPE 6 as the output file, and TAPE 7 as the punch file. In addition three auxiliary files are utilized for transfer of binary data. These files are designated TAPE 1, TAPE 3, and TAPE 4.

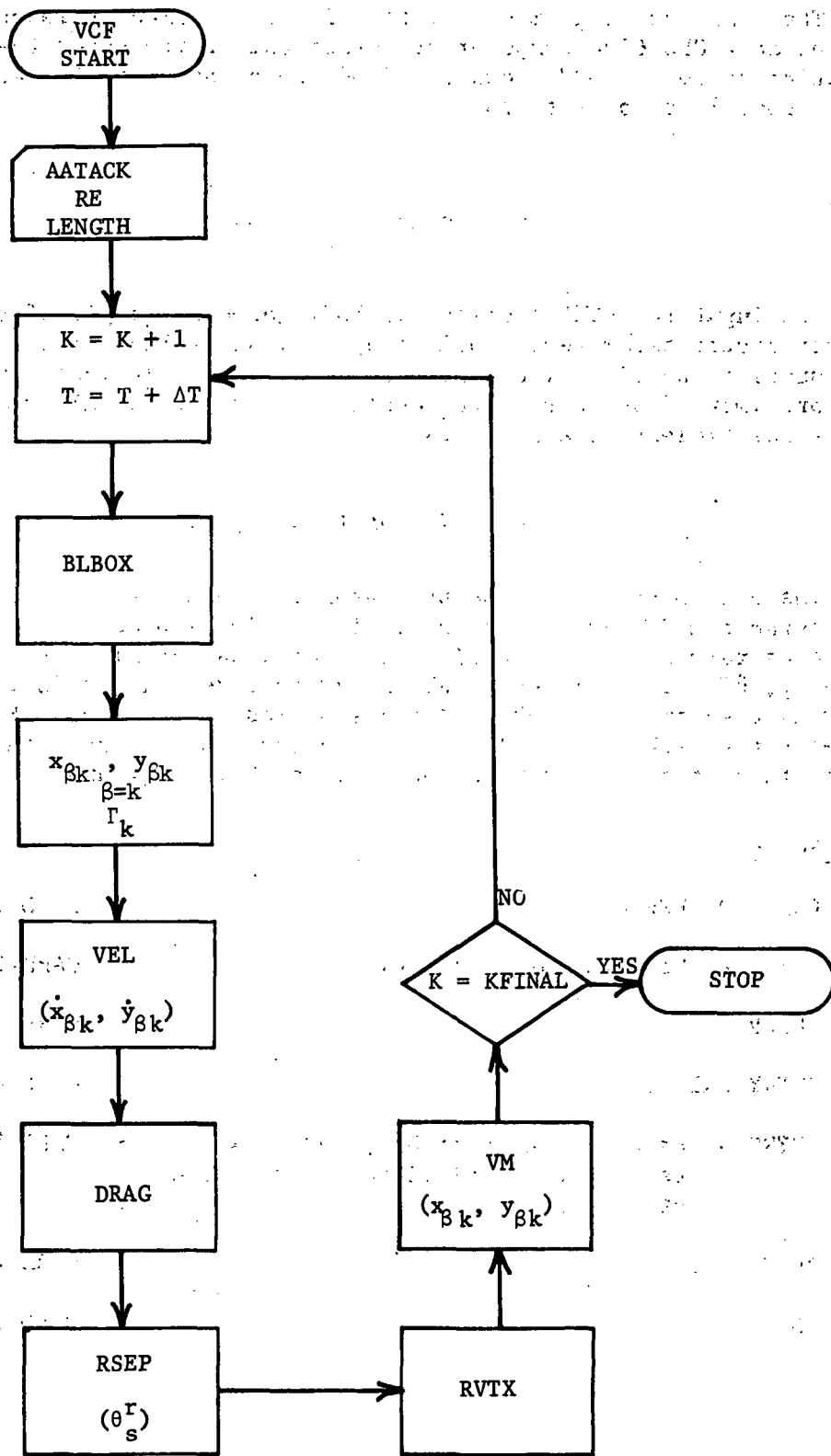
TAPE 1 is used to store the initial boundary layer profile. TAPE 3 and TAPE 4 are used to store intermediate information which is used to continue execution of the program.

These auxiliary files must be rewound and stored on magnetic tape or some other permanent storage file at the termination of each run. To continue the run the files are obtained from permanent file storage, and execution is continued, the necessary data being READ from TAPE 3 or TAPE 4.

VCF OPERATING INSTRUCTIONS

The program deck, geometry input deck, and data deck are loaded in the following sequence: job card, system control cards, end-of-record card, program deck, geometry input deck, end-of-record card, data deck, end-of-file card. The geometry input deck and the data deck are described in the Program Input Data section.

VCF FLOWCHART



ADYNF PROGRAM DESCRIPTION

The computer program was written in the FORTRAN IV language, and was run on a CDC 6500 computer at Purdue University under the MACE operating system. ADYNF requires 55000 words of central memory and takes about 10 seconds to run.

ADYNF PROGRAM INPUT DATA

The input to ADYNF consists of body geometry, angle of attack, characteristic parameters supplied by VCF, and values of c_n and \hat{z} , also supplied by VCF. The number of knots and their locations, used to approximate functions in the evaluation of the normal force and moment coefficients are also read in.

Description of Input Decks

The user must supply the body geometry in the form of a FUNCTION subprogram named RZERO. The value of the dimensional body radius r_0^* , should be returned as a function of dimensional axial distance along the body, \hat{z}^* . In addition ENTRY DRZERO must return $dr_0^*/d\hat{z}^*$. FUNCTION RZERO is a subprogram deck of ADYNF supplied by the user. For a given geometry the deck used in ADYNF is identical to that used in PROGRAM VCF, and in fact, the same deck may be used for both programs

Deck Setup:

FUNCTION RZERO (ZSTAR)	CARD 1
RZERO = (body radius at ZSTAR)	CARD 2
RETURN	CARD 3
ENTRY DRZERO	CARD 4
RZERO = (Rate of change of body radius with respect to axial distance evaluated at ZSTAR)	CARD 5
RETURN	CARD 6
END	CARD 7

Description of Input Cards

CARD 1 - Identification. - Card 1 contains any desired identifying information in columns 1-80.

CARD 2 - Flow parameters, characteristic dimensions. - Card 2 contains 6 real numbers, punched in a 12-column field. Columns 73-80 may be used in any desired manner. Card 2 contains the following.

Columns	Variable	Value	Description
1-12	AATACK		Angle of attack in degrees
13-24	LAMBDA		Moment arm, λ
25-36	LENGTH		Dimensional body length, l
37-48	F		Fineness Ratio, f
49-60	AW		Characteristic length, a
61-72	RW		Maximum Radius, $d/2$

CARD 3 - Number of knots, number of data points. - Card 3 contains 2 integers right justified in a 4 column field. The number of knots refers to the number of cubic polynomials used to obtain an approximating function to the data. A cubic polynomial approximates the data between two knots. Card 3 contains the following.

Columns	Variable	Value	Description
1-4	NOKNOT	4-5	The interval is divided into 3 or 4 segments depending on whether NOKNOT is 4 or 5. The endpoints of the segments are the knots.
5-8	LX		The number of data points, \hat{z} , $C_D(\hat{z})$.

CARD 4,5 ... - Axial distance \hat{z} , Coefficient of drag $C_D(\hat{z})$. - Card 4,5 ... contain values of \hat{z} and $C_D(\hat{z})$, these cards are punched by PROGRAM VCF. The two numbers are punched in the first two 12-column fields of each card.

CARD LAST - Knot positions. - The last card in the input deck to ADYNF contains the location of the knots. Since the nondimensional body length is 1.0, these knot values should be: 0, values contained in the interval [0,1], and 1. The knot values are punched in a 12-column field: 6 to a card. The last card contains the following.

Columns	Variable	Value	Description
1-12	XI(1)	0	Location of first knot
13-24	XI(2)	0 < () < 1	Location of 2nd knot
.	XI(NOKNOT)	1	Location of last knot

ADYNF PROGRAM OUTPUT DATA

Program ADYNF output is printed output processed by a standard 132 character-per-line printer. The program printed output consist of:

- (1) The given data [\hat{z} , $c_n(\hat{z})$]
- (2) The number of knots
- (3) Initial knot locations
- (4) Optimized knot locations (reference 4)
- (5) The cubic coefficients used to approximate the data in each interval
- (6) The errors involved in approximating the data
- (7) The data point \hat{z} and the approximation $c_n(\hat{z})$
- (8) Values of \hat{z} at .05 increments from 0 to 1.0 and the value of the approximation $c_n(\hat{z})$
- (9) Intermediate output from the integration subroutine CADRE (see Appendix II for details)
- (10) The approximation to the integral $\int_0^1 c_n(\hat{z}) d\hat{z}$. The absolute error, and an indication as to the types of singularities involved in the integration. (see APPENDIX II - CADRE for further details)
- (11) The normal force coefficient C_N
- (12) Similar output for the calculation of the moment coefficient $C_{M,\lambda}$.

ADYNF PROGRAM STRUCTURE

ADYNF is a main program which references the subprograms RZRO, FIT, SPLINEB, CADRE and the user input geometry subprogram RZERO. SPLINE, references 3 and 4, and CADRE are "canned" routines obtained from Purdue University's computing center. SPLINE is a deck of subprograms. ADYNF calls SPLINEB which references other subprograms in the SPLINE deck. Those subprograms are considered to be "black boxes," and are not explained in detail in this report. However listings of the SPLINE deck are included in APPENDIX II. Listings of ADYNF, FIT, and RZRO are given in APPENDIX III. ADYNF designates TAPE 5 as input file and TAPE 6 as output file. No other files are used.

ADYNF OPERATING INSTRUCTIONS

The program deck, geometry input deck, and data deck are loaded in the following sequence: job card, system control cards, end-of-record card, program deck, geometry input deck, end-of-record card, data deck, end-of-file card. The geometry input deck and the data deck are described in the Program Input Data section.

APPENDIX I

PROGRAM VCF AND SUBPROGRAM DESCRIPTIONS

This appendix contains a brief outline of the purpose, method, and use of program VCF and its subroutines. The principal variables and constants in each are listed, and identified as input or output data. The subprograms are listed in alphabetical order.

PROGRAM VCF

PURPOSE: To calculate the local normal force distribution coefficient $c_n(\hat{z})$, to predict the line of separation $\theta_s(\hat{z})$, and to provide the point vortex distribution, $(x_{\beta k}, y_{\beta k})$.

METHOD: VCF proceeds in discrete steps of time Δt_k , usually given the value .125. The index K , initially 0, is incremented by 1 for each step in time, thus $K=1$ for $t_1 = t_0 + \Delta t_1$, where t_0 denotes the initial time. The boundary layer equations are integrated using a finite difference solution developed by M. G. Hall, reference 1. The initial velocity distribution is supplied by Wundt's solution, reference 2, to the impulsive start of a circular cylinder. Once the initial conditions have been calculated the subroutines used in the computations are no longer necessary. By writing all of the pertinent data on a FILE, in this instance either TAPE3 or TAPE4, and then replacing the inert subroutine with a dummy subroutine IC, and reading the data back in, the central memory requirement can be reduced after the first few minutes of computation time. Further, since the boundary layer solution employs a finite difference method, storage must be allocated for a finite difference mesh or grid. However, since separation soon occurs, values at mesh points greater than the separation angle are no longer needed. By stopping execution, saving the necessary data, redimensioning the boundary layer grid, and then reading the data back in, and continuing execution, the central memory requirements can be reduced even further. For the case of the ogive cylinder the central memory requirement could be reduced from 110000 words to 65000 words. The boundary layer grid is DIMENSIONED in the main program and values of u^1 , and v^1 are passed as parameters to subprograms with the DIMENSION parameters IDIM, JDIM. Thus if boundary layer grid values are no longer being used because they are in separated flow the variables U, W, UT, and UB are appropriately DIMENSIONED in the main program.

The boundary layer finite difference scheme is applied either to the top half of the cylinder or the bottom. The entire boundary layer solution could be accomplished by a call to BLBOX with appropriate values of velocity on the upper surface being supplied, and then another call to BLBOX with lower surface values. However, since the solution is basically a small time one, and the flow is expected to be symmetrical, it is only necessary to apply the finite difference solution to half of the cylinder.

Once the velocity distribution in the boundary layer is known, and a separation angle has been determined, vorticity is introduced into the outer flow in the form of point vortices. Due to the symmetry of the problem

separate vortex arrays were created for the vortices born from the top separation angle, and for those coming from the bottom separation angle. In addition when backflow velocities were great enough to generate vorticity on the rear portion of the cylinder two additional top and bottom arrays were created for those point vortices originating from the rear separation angles.

Bernoulli's equation yields the pressure distribution on the cylinder, and numerically integrating $C_p \cos \theta$ ($0 < \theta < 2\pi$), gives the drag.

The point vortices existing at time t_k are convected with the fluid and at time $t_{k+1} = t_k + \Delta t_k$ the tangential velocity on the surface (boundary conditions for the boundary layer equations), will be altered. A new boundary layer velocity profile is calculated, a new separation angle determined, and the cycle is repeated. When t_k corresponding to $\hat{z} = 1.0$ is reached, the program stops.

USE:

Input:

INFO	Data identification
AATACK	Angle of attack
RE	Reynolds number based on body length
LENGTH	Dimensional body length
DELT	Finite difference time step
RC	Vortex core cutoff value
SIGMA	Empirical vortex flux factor
KFINAL	Program cycle termination index
TFINAL	Program central processor time termination value
ZFINAL	Nondimensional length termination value
LR	READ continue data from TAPE(LR) if LR=0 no binary continue data is READ
LW	WRITE continue data on TAPE(LW)
KPUN	Vortex positions, velocities, and strengths are punched every KPUN cycles
LEVEL	Printed output level

Output:

PI π

DTOR $\pi/180$

RTOD $180/\pi$

DMAX Maximum body diameter

AW Characteristic length \bar{a} , of the 2DUS flow

F Fineness ratio, f

SA Frontal area, S

RE Re_{2DUS}

KS Initially equal to 1000, KS is set equal to K when the first point vortex is born from the b.l.

KR Initially equal to 1000, KR is set equal to K when the backflow velocity exceeds a value of .1

KT Equal to the number of t.b.l. vortices in the flow at time t_k

KB Equal to the number of b.b.l. vortices in the flow at time t_k

KRT Equal to the number of t.r.s.l. vortices in the flow at time t_k

KRB Equal to the number of b.r.s.l. vortices in the flow at time t_k

INITAL The initial dimension of the θ grid in the b.l.

NDIM Dimension of the vortex arrays, and of the drag array. The number of program cycles must not exceed NDIM

IDIM, JDIM b.l. grid dimension: (θ_i, \bar{r}_j)
 $i = 1, 2, \dots, IDIM$
 $j = 1, 2, \dots, JDIM$

PI2 2π

SQRTPI $\sqrt{\pi}$

SQA	Square of the sine of the angle of attack
SRE	$\sqrt{RE_{2DUS}}$
ISYM	Symmetric mode index ISYM = 1, symmetric mode ISYM = 0, asymmetric mode
THTASYM	Condition on θ_s , if THTASYM is greater than θ_s , the program goes into the asymmetric mode
K	Program time index, initially set equal to 0 at t_0 . K is incremented by 1 for each successive step in time
KTS	Index used in the continuation of a run, if $KTS \geq K$ the b.l. grid is calculated
TK	t_k
TI	t_0
ZHAT	\hat{z}
AK	Radius at time t_k , $a(t_k)$
AKTI	$a(t_0)$
AKDOT	$\dot{a}(t_k)$
CDPI	Initial drag coefficient due to pressure = $2\pi a(t_0) \dot{a}(t_0)$
KMINUS1	$K-1$
TH	$t_{k-1/2}$
AK	Radius at time t_k , $a(t_k)$
AKDOT	$\dot{a}(t_k)$
AADOT	$a(t_k) \dot{a}(t_k)$
AKSQD	$[a(t_k)]^2$
ZHALF	$\hat{z}(t_{k-1/2})$
AKPHALF	$a(t_{k-1/2})$
AKDOTPH	$\dot{a}(t_{k-1/2})$
KTTEMP	Temporary storage of the value of KT

KBTEMP	Temporary storage of the value of KB
U	Boundary layer velocity, $\bar{u}(\bar{r}, t_k, \theta)$
DGAMMA	$(\Gamma)_t$
SMALLM	m_k
CDST	$\int_0^{\theta_s} \tau \sin \theta \, d\theta \quad 0 < \theta_s < \pi$
CLST	$\int_0^{\theta_s} \tau \cos \theta \, d\theta \quad 0 < \theta_s < \pi$
ISEP	An integer denoting the element in the θ_i array, the value of which is the separation angle
TRAPD	Trapezoidal sum used in the evaluation of CDST and CDSB
TRAPL	Trapezoidal sum used in the evaluation of CLST and CLSB
CDSK	Viscous contribution to the drag coefficient = $\frac{1}{2} \int_0^{2\pi} \tau \sin \theta \, a(t) \, d\theta$
TT	Array of t.b.l. vortex diffusion times
TB	Array of b.b.l. vortex diffusion times
TRT	Array of t.r.s.l. vortex diffusion times
TRB	Array of b.r.s.l. vortex diffusion times
X, Y	Arrays of t.b.l. vortex coordinates
XDOT, YDOT	Arrays of t.b.l. vortex velocities ($\dot{u}^\circ, \dot{v}^\circ$)
GAMMA	Array of t.b.l. vortex strengths
XB, YB	Arrays of b.b.l. vortex coordinates
XDOTB, YDOTB	Arrays of b.b.l. vortex velocities

GMAB Array of b.b.l. vortex strengths

XRT, YRT Arrays of t.r.s.l. vortex coordinates

XRDOT, YRDOT Arrays of t.r.s.l. vortex velocities

GMRT Array of t.r.s.l. vortex strengths

XRB, YRB Arrays of b.r.s.l. vortex coordinates

XRDOTB, YRDOTB Arrays of b.r.s.l. vortex velocities

GMRB Array of b.r.s.l. vortex strengths

CDSB $\int_0^{\theta_s} \tau \sin \theta d\theta \quad \pi < \theta_s < 2\pi$

CLSB $\int_0^{\theta_s} \tau \cos \theta d\theta \quad \pi < \theta_s < 2\pi$

THETA θ

THETAS $\theta_s \quad 0 < \theta < \pi$

THETASB $\theta_s \quad \pi < \theta < 2\pi$

THETASR θ_s^r

IPUN Punch index

IPUN ≥ 1 , punch vortex arrays
IPUN < 1 , no vortex array punch

CDPK Drag coefficient due to pressure = 2π

$\frac{1}{2} \int_0^{2\pi} C_p \cos \theta a(t) d\theta$

CDK Drag coefficient, $C_D(t)$

CDN Local normal force distribution coefficient, C_n

TIME Execution time

SUBPROGRAMS CALLED: BLBOX, DQI, DRZRO, NONDIM, PDRAG, RSEP, RVTX, RZRO, SECOND, VEL, VM, WRIT

FUNCTION AN

PURPOSE: To calculate the coefficients a_n, b_n, c_n

METHOD: Given the b.l. velocity distribution the coefficients a_n, b_n, c_n are calculated as outlined in reference 1.

USE: Function reference to

AN (L, N, IDIM, JDIM, U, W)

Input:

L, N The b.l. grid point (θ_i, \bar{r}_j)

IDIM, JDIM b.l. grid dimension (θ_i, \bar{r}_j)
 $i = 1, 2, \dots, IDIM$
 $j = 1, 2, \dots, JDIM$

U $u(\theta_i, t_k, \bar{r}_j)$

W $u(\theta_i, t_k, \bar{r}_j)$

DS

DZ8

DZSQ2 See BLBOX

DZSQ4

Output:

AN Coefficient a_n in the finite difference solution of the b.l. equations

CALLED FROM: BLBOX

REMARKS: Alternate ENTRY points to AN are BN, and CN

SUBROUTINE BLBOX

PURPOSE: To numerically integrate the unsteady b.l. equations

METHOD: A finite difference method is employed to solve the two dimensional unsteady boundary layer equations, (reference 1).

USE: CALL BLBOX (SMALL M, DGAMMA, TAWD, TAWL, U, W, UT, UB, IDIM, JDIM, MODE)

Input:

U The b.l. velocity distribution $u(\theta_i, t_{k-1}, \bar{r}_j)$

UT 2-dimensional storage away for $u(\theta_i, t_{k-1}, \bar{r}_j)$

UB 2-dimensional storage away for $u(\theta_i, t_{k-1}, \bar{r}_j)$
 $\pi < \theta_i < 2\pi$

UTNBIG An array the elements of which are the boundary conditions (t.b.l.), for the finite difference scheme

UBNBIG Boundary conditions (b.b.l.)

IDIM, b.l. grid dimension (θ_i, \bar{r}_j)
 JDIM $i = 1, 2, \dots, \text{IDIM}$
 $j = 1, 2, \dots, \text{JDIM}$

MODE If MODE = 1 appropriate t.b.l. u^i values will be input to the b.l. grid. If MODE = 2 b.b.l. u^i values are input

KTS Index used in the continuation of a run, if $KTS \geq K$ the b.l. grid is calculated

AK $a(t_k)$

AKSQD $[a(t_k)]^2$

AKDOT $\dot{a}(t_k)$

AKPHALF $a(t_{k-1/2})$

IXTRSET An integer denoting the element in the S array, the value of which is the t.b.l. separation angle

IXBRSET An integer denoting the element in the S array, the value of which is the b.b.l. separation angle

NBIG An integer defining the last element in the AN array

Output:

DELS $\Delta\theta$ used to define the b.l. grid $\theta_{i+1} = \theta_i + \Delta\theta$

DS Defining the elements of the array DS as, $\Delta\theta_{i+1} = \theta_{i+1} - \theta_i$, allows for the grid defined to have a variable mesh spacing.

S Array of θ_i values

ST Array of t.b.l. θ_i values

SB Array of b.b.l. θ_i values

DZ b.l. grid spacing $\Delta\bar{r} = .14$

DZ2 $1/2 \Delta\bar{r}$

DZSQ2 $1/2 \Delta\bar{r}^2$

DZSQ4 $1/4 \Delta\bar{r}^2$

ZN Array of \bar{r} values

U The b.l. velocity distribution $u(\theta_i, t_k, \bar{r}_j)$

UT 2-dimensional storage array for $u(\theta_i, t_k, \bar{r}_j)$
 $\pi < \theta_i < 2\pi$

UB 2-dimensional storage array for $u(\theta_i, t_k, \bar{r}_j)$
 $\pi < \theta_i < 2\pi$

TAUS Successive estimates of the surface shear τ

NCYCLE The number of iterative cycles, (reference 1)

DUDR $(\bar{u})_{\bar{r}}$

TAW τ

TAWD $\tau \sin \theta$ evaluated for values of the θ grid

TAWL $\tau \cos \theta$ evaluated for values of the θ grid

THETA θ

THETAS θ_s $0 < \theta < \pi$

THETASB θ_s $\pi < \theta < 2\pi$

SMALLM Location of vortex born from b.l., m_k

DGAMMA $(\Gamma)_t$

CALLED

FROM: VCF

SUBPROGRAMS

CALLED: IC, FREEVTX, POTFLOW, TRID, RITE

REMARKS:

The b.l. subroutine determines the velocity profile for $0 < \theta < \theta_s$ ($0 < \theta_s < \pi$), or for $0 < \theta < \theta_s$ ($\pi < \theta_s < 2\pi$). Thus if the flow is assumed to be symmetric, a finite difference solution is required only on the top half of the cylinder. If the flow is asymmetric, BLBOX is called twice with appropriate top or bottom grid values and boundary conditions being input.

FUNCTION CPC

PURPOSE: To calculate C_p

METHOD: The equations for evaluating C_p are given in part 1

USE: Function reference to

CPC (THTA)

Input:

TT

TB

TRT

TRB

X,
Y

XDOT,
YDOT

GAMMA See VCF

XB,
YB

XDOTB,
YDOTB

GMAB

XRT,
YRT

XRDOT,
YRDOT

GMRT

XRB,
YRB

XRDOTB,
YRDOTB

GMRB

K

KT

KB

See VCF

KRT

KRB

XX

X coordinate of the point on the cylinder surface at which C_p is to be evaluated

YY

Y coordinate of the point on the cylinder surface at which C_p is to be evaluated

NDIM

Dimension of the vortex arrays

Output:

PHIT

$$\phi_{,t} = (\phi_{,t})^V + (\phi_{,t})^P$$

PHIVT

$$(\phi_{,t})^V$$

PHIPT

$$(\phi_{,t})^P$$

PSIKR2

$$u_\circ$$

CPC

Pressure coefficient

CALLED

FROM:

VCF

SUBPROGRAMS

CALLED:

PV, FREEVTX, POTFLOW

REMARKS:

The time derivative of the potential function, $(\phi_{,t})$, see part 1, is computed by first determining the terms due to the point vortices $(\phi_{,t})^V$, adding to that the potential flow solution of a changing radius cylinder in a uniform flow $(\phi_{,t})^P$.

FUNCTION DN

PURPOSE: To determine the coefficient d_n , (reference 1)

METHOD: The equations for the calculation of d_n are given in reference 1

USE: Function reference to
 DN (L, N, IDIM, JDIM, U, W)

Input:

L, The b.l. grid point (θ_i, \bar{r}_j)
 N

IDIM, b.l. grid dimension (θ_i, \bar{r}_j)
 JDIM $i = 1, 2, \dots, IDIM$
 $j = 1, 2, \dots, JDIM$

U $u(\theta_i, t_k, \bar{r}_j)$

W $v(\theta_i, t_k, \bar{r}_j)$

DS

DZ2 see BLBOX

NBIG

DT2 $1/2 \Delta t_k$

Output:

DN Coefficient d_n in the finite difference solution
 of the b.l. equations

CALLED
FROM: BLBOX

SUBPROGRAMS
CALLED: UM, US, UL, AN

SUBROUTINE DQI

PURPOSE: To evaluate $\int_0^{2\pi} C_p \cos \theta d\theta$

METHOD: Numerical integration using Simpsons rule

$$\int_{x_0}^{x_2} f(x) dx \sim \frac{h}{3} [f(x_2) + 4f(x_1) + f(x_0)]$$

USE: CALL DQI (FCT, CK, K, N, ISYM)

Input:

FCT The name of the EXTERNAL FUNCTION SUBPROGRAM used; PDRAG to evaluate drag, PLIFT to evaluate lift.

K Program time index, see VCF

N An even integer which determines the integration stepsize h. The front of the cylinder is divided into 2N equal parts, the front here being $-60^\circ < \theta < 60^\circ$, and the back is divided into 8N equal parts. The stepsize, $h = (60/N)(\pi/180)$.

ISYM An integer either 0 or 1. A value of 1 denotes a symmetric flow, and the limits of integration are $(0, \pi)$ and the value of the integral is doubled. A value of 0 denotes an asymmetric flow and the limits of integration are $(0, 2\pi)$

Output:

CK Approximation to $\int_0^{2\pi} C_p \cos \theta d\theta$ by Simpson's Rule

CALLED FROM: VCF

SUBPROGRAMS CALLED: PDRAG

REMARKS: The value of N used in all testing was 6.

SUBROUTINE FREEVTX

PURPOSE: To calculate vortex induced velocities

METHOD: Each point vortex and its image induce a velocity at every other point in the field. The distance between the field point and the vortex location is determined and induced velocity calculated, see part 1. Summing the effects of all the point vortices yields the vortex induced velocity.

USE: CALL FREEVTX (PSIK1X, PSIK1Y, PSIK1R, IA)

Input:

IA An integer value from 1 to 5.
IA = 1 Field point is a t.b.l. vortex
IA = 2 Field point is a b.b.l. vortex
IA = 3 Field point is on the cylinder
IA = 4 Field point is a t.r.s.l. vortex
IA = 5 Field point is a b.r.s.l. vortex

TT

TB

TRT

TRB

X,
Y

XDOT,
YDOT

See VCF

GAMMA

XB,
YB

XDOTB,
YDOTB

GMAB

XRT,
YRT

XRDOT,
YRDOT

GMR

XRB,
YRB

XRDOTB,
YRDOTB

See VCF

GMRB

KT

KB

TRT

KRB

ALPHA

An integer denoting a particular point vortex,
the coordinates of which is the point at
which the velocity is being calculated.

XX

X coordinate of the point vortex at which
the velocity is being calculated.

YY

Y coordinate of the point vortex at which the
velocity is being calculated.

Output:

PSIK1X $(\hat{v}^{\circ})^v$

PSIK1Y $-(\hat{u}^{\circ})^v$

PSIK1R $(u^{\circ})^v$

CALLED
FROM:

BLBOX, VEL, CPC

SUBPROGRAMS
CALLED:

PSI

SUBROUTINE IC

PURPOSE: To calculate the initial boundary layer profile for the problem of a circular cylinder started impulsively from rest.

METHOD: Wundt's solution for the impulsive start of a circular cylinder, (reference 2).

USE: CALL IC (AKTI, U, IDIM, JDIM, MODE)

Input:

AKTI $a(t_0)$

IDIM, b.l. grid dimension (θ_i, \bar{r}_j)
 JDIM $i = 1, 2 \dots, \text{IDIM}$
 $j = 1, 2 \dots, \text{JDIM}$

MODE MODE = 1 $0 < \theta_i < \pi$
 MODE = 2 $\pi < \theta_i < 2\pi$

Output:

U $u(\theta_i, t_0, \bar{r}_j)$

UTNBIG An array, the elements of which are the boundary conditions (t.b.l.) for the finite difference scheme. $u(\theta_i, t_0, \bar{r}_j)$
 $j = \text{JDIM} \quad 0 < \theta_i < \pi$

UBNBIG Boundary conditions (b.b.l.) $u(\theta_i, t_0, \bar{r}_j)$
 $j = \text{JDIM} \quad \pi < \theta_i < 2\pi$

CALLED FROM: BLBOX

SUBPROGRAMS CALLED: PHIO, PH11, ZTA001, ZTA0011, ZTA02A1, ZTA02B1

REMARKS: Subroutine IC is called once from BLBOX when $t_k = t_0$ (K=1). A dummy subroutine may be substituted thereafter, reducing the central memory requirement. The IC subroutine and the subprograms called from IC are listed in Appendix Ib.

SUBROUTINE NONDIM

PURPOSE: To calculate the characteristic radius \tilde{a} , the maximum diameter d , and the reference area S , for the input body geometry.

METHOD: Values of $r_o^*(\hat{z}^*)$ are obtained from RZERO, the FUNCTION SUBPROGRAM input by the user, for values of \hat{z}^* along the body axis. The maximum $r_o^*(\hat{z}^*)$, doubled, is d . For a closed end body ($r_o^*(\ell) = 0$) the characteristic radius is obtained by using the trapezoidal rule to approximate the integral $1/\ell \int_0^\ell r_o^*(\hat{z}^*) dz^* = \tilde{a}$. For open ended geometries ($r_o^*(\ell) \neq 0$), then $\tilde{a} = d/2$.

USE: CALL NONDIM (DMAX, RW, AW, F, S, L, PI)

Input:

L Dimensional body length, ℓ

PI π

Output:

DMAX Maximum body diameter, d

RW $d/2$

AW Characteristic length, \tilde{a}

F Fineness ratio, ℓ/d

S Frontal area = $\pi d^2/4$

CALLED

FROM: VCF

SUBPROGRAMS

CALLED: RZERO

FUNCTION PDRAG

PURPOSE: To calculate $C_p(\theta) \cos \theta$

METHOD: Function reference to CPC

USE: Function reference to
PDRAG (THETA)

Input:

THETA θ

Output:

PDRAG $C_p(\theta) \cos \theta$

CALLED

FROM: VCF

SUBPROGRAMS

CALLED: CPC

REMARKS: See DQI

SUBROUTINE POTFLOW

PURPOSE: To calculate the velocity at a point in the outer flow due to the uniform inviscid flow about a circular cylinder.

METHOD: Evaluate potential flow solution at a field point

USE: CALL POTFLOW (PSIKXP, PSIKYP, PSIKRP, IA)

Input:

ALPHA Point vortex index

AKSQD $[a(t_k)]^2$

IA An integer value from 1 to 5
IA = 1 Field point is a t.b.l. vortex
IA = 2 Field point is a b.b.l. vortex
IA = 3 Field point is on the cylinder
IA = 4 Field point is a t.r.s.l. vortex
IA = 5 Field point is a b.r.s.l. vortex

X,
Y

XDOT,
YDOT

GAMMA

XB,
YB

XDOTB, YDOTB See VCF

GMAB

XRT,
YRT

XRDOT,
YRDOT

GMRT

XRB,
YRB

XRDOTB,
YRDOTB

See VCF

GMRB

Output:

PSIKXP $(\hat{v}^o)^P$

PSIKYP $-(\hat{u}^o)^P$

PSIKRP $(u^o)^P$

CALLED
FROM:

BLBOX, VEL, CPC

SUBROUTINE PSI

PURPOSE: To sum the induced vortex velocity

METHOD: The equations for the induced velocity of a point vortex are given in part 1.

USE: CALL PSI (X, Y, GMA, NDIM, KI, KF, IA, IB, SUM, SUM1, TK)

Input:

AK $a(t_k)$

AKSQD $[a(t_k)]^2$

XX, Coordinates of the point at which the velocity
YY is being calculated

X, Input arrays of point vortex locations
Y

GMA Input array of point vortex strengths

NDIM Dimension of the input arrays

KI, Sum over KF - KI + 1 vortices
KF

TK Input array of vortex diffusion times

RC r_c

IA Field point index (See FREEVTX)

IB Integer which determines if a point vortex
is to be ignored in the sum

Output:

SUM \tilde{v}° induced by input vortex array

SUM1 $-\tilde{u}^\circ$ induced by input vortex array

**CALLED
FROM:** FREEVTX

SUBROUTINE PV

PURPOSE: To calculate the vortex contribution to C_p
METHOD: Equations for the calculation of ϕ_t are given in part 1.
USE: CALL PV (X, Y, XDT, YDT, GMA, NDIM, KI, KF, SUM, TK)

Input:

AK $a(t_k)$
AKSQD $[a(t_k)]^2$
XX, YY Coordinates of the point on the cylinder at which C_p is to be evaluated
X, Y Input arrays of point vortex locations
XDT, YDT Input arrays of point vortex velocities
GMA Input array of point vortex strengths
NDIM Dimension of the input arrays
KI, KF Sum over $KF - KI + 1$ vortices
TK Input array of vortex diffusion times
RC r_c

Output:

SUM ϕ_t due to input vortex array

CALLED FROM: CPC

SUBROUTINE RITE

PURPOSE: To print the b.l. velocity distribution

METHOD: ENCODE is used to create execution time FORMAT statements, so that only the velocity profile for $0 < \theta < \theta_s$ is printed.

USE: CALL RITE (MT, IDIM, JDIM, U, ZN, NX, S, DEG, ZHAT, DELS, K)

Input:

MT An integer value equal to 1 or 2
 If MT = 1 the initial velocity profile is printed
 If MT = 2 the velocity profile at t_k is printed

IDIM, JDIM b.l. grid dimension (θ_i, \bar{r}_j)
 $i = 1, 2, \dots, IDIM$
 $j = 1, 2, \dots, JDIM$

U $u(\theta_i, t_k, \bar{r}_j)$

ZN Array of \bar{r}_j values $j = 1, 2, \dots, JDIM$

NX An array containing a count of the number of iterative cycles. (reference 1).

S Array of θ_i values $i = 1, 2, \dots, IDIM$

DEG Array of θ_i values in degrees

ZHAT \hat{z}

DELS $\Delta\theta$

K Program cycle index

CALLED

FROM: BLBOX, IC

REMARKS: When MT = 1, u^i is output, when MT = 2 \bar{u} is output.

SUBROUTINE RSEP

PURPOSE: To determine the r.s.l. separation angle

METHOD:
$$\frac{\theta_s^f - \theta_m^f}{\theta_o^f - \theta_m^f} = \frac{\theta_m^r - \theta_s^r}{\theta_m^r - \theta_o^r}$$
 (see part 1)

USE: CALL RSEP (THTASF, THTASR, THTA, UBL, INTX1, INITIAL, MODE)

Input:

THTASF: b.l. separation angle, θ_s^f

THTA: Array of θ_i values $i = 1, 2, \dots, \text{INITIAL}$

UBL: Array of u_i^o (θ_i) values $i = 1, 2, \dots, \text{INITIAL}$

INTX1: An integer denoting the element of the THTA array which is the separation angle

INITIAL: The initial dimension of the b.l. grid, ($\theta_i = \pi$) $i = \text{INITIAL}$

MODE = 1 $0 < \theta_i < \pi$

MODE = 2 $\pi < \theta_i < 2\pi$

Output:

THTASR: r.s.l. separation angle, θ_s^r

CALLED

FROM: VCF

SUBROUTINE RVTX

PURPOSE: To calculate the position and strength of the r.s.l. vortices

METHOD: Subroutine RSEP has determined the r.s.l. separation angle. The strength of the vortex that would be born is $\Gamma = -\Delta t u_0^2/2$. If this value of Γ does not exceed .1 it is stored in arrays. The next time cycle (t_{k+1}), a new θ_s^r is calculated, and if the new separation angle exceeds the previous one then the previous point vortex is aborted. However, if the new θ_s^r is less than the previous value of θ_s^r , the previous point vortex is lumped together with the one now in question. After five time cycles ($t_k < t < t_{k+5}$), or when the lumped vortex strength becomes .1 or greater, the lumped vortex is placed into the outer flow. The angle at which it is placed is the average of the five previous separation angles. See part 1 for equations.

USE: CALL RVTX (X, Y, GMA, THTASR, THTASM, THTAS, UZZ, UZZSQ, GAMA, MODE, KRN, N, NF, IFLAG, BMATL, NDIM, TK)

Input:

AK	$a(t_k)$
DELT	Δt_k
THETAS	θ_s
THTASR	θ_s^r
THTASM	θ_s^r at the time previous to r.s.l. vortex birth
THTAS	Storage array for θ_s^r values
UZZ	Storage array for u_0 values
UZZSQ	Storage array for u_0^2 values
GAMA	Storage array for Γ values
MODE	MODE = 1 $0 < \theta < \pi$ MODE = 2 $\pi < \theta < 2\pi$
N	An integer count of the number of time cycles completed since the previous r.s.l. vortex was placed in the outer flow.

NF Lump parameter, usually $NF = 5$. When $N = NF$ a lumped point vortex is placed in the outer flow.

NDIM Dimension of vortex arrays

TK An array of vortex diffusion times

Output:

GMATL Lumped vortex strength

IFLAG If $IFLAG(N) = 0$, the point vortex aborted
If $IFLAG(N) = 1$, the point vortex is lumped

X, Coordinates of r.s.l. vortex born
Y

GMA Strength of r.s.l. vortex born

SMALLM m_{rk}

KRN The KRN^{th} element of the r.s.l. vortex arrays

CALLER

FROM: VCF

SUBPROGRAMS

CALLER: FREEVTX, POTFLOW

FUNCTION RZRO

PURPOSE: To calculate the nondimensional body radius r_o as a function of nondimensional distance \hat{z} along the body axis.

METHOD: The dimensional quantities are multiplied by characteristic dimensions.

USE: Function reference to
RZRO (ZHAT, L, RW)

Input:

ZHAT	\hat{z}
L	l
RW	$d/2$

Output:

RZRO	r_o
------	-------

CALLED FROM: VCF

SUBPROGRAMS CALLED: RZERO

FUNCTION UAPRX1

PURPOSE: To calculate a first approximation to $u(\theta_{i+1}, t_k, \bar{r}_j)$

METHOD: $u(\theta_{i+1}, t_k, \bar{r}_j) = u(\theta_{i+1}, t_{k-1}, \bar{r}_j) + u(\theta_i, t_k, \bar{r}_j) - u(\theta_i, t_{k-1}, \bar{r}_j)$
 See reference 1, for further explanation.

USE: Function reference to
 UAPRX1 (L, N, IDIM, JDIM, U)

Input:

L, N	The b.l. grid point (θ_i, \bar{r}_j)
IDIM, JDIM	b.l. grid dimension (θ_i, \bar{r}_j) $i = 1, 2, \dots, IDIM$ $j = 1, 2, \dots, JDIM$
U	$u(\theta_i, t_{k-1}, \bar{r}_j)$

Output:

U	First approximation to $u(\theta_{i+1}, t_k, \bar{r}_j)$
---	--

CALLED FROM: BLBOX, DN

REMARKS: Alternate ENTRY points to UAPRX1 are US, UL, and UM

SUBROUTINE VEL

PURPOSE: To calculate the velocity of the point vortices

METHOD: Each point vortex has a velocity due to the uniform flow around a circular cylinder, plus the velocity induced by all of the other point vortices.

USE: CALL VEL (X, Y, XDT, YDT, NDIM, KI, KF, IA)

Input:

X, Y Input arrays of vortex locations

NDIM Dimension of the vortex arrays

KI, KF KI - KI + 1 is the number of vortices in an array

IA Vortex field point index (see FREEVTX)

Output:

XDT, YDT Output arrays of vortex velocities

CALLED

FROM: VCF, RVTX

SUBPROGRAMS

CALLED: FREEVTX, POTFLOW

SUBROUTINE VM

PURPOSE: To calculate the coordinates of the point vortices at time t_{k+1} .

METHOD: Each point vortex travels $\tilde{u}^o \Delta t_k$ in the x direction, and $\tilde{v}^o \Delta t_k$ in the y direction.

USE: CALL VM (X, Y, XDT, YDT, NDIM, KI, KF, IA)

Input:

- X, Y Vortex locations at t_k
- XDT, YDT Input arrays of vortex velocities
- KI, KF KF - KI + 1 is the number of vortices in an array
- IA Vortex field point index (see FREEVTX)

Output:

- R Distance from cylinder to vortex
- XTEMP, YTEMP Temporary storage of the input vortex locations
- X, Y Vortex locations at t_{k+1}

CALLED FROM: VCF

SUBPROGRAMS CALLED: VMFIX

REMARKS: VMFIX is called only if $R < a(t_k)$

SUBROUTINE VMFIX

PURPOSE: To calculate a corrected point vortex location.

METHOD: The Δt_k approximation involved in the motion of the vortices allows for the possibility that a point vortex may cross the cylinder boundary. Since this is physically impossible VMFIX appends the vortex motion and places the point vortex outside of the cylinder. VMFIX calculates the tangent to the cylinder at the point where a vortex has crossed the cylinder boundary. Appended \tilde{u}° , \tilde{v}° are returned to VM such that the point vortex will travel along the tangent line.

USE: CALL VMFIX (X, Y, XDOT, YDOT)

Input:

X, Vortex location inside of cylinder
Y

XDOT, Original \tilde{u}° , \tilde{v}°
YDOT

DEF Some small parameter (.001), applied to the transformations to insure that the appended velocities will result in a new vortex location outside of the cylinder.

Output:

L Quadrant in which the point vortex is located

XDOT, Appended \tilde{u}° , \tilde{v}°
YDOT

**CALLED
FROM:**

VCF

REMARKS:

VM calls VMFIX if $R < a(t_k)$ (see VM), upon returning to VM the vortex location is recalculated starting with the original vortex position.

SUBROUTINE WRIT

PURPOSE: Print, punch, vortex locations, strengths, and velocities.

METHOD: Tape 6 = Output, Tape 7 = Punch

USE: CALL WRIT (X, Y, XDOT, YDOT, GAMMA, XB, YB, GMAB, KIT, KFT, KIB, KFB, K, NDIM, IW)

Input:

X, Y Input array of t.b.l. or t.r.s.l. vortex locations

XDOT, YDOT Input array of t.b.l. or t.r.s.l. vortex velocities

GAMMA Input array of t.b.l. or t.r.s.l. vortex strengths

XB, YB Input array of b.b.l. or b.r.s.l. vortex locations

XDOTB, YDOTB Input array of b.b.l. or b.r.s.l. vortex velocities

GMAB Input array of b.b.l. or b.r.s.l. vortex strengths

KIT, KFT KFT - KIT + 1 = number of t.b.l. or t.r.s.l. vortices to be output

KIB, KFB KFB - KIB + 1 = number of b.b.l. or b.r.s.l. vortices to be output

IW IW = 6 Print
IW = 7 Punch
IW = 9 Write on TAPE 9

CALLED FROM: VCF

APPENDIX Ia

PROGRAM ADYNF AND SUBPROGRAM DESCRIPTIONS

This appendix contains a brief outline of the purpose, method, and use of program ADYNF and subroutine FIT. Subroutine RZRO is described in APPENDIX I. Subroutines SPLINEB and CADRE are not described in this APPENDIX but are listed in APPENDIX II.

PROGRAM ADYNF

PURPOSE: To calculate C_N and $C_{M,\lambda}$

METHOD: Values of $C_D(\hat{z})$ and \hat{z} output from PROGRAM VCF are input on cards to PROGRAM ADYNF. The normal force and moment are given by

$$C_N = (4f/\pi) \int_0^1 c_n d\hat{z}, \text{ and}$$

$$C_{M,\lambda} = - (4f/\pi) \int_0^1 \left[\hat{z} + \frac{r_o}{4f^2} \frac{dr_o}{d\hat{z}} \right] c_n d\hat{z} + \lambda C_N$$

respectively. Approximations to these integrals are evaluated by first obtaining a least squares cubic spline approximation to the integrand function and then by integrating the approximate function. The integration interval is divided into sections, and a cubic polynomial approximates the integrand in each section. The endpoints of the sections are called "knots" the approximating cubics being continuous up to the third derivative at the "knots."

USE:

Input:

INFO	Data Identification
AATAACK	Angle of attack
LAMBDA	Moment arm coefficient
LENGTH	Dimensional body length
F	Fineness Ratio
AW	Characteristic length, \tilde{a}
RW	Maximum Radius, $d/2$
NOKNOT	Number of knots used to segment the interval [0,1]
LX	The number of $C_D(\hat{z})$ values
X	Array of \hat{z} values

PORCD Array of $C_D(\hat{z})$ values
 XI Array of knot positions, the length of the array equals NOKNOT

Output:

ITER Number of sweeps through opt,(reference 4)

XI Knot values after optimization

COEFL Cubic coefficients

ERRL2 Least square error, spline approximation

ERRL1 Average error, spline approximation

ERRL99 Maximum error, spline approximation

XX \hat{z}

FCTL Spline approximation to integrand at \hat{z}

PRINT Scaled error, spline approximation

CN Normal force coefficient C_N

CNI Approximation to the integral $\int_0^1 c_n d\hat{z}$

ERROR Computed absolute error in CNI (See Appendix II - CADRE)

IFLAG An integer between 1 and 5 indicating what difficulties were met with in obtaining an approximation to CNI (See Appendix II - CADRE)

CM Moment coefficient, $C_{M,\lambda}$

CMI Approximation to the integral

$$\int_0^1 \left[\hat{z} + \frac{r_o}{4f^2} \frac{dr_o}{d\hat{z}} \right] c_n d\hat{z}$$

SUBPROGRAMS

CALLED: SPLINEB, CADRE, FIT, RZRO

REMARKS:

VCF punched output \hat{z} , $C_D(\hat{z})$ is input to ADYNF. The first value of \hat{z} output by VCF is $\hat{z} = .03$, and the final value of \hat{z} is usually slightly less than 1.0. It is known that at $\hat{z} = 0$, $C_D = 0$, and by extrapolating the given data to $\hat{z} = 1.0$, two additional data points are input to ADYNF. See the ADYNF PROGRAM INPUT DATA section, and APPENDIX IV - ADYNF SAMPLE INPUT, for further explanation of the input data.

To evaluate the C_N and $C_{M,\lambda}$ integrals, the integrand functions are approximated using cubic splines. This smooths the data, and allows for the integrand to be approximated at any \hat{z} which permits the use of the integration routine CADRE. CADRE has been proven to be a very successful numerical integration scheme and errors in the calculation of C_N , $C_{M,\lambda}$ are believed to be less than 2%. Subprograms RZRO and RZERO are called by ADYNF. For listings and use of CADRE and SPLINEB see Appendix II.

FUNCTION FIT

PURPOSE: To determine the integrand values $[c_n]$ or

$$\left[\hat{z} + \frac{r_0}{4f^2} \frac{dr_0}{d\hat{z}} \right] c_n \text{ for arbitrary values of } \hat{z}, 0 < \hat{z} < 1.$$

METHOD: The cubic coefficients have been supplied by SPLINEB.
The set of coefficients used to approximate the integrand depends upon knot locations and \hat{z} .

USE: Function reference to FIT (X)

Input:

X	\hat{z}
XI	Knot locations
COEFL	Cubic coefficients

Output:

FIT	Cubic spline approximation to the integrand function
-----	--

CALLED
FROM:

ADYNF

REMARKS:

For further information on spline approximations see, references 3 and 4 and SPLINE listing in Appendix II.

APPENDIX Ib

SUBROUTINE IC

This appendix contains a listing of SUBROUTINE IC and subprograms referenced by IC. After the first program cycle is completed SUBROUTINE IC and its subprograms are no longer required, and can be replaced by a dummy SUBROUTINE IC. The coding for the dummy program follows the original SUBROUTINE IC listing.

Description of Parameters

Variable or FUNCTION Subprogram	Description [2]
Q	q
QP	q'
QPP	q''
ETA	η
ZTA001(ETA)	$\zeta'_{00}(n)$
ZTA02A1(ETA)	$\zeta'_{0,2a}(n)$
ZTA02B1(ETA)	$\zeta'_{0,2b}(n)$
PHIO(ETA)	$\Phi_0(n)$
PHI1(ETA)	$\Phi_1(n)$
ZTA011(ETA)	$\zeta'_{01}(n)$


```

SUBROUTINE IC(AKI,U,IDIM,JDIM,MODE)
DIMENSION U(IDIM,2,JDIM)
DIMENSION NX(53),DEG(53)
DIMENSION THTA(53)
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RT00,RC,RMAXSQ,SIGMA,LEVEL
COMMON/BLOCK20/DX,DZ,INTX1,NBIG1
COMMON/BLOCK14/S(53),ST(53),SB(53)
COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB
COMMON/BLBOX2/TAU,PT4,NBIG
COMMON/BLBOX12/ZN(51),ISEP
COMMON/BLBOX13/KTS,IXRSET,IXBRSET,UTNBIG(53),UBNBIG(53)
COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT
COMMON/BLBOX4/A2,A0,B1,C0,D2,D0,E1,E0
COMMON/BLBOX5/A4A,A2A,A0A,P3A,B1A,C2A,C0A,D3A,D1A,E4A,E2A,E0A,F0A
COMMON/BLBOX6/G4A,G2A,G1A,G0A,H3A,H2A,H1A,H0A,I2A,I1A,I0A
COMMON/BLBOX7/K4A,K2A,K1A,K0A,L3A,L2A,L1A,L0A
COMMON/BLBOX8/A4B,A2B,A0B,B3B,B1B,C2B,C0B,D3B,D1B,E4B,E2B,E0B,F0B
COMMON/BLBOX9/G4B,G2B,G1B,G0B,H3B,H2B,H1B,H0B,I2B,I1B,I0B
COMMON/BLBOX10/K4B,K2B,K1B,K0B,L3B,L2B,L1B,L0B
REAL IP, KP, LP
REAL I2, I2A, I2B, I1, I1A, I1B, I0, I0A, I0B
REAL K4, K4A, K4B, K2, K2A, K2B, K1, K1A, K1B, K0, K0A, K0B
REAL L3, L3A, L3B, L2, L2A, L2B, L1, L1A, L1B, L0, L0A, L0B
IF(MODE.EQ.2) GO TO 1
C*****
C
C   WUNDT DATA
C
C*****
A2= 1.
A0= -.5
P1= 1.5
C0= .5
D2= 3.0+4./(3.*PI)
D0= -.5+2./(3.*PI)
F1= 2.+2./(3.*PI)
E0= -2./(3.*SQRT(PI))
A4A=-1.0/3.0
A2A=-1.0
A0A=1.0/4.0
P3A=-2.0/3.0
B1A=-2.0
C2A=-5.0/12.0
C0A=-7.0/6.0
D3A=9.0/40.0
D1A=133.0/240.0
E4A=9.0*SQRT(3.0)/(5.0*PI)
E2A=27.0*SQRT(3.0)/(5.0*PI)
F0A=27.0*SQRT(3.0)/(20.0*PI)
F0A=8.0*SQRT(2.0)/(15.0*SQRT(PI))
G4A=-5.0/6.0+2.0/(9.0*PI)
G2A=-5.0/2.0+2.0/(3.0*PI)
G1A=-4.0/(3.0*SQRT(PI))
G0A=3.0/8.0-1.0/(2.0*PI)
H3A=-13.0/12.0+1.0/(3.0*PI)
H2A=2.0/(3.0*SQRT(PI))
H1A=-73.0/24.0+23.0/(18.0*PI)
H0A=-1.0/(3.0*SQRT(PI))

```

I2A=-1.0/3.0+1.0/(9.0*PI)	IC	60
I1A=1.0/(3.0*SQRT(PI))	IC	61
I0A=-5.0/6.0+4.0/(9.0*PI)	IC	62
K4A=-1.0/2.0-(2.0+27.0*SQRT(3.0))/(15.0*PI)-64.0/(135.0*PI**2)	IC	63
K2A=3.0*K4A	IC	64
Y1A=-4.0/SQRT(PI)-16.0/(9.0*PI*SQRT(PI))	IC	65
K0A=1.0/8.0-(46.0+81.0*SQRT(3.0))/(60.0*PI)-16.0/(45.0*PI**2)	IC	66
L3A=-1.0/2.0-(2.0+27.0*SQRT(3.0))/(30.0*PI)-32.0/(135.0*PI**2)	IC	67
L2A=2.0/(3.0*SQRT(PI))	IC	68
L1A=-19.0/12.0-(22.0+81.0*SQRT(3.0))/(36.0*PI)-16.0/(27.0*PI**2)	IC	69
L0A=(-17.0+8.0*SQRT(2.0))/(15.0*SQRT(PI))-16.0/(15.0*PI*SQRT(PI))	IC	70
A4B=1.0/3.0	IC	71
A2B=0.0	IC	72
A0B=1.0/4.0	IC	73
B3B=7.0/12.0	IC	74
B1B=-3.0/8.0	IC	75
C2B=1.0/3.0	IC	76
C0B=-5.0/12.0	IC	77
D3B=-9.0/160.0	IC	78
D1B=-31.0/320.0	IC	79
E4B=-E4A/4.0	IC	80
E2B=-E2A/4.0	IC	81
E0B=-E0A/4.0	IC	82
F0B=F0A	IC	83
G4B=1.5+2.0/(3.0*PI)	IC	84
G2B=.5+2.0/(3.0*PI)	IC	85
G1B=0.0	IC	86
G0B=5.0/8.0-1.0/(6.0*PI)	IC	87
H3B=7.0/4.0+7.0/(9.0*PI)	IC	88
H2B=0.0	IC	89
H1B=-1.0/8.0+5.0/(6.0*PI)	IC	90
H0B=0.0	IC	91
I2B=0.5+2.0/(9.0*PI)	IC	92
I1B=0.0	IC	93
I0B=-1.0/4.0+2.0/(9.0*PI)	IC	94
K4B=7.0/6.0+(24.0+9.0*SQRT(3.0))/(20.0*PI)+32.0/(45.0*PI**2)	IC	95
K2B=0.5+(136.0+81.0*SQRT(3.0))/(60.0*PI)+32.0/(15.0*PI**2)	IC	96
K1B=0.0	IC	97
K0B=3.0/8.0+(56.0+81.0*SQRT(3.0))/(240.0*PI)+8.0/(15.0*PI**2)	IC	98
L3B=7.0/12.0+(24.0+9.0*SQRT(3.0))/(40.0*PI)+16.0/(45.0*PI**2)	IC	99
L2B=0.0	IC	100
L1B=-5.0/24.0+(40.0+27.0*SQRT(3.0))/(48.0*PI)+8.0/(9.0*PI**2)	IC	101
L0B=(-2.0+8.0*SQRT(2.0))/(15.0*SQRT(PI))-8.0/(45.0*PI*SQRT(PI))	IC	102
C*****	IC	103
C	IC	104
C END WUNDT DATA	IC	105
C	IC	106
C*****	IC	107
1 CONTINUE	IC	108
DO 20 I=1,INTX1	IC	109
IF(MODE.EQ.1) S(I)=ST(I)	IC	110
IF(MODE.EQ.2) S(I)=SB(I)	IC	111
THTA(I)=S(I)	IC	112
NEG(I)=THTA(I)*RTOD		
***** BC Z=0	IC	113
U(I,1,1)=0.0	IC	114
U(I,2,1)=0.0	IC	115
DO 20 J=2,NBIG	IC	116

C***** B.C. THETA=0.0	IC	117
IF(I.NE.1) GO TO 11	IC	118
U(I,1,J)=0.0	IC	119
GO TO 20	IC	120
11 IF(J.NE.NBIG) GO TO 10	IC	121
U(I,1,NBIG)=2.0*SIN(THETA(I))	IC	122
IF(MODE.EQ.1) UTNBIG(I)=U(I,1,NBIG)	IC	123
IF(MODE.EQ.2) UBNBIG(I)=U(I,1,NBIG)	IC	124
GO TO 20	IC	125
10 CONTINUE	IC	126
ETA=ZN(J)/(2.0*SQRT(TI))	IC	127
Q=2.0*SIN(THETA(I))	IC	128
QP=2.0*COS(THETA(I))	IC	129
QPP=-2.0*SIN(THETA(I))	IC	130
U(I,1,J)=Q*(ZTA001(ETA)+TI*QP*ZTA011(ETA)+TI**2*(QP**2*ZTA02A1	IC	131
1(ETA)+Q*QPP*ZTA02B1(ETA)))	IC	132
20 CONTINUE	IC	133
C..... U AT T=TI GOES TO U(BAR) AT T=TI	IC	134
DO 21 I=1,INTX1	IC	136
UTNBIG(I)=AKTI*UTNBIG(I)	IC	137
UBNBIG(I)=AKTI*UBNBIG(I)	IC	138
DO 21 J=1,NBIG	IC	139
21 U(I,1,J)=AKTI*U(I,1,J)	IC	140
IF(MODE.EQ.2) GO TO 9	IC	141
IF(LEVEL.GE.4) CALL RITE(1,IDIM,JDIM,U,ZN,NX,S,DEG,ZHAT,DELS,K)		
9 CONTINUE	IC	157
WRITE(1)(UTNBIG(I),(U(I,1,J),J=1,NBIG),I=1,INTX1)		
RETURN	IC	163
END	IC	164

SUBROUTINE IC(AKTI,U,IDIM,JDIM,MODE)	IC	1
DIMENSION U(IDIM,2,JDIM)	IC	2
COMMON/BL90X13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53)	IC	3
READ(1)(UTNBIG(I),(U(I,1,J),J=1,51),I=1,53)	IC	4
RETURN	IC	5
END	IC	6

```
FUNCTION PHI0(EATA)
PHI0=ERF(EATA)-1.0
RETURN
END
```

```
PHI0 1
PHI0 2
PHI0 3
PHI0 4
```

```
FUNCTION PHI1(EATA)
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DLTA
COMMON/BLBOX2/TAU,PT4,NBIG
PHI1=2.0*EXP(-EATA**2)/SQRT(PI)
RETURN
END
```

```
PHI1 1
PHI1 2
PHI1 3
PHI1 4
PHI1 5
PHI1 6
```

ACTION ZTA001 (EATA)
EATA
A 001-ERF(E)
TURN
ND
1249

ZTA001 1
ZTA001 2
ZTA001 3
ZTA001 4
ZTA001 5

```

FUNCTION ZTA011 (EATA)
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DLTA
COMMON/BLBOX2/TAU,PT4,NBIG
COMMON/BLBOX4/A2,A0,B1,C0,D2,D0,E1,E0
E=E*ATA
A2= 1.0
A0=-0.5
B1= 1.5
C0= 0.5
D2=3.0+4.0/(3.0*PI)
D0=-0.5+2.0/(3.0*PI)
E1= 2.0+2.0/(3.0*PI)
E0=-2.0/(3.0*SQRT(PI))
ZTA011=(A2*E*E+A0)*PHI0(E)**2+B1*E*PHI0(E)*PHI1(E)+C0*PHI1(E)**2
ZTA011=ZTA011+(D2*E*E+D0)*PHI0(E)+(E1*E+E0)*PHI1(E)
RETURN
END

```

```

ZTA011 1
ZTA011 2
ZTA011 3
ZTA011 4
ZTA011 5
ZTA011 6
ZTA011 7
ZTA011 8
ZTA011 9
ZTA01 10
ZTA01 11
ZTA01 12
ZTA01 13
ZTA01 14
ZTA01 15
ZTA01 16
ZTA01 17

```

FUNCTION ZTA02A1(EATA)	ZTA02A 1
COMMON/BLBOX5/A4A,A2A,A0A,B3A,B1A,C2A,C0A,D3A,D1A,E4A,E2A,E0A,F0A	ZTA02A 2
COMMON/BLBOX6/G4A,G2A,G1A,G0A,H3A,H2A,H1A,H0A,I2A,I1A,I0A	ZTA02A 3
COMMON/BLBOX7/K4A,K2A,K1A,K0A,L3A,L2A,L1A,L0A	ZTA02A 4
REAL IP,KP,LP	ZTA02A 5
REAL I2,I2A,I2B,I1,I1A,I1B,I0,I0A,I0B	ZTA02A 6
REAL K4,K4A,K4B,K2,K2A,K2B,K1,K1A,K1B,K0,K0A,K0B	ZTA02A 7
REAL L3,L3A,L3B,L2,L2A,L2B,L1,L1A,L1B,L0,L0A,L0B	ZTA02A 8
E=EATA	ZTA02A 9
P0=PHI0(E)	ZTA02 10
P02=PHI0(E)**2	ZTA02 11
P03=PHI0(E)**3	ZTA02 12
P1=PHI1(E)	ZTA02 13
P12=PHI1(E)**2	ZTA02 14
P13=PHI1(E)**3	ZTA02 15
ET2=E**2	ZTA02 16
ET3=E**3	ZTA02 17
ET4=E**4	ZTA02 18
A4=A4A	ZTA02 19
A2=A2A	ZTA02 20
A0=A0A	ZTA02 21
B3=B3A	ZTA02 22
B1=B1A	ZTA02 23
C2=C2A	ZTA02 24
C0=C0A	ZTA02 25
D3=D3A	ZTA02 26
D1=D1A	ZTA02 27
E4=E4A	ZTA02 28
E2=E2A	ZTA02 29
E0=E0A	ZTA02 30
F0=F0A	ZTA02 31
G4=G4A	ZTA02 32
G2=G2A	ZTA02 33
G1=G1A	ZTA02 34
G0=G0A	ZTA02 35
H3=H3A	ZTA02 36
H2=H2A	ZTA02 37
H1=H1A	ZTA02 38
H0=H0A	ZTA02 39
I2=I2A	ZTA02 40
I1=I1A	ZTA02 41
I0=I0A	ZTA02 42
K4=K4A	ZTA02 43
K2=K2A	ZTA02 44
K1=K1A	ZTA02 45
K0=K0A	ZTA02 46
L3=L3A	ZTA02 47
L2=L2A	ZTA02 48
L1=L1A	ZTA02 49
L0=L0A	ZTA02 50
AP=(A4*ET4+A2*ET2+A0)*P03	ZTA02 51
BP=(B3*ET3+B1*E)*P02*P1	ZTA02 52
CP=(C2*ET2+C0)*P0*P12	ZTA02 53
DP=(D3*ET3+D1*E)*P13	ZTA02 54
SQRT2E=SQRT(2.0)*E	ZTA02 55
SQRT3E=SQRT(3.0)*E	ZTA02 56
EP=(E4*ET4+E2*ET2+E0)*PHI0(SQRT3E)	ZTA02 57
FP=F0*PHI0(SQRT2E)*P1	ZTA02 58

GP= (G4*E⁴+G2*ET²+G1*E+G0) *P02
HP= (H3*ET³+H2*ET²+H1*E+H0) *P0*P1
IP= (I2*ET²+I1*E+I0) *P12
KP= (K4*E⁴+K2*ET²+K1*E+K0) *P0
LP= (L3*ET³+L2*ET²+L1*E+L0) *P1
ZTA02A1= AP+BP+CP+DP+EP+FP+GP+HP+IP+KP+LP
RETURN
END

ZTA02 59
ZTA02 60
ZTA02 61
ZTA02 62
ZTA02 63
ZTA02 64
ZTA02 65
ZTA02 66

FUNCTION ZTA02B1(EATA)	ZTA02B 1
COMMON/BL80X8/A4B,A2B,A0B,B3B,B1B,C2B,C0B,D3B,D1B,E4B,E2B,E0B,F0B	ZTA02B 2
COMMON/BL80X9/G4B,G2B,G1B,G0B,H3B,H2B,H1B,H0B,I2B,I1B,I0B	ZTA02B 3
COMMON/BL80X10/K4B,K2B,K1B,K0B,L3B,L2B,L1B,L0B	ZTA02B 4
REAL IP, KP, LP	ZTA02B 5
REAL I2, I2A, I2B, I1, I1A, I1B, I0, I0A, I0B	ZTA02B 6
REAL K4, K4A, K4B, K2, K2A, K2B, K1, K1A, K1B, K0, K0A, K0B	ZTA02B 7
REAL L3, L3A, L3B, L2, L2A, L2B, L1, L1A, L1B, L0, L0A, L0B	ZTA02B 8
E=EATA	ZTA02B 9
P0=PHI0(E)	ZTA02 10
P02=PHI0(E)**2	ZTA02 11
P03=PHI0(E)**3	ZTA02 12
P1=PHI1(E)	ZTA02 13
P12=PHI1(E)**2	ZTA02 14
P13=PHI1(E)**3	ZTA02 15
ET2=E**2	ZTA02 16
ET3=E**3	ZTA02 17
ET4=E**4	ZTA02 18
A4=A4B	ZTA02 19
A2=A2B	ZTA02 20
A0=A0B	ZTA02 21
B3=B3B	ZTA02 22
B1=B1B	ZTA02 23
C2=C2B	ZTA02 24
C0=C0B	ZTA02 25
D3=D3B	ZTA02 26
D1=D1B	ZTA02 27
E4=E4B	ZTA02 28
E2=E2B	ZTA02 29
E0=E0B	ZTA02 30
F0=F0B	ZTA02 31
G4=G4B	ZTA02 32
G2=G2B	ZTA02 33
G1=G1B	ZTA02 34
G0=G0B	ZTA02 35
H3=H3B	ZTA02 36
H2=H2B	ZTA02 37
H1=H1B	ZTA02 38
H0=H0B	ZTA02 39
I2=I2B	ZTA02 40
I1=I1B	ZTA02 41
I0=I0B	ZTA02 42
K4=K4B	ZTA02 43
K2=K2B	ZTA02 44
K1=K1B	ZTA02 45
K0=K0B	ZTA02 46
L3=L3B	ZTA02 47
L2=L2B	ZTA02 48
L1=L1B	ZTA02 49
L0=L0B	ZTA02 50
AP=(A4*ET4+A2*ET2+A0)*P03	ZTA02 51
BP=(B3*ET3+B1*E)*P02*P1	ZTA02 52
CP=(C2*ET2+C0)*P0*P12	ZTA02 53
DP=(D3*ET3+D1*E)*P13	ZTA02 54
SQRT2E=SQRT(2.0)*E	ZTA02 55
SQRT3E=SQRT(3.0)*E	ZTA02 56
EP=(E4*ET4+E2*ET2+E0)*PHI0(SQRT3E)	ZTA02 57
FP=F0*PHI0(SQRT2E)*P1	ZTA02 58

GP= (G4*ET4+G2*ET2+G1*E+G0)*P02
HP= (H3*ET3+H2*ET2+H1*E+H0)*P0*P1
IP= (I2*ET2+I1*E+I0)*P12
KP= (K4*ET4+K2*ET2+K1*E+K0)*P0
LP= (L3*ET3+L2*ET2+L1*E+L0)*P1
ZTA 02B1= AP+BP+CP+DP+EP+FP+GP+HP+IP+KP+LP
RETURN
END

ZTA02 59
ZTA02 60
ZTA02 61
ZTA02 62
ZTA02 63
ZTA02 64
ZTA02 65
ZTA02 66

APPENDIX II

CANNED SUBROUTINE LISTINGS - CADRE

- SECOND

- SPLINE, ARITH1

- TRID

```

FUNCTION CADRE(F,A,B,AERR,RERR,LEVEL,ERROR,IFLAG) CADRE 1
C THIS FUNCTION RETURNS AN ESTIMATE *CADRE* FOR THE NUMBER CADRE 2
C INT = INTEGRAL OF *F*(X) FROM *A* TO *B* CADRE 3
C WHICH HOPEFULLY SATISFIES CADRE 4
C ABS(INT - *CADRE*) .LE. AMAX1(*AERR*, *RERR* TIMES ABS(INT)). CADRE 5
C THE PROGRAM USES CAUTIOUS ADAPTIVE ROMBERG EXTRAPOLATION. CADRE 6
C IN THIS SCHEME, THE INTEGRAL IS CALCULATED AS THE SUM OF INTEGRALS CADRE 7
C OVER SUITABLY SMALL SUBINTERVALS. ON EACH SUBINTERVAL, AN ESTIMATE CADRE 8
C *VINT*, WITH ESTIMATED ABSOLUTE ERROR *ERRER*, IS FOUND BY CAUTIOUS CADRE 9
C ROMBERG EXTRAPOLATION. IF *ERRER* IS SMALL ENOUGH, *VINT* IS ACCEPT CADRE 10
C ED AND ADDED TO *CADRE*, AND *ERRER* IS ADDED TO *ERROR*. OTHERWISE CADRE 11
C THE SUBINTERVAL IS HALVED, AND EACH HALF IS CONSIDERED SEPARATELY. CADRE 12
C INFORMATION ABOUT THE OTHER HALF BEING TEMPORARILY STACKED. CADRE 13
C CADRE 14
C ***** INPUT ***** CADRE 15
C CADRE 16
C F THE NAME OF A SINGLE-ARGUMENT REAL FUNCTION SUBPROGRAM CADRE 17
C THIS NAME MUST APPEAR IN THE CALLING PROGRAM IN AN CADRE 18
C EXTERNAL STATEMENT. CADRE 19
C A,B THE TWO ENDPONTS OF THE INTERVAL OF INTEGRATION CADRE 20
C AERR CADRE 21
C RERR DESIRED ABSOLUTE AND RELATIVE ERROR IN THE ANSWER CADRE 22
C LEVEL AN INTEGER INDICATING DESIRED LEVEL OF PRINTOUT CADRE 23
C .LE. 1, NO PRINTOUT, CADRE 24
C = 2, SUCCESS OR FAILURE MESSAGE, AND LIST OF SINGULAR- CADRE 25
C ITIES ENCOUNTERED (IF ANY). CADRE 26
C = 3, IN ADDITION, ALL SUBINTERVALS CONSIDERED ARE LISTED CADRE 27
C TOGETHER WITH THE KIND OF REGULAR BEHAVIOUR FOUND CADRE 28
C (IF ANY), CADRE 29
C = 4, IN ADDITION, ALL RATIOS CONSIDERED ARE LISTED AS IS CADRE 30
C INFO ON WHICH DECISION PROCEDURE IS BASED, CADRE 31
C .GE. 5, IN ADDITION, ALL T-TABLES ARE LISTED. CADRE 32
C CADRE 33
C ***** OUTPUT ***** CADRE 34
C CADRE 35
C CADRE ESTIMATE OF THE INTEGRAL, RETURNED VIA THE FUNCTION CALL. CADRE 36
C ERROR ESTIMATED BOUND ON THE ABSOLUTE ERROR OF THE NUMBER *CADRE* CADRE 37
C IFLAG AN INTEGER BETWEEN 1 AND 5 INDICATING WHAT DIFFICULTIES CADRE 38
C WERE MET WITH: SPECIFICALLY CADRE 39
C = 1, ALL IS WELL. CADRE 40
C = 2, ONE OR MORE SINGULARITIES WERE SUCCESSFULLY HANDLED. CADRE 41
C = 3, IN SOME SUBINTERVAL(S), THE ESTIMATE *VINT* WAS ACCEPT CADRE 42
C ED MERELY BECAUSE *ERRER* WAS SMALL, EVEN THOUGH NO CADRE 43
C REGULAR BEHAVIOUR COULD BE RECOGNIZED. CADRE 44
C = 4, FAILURE, OVERFLOW OF STACK *TS* (THIS HAS NEVER HAPPEN CADRE 45
C ED, - SO FAR). CADRE 46
C = 5, FAILURE, TOO SMALL A SUBINTERVAL IS REQUIRED; THIS MAY CADRE 47
C BE DUE TO TOO MUCH NOISE IN THE FUNCTION (RELATIVE TO CADRE 48
C THE GIVEN ERROR REQUIREMENTS) OR DUE TO A PLAIN ORNERY CADRE 49
C INTEGRAND. CADRE 50
C A VERY CAUTIOUS MAN WOULD ACCEPT *CADRE* ONLY IF IFLAG IS 1 OR 2; CADRE 51
C THE MERELY REASONABLE MAN WOULD KEEP THE FAITH EVEN IF IFLAG IS 3. CADRE 52
C THE ADVENTUROUS MAN IS QUITE OFTEN RIGHT IN ACCEPTING *CADRE* CADRE 53
C EVEN IF IFLAG IS 4 OR 5. CADRE 54
C CADRE 55
C ***** LIST OF MAJOR VARIABLES ***** CADRE 56
C CADRE 57
C CUREST BEST ESTIMATE SO FAR FOR CADRE 58
C INT - (INTEGRAL OVER CURRENTLY CONSIDERED SUBINTERVAL). CADRE 59
C FNSIZE MAXIMUM AVERAGE FUNCTION SIZE SO FAR ENCOUNTERED, CADRE 60

```

```

C  ERRR  RELATIVE ERROR REQUIREMENT USED; DERIVED FROM INPUT *PERR*      CADRF 61
C      AND CHOSEN TO LIE BETWEEN .1 AND 10 TIMES *TOLMCH*.                CADRF 62
C  ERRA  = ABS(*AERR*)                                                    CADRF 63
C  STAGE (MORE OR LESS) EQUAL TO  $\frac{1}{2}$  TO THE -(*ISTAGE*)                   CADRF 64
C  THESE FIVE QUANTITIES ARE USED IN THE DETERMINATION OF THE LOCAL      CADRF 65
C  ERROR REQUIREMENT.                                                    CADRF 66
C                                                                              CADRF 67
C  STEPMN MINIMUM SUBINTERVAL LENGTH PERMITTED.                          CADRF 68
C  TS     STACK OF VALUES OF F(x) SO FAR COMPUTED BUT NOT YET           CADRF 69
C        SUCCESSFULLY USED.                                              CADRF 70
C  ISTAGE AN INTEGER INDICATING THE HEIGHT OF THE STACK OF INTERVALS    CADRF 71
C        YET TO BE PROCESSED.                                           CADRF 72
C                                                                              CADRF 73
C        ***** LIST OF PARAMETERS *****                               CADRF 74
C                                                                              CADRF 75
C  TOLMCH DEPENDS ON THE LENGTH OF FLOATING POINT MANTISSA; SHOULD BE    CADRF 76
C        ABOUT 1.E-7 FOR 27 BINARY BIT MANTISSA AND                      CADRF 77
C        ABOUT 1.E-13 FOR 48 BINARY BIT MANTISSA.                       CADRF 78
C  AITLOW SHOULD BE SOMEWHAT GREATER THAN 1.                            CADRF 79
C  H2TOL.                                                                    CADRF 80
C  AITTOL.                                                                    CADRF 81
C  JUMPTL TOLERANCES USED IN THE DECISION PROCESS TO RECOGNIZE          CADRF 82
C        H**2 CONVERGENCE, X**ALPHA TYPE CONVERGENCE, OR                CADRF 83
C        JUMP-TYPE CONVERGENCE OF THE TRAPEZOID SUMS.                   CADRF 84
C  MAXTS.                                                                    CADRF 85
C  MAXTBL.                                                                    CADRF 86
C  MXSTGE ARE THE THREE DIFFERENT UPPER LIMITS FOR THE DIMENSION OF     CADRF 87
C        THE VARIOUS ARRAYS.                                            CADRF 88
C                                                                              CADRF 89
C        ***** PROGRAM LAYOUT *****                                  CADRF 90
C                                                                              CADRF 91
C        INITIALIZATION:                                               CADRF 92
C  5-6   BEGIN WORK ON NEXT SUBINTERVAL                                  CADRF 93
C  9-14  GET NEXT TRAPEZOID SUM                                         CADRF 94
C  15-19 GET RATIOS, PRELIMINARY DECISION PROCEDURE.                  CADRF 95
C  20-   ESTIMATE *VINT* ASSUMING SMOOTH INTEGRAND                     CADRF 96
C  30-   ESTIMATE *VINT* ASSUMING INTEGRAND HAS X**ALPHA TYPE         CADRF 97
C        SINGULARITY                                                  CADRF 98
C  40-   NO LUCK WITH THIS TRAPEZOID SUM. GET NEXT ONE OR GET OUT.    CADRF 99
C  50-   ESTIMATE *VINT* ASSUMING INTEGRAND HAS JUMP                   CADRF100
C  60-   ESTIMATE *VINT* ASSUMING INTEGRAND IS STRAIGHT LINE         CADRF101
C  70-   ESTIMATE *VINT* ASSUMING VARIATION IN INTEGRAND              CADRF102
C        IS MOSTLY NOISE.                                             CADRF103
C  80-   INTEGRATION OVER CURRENT SUBINTERVAL SUCCESSFUL.            CADRF104
C        SET UP NEXT SUBINTERVAL, IF ANY, OR RETURN.                 CADRF105
C  90-   INTEGRATION OVER CURRENT SUBINTERVAL NOT SUCCESSFUL.        CADRF106
C        MARK CURRENT SUBINTERVAL FOR SUBDIVISION AND SET UP        CADRF107
C        NEXT SUBINTERVAL.                                           CADRF108
C  900-  FAILURE.                                                    CADRF109
C                                                                              CADRF110
C        DIMENSION T(10,10),H(10),AIT(10),DIF(10),RN(4),            CADRF111
C        *   TS(2049),IBEGS(30),BEGIN(30),FINIS(30),EST(30)         CADRF112
C        REAL LENGTH, JUMPTL                                           CADRF113
C        LOGICAL H2CCNV,AITKEN,RIGHT,REGLAR,REGLSV(30)                CADRF114
C        DATA TOLMCH,AITLOW,H2TCL,AITTOL,JUMPTL,MAXTS,MAXTBL,MXSTGE  CADRF115
C        * / 2.E-13, 1.1 , .15 , .1 , .01 , 2049, 10 , 30/         CADRF116
C        DATA RN/.71420053,.34662815,.843751..12633046/            CADRF117
C        DATA ALG402 /.3010249956639795/                            CADRF118
C        CADRE = 0.                                                    CADRF119
C        ERRCR = 0.                                                    CADRF120
C        IFLAG = 1                                                    CADRF121

```

```

LENGTH = ABS(B-A)
IF (LENGTH .EQ. 0.) RETURN
ERRR = AMIN1(.1,AMAX1(ABS(RERR), 10.*TOLMCH))
ERRA = ABS(AERR)
STEPMN = AMAX1(LENGTH/FLOAT(2**MXSTGE),
* AMAX1(LENGTH,ABS(A),ABS(B))*TOLMCH)
STAGE = .5
ISTAGE = 1
CUREST = 0.
FNFSIZE = 0.
PREVER = 0.
REGLAR = .FALSE.
C
C THE GIVEN INTERVAL OF INTEGRATION IS THE FIRST INTERVAL CONSIDERED.
C
C BEG = A
C FBEG = F(BEG)/2.
C TS(1) = FBEG
C IBEG = 1
C END = B
C FEND = F(END)/2.
C TS(2) = FEND
C IEND = 2
C
C 5 RIGHT = .FALSE.
C
C INVESTIGATION OF A PARTICULAR SUBINTERVAL BEGINS AT THIS POINT.
C ***** MAJOR VARIABLES *****
C
C BEG, ENDCPOINTS OF THE CURRENT INTERVAL
C END
C FBEG,
C FEND ONE HALF THE VALUE OF F(X) AT THE ENDCPOINTS
C STEP SIGNED LENGTH OF CURRENT SUBINTERVAL
C ISTAGE HEIGHT OF CURRENT SUBINTERVAL IN STACK OF SUBINTERVALS
C
C RIGHT A LOGICAL VARIABLE INDICATING WHETHER CURRENT SUBINTERVAL
C IS RIGHT HALF OF PREVIOUS SUBINTERVAL. NEEDED IN 80FF AND
C 90FF TO DECIDE WHAT INTERVAL TO LOCK AT NEXT.
C
C TS(I), I=IBEG,...,IEND, CONTAINS THE FUNCTION VALUES FOR THIS
C SUBINTERVAL SO FAR COMPUTED. SPECIFICALLY,
C TS(I) = F(BEG + (I-IBEG)/(IEND-IBEG)*STEP), ALL I
C EXCEPT THAT TS(IREG) = FBEG, TS(IEND) = FEND
C
C REGLAR A LOGICAL VARIABLE INDICATING WHETHER OR NOT THE CURRENT
C SUBINTERVAL IS REGULAR (SEE NOTES)
C
C H2CONV A LOGICAL VARIABLE INDICATING WHETHER H**2 CONVERGENCE OF
C THE TRAPEZOID SUMS FOR THIS INTERVAL IS RECOGNIZED,
C
C AITKEN A LOGICAL VARIABLE INDICATING WHETHER CONVERGENCE OF RATIOS
C FOR THIS SUBINTERVAL IS RECOGNIZED
C
C T CONTAINS THE FIRST *L* ROWS OF THE ROMBERG T-TABLE FOR THIS
C SUBINTERVAL IN ITS LOWER TRIANGULAR PART. SPECIFICALLY,
C T(I,1) = TRAPEZOID SUM (WITHOUT THE FACTOR *STEP*)
C ON 2**(I-1) + 1 EQUISPACED POINTS, I=1,...,L,
C T(I,J+1) = T(I,J) + (T(I,J)-T(I-1,J))/(4**J - 1),
C J=2,...,I-1, I=2,...,L.
C
C FURTHER, THE STRICTLY UPPER TRIANGULAR PART OF T CONTAINS
C THE RATIOS FOR THE VARIOUS COLUMNS OF THE T-TABLE.
C SPECIFICALLY,
C T(J,I) = (T(I,J)-T(I-1,J))/(T(I+1,J)-T(I,J)),
C I=J+1,...,I-1, J=1,...,L-2.
C
C FINALLY, THE LAST OR L-TH COLUMN CONTAINS
C T(J,L) = T(L,J) - T(L-1,J), J=1,...,L-1.
C
C 6 STEP = END - BEG

```

CADRE122
CADRE123
CADRE124
CADRE125
CADRE126
CADRE127
CADRE128
CADRE129
CADRE130
CADRE131
CADRE132
CADRE133
CADRE134
CADRE135
CADRE136
CADRE137
CADRE138
CADRE139
CADRE140
CADRE141
CADRE142
CADRE143
CADRE144
CADRE145
CADRE146
CADRE147
CADRE148
CADRE149
CADRE150
CADRE151
CADRE152
CADRE153
CADRE154
CADRE155
CADRE156
CADRE157
CADRE158
CADRE159
CADRE160
CADRE161
CADRE162
CADRE163
CADRE164
CADRE165
CADRE166
CADRE167
CADRE168
CADRE169
CADRE170
CADRE171
CADRE172
CADRE173
CADRE174
CADRE175
CADRE176
CADRE177
CADRE178
CADRE179
CADRE180
CADRE181
CADRE182

```

ASTEP = ABS(STEP)
IF (ASTEP .LT. STEPMN) GO TO 950
IF (LEVEL .GE. 3) WRITE(6,609) BEG,STEP,ISTAGE
609 FORMAT(10H BEG,STEP ,2E16.8,1R)
T(1,1) = FBEG + FEND
TABS = ABS(FBEG) + ABS(FEND)
L = 1
N = 1
H2CONV = .FALSE.
AITKEN = .FALSE.

GO TO 10

C
9 IF (LEVEL .GE. 4) WRITE(6,692) L,T(1,LM1)
10 LM1 = L
L = L + 1

C
CALCULATE THE NEXT TRAPEZOID SUM. T(L,1), WHICH IS BASED ON
C *N2* + 1 EQUISPACED POINTS. HERE. N2 = N*2 = 2**(L-1) .
N2 = N*2
FN = N2
ISTEP = (IEND - IBEG)/N
IF (ISTEP .GT. 1) GO TO 12
II = IEND
IEND = IEND + N
IF (IEND .GT. MAXTS) GO TO 900
HCVN = STEP/FN
III = IEND
DO 11 I=1,N2,2
TS(III) = TS(II)
TS(III-1) = F(END - FLOAT(I)*HCVN)
III = III-2
11 II = II-1
ISTEP = 2
12 ISTEP2 = IBEG + ISTEP/2
SUM = 0.
SUMABS = 0.
DO 13 I=ISTEP2,IEND,ISTEP
SUM = SUM + TS(I)
13 SUMABS = SUMABS + ABS(TS(I))
T(L,1) = T(L-1,1)/2. + SUM/FN
TABS = TABS/2. + SUMABS/FN
ABSI = ASTEP*TABS
N = N2

C
C GET PRELIMINARY VALUE FOR *VINT* FROM LAST TRAPEZOID SUM AND
C UPDATE THE ERROR REQUIREMENT *ERGOAL* FOR THIS SUBINTERVAL.
C THE ERROR REQUIREMENT IS NOT PRORATED ACCORDING TO THE LENGTH OF
C THE CURRENT SUBINTERVAL RELATIVE TO THE INTERVAL OF INTEGRATION,
C BUT ACCORDING TO THE HEIGHT *ISTAGE* OF THE CURRENT SUBINTERVAL
C IN THE STACK OF SUBINTERVALS YET TO BE DONE.
C THIS PROCEDURE IS NOT HACKED BY ANY RIGOROUS ARGUMENT, BUT
C SEEMS TO WORK.
IT = 1
VINT = STEP*T(L,1)
TABTLM = TABS*TOLMCH
FNFSIZE = AMAX1(FNFSIZE,ABS(T(L,1)))
ERGOAL = AMAX1(ASTEP*TOLMCH*FNFSIZE,
* STAGE*AMAX1(ERRA,ERRR*ABS(CUREST+VINT)))

C
COMPLETE ROW L AND COLUMN L OF *T* ARRAY.
FEXTRP = 1.

```

CADRE183
CADRE184
CADRE185
CADRE186
CADRE187
CADRE188
CADRE189
CADRE190
CADRE191
CADRE192
CADRE193
CADRE194
CADRE195
CADRE196
CADRE197
CADRE198
CADRE199
CADRE200
CADRE201
CADRE202
CADRE203
CADRE204
CADRE205
CADRE206
CADRE207
CADRE208
CADRE209
CADRE210
CADRE211
CADRE212
CADRE213
CADRE214
CADRE215
CADRE216
CADRE217
CADRE218
CADRE219
CADRE220
CADRE221
CADRE222
CADRE223
CADRE224
CADRE225
CADRE226
CADRE227
CADRE228
CADRE229
CADRE230
CADRE231
CADRE232
CADRE233
CADRE234
CADRE235
CADRE236
CADRE237
CADRE238
CADRE239
CADRE240
CADRE241
CADRE242
CADRE243


```

DO 14 I=1,LM1
FEXTRP = FEXTRP*4.
T(I,L) = I(L,I) - T(L-1,I)
14 T(L,I+1) = T(L,I) + T(I,L)/(FEXTRP-1.)
ERROR = ASTEP*ABS(T(1,L))
-----
C--- PRELIMINARY DECISION PROCEDURE -----
C
C IF L = 2 AND T(2,1) = I(1,1), GO TO 60 TO FOLLOW UP THE IMPRESSION.
C THAT INTEGRAND IS STRAIGHT LINE.
    IF (L .GT. 2)                GO TO 15
    IF (ABS(I(1,2)) .LE. TABTLM)   GO TO 60
                                GO TO 10
C
CALCULATE NEXT RATIOS FOR COLUMNS 1, ..., L-2 OF T-TABLE
C RATIO IS SET TO ZERO IF DIFFERENCE IN LAST TWO ENTRIES OF
C COLUMN IS ABOUT ZERO.
15 DO 16 I=2,LM1
    DIFF = 0.
    IF (ABS(T(I-1,L)) .GT. TABTLM) DIFF = T(I-1,LM1)/T(I-1,L)
16 T(I-1,LM1) = DIFF
C
C T(1,LM1) IS THE RATIO DERIVED FROM LAST THREE TRAPEZOID SUMS, I.E.,
C  $T(1,LM1) = (T(L-1,1) - T(L-2,1)) / (T(L,1) - T(L-1,1))$ 
C IF THIS RATIO IS ABOUT 4, GO TO 26 FOR ROMBERG EXTRAPOLATION.
C IF THIS RATIO IS ZERO, I.E., IF LAST TWO TRAPEZOID SUMS ARE ABOUT
C EQUAL, GO TO 18 FOR POSSIBLE NOISE CHECK.
C IF THIS RATIO IS ABOUT 2 IN ABSOLUTE VALUE, GO TO 50 WITH THE
C BELIEF THAT INTEGRAND HAS JUMP DISCONTINUITY.
C IF THIS RATIO IS, AT LEAST, ABOUT EQUAL TO THE PREVIOUS RATIO, THEN
C THE INTEGRAND MAY WELL HAVE A NICE INTEGRABLE SINGULARITY.
C GO TO 30 TO FOLLOW UP THIS HUNCH.
    IF (ABS(4. - T(1,LM1)) .LE. H2TOL) GO TO 20
    IF (T(1,LM1) .EQ. 0.)            GO TO 18
    IF (ABS(2. - ABS(T(1,LM1)))) .LT. JUMPTL) GO TO 50
    IF (L .EQ. 3)                    GO TO 9
    H2CONV = .FALSE.
    IF (ABS((T(1,LM1) - T(1,L-2))/T(1,LM1)) .LE. AITTOL)
        GO TO 30
C AT THIS POINT, NO REGULAR BEHAVIOUR WAS DETECTED.
C IF CURRENT SUBINTERVAL IS NOT REGULAR AND ONLY FOUR TRAPEZOID SUMS
C WERE COMPUTED SO FAR, TRY ONE MORE TRAPEZOID SUM.
C IF, AT LEAST, LAST TWO TRAPEZOID SUMS ARE ABOUT EQUAL, THEN
C FAILURE TO RECOGNIZE REGULAR BEHAVIOUR MAY WELL BE DUE TO NOISE.
C GO TO 70 TO CHECK THIS OUT.
C OTHERWISE, GO TO 9 FOR FURTHER SUBDIVISION.
C
17 IF (REGLAR)                    GO TO 18
    IF (L .EQ. 4)                  GO TO 9
18 IF (ERROR .LE. ERGOAL)          GO TO 70
    IF (LEVEL .GE. 4) WRITE (A,695) L,T(1,LM1)
                                GO TO 91
-----
CALTIOUS ROMBERG EXTRAPOLATION -----
C
C THE CURRENT, OR L-TH, ROW OF THE ROMBERG T-TABLE HAS L ENTRIES.
C FOR J=1, ..., L-2, THE ESTIMATE
C  $STEP * T(L, J+1)$ 
C IS BELIEVED TO HAVE ITS ERROR BOUNDED BY
C  $ABS(STEP * (T(L, J) - T(L-1, J))) / (4 ** J - 1)$ 
C IF THE LAST RATIO

```

CADRE244
 CADRE245
 CADRE246
 CADRE247
 CADRE248
 CADRE249
 CADRE250
 CADRE251
 CADRE252
 CADRE253
 CADRE254
 CADRE255
 CADRE256
 CADRE257
 CADRE258
 CADRE259
 CADRE260
 CADRE261
 CADRE262
 CADRE263
 CADRE264
 CADRE265
 CADRE266
 CADRE267
 CADRE268
 CADRE269
 CADRE270
 CADRE271
 CADRE272
 CADRE273
 CADRE274
 CADRE275
 CADRE276
 CADRE277
 CADRE278
 CADRE279
 CADRE280
 CADRE281
 CADRE282
 CADRE283
 CADRE284
 CADRE285
 CADRE286
 CADRE287
 CADRE288
 CADRE289
 CADRE290
 CADRE291
 CADRE292
 CADRE293
 CADRE294
 CADRE295
 CADRE296
 CADRE297
 CADRE298
 CADRE299
 CADRE300
 CADRE301
 CADRE302
 CADRE303
 CADRE304

```

C      T(I,J,LM1) = (T(L-1,J)-T(L-2,J))/(T(L,J)-T(L-1,J))
C FOR COLUMN J OF THE T-TABLE IS ABOUT 4**J.
C THE FOLLOWING IS A SLIGHTLY RELAXED EXPRESSION OF THIS CRITERION.
20 IF (LEVEL .GE. 4) WRITE (6,619) L,T(1,LM1)
619 FORMAT(15,E16.8,5X6#H2CONV)
    IF (H2CONV) GO TO 21
    AITKEN = .FALSE.
    H2CONV = .TRUE.
    IF (LEVEL .GE. 3) WRITE (4,620) L
620 FORMAT(22H H2 CONVERGENCE AT ROW,I3)
21 FEXTRP = 4.
22 IT = IT + 1
    VINT = STEP*T(L,IT)
    ERRER = ABS(STEP/(FEXTRP-1.)*T(IT-1,L))
    IF (ERRER .LE. ERGOAL) GO TO 40
    IF (IT .EQ. LM1) GO TO 40
    IF (T(IT,LM1) .EQ. 0.) GO TO 22
    IF (T(IT,LM1) .LE. FEXTRP) GO TO 40
    IF (ABS(T(IT,LM1)/4.-FEXTRP)/FEXTRP .LT. AITTOL)
    * FEXTRP = FEXTRP*4.
    GO TO 22
-----
C--- INTEGRAND MAY HAVE X**ALPHA TYPE SINGULARITY-----
C RESULTING IN A RATIO OF *SING* = 2***(ALPHA + 1)
30 IF (LEVEL .GE. 4) WRITE (6,629) L,T(1,LM1)
629 FORMAT(15,E16.8,5X6#AITKEN)
    IF (T(1,LM1) .LT. AITLOW) GO TO 91
    IF (AITKEN) GO TO 31
    H2CONV = .FALSE.
    AITKEN = .TRUE.
    IF (LEVEL .GE. 3) WRITE (4,630) L
630 FORMAT(14H AITKEN AT ROW,I3)
31 FEXTRP = T(L-2,LM1)
    IF (FEXTRP .GT. 4.5) GO TO 21
    IF (FEXTRP .LT. AITLOW) GO TO 91
    IF (ABS(FEXTRP-T(L-3,LM1))/T(1,LM1) .GT. H2TOL)
    * GO TO 91
    IF (LEVEL .GE. 3) WRITE (6,631) FEXTRP
631 FORMAT(6H RATIO,F12.8)
    SING = FEXTRP
    FEXTM1 = FEXTRP - 1.
    DO 32 I=2,L
    AIT(I) = T(I,1) + (T(I,1)-T(I-1,1))/FEXTM1
    R(I) = T(1,I-1)
32 DIF(I) = AIT(I) - AIT(I-1)
    IT = 2
33 VINT = STEP*AIT(L)
    IF (LEVEL .LT. 5) GO TO 333
    WRITE (6,632) (R(I+1),I=IT,LM1)
    WRITE (6,632) (AIT(I),I=IT,L)
    WRITE (6,632) (DIF(I+1),I=IT,LM1)
632 FORMAT(1X,8E16.8)
333 ERRER = ERRER/FEXTM1
    IF (ERRER .GT. ERGOAL) GO TO 34
    ALPHA = ALOG10(SING)/ALG400 - 7.
    IF (LEVEL .GE. 2) WRITE (4,633) ALPHA,REG,END
633 FORMAT(11X42HINTEGRAND SHOWS SINGULAR BEHAVIOR OF TYPE
    * 4HX**(F4.2,9H) BETWEEN F5.8,4H ANDE15.8)
    IFLAG = MAX0(IFLAG,2)
    GO TO 40
34 IT = IT + 1

```

```

CADRE305
CADRE306
CADRE307
CADRE308
CADRE309
CADRE310
CADRE311
CADRE312
CADRE313
CADRE314
CADRE315
CADRE316
CADRE317
CADRE318
CADRE319
CADRE320
CADRE321
CADRE322
CADRE323
CADRE324
CADRE325
CADRE326
CADRE327
CADRE328
CADRE329
CADRE330
CADRE331
CADRE332
CADRE333
CADRE334
CADRE335
CADRE336
CADRE337
CADRE338
CADRE339
CADRE340
CADRE341
CADRE342
CADRE343
CADRE344
CADRE345
CADRE346
CADRE347
CADRE348
CADRE349
CADRE350
CADRE351
CADRE352
CADRE353
CADRE354
CADRE355
CADRE356
CADRE357
CADRE358
CADRE359
CADRE360
CADRE361
CADRE362
CADRE363
CADRE364
CADRE365

```

```

IF (IT .EG. LM1) GO TO 40 CADRE366
IF (IT .GT. 3) GO TO 35 CADRE367
H2NEXT = 4. CADRE368
SINGNX = 2.*SING CADRE369
35 IF (H2NEXT .LT. SINGNX) GO TO 36 CADRE370
FEXTRP = SINGNX CADRE371
SINGNX = 2.*SINGNX CADRE372
GO TO 37 CADRE373
36 FEXTRP = H2NEXT CADRE374
H2NEXT = 4.*H2NEXT CADRE375
37 DO 38 I=IT,LM1 CADRE376
R(I+1) = 0. CADRE377
38 IF (ABS(DIF(I+1)) .GT. TABTLM) R(I+1) = DIF(I)/DIF(I+1) CADRE378
IF (LEVEL .GE. 4) WRITE (6,63R) FEXTRP,R(L-1),R(L) CADRE379
638 FORMAT(16F,FEXTRP,*,RATIO5,3E15.8) CADRE380
H2TFEX = -H2TCL*FEXTRP CADRE381
IF (R(L) - FEXTRP .LT. H2TFEX) GO TO 40 CADRE382
IF (R(L-1)-FEXTRP .LT. H2TFEX) GO TO 40 CADRE383
ERRER = ASTEP*ABS(DIF(L)) CADRE384
FEXTM1 = FEXTRP - 1. CADRE385
DO 39 I=IT,L CADRE386
AIT(I) = AIT(I) + DIF(I)/FEXTM1 CADRE387
39 DIF(I) = AIT(I) - AIT(I-1) CADRE388
GO TO 33 CADRE389
C----- CADRE390
CURRENT TRAPEZOID SUM AND RESULTING EXTRAPOLATED VALUES DID NOT GIVE CADRE391
C A SMALL ENOUGH *ERRER*. CADRE392
C IF LESS THAN FIVE TRAPEZOID SUMS WERE COMPUTED SO FAR, TRY NEXT CADRE393
C TRAPEZOID SUM. CADRE394
C OTHERWISE, DECIDE WHETHER TO GO ON OR TO SUBDIVIDE AS FOLLOWS. CADRE395
C WITH T(L,IT) GIVING THE CURRENTLY BEST ESTIMATE, GIVE UP ON DEVELOP CADRE396
C ING THE T-TABLE FURTHER IF L .GT. IT*2. I.E., IF EXTRAPOLATION CADRE397
C DID NOT GO VERY FAR INTO THE T-TABLE. CADRE398
C FURTHER, GIVE UP IF REDUCTION IN *ERRER* AT THE CURRENT RATE CADRE399
C DOES NOT PREDICT AN *ERRER* LESS THAN *ERGOAL* BY THE TIME CADRE400
C *MAXTRL* TRAPEZOID SUMS HAVE BEEN COMPUTED. CADRE401
C ---NOTE--- CADRE402
C HAVING PREVER .LT. ERRER IS AN ALMOST CERTAIN SIGN OF BEGINNING CADRE403
C TROUBLE WITH NOISE IN THE FUNCTION VALUES. HENCE, CADRE404
C A WATCH FOR, AND CONTROL OF, NOISE SHOULD BEGIN HERE. CADRE405
40 FEXTRP = AMAX1(PREVER/ERRER,AITLOW) CADRE406
PREVER = ERRER CADRE407
IF (L .LT. 5) GO TO 10 CADRE408
IF (LEVEL .GE. 3) WRITE (6,641) ERRER,ERGOAL,FEXTRP,IT CADRE409
641 FORMAT(23H ERRER,ERGOAL,FEXTRP,IT,2E15.8,E14.5,I3) CADRE410
IF (L-IT .GT. 2 .AND. ISTAGE .LT. MYSTGE) GO TO 90 CADRE411
IF (ERRER/FEXTRP*(MAXTRL-1) .LT. ERGOAL) GO TO 10 CADRE412
GO TO 90 CADRE413
C----- CADRE414
C--- INTEGRAND HAS JUMP (SEE NOTES) ----- CADRE415
50 IF (LEVEL .GE. 4) WRITE (6,649) L,T(1),LM1) CADRE416
649 FORMAT(15,E16,R,5X4FJUMP) CADRE417
IF (ERRER .GT. ERGOAL) GO TO 90 CADRE418
C NOTE THAT 2*FN = 2*L CADRE419
DIFF = ABS(T(1-L))*2.*FN CADRE420
IF (LEVEL .GE. 2) WRITE (6,657) DIFF,BEG,END CADRE421
650 FORMAT(13X36HINTEGRAND SEEMS TO HAVE JUMP OF SIZEE13.6, CADRE422
* 8H BETWEENE15.R,4H ANDE15.R) CADRE423
GO TO 80 CADRE424
C----- CADRE425
C--- INTEGRAND IS STRAIGHT LINE ----- CADRE426

```

```

C TEST THIS ASSUMPTION BY COMPARING THE VALUE OF THE INTEGRAND AT
C FOUR #RANDOMLY CHOSEN# POINTS WITH THE VALUE OF THE STRAIGHT LINE
C INTERPOLATING THE INTEGRAND AT THE TWO ENDPOINTS OF THE SUB-
C INTERVAL. IF TEST IS PASSED, ACCEPT *VINT*.
60 IF (LEVEL .GE. 4) WRITE (6,660) L
660 FORMAT(15,21X13HSTRAIGHT LINE)
    SLOPE = (FEND-FBEG)*2.
    FBEG2 = FBEG*2.
    DO 61 I=1,4
    DIFF = ABS(F(REG+RN(I)*STEP) - FBEG2-RN(I)*SLOPE)
    IF (DIFF .GT. TABLM) GO TO 72
61 CONTINUE
    IF (LEVEL .GE. 3) WRITE (6,667) REG, END
667 FORMAT(27X43HINTEGRAND SEEMS TO BE STRAIGHT LINE BETWEEN
    * E15.8,4H AND E15.8)
    GO TO 80
-----
C----- NOISE MAY BE DOMINANT FEATURE -----
C ESTIMATE NOISE LEVEL BY COMPARING THE VALUE OF THE INTEGRAND AT
C FOUR #RANDOMLY CHOSEN# POINTS WITH THE VALUE OF THE STRAIGHT LINE
C INTERPOLATING THE INTEGRAND AT THE TWO ENDPOINTS. IF SMALL
C ENOUGH, ACCEPT *VINT*.
70 IF (LEVEL .GE. 4) WRITE (6,670) L,T(1,LM1)
670 FORMAT(15,E16.8,5X5HNOISE)
    SLOPE = (FEND-FBEG)*2.
    FBEG2 = FBEG*2.
    I = 1
71 DIFF = ABS(F(REG+RN(I)*STEP) - FBEG2-RN(I)*SLOPE)
72 ERROR = AMAX1(ERROR,ASTEP*DIFF)
    IF (ERROR .GT. ERGOAL) GO TO 91
    I = I+1
    IF (I .LE. 4) GO TO 71
    IF (LEVEL .GE. 3) WRITE (6,671) REG,END
671 FORMAT(15H NOISE BETWEEN ,E15.8,4H AND,E15.8)
    IFLAG = 3
-----
C----- INTEGRATION OVER CURRENT SUBINTERVAL SUCCESSFUL -----
C ADD *VINT* TO *CADRE* AND *ERROR* TO *ERROR*, THEN SET UP NEXT
C SUBINTERVAL, IF ANY.
80 CADRE = CADRE + VINT
    ERROR = ERROR + ERREF
    IF (LEVEL .LT. 3) GO TO 83
    IF (LEVEL .LT. 5) GO TO 82
    DO 81 I=1,L
81 WRITE (6,692) I,(T(I,J),J=1,L)
82 WRITE (6,682) VINT,ERROR,L,IT
682 FORMAT(12H INTEGRAL IS,E16.8,7H, ERROR,E15.8,9H FROM T(,
    * I1,1H,,I1,jH))
C
83 IF (RIGHT) GO TO 85
    ISTAGE = ISTAGE - 1
    IF (ISTAGE .EQ. 0) RETURN
C
    REGLAR = REGLSV(ISTAGE)
    BEG = BEGIN(ISTAGE)
    END = FINIS(ISTAGE)
    CUREST = CUREST - EST(ISTAGE+1) + VINT
    IEND = IBEG - 1
    FEND = TS(IEND)
    IBEG = IBEGS(ISTAGE)
    GO TO 94

```

CADRE427
 CADRE428
 CADRE429
 CADRE430
 CADRE431
 CADRE432
 CADRE433
 CADRE434
 CADRE435
 CADRE436
 CADRE437
 CADRE438
 CADRE439
 CADRE440
 CADRE441
 CADRE442
 CADRE443
 CADRE444
 CADRE445
 CADRE446
 CADRE447
 CADRE448
 CADRE449
 CADRE450
 CADRE451
 CADRE452
 CADRE453
 CADRE454
 CADRE455
 CADRE456
 CADRE457
 CADRE458
 CADRE459
 CADRE460
 CADRE461
 CADRE462
 CADRE463
 CADRE464
 CADRE465
 CADRE466
 CADRE467
 CADRE468
 CADRE469
 CADRE470
 CADRE471
 CADRE472
 CADRE473
 CADRE474
 CADRE475
 CADRE476
 CADRE477
 CADRE478
 CADRE479
 CADRE480
 CADRE481
 CADRE482
 CADRE483
 CADRE484
 CADRE485
 CADRE486
 CADRE487

85 CUREST = CUREST + VINT		CADRE488
STAGE = STAGE+2.		CADRE489
IEND = IBEG		CADRE490
IBEG = IBEGS(ISTAGE)		CADRE491
END = BEG		CADRE492
BEG = BEGIN(ISTAGE)		CADRE493
FEND = FBEG		CADRE494
FBEG = TS(IBEG)		CADRE495
	GO TO 5	CADRE496
C-----		
C--- INTEGRATION OVER CURRENT SUBINTERVAL IS UNSUCCESSFUL		CADRE497
C MARK SUBINTERVAL FOR FURTHER SUBDIVISION. SET UP NEXT SUBINTERVAL.		CADRE498
		CADRE499
90 REGLAR = .TRUE.		CADRE500
91 IF (ISTAGE .GE. MXSTGE)	GO TO 950	CADRE501
IF (LEVEL .LT. 5)	GO TO 93	CADRE502
DO 92 I=1,L		CADRE503
92 WRITE (6,692) I,(T(I,J),J=1,L)		CADRE504
692 FORMAT(15,7E16,8/3E16,8)		CADRE505
93 IF (RIGHT)	GO TO 95	CADRE506
REGLSV(ISTAGE+1) = REGLAR		CADRE507
BEGIN(ISTAGE) = BEG		CADRE508
IBEGS(ISTAGE) = IBEG		CADRE509
STAGE = STAGE/2.		CADRE510
94 RIGHT = .TRUE.		CADRE511
BEG = (BEG+END)/2.		CADRE512
IBEG = (IBEG+IEND)/2		CADRE513
TS(IBEG) = TS(IBEG)/2.		CADRE514
FBEG = TS(IBEG)		CADRE515
	GO TO 6	CADRE516
95 NNLEFT = IBEG - IBEGS(ISTAGE)		CADRE517
IF (IEND+NNLEFT .GE. MAXTS)	GO TO 900	CADRE518
III = IBEGS(ISTAGE)		CADRE519
II = IEND		CADRE520
DO 96 I=III,IBEG		CADRE521
II = II + 1		CADRE522
96 TS(II) = TS(I)		CADRE523
DO 97 I=IBEG,II		CADRE524
TS(III) = TS(I)		CADRE525
97 III = III + 1		CADRE526
IEND = IEND + i		CADRE527
IBEG = IEND - NNLEFT		CADRE528
FEND = FBEG		CADRE529
FBEG = TS(IBEG)		CADRE530
FINIS(ISTAGE) = END		CADRE531
END = BEG		CADRE532
BEG = BEGIN(ISTAGE)		CADRE533
BEGIN(ISTAGE) = END		CADRE534
REGLSV(ISTAGE) = REGLAR		CADRE535
ISTAGE = ISTAGE + 1		CADRE536
REGLAR = REGLSV(ISTAGE)		CADRE537
EST(ISTAGE) = .VIN		CADRE538
CUREST = CUREST + EST(ISTAGE)		CADRE539
	GO TO 5	CADRE540
C-----		
C--- FAILURE TO HANDLE GIVEN INTEGRATION PROBLEM		CADRE541
900 IF (LEVEL .GE. 2) WRITE (6,6900) BEG, END		CADRE542
6900 FORMAT(37H TOO MANY FUNCTION EVALUATIONS AROUND/		CADRE543
* 10X,E15.8,4H AND,E15.8)		CADRE544
IFLAG = 4		CADRE545
	GO TO 999	CADRE546
950 IFLAG = 5		CADRE547
		CADRE548

```

IF (LEVEL .LT. 2)          GO TO 999
IF (LEVEL .LT. 5)          GO TO 959
DO 958 I=1,L
958 WRITE (6,692) I,(7,(1,J),J=1,L)
959 WRITE (6,6959) BEG, END
6959 FORMAT(12X38HINTEGRAND SHOWS SINGULAR BEHAVIOUR OF
*      20HUNKNOWN TYPE BETWEEN 15.8,4H AND 15.8)
999 CADRE = CUREST + VINJ
      .RETURN
END

```

```

CADRE549
CADRE550
CADRE551
CADRE552
CADRE553
CADRE554
CADRE555
CADRE556
CADRE557
CADRE558

```

IDENT	SECOND		SECON	1
ENTRY	SECOND		SECON	2
SPACE	4		SECON	3
***	SECCND	= RETURN ACCUMULATED CPU TIME IN SECONDS.	SECON	4
*	DAVID S. CODSON.	09/21/71.	SECON	5
	SPACE	4	SECON	6
***	FORTRAN	CALL.	SECON	7
*			SECON	8
*			SECON	9
*	CALL	SECOND (X)	SECON	10
*		X = VARIABLE FOR RETURN OF ACCUMULATED CPU TIME IN	SECON	11
*		SECONDS AS A REAL NUMBER ACCURATE TO	SECON	12
*		MILLISECONDS. THIS ROUTINE USES THE	SECON	13
*		*TIMEMS* REQUEST WHICH IS A NON-STANDARD	SECON	14
*		FEATURE OF THE PURDUE OPERATING SYSTEM.	SECON	15
	TITLE	SECOND = RETURN ACCUMULATED CPL TIME IN SECONDS.	SECON	16
	TITLE	MAIN PROGRAM	SECON	17
SECOND	BSSZ	1 ENTRY/EXIT	SECON	18
	TIMEMS	READ ACCUMULATED CPU TIME IN MILLISECONDS	SECON	19
	SX1	1000 DIVIDE BY 1000	SECON	20
	OX1	X1	SECON	21
	NX1	X1	SECON	22
	PX6	X6	SECON	23
	NX6	X6	SECON	24
	FX6	X6/X1	SECON	25
	SA6	B1 STORE RESULT	SECON	26
	EQ	SECOND RETURN	SECON	27
	SPACE	4	SECON	28
	END		SFCON	29

C	SUBROUTINE SPLINE(NOKNOT)	***** 1
C	NONLINEAR SPLINE APPROXIMATION	SPLIN 2
C	PROGRAM WRITTEN BY CARL DE BOOR AND JOHN RICE	SPLIN 3
C	PURDUE UNIVERSITY	SPLIN 4
C	SUPPORTED BY THE NATIONAL SCIENCE FOUNDATION GP-4052,GP-7163	SPLIN 5
C	PLEASE REPORT ANY CASES OF INOPERATION TO THE AUTHORS.	SPLIN 6
C	THANKS	SPLIN 7
C	**** NUMERICAL ANALYSIS CONTROL ****	SPLIN 8
C	CONTROL PARAMETERS	SPLIN 9
C	ITER	SPLIN 10
C	NO. OF SWEEPS THRU OPT	SPLIN 11
C	BO (IN OPT)	SPLIN 12
C	IMPROVEMENT NEEDED TO REPEAT	SPLIN 13
C	EPSERR (IN SWEEP)	SPLIN 14
C	IMPROVEMENT NEEDED TO REPEAT	SPLIN 15
C	DIST (IN OPT NEAR 30,80)	SPLIN 16
C	KEEPS KNOTS SEPARATED	SPLIN 17
C	INDLP	SPLIN 18
C	NO. OF PASSES THRU OPT	SPLIN 19
C	THE FOLLOWING IS THE MAIN PROGRAM FOR VARYKNOT	SPLIN 20
C	DIMENSION	SPLIN 21
C	INFO(16)	SPLIN 22
C	COMMON INPUT SERVES AS INPUT TO FXDKNT	SPLIN 23
C	SEE FXDKNT FOR DEFINITIONS OF VARIABLES	SPLIN 24
C	COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE	SPLIN 25
C	COMMON OUTPUT SERVES AS OUTPUT FROM FXDKNT	SPLIN 26
C	SEE FXDKNT FOR DEFINITIONS OF VARIABLES	SPLIN 27
C	COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),	SPLIN 28
C	VORDL(28,2),KNCT,LMAX,INTERV	SPLIN 29
C	COMMON OTHER SERVES AS COMMUNICATION BETWEEN OPT,SWEEP AND HERE	SPLIN 30
C	LXI = NUMBER OF INTERIOR KNOTS, LXI1 = LXI+1, LXI2 = LXI+2	SPLIN 31
C	G = NUMERICAL CONTROL VARIABLE USED BETWEEN OPT AND SWEEP	SPLIN 32
C	CHANGE = DITTO	SPLIN 33
C	ERROR = CURRENT VALUE OF THE L-2 ERROR = SQUARED	SPLIN 34
C	ACC = DESIRED ACCURACY OF L-2 ERROR	SPLIN 35
C	XI(28) = ARRAY FOR KNOTS	SPLIN 36
C	COMMON/ OTHER / LXI,LXI1,LXI2,G,CHANGE,ERROR,ACC, XI(28)	SPLIN 37
C	ACC = .1 AND ITER = 4 TO 8 SEEM TO BE GOOD VALUES FOR TYPICAL USES	SPLIN 38
C	ACC = .1	SPLIN 39
C	ITER = 8	SPLIN 40
C	***INFO IS SIMPLY AN IDENTIFICATION OF THE DATA***	SPLIN 41
C	READ IN NO. OF POINTS=LX AND THE DATA XX AND U	SPLIN 42
C	*** IF NOKNOT.GE.2, THEN READ IN LXI2=NOKNOT KNOTS***	SPLIN 43
C	*** OTHERWISE PROGRAM CHOOSES LXI2 =NOKNOT EQUISPACED KNOTS ***	SPLIN 44
C	LXI2 = IABS(NOKNOT)	SPLIN 45
C	IF *NOKNOT* IS .GT. 2, READ IN NOKNOT KNOTS (INCL.BOUNDARY POINTS.)	SPLIN 46
C	601 FORMAT(6F12.6)	SPLIN 47
C	**CHECK ON GIVEN DATA	SPLIN 48
C	AND FROM PRESENTING UNORDERED XX ARRAY	SPLIN 49
C	IERROR = 0	SPLIN 50
C	IF (LX .GE. LXI2+2 .AND. LX .LE. 100) GO TO 3	SPLIN 51
C	WRITE(6,662) LX	SPLIN 52
C	IERROR = 1	SPLIN 53
C	GO TO 7	SPLIN 54
C	3 IF (LXI2 .GE. 3 .AND. LXI2 .LE. 28) GO TO 4	SPLIN 55
C	WRITE(6,660) NOKNOT	SPLIN 56


```

IERROR = 1
4 DO 6 L=2,LX
  IF (XX(L)-XX(L-1)) 5,6,6
5   WRITE(6,664) L,XX(L),U(L)
  IERROR = IERROR + 1
6   CONTINUE
  IF (IERROR .LT. 1) GO TO 14
7   WRITE(6,666) IERROR
  STOP
C
C   **INITIALIZE
14 IF (NOKNOT .GT. 0) GO TO 30
C
C   WHEN NOKNOT IS NEG., INTRODUCE -NOKNOT EQUISPACED KNOTS
  XI(1) = XX(1)
  XI(LXI2) = XX(LX)
  DEL = (XX(LX) - XX(1))/FLOAT(LXI2-1)
  DO 26 J = 3,LXI2
26  XI(J-1) = XI(J-2) + DEL
C
C   SET UP INITIAL APPROXIMATION
30 ADDXI(1) = XI(1)
  ADDXI(2) = XI(LXI2)
  LXI = LXI2-1
  LXI = LXI-1
  MODE = 0
  JADD = LXI2
  DO 35 J = 3,LXI2
35  ADDXI(J) = XI(J-1)
  ERROR = FXDKNT(0)
C
C   ***NOTE. MODE HAS BEEN SET EQUAL TO 1
C   *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
  WRITE(6,611) (L,XX(L),U(L),L=1,LX)
  WRITE(6,612) NOKNOT,ITER
  WRITE(6,900) (XI(I), I=1,LXI2)
900 FORMAT(28H KNOTS PRIOR TO OPTIMIZATION/(9F12.6))
C
C   OPTIMIZE KNOTS
  CALL SWEEP(ITER)
C
  WRITE(6,640)
640 FORMAT(49X,22H*** FINAL OUTPUT ***//)
  MODE = 1
  JADD = 0
  DUMB = FXDKNT(1)
  RETURN
C
610 FORMAT(2I4, / (9F12.8))
611 FORMAT (11H GIVEN DATA// (I4,2F14.8))
612 FORMAT(1H /32H NO. OF INITIAL KNOTS = I3/
1 7H ITER = I3)
660 FORMAT(32H KNOT CONTROL PARAMETER NOKNOT = I3,19H NOT WITHIN BOUNDS
*)
662 FORMAT(26H NO. OF DATA POINTS, LX = I4,44H NO1 WITHIN THE BOUNDS
*BS(NCKNOT)*2 AND 100)
664 FORMAT(12H DATA POINT I4,2F14.8,24H NOT IN ASCENDING ORDER.)
666 FORMAT(23H *** CORRECT INDICATED I3,28H INPUT ERROR(S) AND RESTART
*)
SPLIN 67
SPLIN 68
SPLIN 69
SPLIN 70
SPLIN 71
SPLIN 72
SPLIN 73
SPLIN 74
*** 2
SPLIN 76
SPLIN 77
SPLIN 78
SPLIN 79
SPLIN 80
SPLIN 81
SPLIN 82
SPLIN 83
SPLIN 84
SPLIN 85
SPLIN 86
SPLIN 87
SPLIN 88
SPLIN 89
SPLIN 90
SPLIN 91
SPLIN 92
SPLIN 93
SPLIN 94
SPLIN 95
SPLIN 96
SPLIN 97
SPLIN 98
SPLIN 99
SPLIN100
SPLIN101
SPLIN102
SPLIN103
SPLIN104
SPLIN105
SPLIN106
SPLIN107
SPLIN108
SPLIN109
SPLIN110
SPLIN111
**** 3
SPLIN114
SPLIN115
SPLIN116
SPLIN117
SPLIN118
SPLIN119
SPLIN120
SPLIN121
SPLIN122
SPLIN123
SPLIN124
SPLIN125

```

	SUBROUTINE SWEEP(I TRR)	SPLIN130
C		SPLIN131
C	KVARY, I = INDEX OF KNOT BEING VARIED	SPLIN132
C	SUBROUTINE OPT(I) OPTIMIZES ITH INTERIOR KNOT	SPLIN133
C		SPLIN134
	COMMON/INPUT/LX, XX(100), U(100), JADD, ADDXI(26), MODE	SPLIN135
	COMMON/ OUTPUT / UERROR(100), FCTL(100), XIL(28), COEFL(27,4),	SPLIN136
	VORDL(28,2), KNCT, LMAX, INTERV	SPLIN137
	COMMON/ OTHER / LXI, LXI1, LXI2, C, CHANGE, ERROR, ACC, XI(28)	SPLIN138
C	AT ALL TIMES, ERROR CONTAINS (L2 ERROR)**2 OF CURRENT B.A.	SPLIN139
C		SPLIN140
	ITER = ITRR	SPLIN141
C	**NEXT CARDS SET NUMERICAL ANALYSIS CONTROLS	SPLIN142
	EPSERR = ACC/2.5	SPLIN143
	CHANGE = .4*FLOAT(LXI)	SPLIN144
C		SPLIN145
	10 KVARY = LXI	SPLIN146
	Q = CHANGE/FLOAT(LXI)	SPLIN147
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN148
	C*** WRITE (6,902) ITER,Q	SPLIN149
	C*902 FORMAT (8H ITER, Q, I5, E20.8)	SPLIN150
	CHANGE = 0.	SPLIN151
	PREVER = ERROR	SPLIN152
	MODE = 2	SPLIN153
	JADD = 0	SPLIN154
	KNCT = KNOT - 1	SPLIN155
	DUMB = FXDKNT(0)	SPLIN156
	20 CONTINUE	SPLIN157
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN158
	C*** WRITE (6,900) KVARY	SPLIN159
	C*+00 FORMAT (1H ///8H VARYING, I4, I6H ITH INTERIOR KNOT)	SPLIN160
	C*** WRITE (6,901) ERROR	SPLIN161
	C*901 FORMAT (16H SQ. OF L2-ERROR, E16.6)	SPLIN162
C		SPLIN163
	CALL OPT(KVARY)	SPLIN164
	KVARY = KVARY - 1	SPLIN165
	JADD = JADD + 1	SPLIN166
	IF (JADD .LE. 1) GO TO 22	SPLIN167
	K = JADD	SPLIN168
	DO 21 I = 2, JADD	SPLIN169
	K = K - 1	SPLIN170
	21 ADDXI(K+I) = ADDXI(K)	SPLIN171
	22 ADDXI(1) = XI(KVARY + 2)	SPLIN172
	KNCT = LXI - JADD	SPLIN173
	MODE = 2	SPLIN174
	DUMB = FXDKNT(0)	SPLIN175
	IF (KVARY .NE. 0) GO TO 20	SPLIN176
C	THE LAST CALL TO FXDKNT PRODUCES THE B.A. USING ALL KNOTS.	SPLIN177
C	SINCE THEN ADDXI CONTAINS ALL KNOTS	SPLIN178
	ERROR = DUMB	SPLIN179
C	*** THE FOLLOWING TWO CARDS PRODUCE PRINTED OUTPUT OF L1, L2, L-INF	SPLIN180
C**	JADD = 0	SPLIN181
C**	DUMB = FXDKNT(2)	SPLIN182
C		SPLIN183
C	**IF CHANGE IN ERROR IS BIG ENOUGH MAKE ANOTHER SWEEP, ELSE QUIT	SPLIN184
	IF (PREVER-ERROR .LE. EPSERR*PREVER) GO TO 60	SPLIN185
	ITER = ITER - 1	SPLIN186
C		SPLIN187

C	**CHECK NUMBER OF PASSES THROUGH SWEEP	SPLIN188
	IF(ITER.EQ.0) GO TO 40	SPLIN189
	GC TO 10	SPLIN190
40	CONTINUE	SPLIN191
C		SPLIN192
C	IN FINAL VERSION GO TO 40, GO TO 60 ARE REPLACED BY RETURN	SPLIN193
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN194
C***	WRITE(6,620)	SPLIN195
	RETURN	SPLIN196
60	CONTINUE	SPLIN197
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN198
C***	WRITE(6,610)	SPLIN199
	RETURN	SPLIN200
C*610	FORMAT(54H *** SWEEP DISCONTINUED - INSUFFICIENT CHANGE IN ERROR)	SPLIN201
C*620	FORMAT(36H *** NO. OF ALLOWABLE SWEEPS USED UP)	SPLIN202
	END	SPLIN203
C		SPLIN204
C	*****	SPLIN205
C		SPLIN206

```

SUBROUTINE OPT(I)
  I REFERS TO THE ITH INTERIOR KNOT
  OPT FINDS THE OPTIMAL ITH KNOT BETWEEN THE I-1ST AND I+1ST KNOTS
  THE REMAINING KNOTS ARE HELD FIXED.
  INCLP = A BOUND ON THE NUMBER OF TRIES ALLOWED
  FOR IMPROVEMENT OF THE ITH KNOT
  G = MULTIPLICATION FACTOR WHICH SHOULD DECREASE AS A
  FUNCTION OF THE NO. OF SWEEPS THRU SWEEP
  G IS ALTERED IN SWEEP

COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
  VORDL(28,2),KNCT,LMAX,INTERV
COMMON/ OTHER / LXI,LXI1,LXI2,G,CHANGE,ERROR,ACC, XI(28)

  I = II
  **NUMERICAL ANALYSIS PARAMETERS SET HERE
  INCLP=9
  BD = ACC*ERROR/FLOAT(LXI)
  DIST = .0625
  H = XI(I+2)-XI(I)
  ALCW = XI(I) + DIST*H
  AHIGH = XI(I+2) - DIST*H
  LPCNT= 0
  MODE = 3

  **BEGIN SEARCH = FIND THREE VALUES FOR THE ITH KNOT
  SUCH THAT L2-ERROR AT MIDDLE VALUE, A, IS LESS THAN
  ERROR AT LEFT VALUE, ALEFT, AND AT RIGHT VALUE, ARIGHT
  A = XI(I+1)
  E = FXDKNT(A)
  ALEFT = A + G*(XI(I)-A)
  ELEFT = FXDKNT(ALEFT)
  *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
  C*** ARIGHT = 0.
  C*** ERIGHT = 0.
  C*** WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT
  SGN = SIGN(1.,ELEFT-E)
  IF (SGN.GE.0.)
    GO TO 20
    GO TO 60

  **SEARCHING FOR NEW KNOT TO THE RIGHT
10 ALEFT = A
  ELEFT = E
  A = ARIGHT
  E = ERIGHT
20 ARIGHT = A + G*(XI(I+2)-A)

  **BUFFER TO PREVENT COALESCING OF KNOTS
30 IF (AHIGH.GE.ARIGHT) GO TO 40
  AA = AHIGH
  *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***
  C**** WRITE(6,610) I
  GO TO 199

40 ERIGHT = FXDKNT(ARIGHT)
  *** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***

```

SPLIN207
 SPLIN208
 SPLIN209
 SPLIN210
 SPLIN211
 SPLIN212
 SPLIN213
 SPLIN214
 SPLIN215
 SPLIN216
 SPLIN217
 SPLIN218
 SPLIN219
 SPLIN220
 SPLIN221
 SPLIN222
 SPLIN223
 SPLIN224
 SPLIN225
 SPLIN226
 SPLIN227
 SPLIN228
 SPLIN229
 SPLIN230
 SPLIN231
 SPLIN232
 SPLIN233
 SPLIN234
 SPLIN235
 SPLIN236
 SPLIN237
 SPLIN238
 SPLIN239
 SPLIN240
 SPLIN241
 SPLIN242
 SPLIN243
 SPLIN244
 SPLIN245
 SPLIN246
 SPLIN247
 SPLIN248
 SPLIN249
 SPLIN250
 SPLIN251
 SPLIN252
 SPLIN253
 SPLIN254
 SPLIN255
 SPLIN256
 SPLIN257
 SPLIN258
 SPLIN259
 SPLIN260
 SPLIN261
 SPLIN262
 SPLIN263
 SPLIN264

C****	WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT	SPLIN265
	IF (E.LE.ERIGHT) GO TO 100	SPLIN266
C		SPLIN267
C	**CHECK TO STOP OPT	SPLIN268
	IF(E -ERIGHT.LE.80 .OR. LPCNT .GT. INCLP) GO TO 240	SPLIN269
	50 LPCNT = LPCNT+1	SPLIN270
	IF(SGN.GT.0.) GO TO 10	SPLIN271
C		SPLIN272
C	**SEARCHING FOR NEW KNOT TO THE LEFT	SPLIN273
	60 ARIGHT = A	SPLIN274
	ERIGHT = E	SPLIN275
	A = ALEFT	SPLIN276
	E = ELEFT	SPLIN277
	70 ALEFT = A + Q*(XI(I)-A)	SPLIN278
C		SPLIN279
C		SPLIN280
C	**BUFFER TO PREVENT COALESCING OF KNOTS	SPLIN281
	80 IF (ALEFT.GE.AL0W) GO TO 90	SPLIN282
	AA = AL0W	SPLIN283
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN284
C****	WRITE(6,620) I	SPLIN285
	GO TO 199	SPLIN286
C		SPLIN287
	90 ELEFT = FXDKNT(ALEFT)	SPLIN288
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN289
C****	WRITE (6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT	SPLIN290
	IF (E.LE.ELEFT) GO TO 100	SPLIN291
C		SPLIN292
C	**CHECK TO STOP OPT	SPLIN293
	IF(E - ELEFT.LE.80 .OR. LPCNT .GT. INCLP . GO TO 230	SPLIN294
	GO TO 50	SPLIN295
C		SPLIN296
C	**REQUIRED 3 VALUES HAVE BEEN FOUND	SPLIN297
C	FOLLOWING CODE FINDS PT. AT WHICH MIN OF PARABOLA CURVE PASSIN	SPLIN298
C	THRU THE ERROR VALUES AT THE PTS ALEFT, A, ARIGHT OCCURS	SPLIN299
	100 DXLEFT = ALEFT - A	SPLIN300
	DXRGHT = ARIGHT - A	SPLIN301
	DYLEFT = (ELEFT-E)/DXLEFT	SPLIN302
	DYRGHT = (ERIGHT-E)/DXRGHT	SPLIN303
	DIFF = DYLEFT - DYRGHT	SPLIN304
	IF (DIFF .EQ. 0.) GO TO 200	SPLIN305
	DEL = .5/DIFF*(DXRGHT*DYLEFT-DXLEFT*DYRGHT)	SPLIN306
	EPRED = E+DEL*(DYRGHT+(DXRGHT-DEL)/(ARIGHT-ALEFT)*DIFF)	SPLIN307
	ABEST = A + DEL	SPLIN308
	EBEST = FXDKNT(ABEST)	SPLIN309
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN310
C****	WRITE (6,900) ELEFT,EBEST,ERIGHT,ALEFT,ABEST,ARIGHT	SPLIN311
C		SPLIN312
C	**DETERMINE WHETHER ABEST GIVES BEST APPRX AND MAKE APPROPRIATE	SPLIN313
C	SWITCHING OF THE AI'S DEPENDING ON SIGN OF DEL	SPLIN314
	IF (EBEST.LE.E)	SPLIN315
	IF(DEL)110,20,120	SPLIN316
	110 ALEFT = ABEST	SPLIN317
	ELEFT = EBEST	SPLIN318
	GO TO 170	SPLIN319
	120 ARIGHT = ABEST	SPLIN320
	ERIGHT = EBEST	SPLIN321
	GO TO 170	SPLIN322

130	IF (DEL) 140, 200, 150	SPLIN323
140	ARIGHT = A	SPLIN324
	ERIGHT = E	SPLIN325
	GO TO 160	SPLIN326
150	ALEFT = A	SPLIN327
	ELEFT = E	SPLIN328
160	A = ABEST	SPLIN329
	E = EBEST	SPLIN330
C		SPLIN331
C	**FOLLOWING TESTS DETERMINE WHETHER OR NOT TO	SPLIN332
C	REITERATE PARABOLA MINIMIZATION PHASE	SPLIN333
170	IF (ABS(EPRED-EBEST).LT.5.*80) GO TO 210	SPLIN334
	IF(LPCNT.GT.INDLP) GO TO 200	SPLIN335
	LPCNT = LPCNT+1	SPLIN336
	GO TO 100	SPLIN337
C		SPLIN338
199	ETRY = FXDKNT(AA)	SPLIN339
	IF (E.LT.ETRY) GO TO 200	SPLIN340
	A = AA	SPLIN341
	E = ETRY	SPLIN342
200	CHANGE = CHANGE + ABS(A - XI(I+1))/H	SPLIN343
	XI(I+1) = A	SPLIN344
	ERROR = E	SPLIN345
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN346
C***	WRITE(6,900) ELEFT,E,ERIGHT,ALEFT,A,ARIGHT	SPLIN347
	RETURN	SPLIN348
C		SPLIN349
C	IN FINAL VERSION GO TO 210, IS REPLACED BY GO TO 200	SPLIN350
210	CONTINUE	SPLIN351
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN352
C***	WRITE(6,640) LPCNT	SPLIN353
	GO TO 200	SPLIN354
230	A = ALEFT	SPLIN355
	E = ELEFT	SPLIN356
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN357
C***	WRITE(6,640) LPCNT	SPLIN358
	GO TO 200	SPLIN359
240	A = ARIGHT	SPLIN360
	E = ERIGHT	SPLIN361
C	*** THIS IS TEMPORARY DEBUGGING AND TESTING OUTPUT ***	SPLIN362
C***	WRITE(6,640) LPCNT	SPLIN363
	GO TO 200	SPLIN364
C*610	FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (,I2,35H) MS	SPLIN365
C****	MOVED TOO CLOSE TO RIGHT NEIGHBOR)	SPLIN366
C*620	FORMAT(46H *** OPT DISCONTINUED - KNOT BEING OPTIMIZED (,I2,34H) MS	SPLIN367
C****	MOVED TOO CLOSE TO LEFT NEIGHBOR)	SPLIN368
C*640	FORMAT(24H *** OPT DISCONTINUED AT,I4,31H - INSUFFICIENT CHANGE IN	SPLIN369
C***	* ERROR)	SPLIN370
C*900	FORMAT(25H PARABOLA - ERROR VALUES ,3E20.6/12X,13HAI VALUES ,	SPLIN371
C***	1 3E20.6)	SPLIN372
	END	SPLIN373
C	-----	SPLIN374
		SPLIN375
		SPLIN376

```

FUNCTION FXDKNT (ARG)
C          THE FUNCTION RETURNS THE SQUARE OF THE L2-ERROR
LOGICAL MODES
C** IT MAY BE NECESSARY ON SOME SYSTEMS TO MENTION ALL COMMON BLOCKS
C LISTED HERE IN THE PROGRAM CALLING *FXDKNT*, TOO, TO INSURE THAT
C THE INFO IN THESE BLOCKS DOES NOT DIE BETWEEN CALLS TO *FXDKNT*
COMMON / WANDT / TREND(100),TRPZWT(100),PRINT(100)
COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
C      U(L) = FCT TO BE APPR AT XX(L), L=1,LX.
C      XX(L) IS ASSUMED TO BE NONDECREASING WITH L
C      ADDXI(I) = I-TH KNOT TO BE INTRODUCED, I=1,JADD
C      MODE = 0,1,2,3. SEE COMMENTS BELOW (AND IN NUBAS)
COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),
*      VORDL(28,2),KNOT,LMAX,INTERV
C      UERROR(L) = ERROR OF B.L2 A. TO U, L=1,LX
C      KNOT = CURRENT NO. OF KNOTS (INCL BDRY KNOTS)
C      INTERV = KNOT - 1 = CURRENT NO. OF INTERVALS (POL.PIECES)
C      XIL(K),K=1,KNOT, CURRENT (ORDERED) SET OF KNOTS
C      THE MAXIMUM ERROR OCCURS AT XX(LMAX)
C      IF ARG=1, FCTL(L) CONTAINS THE CURRENT B.AB TO U AT XX(L)
C      COEFL(I,.) CONTAINS THE POL. COEF. ON I-TH INTERVAL FOR B.A.
C      VORDL(I,.) CONTAINS VALUE AND DERIV. OF B.A. AT XIL(I)
COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST
C** A CHANGE IN THE COLUMN LENGTH OF *FCT* FORCES CHANGE IN ST.NO.69
C IN *NUBAS* .
C      FCT (L,M) = BASIS FCT M AT XX(L)
C      VORD(M,K,L) CONTAINS THE ORDS (L=1) AND SLOPES (L=2) OF FCT M
C      AT THE KNOT INTRODUCED AS K-TH. CORRELATION TO ORDERING OF
C      KNOTS BY SIZE IS DONE VIA IORDER, I.E., ORD AND SLOPE AT
C      XIL(K) ARE IN VORD(M,IORDER(K),.).
C      BC(I) = COORDINATE OF U (AND OF B.A. TO U) WRTO I-TH O.N.FCTS
C      ILAST = CURRENT NO. OF BASIS FCTNS
COMMON/ LASTB /IORDER(28),INSIRT(30),XKNOT
C      THE FCT ILAST (TO BE) INTRODUCED LAST HAS ADDITIONAL KNOT
C      XKNOT, THE KNOT JUST INTRO-
C      DUCED HAS INDEX INSERT IN XIL, INSERT IS SAVED IN INSIRT(ILAS
C      FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).
C      ***LOCAL VARIABLES
COMMON /LOCAL/ XSCALE,KNOTSV,ERBUT1,CUBERR(100),WEIGHT(100),MODE3
C      XSCALE = XX(LX) - XX(1), USED TO NORMALIZE INNER PRODUCT
C      = LENGTH OF THE INTERVAL OF INTEGRATION
C      KNOTSV = NO. OF KNOTS USED IN MOST RECENT CALL TO FXDKNT
C      ERBUT1 = SQ. OF L2-ERROR OF APPR USING ALL BUT THE ONE
C      KNOT BEING VARIED ( USED IN MODE = 3)
C      CUBERR = UERROR OF B.A. BY CUBIC POL-S (NEEDED FOR MODE = 2)
C      MODE3 = TRUE OR FALSE DEP. ON WHETHER PREV. CALL WAS IN
C      MODE=3 OR NOT
EQUIVALENCE (IPRINT,CHANGE)
C      ARG IS EITHER FIXED POINT (MODE.NE.3) TO PICK PRINT-OUT OPTIS
C      OR IS FLOATING POINT (MODE=3) TO GIVE NEW VALUE OF KNOT VARIS
CHANGE = ARG
IF (MODE.GT.0)          GO TO 29
C-----
C      *** MODE=0* COMPUTE BASIS FCT 1 THROUGH 4 AND B.A. TO U WRTO THE
C      THEN SET MODE = 1 AND PUT UERROR INTO U.
C      XSCALE = XX(LX) - XX(1)
C      DO 10 I=5,30
10 INSIRT(I) = 0

```

	DO 11 L=1,LX	SPLIN435
	UERROR(L) = U(L)	SPLIN436
	TREND(L) = T(XX(L))	SPLIN437
11	WEIGHT(L) = W(XX(L))	SPLIN438
	DO 12 L=2,LX	SPLIN439
12	TRPZNT(L-1) = (XX(L)-XX(L-2))/2.*WEIGHT(L-1)	SPLIN440
	TRPZNT(1) = (XX(2)-XX(1))/2.*WEIGHT(1)	SPLIN441
	TRPZNT(LX) = (XX(LX)-XX(LX-1))/2.*WEIGHT(LX)	SPLIN442
		SPLIN443
	XIL(1) = ADDXI(1)	SPLIN444
	XIL(2) = ADDXI(2)	SPLIN445
	IORDER(1) = 1	SPLIN446
	IORDER(2) = 2	SPLIN447
	KNOT = 2	SPLIN448
	INTERV = I	SPLIN449
	DO 19 I=1,4	SPLIN450
	ILAST = I	SPLIN451
	CALL NUBAS	SPLIN452
	DO 19 L=1,LX	SPLIN453
19	UERROR(L) = UERROR(L) - BC(I)*FCT(L,I)	SPLIN454
		SPLIN455
	MODE = 1	SPLIN456
	DO 20 L = 1,LX	SPLIN457
20	CUBERR(L) = UERROR(L)	SPLIN458
	IF (JADD.LE.2), ONLY B.A. BY CUBICS IS COMPUTED	SPLIN459
	OTHERWISE, ADDXI(I), I.GT.2, CONTAINS ADDITIONAL KNOTS	SPLIN460
	JADD = JADD - 2	SPLIN461
	IF (JADD.LE.0) GO TO 60	SPLIN462
	DO 21 I=1,JADD	SPLIN463
21	ADDXI(I) = ADDXI(I+2)	SPLIN464
	GO TO 51	SPLIN465
		SPLIN466
	GO TO (40,40,30),MODE	SPLIN467
		SPLIN468
	*** MODE=3 *** MERELY REPLACE THE LAST KNOT INTRODUCED BY	SPLIN469
	CHANGE AND RECOMPUTE L2 ERROR. CHANGE ENTERS	SPLIN470
	VIA THE ARGUMENT JPRINT = CHANGE.	SPLIN471
	THIS MODE SHOULD BE USED FOR	SPLIN472
	MINIMIZING THE L2-ERROR WRTO THE KNOT	SPLIN473
	INTRODUCED LAST AS IT MINIMIZES THE COMP WORK	SPLIN474
	IF MODE3 = TRUE (I.E., THE PRECEDING CALL TO FXDKNT	SPLIN475
	WAS IN MODE=3), THE PROGR WILL ASSUME THAT CHANGES	SPLIN476
	HAS THE SAME ORDER REL TO THE OTHER KNOTS AS THE	SPLIN477
	PREV INTRODUCED VALUE FOR KNOT. OTHERWISE	SPLIN478
	IF MODE3=FALSE (THE PRECEDING CALL WAS IN SOME OTHER	SPLIN479
	, A FCT IS ADDED WITH CHANGE AS THE ADD. KNOT.	SPLIN480
	UERROR IS ASSUMED TO CONTAIN ERROR OF B.A. TO U	SPLIN481
	ALL PREV FCTNS. **NOTE** IF THE NEXT CALL TO FXD	SPLIN482
	IS IN A MODE OTHER THAN 3, THE CHANGE PROPOSED	SPLIN483
	NOW WILL BE MADE PERMANENT.	SPLIN484
		SPLIN485
30	XKNOT = CHANGE	SPLIN486
	IF (MODE3) GO TO 35	SPLIN487
	MODE3 = .TRUE.	SPLIN488
	ERBUT1 = FXDKNT	SPLIN489
	MODE = 2	SPLIN490
	CALL NUBAS	SPLIN491
	KNOTSV = KNOT	SPLIN492
	MODE = 3	


```

35 CALL NUBAS
36 FXDKNT = ABSTERBUTI - BC(ILAST)/XSCALE*BC(ILAST)
RETURN
C -----
C ***MODE=1,2*** RETAIN THE FIRST KNOT KNOTS INTRODUCED EARLIER SPLIN493
C (HENCE THEIR CORRESP FCTNS) BUT REPLACE FURTHER SPLIN494
C FCTNS (IF ANY) BY FCTNS HAVING ADDITIONAL SPLIN495
C KNOTS ADDXI(I), I=1, JADD) HENCE SPLIN496
C IF KNOT.LT.KNOTSV (=NO. OF KNOTS USED IN PREV CALLS) SPLIN497
C 40 THROUGH 49 RESTORES ARRAYS IORDER, XIL, UERROR TO THE STATE OSPLIN500
C ILAST = KNOT + 2, INVERTING THE ACTION OF DO 11 ... TO 14 IN SPLIN501
C 40 IF (KNOT.LT.KNOTSV) GO TO 42 SPLIN502
C KNOT = KNOTSV SPLIN503
C IF (.NOT.MODE3) GO TO 50 SPLIN504
C DO 41 L=1, LX SPLIN505
C 41 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L, ILAST) SPLIN506
C GO TO 49 SPLIN507
C 42 DO 43 L=1, LX SPLIN508
C 43 UERROR(L) = CUBERR(L) SPLIN509
C IF (KNOT.LE.2) GO TO 48 SPLIN510
C IDUM = KNOT + 1 SPLIN511
C DO 45 IO=IDUM, KNOTSV SPLIN512
C INSERT = INSIRT(ILAST) SPLIN513
C ILM3 = ILAST - 3 SPLIN514
C DO 44 K=INSERT, ILM3 SPLIN515
C IORDER(K) = IORDER(K+1) SPLIN516
C 44 XIL(K) = XIL(K+1) SPLIN517
C 45 ILAST = ILAST-1 SPLIN518
C DO 47 I=5, ILAST SPLIN519
C DO 47 L=1, LX SPLIN520
C 47 UERROR(L) = UERROR(L) - BC(I)*FCT(L, I) SPLIN521
C GO TO 49 SPLIN522
C 48 XIL(2) = XIL(ILAST-2) SPLIN523
C IORDER(2) = 2 SPLIN524
C KNOT = 2 SPLIN525
C 49 IF (JADD.GT.0) GO TO 51 SPLIN526
C ILAST = KNOT + 2 SPLIN527
C INTERV = KNOT - 1 SPLIN528
C GO TO 60 SPLIN529
C ***MODE=1,2*** ADD JADD BASIS FCTNS, I.E., FOR IO=1, JADD, SPLIN530
C CONSTRUCT FCT ILAST WITH ONE MORE KNOT, VIZ. SPLIN531
C XKNOT=ADDXI(IO), THAN THE PREVIOUS LAST FCT, SPLIN532
C ORTHONORMALIZE IT OVER ALL PREVIOUS FCTNS, THEN SPLIN533
C COMPUTE THE COORDINATE BC(ILAST) OF U WRTO IT, SPLIN534
C SUBTRACT OUT ITS COMPONENT FROM UERROR. SPLIN535
C 50 IF (JADD.LE.0) GO TO 51 SPLIN536
C 51 DO 52 IO=1, JADD SPLIN537
C XKNOT = ADDXI(IO) SPLIN538
C CALL NUBAS SPLIN539
C DO 52 L=1, LX SPLIN540
C 52 UERROR(L) = UERROR(L) - BC(ILAST)*FCT(L, ILAST) SPLIN541
C 60 FXDKNT= DOT(31, 2)/XSCALE SPLIN542
C KNOTSV = KNOT SPLIN543
C 61 MODE3 = .FALSE. SPLIN544
C IF (IPRINT.EQ.0) RETURN SPLIN545
C RETURN SPLIN546
C RETURN SPLIN547
C RETURN SPLIN548
C RETURN SPLIN549
C RETURN SPLIN550

```

```

C          VARIOUS PRINTING IS DONE DEP ON THE ARG = IPRINT
C                                     GO TO (70,80,90),IPRINT
C
C          COMPUTE COEFFICIENTS OF B.A. AND PRINT
C          ****          BEST APPROXIMATION PRINTOUT          ****
C          FORMAT IS
C          KNOTS XI(J)          CUBIC COEFFICIENTS P(I,J) IN
C          INTERVAL (XI(J), XI(J+1))
C          ERROR CURVE (SCALED)
C
C          THE FOLLOWING FORTRAN CODE FINDS VALUES AT X OF THE
C          APPROXIMATION FROM THIS OUTPUT----
C          I=LXI2
C          1 A=X-XI(I)
C          IFTA) 2, 4, 4
C          2 I=I-1
C          IF(I) 3, 3, 1
C          3 I=I
C          4 V=P(I,1)+A*(P(I,2)+A*(P(I,3)+A*(P(I,4))))
C          ..... FOR A SUBPROGRAM USE COMMON/OUTPUT/..., COMMON/OTHER/...,
C          ..... THE P BECOME COEFL(I,1),COEFL(I,2),COEFL(I,3),COEFL(I,4)
C
C          70 WRITE(6,610)
C          DO 72 I=1,KNOT
C          ILOC = IORDER(I)
C          DO 72 L=1,2
C          72 VORD(I,L) = -ARITH(0.,ILAST,80,I,VORD(I,ILOC,L),I)
C          SEE COMMENT IN *DOT* ABOUT THE *ARITH* ROUTINE.
C          CALL EVAL
C          DO 73 I=1,INTERV
C          WRITE(6,620) I,XIL(I)
C          73 WRITE (6,630) (J,COEFL(I ,J),J=1,4)
C          WRITE (6,620) KNOT,XIL(KNOT)
C          610 FORMAT(42X,5HKNOTS,22X,18HCUBIC COEFFICIENTS//)
C          620 FORMAT(35X, 3HXI(, I2, 3H) =, F12.6)
C          630 FORMAT(67X,2HC(,I1,3H) =,E16.6)
C
C          **COMPUTE L2, L1, MAX ERRORS AND PRINT
C          80 ERRL2 = SQRT(FXOKNT)
C          ERRL1 = 0.
C          ERRL99= 0.
C          DO 82 L=1,LX
C          DIF = ABS(UERROR(L)*WEIGHI(L))
C          IF(ERRL99.GT.DIF)          GO TO 81
C          LMAX = L
C          ERRL99 = DIF
C          81 ERRL1 = ERRL1+ DIF
C          82 CONTINUE
C          ERRL1 = ERRL1/FLOAT(LX)
C          WRITE(6,623) ERRL2, ERRL1, ERRL99,XX(LMAX)
C          *** THE FOLLOWING CARD IS TEMPORARY
C          GO TO (90,96,96),IPRINT
C
C          ** SCALE ERROR CURVE AND PRINT
C          90 IE = 0
C          SCALE = 1.
C          IF (ERRL99.GE.10.)          GO TO 92
C          DO 91 IE=1,9

```

```

SPLIN551
SPLIN552
SPLIN553
SPLIN554
SPLIN555
SPLIN556
SPLIN557
SPLIN558
SPLIN559
SPLIN560
SPLIN561
SPLIN562
SPLIN563
SPLIN564
SPLIN565
SPLIN566
SPLIN567
SPLIN568
SPLIN569
SPLIN570
SPLIN571
SPLIN572
SPLIN573
SPLIN574
SPLIN575
SPLIN576
SPLIN577
SPLIN578
SPLIN579
SPLIN580
SPLIN581
SPLIN582
SPLIN583
SPLIN584
SPLIN585
SPLIN586
SPLIN587
SPLIN588
SPLIN589
SPLIN590
SPLIN591
SPLIN592
SPLIN593
SPLIN594
SPLIN595
SPLIN596
SPLIN597
SPLIN598
SPLIN599
SPLIN600
SPLIN601
SPLIN602
SPLIN603
SPLIN604
SPLIN605
SPLIN606

```

```

SCALE = SCALE*10.
IF (ERR199*SCALE.GE.10.) GO TO 92
91 CONTINUE
92 DO 93 L=1,LX
93 PRINT (L) = UERROR(L)*SCALE
GO TO (94,95,95), IPRINT
94 WRITE (6,621) IE, (L,XX(L),FCTL(L),PRINT(L),L=1,LX)
GO TO 96
95 WRITE (6,622) IE, (L,XX(L),PRINT(L),L=1,LX)
96 RETURN
621 FORMAT(1H //45X,36HAPPROXIMATION AND SCALED ERROR CURVE/38X,
*10HDATA POINT,7X,13HAPPROXIMATION,3X,16HDEVIATION X 10E+,I1/
*(31X,I4,F16.8,F16.8,F17.6))
622 FORMAT(1H //58X, 11HERROR CURVE/38X, 10HDATA POINT, 23X,
116HDEVIATION X 10E+,I1/(31X,I4,F16.8,16X,F17.6))
623 FORMAT(1H ///40X20HLEAST SQUARE ERROR =,E20.6/
1 40X20HAVERAGE ERROR =,E20.6/
2 40X20HMAXIMUM ERROR =,E20.6,3H AT,F12.6///)
END
C
C*****
C

```

```

SPLIN617
SPLIN618
SPLIN619
SPLIN610
SPLIN611
SPLIN612
SPLIN613
SPLIN614
SPLIN615
SPLIN616
SPLIN617
SPLIN618
SPLIN619
SPLIN620
SPLIN621
SPLIN622
SPLIN623
SPLIN624
SPLIN625
SPLIN626
SPLIN627
SPLIN628

```

C C C C	SUBROUTINE INTERP COMPUTE THE SLOPES VORDL(I,2), I=2,KNOT-1 AT INTERIOR KNOTS OF CUBIC SPLINE FOR GIVEN VALUES VORDL(I,1),I=1,KNOT, AT ALL THE KNOTS AND GIVEN BOUNDARY DERIVATIVES DIMENSION D(28), DIAG(28) COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4), VORDL(28,2),KNOT,LMAX,INTERV DATA DIAG(1),D(1)/1.,0.7 DO 10 M=2,KNOT D(M) = XIL(M) - XIL(M-1) 10 DIAG(M) = (VORDL(M,1)-VORDL(M-1,1))/D(M) DO 20 M=2,INTERV VORDL(M,2) = 3.*(D(M)*DIAG(M+1) + D(M+1)*DIAG(M)) 20 DIAG(M) = 2.*(D(M)+D(M+1)) DO 30 M=2,INTERV G = -D(M+1)/DIAG(M-1) DIAG(M) = DIAG(M) + G*D(M-1) 30 VORDL(M,2) = VORDL(M,2) + G*VORDL(M-1,2) NJ = KNOT DO 40 M=2,INTERV NJ = NJ - 1 40 VORDL(NJ,2) = (VORDL(NJ,2) - D(NJ)*VORDL(NJ+1,2))/DIAG(NJ) RETURN END	SPLIN629 SPLIN630 SPLIN631 SPLIN632 SPLIN633 SPLIN634 SPLIN635 SPLIN636 SPLIN637 SPLIN638 SPLIN639 SPLIN640 SPLIN641 SPLIN642 SPLIN643 SPLIN644 SPLIN645 SPLIN646 SPLIN647 SPLIN648 SPLIN649 SPLIN650 SPLIN651 SPLIN652 SPLIN653 SPLIN654 SPLIN655 SPLIN656
------------------	--	--

```

FUNCTION DOT (M,INDEX)
C   COMPUTE INNER PRODUCT OF FCT M WITH FCT ILAST (INDEX=1) OR
C   UERROR (INDEX=2)
COMMON / WANDT / TREND(100),TRPZWT(100),G(100)
COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE
COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(26),COEFL(27,4),
      VORDL(26,2),KNOT,LMAX,INTERV
COMMON/ BASIS /FCT(100,30),VORD(30,26,2),BC(30),ILAST
      GO TO (10,30),INDEX
10 IF (M.EQ.ILAST)
   DO 11 L=1,LX
11 G(L) = FCT(L,M)*FCTL(L)
      GO TO 80
20 DO 21 L=1,LX
21 G(L) = FCTL(L)*FCTL(L)
      GO TO 80
30 IF (M.EQ.31)
   DO 31 L=1,LX
31 G(L) = FCTL(L)*UERROR(L)
      GO TO 80
40 DO 41 L=1,LX
41 G(L) = UERROR(L)*UERROR(L)
C
C EFFICIENTLY PROGRAMMED DOUBLE PRECISION ACCUMULATION OF SCALAR
C PRODUCTS IS CALLED FOR HERE. AT PURDUE, WE USE
C   D = ARITH1(C,N,A,IA,B,IB)
C WHICH RETURNS THE VALUE OF
C   D = C - SUM(A(1+J*IA) * B(1+J*IB), J=0,...,N-1)
C
80 DOT = -ARITH1(0.,LX,G,1,TRPZWT,1)
      RETURN
END

```

```

SPLIN657
SPLIN658
SPLIN659
SPLIN660
SPLIN661
SPLIN662
SPLIN663
SPLIN664
SPLIN665
SPLIN666
SPLIN667
SPLIN668
SPLIN669
SPLIN670
SPLIN671
SPLIN672
SPLIN673
SPLIN674
SPLIN675
SPLIN676
SPLIN677
SPLIN678
SPLIN679
SPLIN680
SPLIN681
SPLIN682
SPLIN683
SPLIN684
SPLIN685
SPLIN686
SPLIN687
SPLIN688
SPLIN689
SPLIN690
SPLIN691

```

```

C *****
C
C

```

C	SUBROUTINE EVAL	SPLIN692
C	COMPUTE POL. COEFF COEFL(I,K) OF FCT ILAST FROM VORDL,	SPLIN693
C	THEN COMPUTE FCTL(L) = (FCT ILAST)*TREND AT XX(L),L=1,LX	SPLIN694
	COMMON / WANDT / TREND(100),TRPZWT(100),G(100)	SPLIN695
	COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE	SPLIN696
	COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),	SPLIN697
	* VORDL(28,2),KNOT,LMAX,INTERV	SPLIN698
	DO IL I=1,INTERV	SPLIN699
	COEFL(I,1) = VORDL(I,1)	SPLIN700
	COEFL(I,2) = VORDL(I,2)	SPLIN701
	DX = XIL(I+1) - XIL(I)	SPLIN702
	DUM1 = (VORDL(I+1,1)-VORDL(I,1))/DX	SPLIN703
	DUM2 = VORDL(I,2)+VORDL(I+1,2)-2.*DUM1	SPLIN704
	COEFL(I,3) = (DUM1-DUM2-VORDL(I,2))/DX	SPLIN705
C	10 COEFL(I,4) = DUM2/DX/DX	SPLIN706
	J = 1	SPLIN707
	ISWTCH = 1	SPLIN708
	DO 20 L=1,LX	SPLIN709
	11 IF (J.EQ.INTERV) GO TO (II,13),ISWTCH	SPLIN710
	IF (XX(L).LT.XIL(J+1)) GO TO 12	SPLIN711
	J = J + 1 GO TO 13	SPLIN712
	GO TO 11	SPLIN713
	12 ISWTCH = 2	SPLIN714
	13 DX = XX(L) - XIL(J)	SPLIN715
	20 FCTL(L) = (COEFL(J,1)+DX*(COEFL(J,2)+DX*(COEFL(J,3)	SPLIN716
	+DX*COEFL(J,4)))*TREND(L)	SPLIN717
	* RETURN	SPLIN718
	END	SPLIN719
C		SPLIN720
C		SPLIN721
C		SPLIN722
		SPLIN723
		SPLIN724
		SPLIN725

	SUBROUTINE NUBAS	SPLIN726
	COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE	SPLIN727
	COMMON/ OUTPUT /UERROR(100),FCTL(100),XIL(28),COEFL(27,4),	SPLIN728
	VORDL(28,2),KNOT,LMAX,INTERV	SPLIN729
	COMMON/ BASIS /FCT(100,30),VORD(30,28,2),BC(30),ILAST	SPLIN730
	COMMON/ LASTB /IORDER(28),INSIRT(30),XKNOT	SPLIN731
C	COEF(IC,.) CONTAINS THE POL COEFFICIENTS OF FCT M FOR INTER-	SPLIN732
C	VAL TO THE RIGHT OF XI(IC), IC=ICH,ICH*M-3,	SPLIN733
C	WITH ICH = M*(M-7)/2 + 10 (WITH OBVIOUS MODS FOR M.LE.4)	SPLIN734
C	THE FCT ILAST (TO BE) INTRODUCED LAST, HAS ITS VALUES AT THE	SPLIN735
C	POINTS XX(L) IN FCTL(L), HAS FIRST INDEX ICLASS	SPLIN736
C	IN COEF AND XI, HAS ADDITIONAL KNOT XKNOT, THE KNOT KNOTS	SPLIN737
C	FOR IT ARE CONTAINED, IN INCREASING ORDER, IN XIL,ITS COR-	SPLIN738
C	RESPONDING ORDS AND SLOPES ARE IN VORDL, THE KNOT JUST INTRO-	SPLIN739
C	DUCE HAS INDEX INSERT IN XIL,INSERT IS SAVED IN INSIRT(ICLASS	SPLIN740
C	FOR POSSIBLE REPLACEMENT OF KNOTS LATER ON (SEE MODE=2,3).	SPLIN741
	DIMENSION TEMP(30),XI(381),COEF(381,4)	SPLIN742
	IF (MODE.GT.0) GO TO 8	SPLIN743
C	***CONSTRUCT FCT ILAST FOR ILAST.LE.4	SPLIN744
	XI(ILAST) = XI(1)	SPLIN745
	ICLAST = ILAST	SPLIN746
	ILM1 = ILAST-1	SPLIN747
	IF(ILAST.GT.2) GO TO 7	SPLIN748
	IF(ILAST.EQ.2) GO TO 6	SPLIN749
C	FIRST BASIS FCT IS A CONSTANT	SPLIN750
	VORDL(1,1)=1.	SPLIN751
	VORDL(2,1)=1.	SPLIN752
	VORDL(1,2)=0.	SPLIN753
	VORDL(2,2)=0.	SPLIN754
	GO TO 67	SPLIN755
C	SECOND BASIS FCT IS A STRAIGHT LINE	SPLIN756
	6 VORDL(2,2) = VORDL(1,1)/(XIL(2) - XIL(1))*2.	SPLIN757
	VORDL(1,2) = -VORDL(2,2)	SPLIN758
C		SPLIN759
	7 VORDL(2,1) = - VORDL(2,1)	SPLIN760
	VORDL(2,2) = - VORDL(2,2)	SPLIN761
	GO TO 59	SPLIN762
C	8	SPLIN763
	GO TO (10,10,14),MODE	SPLIN764
C	***SET UP CONSTANTS DEP.ON ILAST. INSERT NEW KNOT INTO XIL	SPLIN765
C	AND UPDATE VORD FOR FCT M,M=1,ILAST-1	SPLIN766
	10 KNOT = KNOT + 1	SPLIN767
	ILAST = KNOT + 2	SPLIN768
	ICLAST = ILAST*(ILAST-7)/2 + 10	SPLIN769
	ILM1 = ILAST-1	SPLIN770
	INTERV = KNOT - 1	SPLIN771
	DO 11 INSERT=2,INTERV	SPLIN772
	IF (XKNOT.LT.XIL(INSERT)) GO TO 12	SPLIN773
	11 CONTINUE	SPLIN774
	GO TO 95	SPLIN775
	12 IF (XKNOT.LE.XIL(INSERT-1)) GO TO 95	SPLIN776
	IO = KNOT	SPLIN777
	DO 13 L=INSERT,INTERV	SPLIN778
	IO = IO - 1	SPLIN779
	XIL(IO+1) = XIL(IO)	SPLIN780
	13 IORDER(IO+1) = IORDER(IO)	SPLIN781
	IORDER(INSERT) = KNOT	SPLIN782
C		SPLIN783

	RETURN	SPLIN900
C		SPLIN901
C	*** THIS OUTPUT INDICATES A FAILURE CONDITION ***	SPLIN902
	95 WRITE (6,950) XKNOT,ILAST	SPLIN903
	950 FORMAT (15H *** NEW KNOT,E20.8,13H FOR FUNCTION,I3,50H OUT OF BOS	SPLIN904
	*UNDS OR COINCIDENT WITH A PREVIOUS KNOT./36H *** EXECUTION CANNOS	SPLIN905
	*T BE CONTINUED)	SPLIN906
	STOP	SPLIN907
C		SPLIN908
	END	SPLIN909
C		SPLIN910
C	*****TREND AND WEIGHT FUNCTIONS*****	SPLIN911
C		SPLIN912

FUNCTION T(Z)
T = 1.
RETURN
END

SPLIN913
SPLIN914
SPLIN915
SPLIN916
SPLIN917

FUNCTION W(Z)
W = 1.
RETURN
END

SPLIN918
SPLIN919
SPLIN920
SPLIN922
SPLIN923

C-----

IDENT ARITH1
 ARITH1 - ARITHMETIC PACKAGE FOR LINEQ1 AND DTMNT1.

DAVID S. DODSON, 06/01/70.

FUNCTION.

GIVEN REAL SCALAR C AND N-VECTORS A AND B, THIS PACKAGE COMPUTES THE FUNCTION:

$$ARITH1 = C - \sum_{I=1}^N A(I) * B(I)$$

USAGE.

FORTRAN FUNCTION REFERENCE TO ARITH1 OF FORM:

Y=ARITH1(C,N,A,KA,R,KB)

WHERE: A AND B ARE THE NAMES OF THE TWO VECTORS AND
 KA AND KB ARE THE INCREMENTS BETWEEN SUCCESSIVE
 ELEMENTS OF THE A AND B VECTORS IN MEMORY.

COMPATABILITY.

THIS ROUTINE IS EQUIVALENT TO THE FORTRAN SUBPROGRAM:

```

FUNCTION ARITH1 (C,N,A,KA,B,KB)
  DOUBLE PRECISION T
  REAL A(KA,N),B(KB,N)
  T=DBLE(C)
  IF(N.EQ.0)GO TO 5
  DO 4 I=1,N
  4 T=T-DBLE(A(I,I))*DBLE(B(I,I))
  5 ARITH1=T
  RETURN
  END
  
```

EJECT
 ENTRY ARITH1

LOOP

```

SA1 B3          FETCH NEXT A
SB3 B3+B4
SA2 B5          FETCH NEXT B
SB5 B5+B6
FX0 X1*X2      (X0,X1) = (X1) * (X2)
DX1 X1*X2
FX2 X6-X0      (X6,X7) = (X6,X7) - (X0,X1)
DX3 X6-X0
FX0 X7-X1
NX2 X2
FX1 X0+X3
FX0 X1+X2
NX3 X0
DX1 X1+X2
  
```

ARITH 1
 ARITH 2
 ARITH 3
 ARITH 4
 ARITH 5
 ARITH 6
 ARITH 7
 ARITH 8
 ARITH 9
 ARITH 10
 ARITH 11
 ARITH 12
 ARITH 13
 ARITH 14
 ARITH 15
 ARITH 16
 ARITH 17
 ARITH 18
 ARITH 19
 ARITH 20
 ARITH 21
 ARITH 22
 ARITH 23
 ARITH 24
 ARITH 25
 ARITH 26
 ARITH 27
 ARITH 28
 ARITH 29
 ARITH 30
 ARITH 31
 ARITH 32
 ARITH 33
 ARITH 34
 ARITH 35
 ARITH 36
 ARITH 37
 ARITH 38
 ARITH 39
 ARITH 40
 ARITH 41
 ARITH 42
 ARITH 43
 ARITH 44
 ARITH 45
 ARITH 46
 ARITH 47
 ARITH 48
 ARITH 49
 ARITH 50
 ARITH 51
 ARITH 52
 ARITH 53
 ARITH 54
 ARITH 55
 ARITH 56
 ARITH 57
 ARITH 58
 ARITH 59
 ARITH 60

NX2
 FX6
 DX7
 SB2
 NZ
 ARITH1
 RSSZ
 SA1
 BX6
 MX7
 DX7
 SB1
 SA1
 SB2
 ZR
 SA1
 SB4
 SA1
 SB6
 EQ
 SPACE
 END

X1
 X2+X3
 X2+X3
 B2-B1
 B2*LOOP

COUNT TERM
 LOOP TO COMPUTE INNER PRODUCT

ENTRY/EXIT
 (X6,X7) = DBLE(C)

(B1) = 1
 (B2) = N

RETURN IF N = 0

(B4) = KA

(B6) = KB

LOOP
 4

ARITH 61
 ARITH 62
 ARITH 63
 ARITH 64
 ARITH 65
 ARITH 66
 ARITH 67
 ARITH 68
 ARITH 69
 ARITH 70
 ARITH 71
 ARITH 72
 ARITH 73
 ARITH 74
 ARITH 75
 ARITH 76
 ARITH 77
 ARITH 78
 ARITH 79
 ARITH 80
 ARITH 81
 ARITH 82

APPENDIX III

PROGRAM LISTING - VCF

- ADYNF

- SUBROUTINES

PROGRAM VCF8 (INPUT, OUTPUT, PUNCH, TAPE3, TAPE4,	VCF8	2
1 TAPE5=INPUT, TAPE6=OUTPUT, TAPE7=PUNCH)		
DIMENSION U(32, 2, 51), W(32, 51), UT(32, 51), UB(32, 51)	VCF8	4
DIMENSION INFO(16)	VCF8	5
DIMENSION THASRT(6), UZZT(6), UZZSQ(6), GAMART(6), IFLGT(6)	VCF8	6
DIMENSION THASRB(6), UZZB(6), UZZSQB(6), GAMARB(6), IFLGB(6)	VCF8	7
DIMENSION COK(100), COPK(100), CDSK(100)	VCF8	8
DIMENSION CLSK(100)	VCF8	9
DIMENSION TAWD(53), TAWL(53)	VCF8	10
COMMON ALPHA, PI, PI2, RE, SRE, SQRTPI, DTOR, RTOD, RC, RCHXSQ, SIGMA, LEVEL	VCF8	11
COMMON/KAYS/K, KS, KR, KT, KB, KRT, KRB	VCF8	12
COMMON/ELIPS2/AKPHALF, AKDOTPH	VCF8	13
COMMON/BLOCK1/X(100), Y(100), XB(100), YB(100), XRT(100), YRT(100),	VCF8	14
1 XRB(100), YRB(100)	VCF8	15
COMMON/BLOCK2/GAMMA(100), GMAB(100), GMRT(100), GMRB(100)	VCF8	16
COMMON/BLOCK3/XDOT(100), YDOT(100), XDOTB(100), YDOTB(100),	VCF8	17
1 XRDOT(100), YRDOT(100), XRDOTB(100), YRDOTB(100)	VCF8	18
COMMON/BLOCK4/TT(100), TR(100), TRT(100), TRB(100)	VCF8	19
COMMON/BLOCK5/AK, AKSQD, AKHALF, AKDOT	VCF8	20
COMMON/BLOCK10/THETAS, THETASB, THETA	VCF8	21
COMMON/BLOCK11/DS(52), DZ2, DZ8, DZSQ, DZSQ2, DZSQ4, DT2	VCF8	22
COMMON/BLOCK20/DX, DZ, INTX1, NBIG1	VCF8	23
COMMON/BLOCK30/T, TI, DELT, DELTT, DELTB	VCF8	24
COMMON/BLBOX2/TAU, PT4, NBIG	VCF8	25
COMMON/BLBOX12/ZN(51), ISEP	VCF8	26
COMMON/BLBOX13/KTS, IXRSET, IXBRSET, UTNBIG(53), UBNBIG(53)	VCF8	27
COMMON/BLBOX14/ZHAT, AKTI	VCF8	28
EXTERNAL CPC, DRAG, PLIFT	VCF8	29
INTEGER ALPHA	VCF8	30
REAL NOR, NU, MU	VCF8	31
REAL L, LB	VCF8	32
REAL LENGTH	VCF8	33
C LENGTH=DIMENSIONAL LENGTH	VCF8	34
C AATAACK=ANGLE OF ATTACK IN DEGREES	VCF8	35
C A FUNCTION SUBPROGRAM NAMED RZERO (ZSTAR) MUST BE	VCF8	36
C SUPPLIED BY THE USER TO ENTER THE BODY GEOMETRY.	VCF8	37
C ZSTAR IS THE DIMENSIONAL DISTANCE ALONG THE BODY AXIS,	VCF8	38
C AND RZERO IS THE CORRESPONDING DIMENSIONAL RADIUS.	VCF8	39
C IF THE BODY GEOMETRY IS IN TABULAR FORM THEN RZEROTZSTART	VCF8	40
C IS AN APPROXIMATING FUNCTION	VCF8	41
C V=FREE STREAM VELOCITY	VCF8	42
C NU=KINEMATIC VISCOSITY	VCF8	43
C INFO DATA IDENTIFICATION	VCF8	44
READ(5, 705) (INFO(I), I=1, 16)	VCF8	45
WRITE(6, 605) (INFO(I), I=1, 16)	VCF8	46
READ(5, 706) AATAACK, RE, LENGTH	VCF8	47
READ(5, 706) DELT, RC, SIGMA	VCF8	48
READ(5, 708) KFINAL, TFINAL, ZFINAL	VCF8	49
READ(5, 709) LR, LW, LEVEL, KPUN	VCF8	50
PI=4.0*ATAN(1.0)	VCF8	51
DTOR=PI/180.0	VCF8	52
RTOD=180.0/PI	VCF8	53
CALL NONDIM(DMAX, RW, AW, F, SA, LENGTH, PI)	VCF8	54
WRITE(6, 611) DMAX, AW, F, SA	VCF8	55
WRITE(6, 606) AATAACK, RE	VCF8	56
AATAACK=DTOR*AATAACK	VCF8	57
RE=PE*SIN(AATAACK)*AW/LENGTH	VCF8	58

	WRITE(6,601) RE, DELT, RC, SIGMA, KFINAL, TFINAL, ZFINAL, LR, LW, LEVEL, KPUN	VCF8	59
	FTA=F*TAN(AATACK)	VCF8	60
C OUTPUT PARAMETERS	VCF8	61
	IPUN=0	VCF8	62
C COUNTERS	VCF8	63
	KT=KB=0	VCF8	64
	KRT=KRB=0	VCF8	65
	KR=KS=KSB=1000	VCF8	66
	IXTRSET=IXBRSET=INITAL=53	VCF8	67
C NDIM IS THE K DIMENSION	VCF8	68
C NDIM MUST BE PUT IN FREEVTX, CPC, AND RVTX	VCF8	69
	NDIM=100	VCF8	70
C IDIM=DIMENSION OF THETA GRID IN B.L. ROUTINE	VCF8	71
C JDIM= DIMENSION OF R GRID IN B.L. ROUTINE	VCF8	72
	IDIM=32		
	JDIM=51	VCF8	74
C CONSTANTS	VCF8	75
	PI2=2.0*PI	VCF8	76
	PINF=132.3/.1189	VCF8	77
	SQRTPI=SQRT(PI)	VCF8	78
	SQA=SIN(AATACK)**2	VCF8	79
	DELT=.125	VCF8	80
	SRE=SQRT(RE)	VCF8	81
C ISYM=1 IS THE SYMMETRIC CASE	VCF8	82
C ISYM=0 IS THE ASYMMETRIC CASE	VCF8	83
	ISYM=1	VCF8	84
	THTASYM=0.0	VCF8	85
C	***** INITIAL CONDITIONS	VCF8	86
C	***** START OF LOOP	VCF8	87
	K=0	VCF8	88
	IF(LR.EQ.0) GO TO 219	VCF8	89
	READ(LR) K, KS, KR, KT, KB, KRT, KRB, T, DELT, NBIG, IXTRSET, IXBRSET	VCF8	90
	READ(LR) (UTNBIG(I), (UT(I,J), J=1, NBIG), I=1, IXTRSET)	VCF8	91
	IF(ISYM.EQ.0) READ(LR) (UBNBIG(I), (UB(I,J), J=1, NBIG), I=1, IXBRSET)	VCF8	92
	IF(K.LT.KS) GO TO 219	VCF8	93
	READ(LR) (X(I), Y(I), XDOT(I), YDOT(I), GAMMA(I), TT(I), I=1, KT)	VCF8	94
	READ(LR) (XB(I), YB(I), XDOTB(I), YDOTB(I), GMAB(I), TB(I), I=1, KB)	VCF8	95
	IF(KRT.EQ.0) GO TO 220	VCF8	96
	READ(LR) (XRT(I), YRT(I), XRDOT(I), YRDOT(I), GMRT(I), TRT(I), I=1, KRT)	VCF8	97
	READ(LR) (XRB(I), YRB(I), XRDOTB(I), YRDOTB(I), GMRB(I), TRB(I),	VCF8	98
	1I=1, KRB)	VCF8	99
220	CONTINUE	VCF8	100
	IF(K.LT.KR) GO TO 219	VCF8	101
	READ(LR) THTASMT, THTASRT, UZZT, UZZSQT, GAMART, IFLGT, THTASMB,	VCF8	102
	1THTASRB, UZZB, UZZSQB, GAMARB, IFLGB, NT, NB, NFT, NFB, GMTLT, GMTLB	VCF8	103
219	CONTINUE	VCF8	104
	IF((K/KPUN)*KPUN.EQ.K) IPUN=2	VCF8	105
	KTS=K+1	VCF8	106
	IF(K.NE.0) GO TO 69	VCF8	107
	T=TI=(RW/AW)*2.0*FTA*.03	VCF8	108
	ZHAT=T*(AW/(RW*2.0*FTA))	VCF8	109
	AK=RW/AW*RZRO(ZHAT, LENGTH, RW)	VCF8	110
	AKT I=AK	VCF8	111
	AKDOT=DRZRO(ZHAT, LENGTH, RW)/(2.0*FTA)	VCF8	112
	CDPI=PI2*AK*AKDOT	VCF8	113
	WRITE(6,620) T, ZHAT, AK, AKDOT, CDPI	VCF8	114
69	K=K+1	VCF8	115
	KMI=K-1	VCF8	116

T=T+DELT	VCF8 117
TH=T-DELT/2.0	VCF8 118
ZHAT=T*(AW/(RW*2.0*FTA))	VCF8 119
AK=RW/AW*RZRO(ZHAT,LENGTH,RW)	VCF8 120
AKDOT=DRZRO(ZHAT,LENGTH,RW)/(2.0*FTA)	VCF8 121
AADOT=AK*AKDOT	VCF8 122
AKSQ=AK**2	VCF8 123
ZHALF=TH*(AW/(RW*2.0*FTA))	VCF8 124
AKPHALF=AKHALF=RW/AW*RZRO(ZHALF,LENGTH,RW)	VCF8 125
AKDOTPH=DRZRO(ZHALF,LENGTH,RW)/(2.0*FTA)	VCF8 126
C..... RCMAXSQ AND RCSQ ARE THE MAX CORE RADIUS SQUARED AND CORE	VCF8 127
C..... RADIUS SQUARED	VCF8 128
IF(KS.LE.K)RCMAXSQ=5.04*T/RE	VCF8 129
IF(KS.LE.K)RCMAX=SQRT(RCMAXSQ)	VCF8 130
KTTEMP=KT	VCF8 131
KBTEMP=KB	VCF8 132
KT=KT+1	VCF8 133
KB=KB+1	VCF8 134
CALL BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,I)	VCF8 135
KT=KTTEMP	VCF8 136
KB=KBTEMP	VCF8 137
C..... SHEAR DRAG	VCF8 138
CDS T=0.0	VCF8 139
CLST=0.0	VCF8 140
INTX=ISEP-1	VCF8 141
DO 1 I=1,INTX	VCF8 142
TRAPD=DS(I)/2.0*(TAWD(I)+TAWD(I+1))	VCF8 143
TRAPL=DS(I)/2.0*(TAWL(I)+TAWL(I+1))	VCF8 144
CDS T=CDS T+TRAPD	VCF8 145
1 CLST=CLST+TRAPL	VCF8 146
CDSK(K)=.50*AK*CDS T	VCF8 147
CLSK(K)=.50*AK*CLST	VCF8 148
C..... UPPER HALF OF CYLINDER	VCF8 149
2 IF(K.LT.KS) GO TO 3	VCF8 150
KT=KT+1	VCF8 151
TT(KT)=T	VCF8 152
GAMMA(KT)=DELT*DGAMMA	VCF8 153
GAMMA(KT)=SIGMA*GAMMA(KT)	VCF8 154
X(KT)=(AK+SMALLM)*COS(THETAS)	VCF8 155
Y(KT)=(AK+SMALLM)*SIN(THETAS)	VCF8 156
IF(ISYM.EQ.0) GO TO 3	VCF8 157
KB=KB+1	VCF8 158
XB(KB)=X(KT)	VCF8 159
YB(KB)=-Y(KT)	VCF8 160
GMA B(KB)=-GAMMA(KT)	VCF8 161
TB(KB)=T	VCF8 162
3 CONTINUE	VCF8 163
IF(THETAS.LT.THTASYM.OR.ISYM.EQ.0) GO TO 100	VCF8 164
GO TO 101	VCF8 165
C***** SYMMETRIC CASE	VCF8 166
100 KBTEMP=KB	VCF8 167
KB=KB+1	VCF8 168
CALL BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,2)	VCF8 169
KB=KBTEMP	VCF8 170
C..... ASYMMETRIC CASE	VCF8 171
C..... LOWER HALF OF CYLINDER	VCF8 172
C..... SHEAR DRAG	VCF8 173
CDSB=0.0	VCF8 174

2	CLS B=0.0	VCF8 175
3	INT X=ISEP-1	VCF8 176
4	DO 5 I=1,INTX	VCF8 177
5	TRAPD=DS(I)/2.0*(TAWD(I)+TAWD(I+1))	VCF8 178
6	TRAPL=DS(I)/2.0*(TAWL(I)+TAWL(I+1))	VCF8 179
7	CDSB=CDSB+TRAPD	VCF8 180
8	CLSB=CLSB+TRAPL	VCF8 181
9	CDSK(K)=CDSK(K)-.50*AK*CDSB	VCF8 182
10	CLSK(K)=CLSK(K)-.50*AK*CLSB	VCF8 183
11	IF(K.LT.KSB) GO TO 7	VCF8 184
12	KB=KB+1	VCF8 185
13	TB(KB)=T	VCF8 186
14	GMA B(KB)=-DELT*DGAMMA	VCF8 187
15	GMA P(KB)=SIGMA*GMA B(KB)	VCF8 188
16	XB(KB)=(AK+SMALLM)*COS(THETASB)	VCF8 189
17	YB(KB)=(AK+SMALLM)*SIN(THETASB)	VCF8 190
18	CONTINUE	VCF8 191
19	GO TO 9	VCF8 192
20	101 CONTINUE	VCF8 193
21	C SYMMETRIC CASE	VCF8 194
22	CDSK(K)=2.0*CDSK(K)	VCF8 195
23	CLSK(K)=6.0	VCF8 196
24	DO 102 I=1,IXTRSET	VCF8 197
25	UBNBIG(I)=-UTNBIG(I)	VCF8 198
26	DO 102 J=1,NBIG	VCF8 199
27	102 UB(I,J)=-UT(I,J)	VCF8 200
28	IXBPSET=IXTRSET	VCF8 201
29	THETASB=PIZ-THETAS	VCF8 202
30	KSB=KS	VCF8 203
31	KB=KT	VCF8 204
32	CONTINUE	VCF8 205
33	C XDOT,YDOT CALCULATIONS	VCF8 206
34	IF(K.LT.KS) GO TO 85	VCF8 207
35	CALL VEL(X,Y,XDOT,YDOT,NDIM,1,KT,1)	VCF8 208
36	IF(ISYM.EQ.0) GO TO 81	VCF8 209
37	DO 80 I=1,KT	VCF8 210
38	XDOTB(I)=XDOT(I)	VCF8 211
39	YDOTB(I)=-YDOT(I)	VCF8 212
40	80 CONTINUE	VCF8 213
41	GO TO 82	VCF8 214
42	81 CONTINUE	VCF8 215
43	CALL VEL(XB,YB,XDOTB,YDOTB,NDIM,1,KB,2)	VCF8 216
44	82 CONTINUE	VCF8 217
45	IF(KRT.EQ.0) GO TO 85	VCF8 218
46	CALL VEL(XRT,YRT,XRDOT,YRDOT,NDIM,1,KRT,4)	VCF8 219
47	IF(ISYM.EQ.0) GO TO 83	VCF8 220
48	DO 83 I=1,KRT	VCF8 221
49	XRDOTB(I)=XRDOT(I)	VCF8 222
50	YRDOTB(I)=-YRDOT(I)	VCF8 223
51	83 CONTINUE	VCF8 224
52	GO TO 85	VCF8 225
53	84 CONTINUE	VCF8 226
54	CALL VEL(XRB,YRB,XRDOTB,YRDOTB,NDIM,1,KRB,5)	VCF8 227
55	85 CONTINUE	VCF8 228
56	C END XDOT,YDOT CALCULATIONS	VCF8 229
57	C PRINTOUT	VCF8 230
58	C	VCF8 231
59	IPUN=IPUN-1	VCF8 232

IF(K/KPUN)*KPUN.EQ.K) IPUN=2	VCF8 233
IW=6	VCF8 234
IF(IPUN.GE.1) IW=7	VCF8 235
IF(K.GE.KS) WRITE(6,600)	VCF8 236
IF(K.GE.KS) CALL WRIT(X,Y,XDOT,YDOT,GAMMA,XB.YB,XDOTB,YDOTB,GHAB,	VCF8 237
11,KT,1,KB,K,NDIM,IW)	VCF8 238
IF(KRT.GT.0) WRITE(6,602)	VCF8 239
IF(KRT.GT.0) CALL WRIT(XRT,YRT,XRDOT,YRDOT,GMRT,XRB,YRB,XRDOTB,	VCF8 240
*YRDOTB,GMRB,1,KRT,1,KRB,K,NDIM,IW)	VCF8 241
C E NO PRINTOUT	VCF8 242
CDP=4.0*PI*AKDOT	VCF8 243
SEPDEG=THETAS*RTOD	VCF8 244
IF(K.GE.KS.AND.LEVEL.GE.4) WRITE(6,622)T,ZHAT,AK,AKDOT,SEPDEG	VCF8 245
IF(K.GE.KS) CALL DQI(PDRAG,CDP,K,5,ISYM)	VCF8 246
CDPK(K)=.5*AK*CDP	VCF8 247
CDK(K)=CDSK(K)+CDPK(K)	VCF8 248
CDN=(AW/RW)*CDK(K)*SQA	VCF8 249
WRITE(6,603)K,CDPK(K),CDSK(K),GDK(K),CDN	VCF8 250
IF(KPUN.NE.0) WRITE(7,755)ZHAT,CDK(K),T,K	VCF8 251
IF(K.LT.KR) GO TO 78	VCF8 252
KTEMP=KRT	VCF8 253
CALL RSEP(THETAS,THTASR,ST,UTNBIG,IXTRSET,INITAL,1)	VCF8 254
CALL RVTX(XRT,YRT,GMRT,THTASR,THTASMT,THTASRT,UZZT,UZZSQT,GAMART,	VCF8 255
11,KRT,NT,NFT,IFLGT,GMTLT,NDIM,TRT)	VCF8 256
IF(KTEMP.EQ.KRT) GO TO 78	VCF8 257
KRB=KRT	VCF8 258
TRB(KRB)=TRT(KRT)	VCF8 259
GMRB(KRB)=-GMRT(KRT)	VCF8 260
XRB(KRB)=XRT(KRT)	VCF8 261
YRB(KRB)=-YRT(KRT)	VCF8 262
CALL VEL(XRT,YRT,XRDOT,YRDOT,NDIM,KRT,KRT,4)	VCF8 263
XRDOTB(KRB)=XRDOT(KRT)	VCF8 264
YRDOTB(KRB)=-YRDOT(KRT)	VCF8 265
IF(ISYM.EQ.1) GO TO 79	VCF8 266
KTEMP=KRB	VCF8 267
CALL RSEP(THETASB,THTASR,SB,UBNBIG,IXBRSET,INITAL,2)	VCF8 268
CALL RVTX(XRB,YRB,GMRB,THTASR,THTASMB,THTASRB,UZZB,UZZSQB,GAMARB,	VCF8 269
12,KRB,NB,NFB,IFLGB,GMTLB,NDIM,TRB)	VCF8 270
IF(KTEMP.EQ.KRB) GO TO 78	VCF8 271
CALL VEL(XRB,YRB,XRDOTB,YRDOTB,NDIM,KRB,KRB,5)	VCF8 272
79 CONTINUE	VCF8 273
78 CONTINUE	VCF8 274
C VORTEX MOTION	VCF8 275
IF(K.LT.KS) GO TO 95	VCF8 276
CALL VM(X,Y,XDOT,YDOT,NDIM,1,KT,DELT)	VCF8 277
IF(ISYM.EQ.0) GO TO 91	VCF8 278
DO 90 I=1,KT	VCF8 279
XB(I)=X(I)	VCF8 280
YB(I)=-Y(I)	VCF8 281
90 CONTINUE	VCF8 282
GO TO 92	VCF8 283
91 CONTINUE	VCF8 284
CALL VM(XB,YB,XDOTB,YDOTB,NDIM,1,KB,DELT)	VCF8 285
92 CONTINUE	VCF8 286
IF(KRT.EQ.0) GO TO 95	VCF8 287
CALL VM(XRT,YRT,XRDOT,YRDOT,NDIM,1,KRT,DELT)	VCF8 288
IF(ISYM.EQ.0) GO TO 94	VCF8 289
DO 93 I=1,KRT	VCF8 290

1	XRB(I)=XRT(I)	VCF8 291
2	YRB(I)=-YRT(I)	VCF8 292
93	CONTINUE	VCF8 293
	GO TO 95	VCF8 294
94	CONTINUE	VCF8 295
	CALL VM(XRB,YRB,XRDOTB,YRDOTB,NDIM,1,KRB,DELT)	VCF8 296
95	CONTINUE	VCF8 297
C.....	END VORTEX MOTION	VCF8 298
	CALL SECOND(TIME)	VCF8 299
	WRITE(6,612) TIME	VCF8 300
	IF(K.EQ.KFINAL.OR.TIME.GE.TFINAL.OR.ZHAT.GE.ZFINAL) GO TO 221	VCF8 301
	GO TO 69	VCF8 302
221	CONTINUE	VCF8 303
	WRITE(LW) K,KS,KR,KT,KB,KRT,KRB,T,DELT,NBIG,IXTRSET,IXBRSET	VCF8 304
	WRITE(LW) (UTNBIG(I), (UT(I,J), J=1,NBIG), I=1, IXTRSET)	VCF8 305
	IF(ITYM.EQ.0) WRITE(LW) (UBNBIG(I), (UB(I,J), J=1,NBIG), I=1, IXBRSET)	VCF8 306
	IF(K.LT.KS) GO TO 223	VCF8 307
	WRITE(LW) (X(I), Y(I), XDOT(I), YDOT(I), GAMMA(I), TT(I), I=1, KT)	VCF8 308
	WRITE(LW) (XB(I), YB(I), XDOTB(I), YDOTB(I), GMAB(I), TB(I), I=1, KB)	VCF8 309
	IF(KRT.EQ.0) GO TO 222	VCF8 310
	WRITE(LW) (XRT(I), YRT(I), XRDOT(I), YRDOT(I), GMRT(I), TRT(I), I=1, KRT)	VCF8 311
	WRITE(LW) (XRB(I), YRB(I), XRDOTB(I), YRDOTB(I), GMRB(I), TRB(I),	VCF8 312
	I=1, KRB)	VCF8 313
222	CONTINUE	VCF8 314
	IF(K.LT.KR) GO TO 223	VCF8 315
	WRITE(LW) THTASMT, THTASRT, UZZT, UZZS, GAMART, IFLGT, THTASMB,	VCF8 316
	1 THTASRB, UZZB, UZZSOB, GAMARB, IFLGB, NT.NB, NFB, GMTLT, GMTLB	VCF8 317
223	CONTINUE	VCF8 318
600	FORMAT(/ /50X*POINT VORTEX LOCATIONS*/27X*TOP BOUNDARY LAYER*44X	VCF8 319
	1*BOTTOM BOUNDARY LAYER*//,	VCF8 320
	1 * A K*6X*X(A)*8X*Y(A)*7X*XDOT(A)*4X*YDOT(A)*6X	VCF8 321
	1*GAMMA(A)*3X*XB(A)*7X*YB(A)*6X*XDOTB(A)*4X*YDOTB(A)*6X*GMAB(A)*	VCF8 322
601	FORMAT(* 2DUS REYNOLDS NO.=*F11.2, / / * 2DUS PARAMETERS*/	VCF8 323
	1*DELT=*F5.3, / * RC=*F5.3, / * SIGMA=*F5.3, / / * PROGRAM CONTROL*7	VCF8 324
	2* KFINAL=*I3, / * TFINAL=*F6.1, / * ZFINAL=*F5.3, / * LR=*I3, / * LW=*I3, /	VCF8 325
	4* LEVEL=*I3, / * KPUN=*I3)	VCF8 326
602	FORMAT(/ /25X*TOP REAR SHEAR LAYER*44X*BOTTOM REAR SHEAR LAYER*//,	VCF8 327
	1 * A K*6X*XRT(A)*6X*YRT(A)*5X*XRDOT(A)*3X*YRDOT(A)*5X	VCF8 328
	1*GMRT(A)*4X*XRB(A)*5X*YRB(A)*4X*XRDOTB(A)*2X*YRDOTB(A)*4X	VCF8 329
	1*GMAB(A)*)	VCF8 330
603	FORMAT(/ / * K=*I3, / * CDPK=*F12.6, / * CDSK=*F12.6, / * CDK=*F12.6,	VCF8 331
	1/* CON=*F12.6)	VCF8 332
605	FORMAT(1H1,20X,16A5 //)	VCF8 333
606	FORMAT(* ANGLE OF ATTACK=*F5.1, * DEGREES*/ * 3DS REYNOLDS NO.=*F11.	VCF8 334
	12)	VCF8 335
611	FORMAT(/ /42X*OMAX=*F7.4, /42X* AM=*F7.4, /42X* F=*F7.4, /42X,	VCF8 336
	1* S=*F7.4, //)	VCF8 337
612	FORMAT(* ELAPSED TIME=*F12.6)	VCF8 338
620	FORMAT(* TI=*F12.6, * ZHAT(TI)=*F12.6, / * AK(TI)=*	VCF8 339
	1F12.6, * AKDOT(TI)=*F12.6, * CDPI=*F12.6)	VCF8 340
622	FORMAT(1H1,40X*PRESSURE DISTRIBUTION*/14X*T=*F7.4, 5X*ZHAT=*F7.4, 5X	VCF8 341
	1*AK=*F7.4, 5X*AKDOT=*F7.4, 5X*THETAS=*F7.2, / /3X*DEG*8X*PHIVT*8X	VCF8 342
	2*PHIPT*8X*2(PHIT)*6X*-PSIKSQ0*5X*CPK*10X*PDRAG*8X*UTAN*)	VCF8 343
705	FORMAT(16A5)	VCF8 344
706	FORMAT(3F12.6)	VCF8 345
708	FORMAT(I3,9X,2F12.6)	VCF8 346
709	FORMAT(4I2)	VCF8 347
755	FORMAT(3F12.6,I3)	VCF8 348

```

FUNCTION AN(L,N, IDIM, JDIM, U, W)
DIMENSION U(IDIM,2, JDIM), W(IDIM, JDIM)
COMMON/ALOCK11/DS(52), DZ2, DZ8, DZSQ, DZSQ2, DZSQ4, DT2
AN=DZ8*W(L,N)-DZSQ4
RETURN
ENTRY BN
DX4=.25/DS(L)
BN=DT2+DX4*(U(L+1,2,N)+U(L+1,1,N))+DZSQ2
RETURN
ENTRY CN
CN=-DZ8*W(L,N)-DZSQ4
RETURN
END

```

```

AN 1
AN 2
AN 3
AN 4
AN 5
AN 6
AN 7
AN 8
AN 9
AN 10
AN 11
AN 12
AN 13

```

SUBROUTINE BLBOX(SMALLM,DGAMMA,TAWD,TAWL,U,W,UT,UB,IDIM,JDIM,MODE) BLBOX	1
DIMENSION U(IDIM,2,JDIM),W(IDIM,JDIM),UT(IDIM,JDIM),UB(IDIM,JDIM) BLBOX	2
DIMENSION SUP(49),SUB(49),DIAG(49),R(49) BLBOX	3
DIMENSION TAWD(53),TAWL(53) BLBOX	4
DIMENSION THTA(53) BLBOX	5
DIMENSION NX(53),DEG(53) BLBOX	6
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCHXSQ,SIGMA,LEVEL BLBOX	7
COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB BLBOX	8
COMMON/ELIPS2/AKPHALF,AKDOTPH BLBOX	9
COMMON/BLOCK5/AK,AKSQ,AKHALF,AKDOT BLBOX	10
COMMON/BLOCK10/THETAS,THETASB,THETA BLBOX	11
COMMON/BLOCK11/DS(52),DZ2,DZ8,DZSQ,DZSQ2,DZSQ4,DT2 BLBOX	12
COMMON/BLOCK28/OX,DZ,INTX1,NBIG1 BLBOX	13
COMMON/BLOCK30/T,TI,DELT,DELTB BLBOX	14
COMMON/BLBOX2/TAU,PT4,NBIG BLBOX	15
COMMON/BLBOX12/ZN(51),ISEP BLBOX	16
COMMON/BLBOX13/KTS,IXTRSET,IXBRSET,UTNBIG(53),UBNBIG(53) BLBOX	17
COMMON/BLBOX14/ZHAT,AKTI BLBOX	18
COMMON/BLOCK14/S(53),ST(53),SB(53) BLBOX	19
C***** MODE1 IMPLIES 0 DEGREES.LT.THETA.LT. 80 DEGREES BLBOX	20
C***** MODE2 IMPLIES -80 DEGREES.LT.THETA.LT. 0 DEGREES BLBOX	21
C***** RBAR CORRESPONDS TO Z BLBOX	22
C***** THETA CORRESPONDS TO X BLBOX	23
LEVEL=5 BLBOX	24
IF(K.GT.KTS.OR.MODE.EQ.2) GO TO 9 BLBOX	25
C***** INITIAL DETERMINES INITIAL GRID POINTS BLBOX	26
RTOD=180.0/PI BLBOX	27
DTOR=PI/180.0 BLBOX	28
INITAL=53 BLBOX	29
INTX1=IXTRSET BLBOX	30
IF(IXBRSET.GT.IXTRSET) INTX1=IXBRSET BLBOX	31
DELS=5.0*DTOR BLBOX	32
ST(1)=0.0 BLBOX	33
SB(1)=PI2 BLBOX	34
DO 19 I=2,INITAL BLBOX	35
IF(I.LE.15.OR.I.GE.36) GO TO 18 BLBOX	36
ST(I)=ST(I-1)+DELS/5.0 BLBOX	37
SB(I)=SB(I-1)-DELS/5.0 BLBOX	38
GO TO 19 BLBOX	39
18 ST(I)=ST(I-1)*DELS BLBOX	40
SB(I)=SB(I-1)-DELS BLBOX	41
19 CONTINUE BLBOX	42
Z0=0.0 BLBOX	43
INTZ=50 BLBOX	44
INTZ1=INTZ+1 BLBOX	45
NBIG1=INTZ BLBOX	46
NBIG=INTZ1 BLBOX	47
DZ=.14 BLBOX	48
DZ2=0.5/DZ BLBOX	49
DZ8=0.125/DZ BLBOX	50
DZSQ=1.0/(DZ*DZ) BLBOX	51
DZSQ2=0.5/(DZ*DZ) BLBOX	52
DZSQ4=0.25/(DZ*DZ) BLBOX	53
ZN(1)=Z0 BLBOX	54
C***** RBAR VARIATION BLBOX	55
DO 22 J=2,NBIG BLBOX	56
RJ=J BLBOX	57
Z=Z0+(RJ-1.0)*DZ BLBOX	58


```

72  ZN(J)=Z
    E=10.0** -5
    THE TAS=ST(IXTRSET)
    THE TASB=SB(IXBRSET)
C***** THETA VARIATION
9   CONTINUE
C***** TOP HALF
    INTX1=ISEP=IXTRSET
    IF(MODE.EQ.2)INTX1=ISEP=IXBRSET
    INTX=INTX1-1
    IF(K.EQ.1)CALL IC(AKI,U,IDIM,JOIM,MODE)
    DO 200 I=1,INITAL
    GO TO(204,206) MODE
204 S(I)=ST(I)
    GO TO 208
206 S(I)=SB(I)
208 THTA(I)=S(I)
    DEG(I)=THTA(I)*RTOD
    IF(I.EQ.1) GO TO 213
    DS(I-1)=S(I)-S(I-1)
    DS(I-1)=DS(I-1)*AKHALF**2
C***** B.C. RBAR=0
213 IF(I.GT.INTX1) GO TO 214
    U(I,1,1)=0.0
    U(I,2,1)=0.0
    JI=2
    GO TO 215
214 JI=NBIG
215 DO 200 J=JI,NBIG
C***** B.C. THETA=0
    IF(1.NE.1) GO TO 201
    U(I,1,J)=0.0
    U(I,2,J)=0.0
    GO TO 200
201 IF(J.NE.NBIG) GO TO 202
C***** B.C. RBAR=INF
    THETA=THTA(I)
    CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)
    CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)
    IF(I.GT.INTX1) GO TO 203
    U(I,2,NBIG)=AK*(PSIK1R+PSIKRP)
    GO TO (205,207) MODE
203 GO TO (211,212) MODE
205 U(I,1,NBIG)=UTNBIG(I)
211 UTNBIG(I)=AK*(PSIK1R+PSIKRP)
    GO TO 200
207 U(I,1,NBIG)=UBNBIG(I)
212 UBNBIG(I)=AK*(PSIK1R+PSIKRP)
    GO TO 200
202 IF(K.EQ.1) GO TO 200
    GO TO(209,210) MODE
209 U(I,1,J)=U(I,J)
    GO TO 200
210 U(I,1,J)=UB(I,J)
200 CONTINUE
40  CONTINUE
    DT2=.5/DELT
    DO 180 I=1,INTX

```

```

BLBOX 59
BLBOX 60
BLBOX 61
BLBOX 62
BLBOX 63
BLBOX 64
BLBOX 65
BLBOX 66
BLBOX 67
BLBOX 68
BLBOX 69
BLBOX 70
BLBOX 71
BLBOX 72
BLBOX 73
BLBOX 74
BLBOX 75
BLBOX 76
BLBOX 77
BLBOX 78
BLBOX 79
BLBOX 80
BLBOX 81
BLBOX 82
BLBOX 83
BLBOX 84
BLBOX 85
BLBOX 86
BLBOX 87
BLBOX 88
BLBOX 89
BLBOX 90
BLBOX 91
BLBOX 92
BLBOX 93
BLBOX 94
BLBOX 95
BLBOX 96
BLBOX 97
BLBOX 98
BLBOX 99
BLBO 100
BLBO 101
BLBO 102
BLBO 103
BLBO 104
BLBO 105
BLBO 106
BLBO 107
BLBO 108
BLBO 109
BLBO 110
BLBO 111
BLBO 112
BLBO 113
BLBO 114
BLBO 115
BLBO 116

```

	NCYCLE=0	BLBO 117
	TAUNEW=0.0	BLBO 118
	W(I,1)=0.0	BLBO 119
50	NCYCLE=NCYCLE+1	BLBO 120
	DO 140 J=2,NBIG1	BLBO 121
	TAUS=TAUNEW	BLBO 122
	Z=ZN(J)	BLBO 123
90	IF (NCYCLE.GT.1) GO TO 100	BLBO 124
	U(I+1,2,J)=UAPRX1(I,J,IDIM,JDIM,U)	BLBO 125
100	DZDX=0Z/(4.0*DS(I))	BLBO 126
	W(I,J)=W(I,J-1)-DZDX*(U(I+1,2,J)+U(I+1,2,J-1)+	BLBO 127
	UL(I,J,IDIM,JDIM,U)+UL(I,J-1,IDIM,JDIM,U))	BLBO 128
	W(I,J)=W(I,J)-DZ*AKDOTPH/AKPHALF	BLBO 129
	IF(J.EQ.2) GO TO 110	BLBO 130
	SUB(J-2)=CN(I,J,IDIM,JDIM,U,W)	BLBO 131
110	DIAG(J-1)=BN(I,J,IDIM,JDIM,U,W)	BLBO 132
	IF (J.EQ.NBIG1) GO TO 120	BLBO 133
	SUP(J-1)=AN(I,J,IDIM,JDIM,U,W)	BLBO 134
120	R(J-1)=DN(I,J,IDIM,JDIM,U,W)	BLBO 135
140	CONTINUE	BLBO 136
160	CALL TRID(49,SUP,SUB,DIAG,R)	BLBO 137
	DO 170 JJ=2,NBIG1	BLBO 138
	U(I+1,2,JJ)=R(JJ-1)	BLBO 139
	IF(MODE.EQ.1) UT(I+1,JJ)=R(JJ-1)	BLBO 140
	IF(MODE.EQ.2) UB(I+1,JJ)=R(JJ-1)	BLBO 141
170	CONTINUE	BLBO 142
C	***** WALL FRICTION AT EACH X STATION	BLBO 143
	TAUNEW=2.0*U(I+1,2,2)/DZ	BLBO 144
	IF(NCYCLE.GE.20) GO TO 179	BLBO 145
	IF(ABS(TAUNEW-TAUS).GT.E.OR.NCYCLE.LE.2) GO TO 50	BLBO 146
179	CONTINUE	BLBO 147
	NX(I+1)=NCYCLE	BLBO 148
180	CONTINUE	BLBO 149
C.....	UI GOES TO UBAR	BLBO 150
C.....	UI REMAINS IN UT AND UB ARRAYS	BLBO 151
	DO 182 I=1,INTX1	BLBO 152
	DO 182 J=1,NBIG	BLBO 153
182	U(I,2,J)=U(I,2,J)/AK	BLBO 154
	IF(MODE.EQ.2) GO TO 28	BLBO 155
	IF(LEVEL.GE.4)CALL RITE(2,IDIM,JDIM,U,ZN,NX,S,DEG,ZHAT,DELS,K)	BLBO 156
28	CONTINUE	BLBO 157
C	.SEARCH ALONG SURFACE FOR ZERO SHEAR POINT	BLBO 158
300	DO 302 I=1,INTX1	BLBO 159
	DUDR=(U(I,2,2)-U(I,2,1))/DZ	BLBO 160
	TAW=(2.0/SRE)*DUDR	BLBO 161
	GO TO(305,306) MODE	BLBO 162
305	IF(I.EQ.1.OR.I.EQ.INITAL.OR.TAW.GT.0.0) GO TO 301	BLBO 163
	ISEP=I-1	BLBO 164
	GO TO 303	BLBO 165
306	IF(I.EQ.1.OR.I.EQ.INITAL.OR.TAW.LT.0.0) GO TO 301	BLBO 166
	ISEP=I-1	BLBO 167
	GO TO 303	BLBO 168
301	TAWL(I)=TAW*COS(THTA(I))	BLBO 169
302	TAWD(I)=TAW*SIN(THTA(I))	BLBO 170
303	CONTINUE	BLBO 171
C	.FLOW HAS NOT SEPARATED... SKIP W CALC	BLBO 172
	IF(ISEP.EQ.INITAL) GO TO 309	BLBO 173
C	.FLOW HAS SEPARATED ... SEPARATION POINT	BLBO 174

C		.DEFINED 5 DEGREES UPSTREAM	BLBO 175
	310	IF(MODE.NE.1) GO TO 304	BLBO 176
C		.TOP	BLBO 177
		IXRSET=ISEP	BLBO 178
		THE TAS=THTA(ISEP)	BLBO 179
		IF(KS.EQ.1000) KS=K	BLBO 180
		GO TO 307	BLBO 181
C		.BOTTOM	BLBO 182
	304	IXRSET=ISEP	BLBO 183
		THE TASB=THTA(ISEP)	BLBO 184
		IF(KSB.EQ.1000) KSB=K	BLBO 185
	307	IFLAG=0	BLBO 186
		PSIKR=U(ISEP,2,NBIG)	BLBO 187
		U00=ABS(PSIKR)	BLBO 188
		SMALLM=DELT*U00/PI2	BLBO 189
		DGAMMA=(U00**2)/2.0	BLBO 190
	309	CONTINUE	BLBO 191
		WRITE(6,602) DGAMMA, SMALLM	BLBO 192
	602	FORMAT(//* DGAMMA=*F12.6,5X*SMALLM=*F12.6)	BLBO 193
		RETURN	BLBO 194
		END	BLBO 195

FUNCTION CPC(THTA)	CPC	1
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL	CPC	2
COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB	CPC	3
COMMON/PRSURE/XX,YY,SINE,COSINE	CPC	4
COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),	CPC	5
1 XRB(100),YRB(100)	CPC	6
COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)	CPC	7
COMMON/BLOCK3/XDOT(100),YDOT(100),XDOTB(100),YDOTB(100),	CPC	8
1 XRDOT(100),YRDOT(100),XRDOTB(100),YRDOTB(100)	CPC	9
COMMON/BLOCK4/TT(100),TB(100),TRT(100),TRB(100)	CPC	10
COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	CPC	11
COMMON/BLOCK7/PHIVT,PHIPT,P1,P2,P3,P4	CPC	12
COMMON/BLOCK10/THETAS,THETASB,THETA	CPC	13
INTEGER A	CPC	14
REAL L,LB	CPC	15
THETA=THTA	CPC	16
NDIM=100	CPC	17
PHIVT=0.0	CPC	18
SINE=SIN(THETA)	CPC	19
COSINE=COS(THETA)	CPC	20
IF(KT.LE.1) GO TO 10	CPC	21
XX=AK*COSINE	CPC	22
YY=AK*SINE	CPC	23
KMINUS1=KT-1	CPC	24
CALL PV(X,Y,XDOT,YDOT,GAMMA,NDIM,1,KMINUS1,SUM,TT)	CPC	25
PHIVT=PHIVT+SUM	CPC	26
IF(KRT.EQ.0) GO TO 10	CPC	27
CALL PV(XRT,YRT,XRDOT,YRDOT,GMRT,NDIM,1,KRT,SUM,TRT)	CPC	28
PHIVT=PHIVT+SUM	CPC	29
10 IF(KB.LE.1) GO TO 20	CPC	30
KMINUS1=KB-	CPC	31
CALL PV(XB,YB,XDOTB,YDOTB,GMAB,NDIM,1,KMINUS1,SUM,TB)	CPC	32
PHIVT=PHIVT+SUM	CPC	33
IF(KRB.EQ.0) GO TO 20	CPC	34
CALL PV(XRB,YRB,XRDOTB,YRDOTB,GMRB,NDIM,1,KRB,SUM,TRB)	CPC	35
PHIVT=PHIVT+SUM	CPC	36
20 CONTINUE	CPC	37
PHIPT=2.0*AKDOT*COSINE	CPC	38
PHIT=PHIPT+PHIVT	CPC	39
P1=2.0*PHIT	CPC	40
CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)	CPC	41
CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)	CPC	42
PSIKR2=PSIK1R+PSIKRP	CPC	43
IF(PSIKR2.LE.-.1.AND.KR.EQ.1000) KR=K	CPC	44
P4=PSIKR2	CPC	45
PSIKSQD=PSIKR2**2	CPC	46
P2=-PSIKSQD	CPC	47
CPC=2.0*PHIT->SIKSQD	CPC	48
P3=CPC	CPC	49
RETURN	CPC	50
END	CPC	51

FUNCTION DN(L,N, IDIM, JDIM, U, W)	DN	1
DIMENSION U(IDIM, 2, JDIM), W(IDIM, JDIM)	DN	2
COMMON/BLBOX2/TAU, PT4, NBIG	DN	3
COMMON/BLOCK11/DS(52), DZ2, DZ8, DZSQ, DZSQ2, DZSQ4, DT2	DN	4
COMMON/BLOCK20/DX, DZ, INTX1, NBIG1	DN	5
DX8 = .125/DS(L)	DN	6
DX2 = .5/DS(L)	DN	7
DN = DX8 * U(L+1, 2, N) ** 2 - DT2 * UM(L, N, IDIM, JDIM, U) -	DN	8
1 DX2 * US(L, N, IDIM, JDIM, U) * UL(L, N, IDIM, JDIM, U)	DN	9
DN = DN + DT2 * (U(L+1, 2, NBIG) + UM(L, NBIG, IDIM, JDIM, U))	DN	10
DN = DN + DX2 * (U(L+1, 2, NBIG) / 4.0 + US(L, NBIG, IDIM, JDIM, U))	DN	11
1 * (U(L+1, 2, NBIG) + UL(L, NBIG, IDIM, JDIM, U))	DN	12
UNS1 = US(L, N+1, IDIM, JDIM, U)	DN	13
UNS2 = US(L, N+1, IDIM, JDIM, U) - 2.0 * US(L, N, IDIM, JDIM, U)	DN	14
IF(N.EQ.1) GO TO 30	DN	15
UNS1 = UNS1 - US(L, N-1, IDIM, JDIM, U)	DN	16
UNS2 = UNS2 + US(L, N-1, IDIM, JDIM, U)	DN	17
30 DN = DN - DZ2 * UNS1 * W(L, N) + DZSQ * UNS2	DN	18
IF (N.LT.NBIG1) GO TO 40	DN	19
DN = DN - AN(L, NBIG1, IDIM, JDIM, U, W) * U(L+1, 2, NBIG)	DN	20
40 RETURN	DN	21
END	DN	22

	SUBROUTINE DQI(FCT,CK,K,N,ISYM)	DQI	1
	. THE FRONT OF THE CYLINDER IS DIVIDED INTO 2N EQUAL PARTS	DQI	2
C	. THE FRONT IS DEFINED AS -60 DEG.LE.THETA.LE.60 DEG	DQI	3
C	. THE BACK OF THE CYLINDER IS DIVIDED INTO 8N EQUAL PARTS	DQI	4
C	. N MUST BE AN EVEN INTEGER	DQI	5
	PI=4.0*ATAN(1.0)	DQI	6
	IFLAG=0	DQI	7
	XX=0.0	DQI	8
	SIMP=0.0	DQI	9
	H=6.0/N*PI/180.0	DQI	10
	NB=2*N	DQI	11
	H3=H/3.0	DQI	12
	NF=N/2	DQI	13
	F0=FCT(XX)	DQI	14
10	DO 1 I=1,NF	DQI	15
	XX=XX+H	DQI	16
	F1=FCT(XX)	DQI	17
	XX=XX+H	DQI	18
	F2=FCT(XX)	DQI	19
	SIMP=SIMP+H3*(F2+4.0*F1+F0)	DQI	20
1	F0=F2	DQI	21
	IF(IFLAG.EQ.1) GO TO 5	DQI	22
	H=H/2.0	DQI	23
	H3=H/3.0	DQI	24
20	DO 2 I=1,NB	DQI	25
	XX=XX+H	DQI	26
	F1=FCT(XX)	DQI	27
	XX=XX+H	DQI	28
	F2=FCT(XX)	DQI	29
	SIMP=SIMP+H3*(F2+4.0*F1+F0)	DQI	30
2	F0=F2	DQI	31
	IF(IFLAG.EQ.0) GO TO 30	DQI	32
	H=H*2.0	DQI	33
	H3=H/3.0	DQI	34
	GO TO 10	DQI	35
30	KMINUS1=K-1	DQI	36
	IF(ISYM.EQ.0) GO TO 4	DQI	37
	CK=2.0*SIMP	DQI	38
	RETURN	DQI	39
4	IFLAG=1	DQI	40
	GO TO 20	DQI	41
5	CK=SIMP	DQI	42
	RETURN	DQI	43
	END	DQI	44

	SUBROUTINE FREEVTX(PSIK1X,PSIK1Y,PSIK1R,IA)	FREEVT 1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCHXSQ,SIGMA,LEVEL	FREEVT 2
	COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB	FREEVT 3
	COMMON/BLOCK1/X(100),Y(100),XB(100),YB(100),XRT(100),YRT(100),	FREEVT 4
	1 XRB(100),YRB(100)	FREEVT 5
	COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)	FREEVT 6
	COMMON/BLOCK4/TT(100),TB(100),TRT(100),TRB(100)	FREEVT 7
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	FREEVT 8
	COMMON/BLOCK10/THETAS,THETASB,THETA	FREEVT 9
	COMMON/VTXI/XX,YY	FREEVT 10
	INTEGER A,B,ALPHA	FREEVT 11
	REAL LB,L	FREEVT 12
	NDIM=100	FREEVT 13
	PSIK1R=0.0	FREEVT 14
	PSIK1X=0.0	FREEVT 15
	PSIK1Y=0.0	FREEVT 16
	GO TO(1,2,3,4,5) IA	FREEVT 17
1	XX=X(ALPHA)	FREEVT 18
	YY=Y(ALPHA)	FREEVT 19
C DERIVATIVE OF PSI W.R.T. X	FREEVT 20
C VORTEX MOTION TOP	FREEVT 21
	GO TO 20	FREEVT 22
2	XX=XB(ALPHA)	FREEVT 23
	YY=YB(ALPHA)	FREEVT 24
C DERIVATIVE OF PSI W.R.T. Y	FREEVT 25
C VORTEX MOTION BOTTOM	FREEVT 26
	GO TO 20	FREEVT 27
4	XX=XRT(ALPHA)	FREEVT 28
	YY=YRT(ALPHA)	FREEVT 29
	GO TO 20	FREEVT 30
5	XX=XRB(ALPHA)	FREEVT 31
	YY=YRB(ALPHA)	FREEVT 32
20	IF(KT.LE.1) GO TO 23	FREEVT 33
	KMINUS1=KT-1	FREEVT 34
	CALL PSI(X,Y,GAMMA,NDIM,1,KMINUS1,IA,1,SUM,SUM1,TT)	FREEVT 35
	PSIK1X=PSIK1X+SUM	FREEVT 36
	PSIK1Y=PSIK1Y+SUM	FREEVT 37
	IF(KRT.EQ.0) GO TO 23	FREEVT 38
	CALL PSI(XRT,YRT,GMRT,NDIM,1,KRT,IA,4,SUM,SUM1,TRT)	FREEVT 39
	PSIK1X=PSIK1X+SUM	FREEVT 40
	PSIK1Y=PSIK1Y+SUM	FREEVT 41
23	IF(KB.LE.1) RETURN	FREEVT 42
	KMINUS1=KB-1	FREEVT 43
	CALL PSI(XB,YB,GMAB,NDIM,1,KMINUS1,IA,2,SUM,SUM1,TB)	FREEVT 44
	PSIK1X=PSIK1X+SUM	FREEVT 45
	PSIK1Y=PSIK1Y+SUM	FREEVT 46
	IF(KRB.EQ.0) GO TO 51	FREEVT 47
	CALL PSI(XRB,YRB,GMRB,NDIM,1,KRB,IA,5,SUM,SUM1,TRB)	FREEVT 48
	PSIK1X=PSIK1X+SUM	FREEVT 49
	PSIK1Y=PSIK1Y+SUM	FREEVT 50
51	CONTINUE	FREEVT 51
	RETURN	FREEVT 52
3	XX=AK*COS(THETA)	FREEVT 53
	YY=AK*SIN(THETA)	FREEVT 54
C DERIVATIVE OF PSI W.R.T. R ON SURFACE	FREEVT 55
	IF(KT.LE.1) GO TO 34	FREEVT 56
	KMINUS1=KT-1	FREEVT 57
	CALL PSI(X,Y,GAMMA,NDIM,1,KMINUS1,IA,IB,SUM,SUM1,TT)	FREEVT 58

	PSIK1R=PSIK1R+SUM	FREEV 59
	IF(KRT.EQ.0) GO TO 34	FREEV 60
	CALL PSIONE(XRT,YRT,GMRT,NDIM,1,KRT,IA,IB,SUM,SUM1,TRT)	FREEV 61
	PSIK1R=PSIK1R+SUM	FREEV 62
34	IF(KB.LE.1) RETURN	FREEV 63
	KMINUS1=KB-1	FREEV 64
	CALL PSIONE(XB,YB,GMAB,NDIM,1,KMINUS1,IA,IB,SUM,SUM1,TB)	FREEV 65
	PSIK1R=PSIK1R+SUM	FREEV 66
	IF(KRB.EQ.0) GO TO 39	FREEV 67
	CALL PSIONE(XRB,YRB,GMRB,NDIM,1,KRB,IA,IB,SUM,SUM1,TRB)	FREEV 68
	PSIK1R=PSIK1R+SUM	FREEV 69
39	CONTINUE	FREEV 70
	RETURN	FREEV 71
	END	FREEV 72

	SUBROUTINE NONDIM(DMAX,RW,AW,F,S,L,PI)	NONDIM 1
	DIMENSION RZ(20)	NONDIM 2
	REAL L	NONDIM 3
	IEND=20	NONDIM 4
	DZSTAR=L/IEND	NONDIM 5
	WRITE(6,51)L	NONDIM 6
51	FORMAT(17//38X*BODY GEOMETRY(DIMENSIONAL LENGTH=*F7.3,*)*7/35X,	NONDIM 7
	1*ZSTAR*11X*RZERO(ZSTAR)*)	NONDIM 8
	ZSTAR=0.0	NONDIM 9
	DO 1 I=1,IEND	NONDI 10
	RZ(I)=RZERO(ZSTAR)	NONDI 11
	WRITE(6,50)ZSTAR,RZ(I)	NONDI 12
50	FORMAT(35XF7.4,13XF7.4)	NONDI 13
1	ZSTAR=ZSTAR+DZSTAR	NONDI 14
C SEARCH FOR MAX DIAMETER	NONDI 15
	DMAX=0.0	NONDI 16
	DO 2 I=2,IEND	NONDI 17
	DMAXN=AMAX1(RZ(I-1),RZ(I))	NONDI 18
	IF(DMAXN.LE.DMAX) GO TO 2	NONDI 19
	DMAX=DMAXN	NONDI 20
2	CONTINUE	NONDI 21
	DMAX=2.0*DMAX	NONDI 22
	RW=DMAX/2.0	NONDI 23
	F=L/DMAX	NONDI 24
	S=PI*DMAX**2/4.0	NONDI 25
	IF((RZERO(L)-.801).GT.0.0) GO TO 4	NONDI 26
	AW=0.0	NONDI 27
	DO 3 I=1,IEND	NONDI 28
	TRAP=DZSTAR/2.0*(RZ(I)+RZ(I+1))	NONDI 29
3	AW=AW+TRAP	NONDI 30
	AW=(1.0/L)*AW	NONDI 31
	GO TO 5	NONDI 32
4	AW=DMAX/2.0	NONDI 33
5	CONTINUE	NONDI 34
	RETURN	NONDI 35
	END	NONDI 36

	FUNCTION PDRAG(THETA)	PDRAG 1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RMAXSQ,SIGMA,LEVEL	PDRAG 2
	COMMON/BLOCK7/PHIVT,PHIPT,P1,P2,P3,P4	PDRAG 3
	DEG=THETA*180.0/PI	PDRAG 4
	CPK=CPC(THETA)	PDRAG 5
	PDRAG=CPK*COS(THETA)	PDRAG 6
	WRITE(6,50)DEG,PHIVT,PHIPT,P1,P2,CPK,PDRAG,P4	PDRAG 7
50	FORMAT(1X,F7.2,7(1X,F12.6))	PDRAG 8
	RETURN	PDRAG 9
	ENTRY PLIFT	PDRAG 10
	CPK=CPC(THETA)	PDRAG 11
	PLIFT=CPK*SIN(THETA)	PDRAG 12
	RETURN	PDRAG 13
	END	PDRAG 14

	SUBROUTINE POTFLOW(PSIKXP,PSIKYP,PSIKRP,IA)	POTFLO 1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,KC,RCHXSQ,SIGMA,LEVEL	POTFLO 2
	COMMON/BLOCK1/X(100),Y(100),X3(100),YB(100),XRT(100),YRT(100),	POTFLO 3
	IXRB(100),YRB(100)	POTFLO 4
	COMMON/BLOCK2/GAMMA(100),GMAB(100),GMRT(100),GMRB(100)	POTFLO 5
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDGT	POTFLO 6
	COMMON/BLOCK10/THETAS,THETASB,THETA	POTFLO 7
	COMMON/BLOCK20/DX,DZ,INTX1,NBIG1	POTFLO 8
	INTEGER ALPHA,A	POTFLO 9
	REAL L,LB	POTFL 10
	GO TO(1,2,3,4,5) IA	POTFL 11
1	L=X(ALPHA)**2+Y(ALPHA)**2	POTFL 12
	PSIKXP=2.0*X(ALPHA)*Y(ALPHA)/L**2	POTFL 13
	PSIKYP=AKSQD*PSIKXP	POTFL 14
	PSIKRP=1.0+AKSQD*(Y(ALPHA)**2-X(ALPHA)**2)/L**2	POTFL 15
	RETURN	POTFL 16
2	LB=XB(ALPHA)**2+YB(ALPHA)**2	POTFL 17
	PSIKXP=2.0*XB(ALPHA)*YB(ALPHA)/LB**2	POTFL 18
	PSIKYP=AKSQD*PSIKXP	POTFL 19
	PSIKRP=1.0+AKSQD*(YB(ALPHA)**2-XB(ALPHA)**2)/LB**2	POTFL 20
	RETURN	POTFL 21
3	PSIKRP=2.0*SIN(THETA)	POTFL 22
	RETURN	POTFL 23
4	L=XRT(ALPHA)**2+YRT(ALPHA)**2	POTFL 24
	IF(L.EQ.0.0) RETURN	POTFL 25
	PSIKXP=2.0*XRT(ALPHA)*YRT(ALPHA)/L**2	POTFL 26
	PSIKYP=AKSQD*PSIKXP	POTFL 27
	PSIKRP=1.0+AKSQD*(YRT(ALPHA)**2-XRT(ALPHA)**2)/L**2	POTFL 28
	RETURN	POTFL 29
5	LB=XR(B(ALPHA)**2+YRB(ALPHA)**2	POTFL 30
	IF(LB.EQ.0.0) RETURN	POTFL 31
	PSIKXP=2.0*XR(B(ALPHA)*YRB(ALPHA)/LB**2	POTFL 32
	PSIKYP=AKSQD*PSIKXP	POTFL 33
	PSIKRP=1.0+AKSQD*(YRB(ALPHA)**2-XRB(ALPHA)**2)/LB**2	POTFL 34
	RETURN	POTFL 35
	END	POTFL 36

	SUBROUTINE PSI(X,Y,GMA,NDIM,KI,KF,IA,IB,SUM,SUM1,TK)	PSI	1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL	PSI	2
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDGT	PSI	3
	COMMON/BLOCK30/T,TT,DELT,DELTT,DELTB	PSI	4
	COMMON/VTX1/XX,YY	PSI	5
	DIMENSION X(NDIM),Y(NDIM),GMA(NDIM),TK(NDIM)	PSI	6
	REAL L	PSI	7
	INTEGER ALPHA,A	PSI	8
	SUM=SUM1=0.0	PSI	9
	DO 1 A=KI,KF	PSI	10
	IF(ALPHA.EQ.A.AND.IA.EQ.IB) GO TO 1	PSI	11
	IFLAG=0	PSI	12
	L=X(A)**2+Y(A)**2	PSI	13
	R1=(XX-X(A))**2+(YY-Y(A))**2	PSI	14
	R2=(XX-X(A)*AKSQD/L)**2+(YY-Y(A)*AKSQD/L)**2	PSI	15
	ARE=XX**2+YY**2	PSI	16
	IF(R1.GT.RCMAXSQ) GO TO 2	PSI	17
	RCSQ=5.04*(T-TK(A))/RE	PSI	18
	IF(R1.GT.RCSQ) GO TO 2	PSI	19
C POTENTIAL VORTEX APPROX INVALID	PSI	20
	IFLAG=1	PSI	21
	GMRSET=GMA(A)	PSI	22
	GMA(A)=0.0	PSI	23
2	CONTINUE	PSI	24
	SUM=SUM+(GMA(A)/PI2)*((XX-X(A))/R1+XX/ARE-(XX-X(A)*AKSQD/L)/R2)	PSI	25
	SUM1=SUM1+(GMA(A)/PI2)*((YY-Y(A))/R1+YY/ARE-(YY-Y(A)*AKSQD/L)/R2)	PSI	26
	IF(IFLAG.EQ.1)GMA(A)=GMRSET	PSI	27
1	CONTINUE	PSI	28
	RETURN	PSI	29
	ENTRY PSIONE	PSI	30
	SUM=0.0	PSI	31
	DO 3 A=KI,KF	PSI	32
	IF(X(A).EQ.0.0.AND.Y(A).EQ.0.0) GO TO 3	PSI	33
	IFLAG=0	PSI	34
	L=X(A)**2+Y(A)**2	PSI	35
	R1=(XX-X(A))**2+(YY-Y(A))**2	PSI	36
	RCSQ=RC*RC	PSI	37
	IF(R1.GE.RCSQ) GO TO 4	PSI	38
C POTENTIAL VORTEX APPROX INVALID	PSI	39
	IFLAG=1	PSI	40
	GMRSET=GMA(A)	PSI	41
	GMA(A)=0.0	PSI	42
4	CONTINUE	PSI	43
	SUM=SUM+(GMA(A)/(PI2*AK))*((AKSQD-L)/R1+1.0)	PSI	44
	IF(IFLAG.EQ.1)GMA(A)=GMRSET	PSI	45
3	CONTINUE	PSI	46
	RETURN	PSI	47
	END	PSI	48

	SUBROUTINE PV(X,Y,XDT,YDT,GMA,NDIM,KI,KF,SUM,TK)	PV	1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RMAXSQ,SIGMA,LEVEL	PV	2
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	PV	3
	COMMON/PRSURE/XX,YY,SINE,COSINE	PV	4
	COMMON/BLOCK30/T, TI, DELT, DELTT, DELTB	PV	5
	DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM),GMA(NDIM)	PV	6
	DIMENSION TK(NDIM)	PV	7
	REAL L	PV	8
	INTEGER A	PV	9
	SUM=0.0	PV	10
	DO 1 A=KI,KF	PV	11
	L=X(A)**2+Y(A)**2	PV	12
	IFLAG=0	PV	13
	CAPA=(X(A)*COSINE+Y(A)*SINE)	PV	14
	R1=AKSQD+L-2.0*AK*CAPA	PV	15
	R2=(AKSQD*L+AKSQD**2-2.0*AK**3*CAPA)/L	PV	16
	RCSQ=RC*RC	PV	17
	IF(R1.GE.RCSQ) GO TO 34	PV	18
	SQR1=SQRT(R1)	PV	19
	WRITE(6,50)A,SQR1	PV	20
50	FORMAT(* POTENTIAL VORTEX APPROX INVALID ALPHA=*I3,5X,	PV	21
	I*CORE RADIUS=*F12.6)	PV	22
C POTENTIAL VORTEX APPROX INVALID	PV	23
	IFLAG=1	PV	24
	GMRSET=GMA(A)	PV	25
	GMA(A)=0.0	PV	26
34	CONTINUE	PV	27
	CAPX=(AKSQD*XDT(A)+2.0*AK*X(A)*AKDOT)/L	PV	28
	I-(2.0*AKSQD*X(A)/L**2)*(X(A)*XDT(A)+Y(A)*YDT(A))	PV	29
	CAPY=(AKSQD*YDT(A)+2.0*AK*Y(A)*AKDOT)/L	PV	30
	I-(2.0*AKSQD*Y(A)/L**2)*(X(A)*XDT(A)+Y(A)*YDT(A))	PV	31
	SUM=SUM+(GMA(A)/PI2)*((YDT(A)*(XX-X(A))-XDT(A)*(YY-Y(A)))/R1	PV	32
	I-(CAPY*(XX-AKSDQ*X(A)/L)-CAPX*(YY-AKSDQ*Y(A)/L))/R2)	PV	33
	IF(IFLAG.EQ.1) GMA(A)=GMRSET	PV	34
1	CONTINUE	PV	35
	RETURN	PV	36
	END	PV	37

SUBROUTINE RITE(MT, IDIM, JDIM, U, ZN, NX, S, DEG, ZHAT, DELS, K)	RITE	1
DIMENSION U(IDIM, 2, JDIM)	RITE	2
DIMENSION FOR601(4), FOR560(3), FOR561(3), FOR580(2)	RITE	3
DIMENSION NX(53), DEG(53)	RITE	4
DIMENSION S(53), ZN(53)	RITE	5
COMMON/BLOCK20/DX, DZ, INTX1, NBIG1	RITE	6
COMMON/BLOCK30/T, TI, DELT, DELTT, DELTB	RITE	7
COMMON/BLOCK5/AK, AKSQD, AKHALF, AKDOT	RITE	8
COMMON/BLBOX2/TAU, PT4, NBIG	RITE	9
MI=INTX1/12	RITE	10
I=1	RITE	11
M=1	RITE	12
IF(MT.EQ.0) GO TO 35	RITE	13
MM=12	RITE	14
IF(MT.EQ.1) WRITE(6, 549) TI	RITE	15
IF(MT.EQ.2) WRITE(6, 550) K, T, ZHAT, AK, AKDOT	RITE	16
WRITE(6, 600) (NX(I), I=1, 12)	RITE	17
GO TO 32	RITE	18
31 WRITE(6, 601) (NX(I), I=M, MM)	RITE	19
32 WRITE(6, 560) (S(I), I=M, MM)	RITE	20
WRITE(6, 561) (DEG(I), I=M, MM)	RITE	21
WRITE(6, 580) (ZN(J), (U(I, MT, J), I=M, MM), J=1, NBIG)	RITE	22
I=I+1	RITE	23
M=MM+1	RITE	24
MM=M+11	RITE	25
IF(I.LE.MI) GO TO 31	RITE	26
IF(M-1.EQ.INTX1) GO TO 33	RITE	27
35 MM=INTX1	RITE	28
N=(MM-M)+1	RITE	29
NN=N+1	RITE	30
ENCODE(33, 1000, FOR601) N	RITE	31
ENCODE(29, 1004, FOR560) N	RITE	32
ENCODE(30, 1005, FOR561) N	RITE	33
ENCODE(16, 1002, FOR580) NN	RITE	34
IF(MT.EQ.0) WRITE(6, 550) K, T, ZHAT, DELT, DELS, DZ	RITE	35
WRITE(6, FOR601) (NX(I), I=M, MM)	RITE	36
WRITE(6, FOR560) (S(I), I=M, MM)	RITE	37
WRITE(6, FOR561) (DEG(I), I=M, MM)	RITE	38
WRITE(6, FOR580) (ZN(J), (U(I, MT, J), I=M, MM), J=1, NBIG)	RITE	39
33 CONTINUE	RITE	40
RETURN	RITE	41
549 FORMAT(1H1, 40X*BOUNDARY LAYER VELOCITY DISTRIBUTION (UI) TI=* 1F5.3, /)	RITE	42
550 FORMAT(1H1, 50X*BOUNDARY LAYER VELOCITY DISTRIBUTION*/30X* K=*I2, 5XRITE 1*T=*F6.3, 5X*ZHAT=*F6.3, 5X*AK=*F6.3, 5X*AKDOT=*F6.3)	RITE	43
560 FORMAT(* R+ (RAD)*, 1X, 12(F9.5, 1X))	RITE	44
561 FORMAT(* (DEG)*, 1X, 12(F9.5, 1X)/)	RITE	45
580 FORMAT(1X, 13(F9.5, 1X))	RITE	46
600 FORMAT(/* NCYCLE *, 12(4X, I2, 4X)/)	RITE	47
601 FORMAT(///* NCYCLE *, 12(4X, I2, 4X)/)	RITE	48
1000 FORMAT(18H(///* NCYCLE *, I2, 13H(4X, I2.4X)/))	RITE	49
1002 FORMAT(4H(1X, I2, 10H(F9.5, 1X)))	RITE	50
1004 FORMAT(17H(* R+ (RAD)*, 1X, I2, 10H(F9.5, 1X)))	RITE	51
1005 FORMAT(17H(* (DEG)*, 1X, I2, 11H(F9.5, 1X)/)	RITE	52
END	RITE	53
	RITE	54
	RITE	55

	SUBROUTINE RSEP(THTASF, THTASR, THTA, UBL, INTX1, INITAL, MODE)	RSEP	1
	COMMON ALPHA, PI, PI2, RE, SRE, SQRTPI, DTOR, RTOD, RC, RCMAXSQ, SIGMA, LEVEL	RSEP	2
	DIMENSION THTA(53), UBL(53)	RSEP	3
	UFMX=0.0	RSEP	4
C SEARCH B.L. FOR MAX VELOCITY	RSEP	5
	DO 1 I=2, INTX1	RSEP	6
	UFMXN=AMAX1(UBL(I-1), UBL(I))	RSEP	7
	IF(UFMXN.LE.UFMX) GO TO 1	RSEP	8
	IMAX=I-1	RSEP	9
	IF(UFMXN.GT.UBL(I-1)) IMAX=I	RSEP	10
	UFMX=UFMXN	RSEP	11
1	CONTINUE	RSEP	12
	THTAMXF=THTA(IMAX)	RSEP	13
	URMX=0.0	RSEP	14
C SEARCH S.L. FOR MAX BACKFLOW VELOCITY	RSEP	15
	ISLS=INTX1+2	RSEP	16
	DO 2 I=ISLS, INITAL	RSEP	17
	URMXN=AMIN1(UBL(I-1), UBL(I))	RSEP	18
	IF(URMXN.GE.URMX) GO TO 2	RSEP	19
	JMAX=I-1	RSEP	20
	IF(URMXN.LT.UBL(I-1)) JMAX=I	RSEP	21
	URMX=URMXN	RSEP	22
2	CONTINUE	RSEP	23
	THTAMXR=THTA(JMAX)	RSEP	24
C SEARCH S.L. FOR ZERO VELOCITY	RSEP	25
	ISLE=INIVAL-1	RSEP	26
	DO 4 I=ISLS, ISLE	RSEP	27
	UZN=AMIN1(ABS(UBL(I-1)), ABS(UBL(I)))	RSEP	28
	IF(I.EQ.ISLS) GO TO 3	RSEP	29
	IF(UZN.GE.UZ) GO TO 4	RSEP	30
3	KMAX=I-1	RSEP	31
	IF(UZN.LT.ABS(UBL(I-1))) KMAX=I	RSEP	32
	UZ=UZN	RSEP	33
4	CONTINUE	RSEP	34
	THTAZRO=THTA(KMAX)	RSEP	35
C CALCULATE S.L. SEPARATION POINT	RSEP	36
	X=(THTAZRO-THTAMXF)/(THTASF-THTAMXF)	RSEP	37
	THTASR=THTAMXR-(THTAMXR-THTAZRO)/X	RSEP	38
	THTAMXF=THTAMXF*RTOD	RSEP	39
	THTAMXR=THTAMXR*RTOD	RSEP	40
	THTAZRO=THTAZRO*RTOD	RSEP	41
	THTASRD=THTASR*RTOD	RSEP	42
	WRITE(6, 50) UFMX, THTAMXF, URMX, THTAMXR, UZ, THTAZRO, THTASRD	RSEP	43
50	FORMAT(1H1, * MAX VELOCITY=*F6.3, * AT THETA=*F5.1, * DEGREES*/	RSEP	44
	1* MAX BACKFLOW VEL=*F6.3, * AT THETA=*F5.1, * DEGREES*/	RSEP	45
	2* MIN TANGENTIAL VELOCITY BETWEEN MAX VELOCITY AND MAX BACKFLOW VERSEP	RSEP	46
	3LOCITY=*F6.3, * AT IHETA=*F5.1, * DEGREES*/ * REAR SEPARATION ANGLE=*RSEP	RSEP	47
	4F5.3, * DEGREES*)	RSEP	48
	RETURN	RSEP	49
	END	RSEP	50

	SUBROUTINE RVTX(X,Y,GMA,THTASR,THTASM,THTAS,UZZ,UZZSQ,GAMA,	RVTX	1
1	MODE,KRN,N,NF,IFLAG,GMATL,NDIM,TK)	RVTX	2
	DIMENSION X(NDIM),Y(NDIM),GMA(NDIM),TK(NDIM)	RVTX	3
	DIMENSION THTAS(6),UZZ(6),UZZSQ(6),GAMA(6),IFLAG(6)	RVTX	4
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSQ,SIGMA,LEVEL	RVTX	5
	COMMON/KAYS/K,KS,KR,KT,KB,KRT,KRB	RVTX	6
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	RVTX	7
	COMMON/BLOCK10/THTAS,THTASB,THETA	RVTX	8
	COMMON/BLOCK30/T,TI,DELT,DELTT,DELTB	RVTX	9
	IF(K.NE.KR) GO TO 1	RVTX	10
	NF=5	RVTX	11
	N=N	RVTX	12
	GMATL=0.	RVTX	13
1	N=N+1	RVTX	14
	THTAS(N)=THTASR	RVTX	15
	THETA=THTAS(N)	RVTX	16
	CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,3)	RVTX	17
	CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,3)	RVTX	18
	UZZ(N)=PSIK1R+PSIKRF	RVTX	19
	THTAP=THTAS(N)*RTOD	RVTX	20
	WRITE(6,600)N,THTAP,UZZ(N)	RVTX	21
600	FORMAT(77* RVTX SUBROUTINE N=*I3,2X*THETAS(N)=*F12.6,	RVTX	22
	12X*U00(N)=*F12.6)	RVTX	23
	UZZ(N)=ABS(UZZ(N))	RVTX	24
	UZZSQ(N)=UZZ(N)*UZZ(N)	RVTX	25
	GAMA(N)=-DELT*UZZSQ(N)/2.0	RVTX	26
	IF(MODE.EQ.2)GAMA(N)=-GAMA(N)	RVTX	27
	WRITE(6,601)GAMA(N)	RVTX	28
601	FORMAT(* GAMA(N)=*F12.6)	RVTX	29
	IF(N.EQ.NF+1) GO TO 10	RVTX	30
	IFLAG(N-1)=0	RVTX	31
	IF(THTAS(N)-THTAS(N-1))3,4,2	RVTX	32
2	IF(MODE.EQ.1) IFLAG(N-1)=1	RVTX	33
	GO TO 5	RVTX	34
3	IF(MODE.EQ.2) IFLAG(N-1)=1	RVTX	35
	GO TO 5	RVTX	36
4	IFLAG(N-1)=1	RVTX	37
5	CONTINUE	RVTX	38
	IF(IFLAG(N-1).EQ.1) GMATL=GMATL+GAMA(N-1)	RVTX	39
	WRITE(6,602)GMATL,N,IFLAG(N-1)	RVTX	40
602	FORMAT(/* GAMA CHECK SUM=*F12.6,5X*N=*I3,5X*IFLAG(N-1)=*I3)	RVTX	41
	AGMATL=ABS(GMATL)	RVTX	42
	IF(AGMATL.LE..1) GO TO 6	RVTX	43
	IF(GMATL.LE..1) GO TO 6	RVTX	44
	NF=N	RVTX	45
	GO TO 7	RVTX	46
6	IF(N.LT.5) RETURN	RVTX	47
7	THTASM=THTAS(N)	RVTX	48
	GMATL=0.	RVTX	49
	RETURN	RVTX	50
10	IF(K.EQ.KR) GO TO 2000	RVTX	51
	IFLAG(NF)=0	RVTX	52
	IF(THTAS(N)-THTASM)30,40,20	RVTX	53
20	IF(MODE.EQ.1) IFLAG(NF)=1	RVTX	54
	GO TO 50	RVTX	55
30	IF(MODE.EQ.2) IFLAG(NF)=1	RVTX	56
	GO TO 50	RVTX	57
40	IFLAG(NF)=1	RVTX	58

50	CONTINUE	RVTX 59
	THTAP=THTAS(N)*RTOD	RVTX 60
	THTAPM=THTASM*RTOD	RVTX 61
	WRITE(6,603)N,NF,IFLAG(NF),THTAP,THTAPM	RVTX 62
603	FORMAT(/* OVERLAP CHECK N=*I3,5X*NF=*I3,5X*IFLAG(NF)=*I3,	RVTX 63
	15X*THTAS(N)=*F12.6,5X*THTASM=*F12.6)	RVTX 64
	U00=0.0	RVTX 65
	U00SQ=0.0	RVTX 66
	GAM=0.	RVTX 67
	THTASR=0.	RVTX 68
	RN=0.	RVTX 69
	N=1	RVTX 70
100	IF(IFLAG(N).EQ.0) GO TO 200	RVTX 71
	RN=RN+1.0	RVTX 72
	U00=U00+UZZ(N)	RVTX 73
	U00SQ=U00SQ+UZZSQ(N)	RVTX 74
	GAM=GAM+GAMA(N)	RVTX 75
	THTASR=THTASR+THTAS(N)	RVTX 76
	WRITE(6,604)N,U00,U00SQ,GAM,THTASR	RVTX 77
604	FORMAT(/* LUMP SUMS N=*I3,2X*U00=*F12.6,2X*U00SQ=*F12.6,	RVTX 78
	12X*GAM=*F12.6,2X*THTASR=*F12.6)	RVTX 79
200	IF(N.EQ.NF) GO TO 1000	RVTX 80
	N=N+1	RVTX 81
	GO TO 100	RVTX 82
1000	CONTINUE	RVTX 83
	SMALLM=(RN*DELT/PI2)*(U00SQ/U00)	RVTX 84
	THTASR=THTASR/RN	RVTX 85
	KRN=KRN+1	RVTX 86
	X(KRN)=(AK+SMALLM)*COS(THTASR)	RVTX 87
	Y(KRN)=(AK+SMALLM)*SIN(THTASR)	RVTX 88
	GMA(KRN)=GAM	RVTX 89
	GMA(KRN)=SIGMA*GMA(KRN)	RVTX 90
	THTAP=THTASR*RTOD	RVTX 91
	TK(KRN)=T	RVTX 92
	WRITE(6,605)SMALLM,THTAP,KRN,X(KRN),Y(KRN),GMA(KRN)	RVTX 93
605	FORMAT(/** RVTX BIRTH SMALLM=*F12.6,5X*THTASR(AVERAGE)=*F12.6, /	RVTX 94
	15X*KRN=*I3,2X*X(KRN)=*F12.6,2X*Y(KRN)=*F12.6,2X*GMA(KRN)=*F12.6)	RVTX 95
2000	UZZ(1)=UZZ(NF+1)	RVTX 96
	UZZSQ(1)=UZZSQ(NF+1)	RVTX 97
	GAMA(1)=GAMA(NF+1)	RVTX 98
	THTAS(1)=THTAS(NF+1)	RVTX 99
	N=1	RVTX 100
	NF=5	RVTX 101
	RETURN	RVTX 102
	END	RVTX 103

```
FUNCTION RZRO(ZHAT,L,RW)
REAL L
RZRO=RZERO(ZHAT*L)/RW
RETURN
ENTRY DRZRO
RZRO=(L/RW)*DRZERO(ZHAT*L)
RETURN
END
```

```
RZRO 1
RZRO 2
RZRO 3
RZRO 4
RZRO 5
RZRO 6
RZRO 7
RZRO 8
```

FUNCTION UAPRX1(L,N,IDIM,JDIM,U)	UAPRX1 1
DIMENSION U(IDIM,2,JDIM)	UAPRX1 2
*** *****FIRST APPROXIMATION TO U(L+1,2,N)	UAPRX1 3
UAPRX1=U(L+1,1,N)+U(L,2,N)-U(L,1,N)	UAPRX1 4
RETURN	UAPRX1 5
ENTRY US	UAPRX1 6
US=(U(L+1,1,N)+U(L,2,N)+U(L,1,N))/4.0	UAPRX1 7
RETURN	UAPRX1 8
ENTRY UL	UAPRX1 9
UL=U(L+1,1,N)-U(L,2,N)-U(L,1,N)	UAPRX 10
RETURN	UAPRX 11
ENTRY UM	UAPRX 12
UM=U(L,2,N)-U(L+1,1,N)-U(L,1,N)	UAPRX 13
RETURN	UAPRX 14
END	UAPRX 15

SUBROUTINE VEL(X,Y,XDT,YDT,NDIM,KI,KF,IA)	VEL	1
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCHMAXSQ,SIGMA,LEVEL	VEL	2
COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	VEL	3
DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM)	VEL	4
INTEGER ALPHA	VEL	5
REAL L	VEL	6
DO 1 ALPHA=KI,KF	VEL	7
L=X(ALPHA)**2+Y(ALPHA)**2	VEL	8
CALL FREEVTX(PSIK1X,PSIK1Y,PSIK1R,IA)	VEL	9
CALL POTFLOW(PSIKXP,PSIKYP,PSIKRP,IA)	VEL	10
PSIKX=PSIKXP+PSIK1X+AK*AKDOT*Y(ALPHA)/L	VEL	11
PSIKY=PSIKYP+PSIK1Y-AK*AKDOT*X(ALPHA)/L	VEL	12
XDT(ALPHA)=-PSIKY	VEL	13
YDT(ALPHA)=PSIKX	VEL	14
CONTINUE	VEL	15
1 RETURN	VEL	16
END	VEL	17

	SUBROUTINE VM(X,Y,XDT,YDT,NDIM,KI,KF,DT)	VM	1
	COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RMAXSQ,SIGMA,LEVEL	VM	2
	COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT	VM	3
	DIMENSION X(NDIM),Y(NDIM),XDT(NDIM),YDT(NDIM)	VM	4
	INTEGER ALPHA	VM	5
	DO 1 ALPHA=KI,KF	VM	6
	XTEMP=X(ALPHA)	VM	7
	YTEMP=Y(ALPHA)	VM	8
	X(ALPHA)=X(ALPHA)+DT*XDT(ALPHA)	VM	9
	Y(ALPHA)=Y(ALPHA)+DT*YDT(ALPHA)	VM	10
	R=SQRT(X(ALPHA)**2+Y(ALPHA)**2)	VM	11
	IF(R.GT.AK) GO TO 1	VM	12
	WRITE(6,600)X(ALPHA),Y(ALPHA)	VM	13
	CALL VMFIX(XTEMP,YTEMP,XDT(ALPHA),YDT(ALPHA))	VM	14
	X(ALPHA)=XTEMP+DT*XDT(ALPHA)	VM	15
	Y(ALPHA)=YTEMP+DT*YDT(ALPHA)	VM	16
	WRITE(6,601)X(ALPHA),Y(ALPHA)	VM	17
1	CONTINUE	VM	18
600	FORMAT(//* BEFORE VMFIX(X(ALPHA),Y(ALPHA))=(*F8.5,*,*F8.5,*)*)	VM	19
601	FORMAT(//* AFTER VMFIX(X(ALPHA),Y(ALPHA))=(*F8.5,*,*F8.5,*)*)	VM	20
	RETURN	VM	21
	END	VM	22

```

SUBROUTINE VMFIX(X,Y,XDOT,YDOT) VMFIX 1
COMMON ALPHA,PI,PI2,RE,SRE,SQRTPI,DTOR,RTOD,RC,RCMAXSO,SIGMA,LEVEL VMFIX 2
COMMON/BLOCK5/AK,AKSQD,AKHALF,AKDOT VMFIX 3
COMMON/BLOCK10/THETAS,THETASB,THETA VMFIX 4
DEF=.001 VMFIX 5
IF(X.NE.0) GO TO 10 VMFIX 6
L=1 VMFIX 7
ANGLE=PI/2.0 VMFIX 8
IF(Y.LT.0)L=2 VMFIX 9
IF(Y.LT.0) ANGLE=(3.0/2.0)*PI VMFIX 10
GO TO 30 VMFIX 11
10 IF(Y.NE.0) GO TO 20 VMFIX 12
L=1 VMFIX 13
ANGLE=0.0 VMFIX 14
IF(X.LT.0)L=2 VMFIX 15
IF(X.LT.0) ANGLE=PI VMFIX 16
GO TO 30 VMFIX 17
20 IF(X.GT.0.0.AND.Y.GT.0.0)L=1 VMFIX 18
IF(X.LT.0.0.AND.Y.GT.0.0)L=2 VMFIX 19
IF(X.LT.0.0.AND.Y.LT.0.0)L=3 VMFIX 20
IF(X.GT.0.0.AND.Y.LT.0.0)L=4 VMFIX 21
ANGLE=ATAN(Y/X) VMFIX 22
30 CONTINUE VMFIX 23
SINE=SIN(ANGLE) VMFIX 24
COSINE=COS(ANGLE) VMFIX 25
GO TO(1,2,3,4),L VMFIX 26
1 X=(AK+DEF)*COSINE VMFIX 27
Y=(AK+DEF)*SINE VMFIX 28
XDOT=XDOT*SINE VMFIX 29
YDOT=YDOT*COSINE VMFIX 30
U=-XDOT+YDOT VMFIX 31
XDOT=-U*SINE VMFIX 32
YDOT=U*COSINE VMFIX 33
GO TO 5 VMFIX 34
2 X=- (AK+DEF)*COSINE VMFIX 35
Y=- (AK+DEF)*SINE VMFIX 36
XDOT=-XDOT*SINE VMFIX 37
YDOT=YDOT*COSINE VMFIX 38
U=-XDOT-YDOT VMFIX 39
XDOT=U*SINE VMFIX 40
YDOT=-U*COSINE VMFIX 41
GO TO 5 VMFIX 42
3 X=- (AK+DEF)*COSINE VMFIX 43
Y=- (AK+DEF)*SINE VMFIX 44
XDOT=XDOT*SINE VMFIX 45
YDOT=YDOT*COSINE VMFIX 46
U=XDOT-YDOT VMFIX 47
XDOT=U*SINE VMFIX 48
YDOT=-U*COSINE VMFIX 49
GO TO 5 VMFIX 50
4 X=(AK+DEF)*COSINE VMFIX 51
Y=(AK+DEF)*SINE VMFIX 52
XDOT=-XDOT*SINE VMFIX 53
YDOT=YDOT*COSINE VMFIX 54
U=XDOT+YDOT VMFIX 55
XDOT=-U*SINE VMFIX 56
YDOT=U*COSINE VMFIX 57
GO TO 5 VMFIX 58

```

```
5 CONTINUE
  RTOD=180.0/PI
  ADEG=ANGLE*RTOD
  WRITE(6,50)ALPHA,ADEG,X,Y,XDOT,YDOT
50  FORMAT (// * VMFIX*2X*ALPHA=*I3,2X*ANGLE=*F12.6,/* X=*F12.6,
  12X*Y=*F12.6,2X*XDOT=*F12.6,2X*YDOT=*F12.6)
  RETURN
  END
```

```
VMFIX 59
VMFIX 60
VMFIX 61
VMFIX 62
VMFIX 63
VMFIX 64
VMFIX 65
VMFIX 66
```

	SUBROUTINE WRIT(X,Y,XDOT,YDOT,GAMMA,XB,YB,XDOTB,YDOTB,GMAB,	WRIT	1
	1KIT,KFT,KIB,KFB,K,NDIM,IW)	WRIT	2
	DIMENSION X(NDIM),Y(NDIM),XDOT(NDIM),YDOT(NDIM),GAMMA(NDIM),	WRIT	3
	1XB(NDIM),YB(NDIM),XDOTB(NDIM),YDOTB(NDIM),GMAB(NDIM)	WRIT	4
	IF(KFT-KFB) 1,2,2	WRIT	5
1	KF=KFB	WRIT	6
	GO TO 3	WRIT	7
2	KF=KFT	WRIT	8
3	CONTINUE	WRIT	9
	IF(KIT.GT.KIB) GO TO 17	WRIT	10
	DO 16 I=KIT,KF	WRIT	11
	IF(I.GE.KSB) GO TO 12	WRIT	12
11	WRITE(6,601) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I)	WRIT	13
	IF(IW.NE.7.AND.IW.NE.9) GO TO 16	WRIT	14
	WRITE(IW,701) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I)	WRIT	15
	GO TO 16	WRIT	16
12	IF(KFT.NE.KFB) GO TO 14	WRIT	17
13	WRITE(6,603) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),	WRIT	18
	1XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)	WRIT	19
	IF(IW.NE.7.AND.IW.NE.9) GO TO 16	WRIT	20
	WRITE(IW,703) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),XB(I),YB(I),	WRIT	21
	1XDOTB(I),YDOTB(I),GMAB(I)	WRIT	22
	GO TO 16	WRIT	23
14	IF(KFT.LT.KFB) GO TO 15	WRIT	24
	IF(I.LE.KFB) GO TO 13	WRIT	25
	GO TO 11	WRIT	26
15	IF(I.LE.KFT) GO TO 13	WRIT	27
16	CONTINUE	WRIT	28
	GO TO 20	WRIT	29
17	CONTINUE	WRIT	30
	DO 19 I=KIB,KF	WRIT	31
	IF(I.GE.KS) GO TO 18	WRIT	32
	WRITE(6,602) I,K,XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)	WRIT	33
	IF(IW.NE.7.AND.IW.NE.9) GO TO 19	WRIT	34
	WRITE(IW,702) I,K,XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)	WRIT	35
	GO TO 19	WRIT	36
18	WRITE(6,603) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),	WRIT	37
	1XB(I),YB(I),XDOTB(I),YDOTB(I),GMAB(I)	WRIT	38
	IF(IW.NE.7.AND.IW.NE.9) GO TO 19	WRIT	39
	WRITE(IW,703) I,K,X(I),Y(I),XDOT(I),YDOT(I),GAMMA(I),XB(I),YB(I),	WRIT	40
	1XDOTB(I),YDOTB(I),GMAB(I)	WRIT	41
19	CONTINUE	WRIT	42
20	CONTINUE	WRIT	43
601	FORMAT(2X13,2X13,5(2X,F10.6))	WRIT	44
602	FORMAT(2X13,2X13,60X,5(2X,F10.6))	WRIT	45
603	FORMAT(2X13,2X13,10(2X,F10.6))	WRIT	46
701	FORMAT(2I4,5F7.3)	WRIT	47
702	FORMAT(2I4,35X,4F7.3)	WRIT	48
703	FORMAT(2I4,10F7.3)	WRIT	49
	RETURN	WRIT	50
	END	WRIT	51

PROGRAM ADYNF (INPUT, OUTPUT, TAPES=INPUT, TAPE6=OUTPUT)	ADYNF	1
COMMON/INPUT/LX, XX(100), U(100), JADD, ADDXI(26), MODE	ADYNF	2
COMMON/OUTPUT/NEPDR(100), FCTI(100), XIL(28), COEFL(27,4),	ADYNF	3
* VORDL(28,2), KNOT, LMAX, INTERV	ADYNF	4
COMMON/OTHER/LXI, LXI1, LXI2, 0, CHANGE, ERROR, ACC, XI(28)	ADYNF	5
DIMENSION INFO(16), X(100), Z(100), PORCD(100), CDN(100)	ADYNF	6
DIMENSION Y(100)	ADYNF	7
REAL L	ADYNF	8
REAL LENGTH, LAMBDA	ADYNF	9
EXTERNAL FIT	ADYNF	10
C INFO DATA IDENTIFICATION	ADYNF	11
READ(5,605) (INFO(I), I=1,16)	ADYNF	12
605 FORMAT(16A5)	ADYNF	13
WRITE(6,651) (INFO(I), I=1,16)	ADYNF	14
651 FORMAT(1H1,20X,16A5 //)	ADYNF	15
READ(5,652) AATAACK, LAMBDA, LENGTH, F, AW, RW	ADYNF	16
652 FORMAT(6F12.6)	ADYNF	17
C READ NO. OF KNOTS, NO. OF POINTS, X CO-ORD, Y CO-ORD	ADYNF	18
READ(5,610) NOKNOT, LX, (X(I), PORCD(I), I=1, LX)	ADYNF	19
610 FORMAT(2I4,/(2F12.8))	ADYNF	20
C IF NOKNOT.GT.0 READ IN KNOT POSITIONS	ADYNF	21
LXI2=1ABS(NOKNOT)	ADYNF	22
IF(NOKNOT.GT.0) READ(5,607) (XI0J), J=1, LXI2)	ADYNF	23
607 FORMAT(6F12.6)	ADYNF	24
C ELLIPSOID PARAMETERS	ADYNF	25
PI=4.0*ATAN(1.0)	ADYNF	26
DTOR=PI/180.0	ADYNF	27
RTCD=180.0/PI	ADYNF	28
AATAACK=AATAACK*DTOR	ADYNF	29
FIA=F*TAN(AATAACK)	ADYNF	30
COEFN=4.0*F/PI	ADYNF	31
COEFM=-4.0*F/PI	ADYNF	32
COEF1=AW/RW*SIN(AATAACK)**2	ADYNF	33
C CALCULATE CDN, ZHAT	ADYNF	34
C GET SPLINE X CO ORD XX(I), Y CO ORD U(I)	ADYNF	35
DO 1 I=1, LX	ADYNF	36
Z(I)=X(I)	ADYNF	37
CDN(I)=COEF1*PORCD(I)	ADYNF	38
U(I)=CDN(I)	ADYNF	39
1 XX(I)=Z(I)	ADYNF	40
CALL SPLINEB(NOKNOT)	ADYNF	41
C TEMPORARY DEBUGGING DATA PLOT Y. VS. X	ADYNF	42
WRITE(6,62)	ADYNF	43
DO 3 I=1,21	ADYNF	44
X(I)=.05*(I-1)	ADYNF	45
Y(I)=FIT(X(I))	ADYNF	46
3 WRITE(6,61) X(I), Y(I)	ADYNF	47
61 FORMAT(F12.6,4X,F12.6)	ADYNF	48
62 FORMAT(1H1,6X*Z*13X*CDN(Z)*)	ADYNF	49
CNI=CAURE(FIT,0.0,1.0,.01,.01,3,ERR,IFLAG)	ADYNF	50
CN=COEFN*CNI	ADYNF	51
WRITE(6,60) CN, CNI, ERR, IFLAG	ADYNF	52
60 FORMAT(/////* CN=*F12.6,/* CNI=*F12.6,5X*ERROR=*F12.8,	ADYNF	53
15X*IFLAG=*12)	ADYNF	54
DO 2 I=1, LX	ADYNF	55
CDN(I)=(Z(I)*RZRO(Z(I),LENGTH,RW)*DRZRO(Z(I),LENGTH,RW)/	ADYNF	56
1(4.0*F**2))*CDN(I)	ADYNF	57
XX(I)=Z(I)	ADYNF	58

2	U(I)=CDN(I)	ADYNF 59
	CALL SPLINEB(NOKNOT)	ADYNF 60
C TEMP DEBUG DATA SPLINE Y VS X	ADYNF 61
	WRITE(6,63)	ADYNF 62
	DO 4 I=1,21	ADYNF 63
	X(I)=.05*(I-1)	ADYNF 64
	Y(I)=FIT(X(I))	ADYNF 65
4	WRITE(6,61)X(I),Y(I)	ADYNF 66
63	FORMAT(1H1,6X*2*13X*(,5-2)CDN*)	ADYNF 67
	CM=CADRE(FIT,0,0,1,0,.01,.01,3,ERR,IFLAG)	ADYNF 68
	CM=COEFM*CM+LAMBDA*CN	ADYNF 69
	WRITE(6,65)CM,CM,ERR,IFLAG	ADYNF 70
65	FORMAT(///// * CM=*F12.6,/* CM=*F12.6,5X*ERROR=*F12.8,	ADYNF 71
	15X*IFLAG=*I2)	ADYNF 72
	END	ADYNF 73

	FUNCTION FIT(X)	FIT	1
	COMMON/INPUT/LX,XX(100),U(100),JADD,ADDXI(26),MODE	FIT	2
	COMMON/OUTPUT/UERROR(100),FCIL(100),XIL(28),COEFL(27,4),	FIT	3
	VORDL(28,2),KNOT,LMAX,INTERV	FIT	4
	COMMON/OTHER/LXI,LXII,LXIZ,0,CHANGE,ERROR,ACC,XI(28)	FIT	5
	I=LXIZ	FIT	6
1	A=X-XI(I)	FIT	7
	IF(A)2,2,4	FIT	8
7	I=I-1	FIT	9
	IF(I)3,3,1	FIT	10
3	I=1	FIT	11
4	FIT=COEFL(I,1)+A*(COEFL(I,2)+A*(COEFL(I,3)+A*COEFL(I,4)))	FIT	12
	RETURN	FIT	13
	END	FIT	14

APPENDIX IV

SAMPLE CASE

SAMPLE
 FUNCTION RZERO
 (INPUT BODY GEOMETRY)

	FUNCTION RZERO(ZSTAR)	RZERO 1
	REAL LN	RZERO 2
	D=4.7	RZERO 3
	LN=14.1	RZERO 4
	R=43.475	RZERO 5
	IF(ZSTAR.GE.LN) GO TO 1	RZERO 6
	RZERO=SQRT(R**2-(ZSTAR-LN)**2)-(R-D/2.0)	RZERO 7
	GO TO 2	RZERO 8
1	RZERO=D/2.0	RZERO 9
2	CONTINUE	RZERO 10
	RETURN	RZERO 11
	ENTRY DRZERO	RZERO 12
	IF(ZSTAR.GE.LN) GO TO 3	RZERO 13
	RZERO=(LN-ZSTAR)/SQRT(R**2-(ZSTAR-LN)**2)	RZERO 14
	GO TO 4	RZERO 15
3	RZERO=0.0	RZERO 16
4	CONTINUE	RZERO 17
	RETURN	RZERO 18
	END	RZERO 19

VCF SAMPLE INPUT
CASE ... OGIVE CYLINDER ... ANGLE OF ATTACK (15 DEG)

	*****	VCF	SAMPLE	OUTPUT	*****	
15.0		4725600.0	50.47A			CARD 1
.125		.05	.6			CARD 2
39		90.0	1.0			CARD 3
3 4 5 5						CARD 4
						CARD 5

140

BODY GEOMETRY(DIMENSIONAL LENGTH= 50.47A)

ZSTAR	RZERO(ZSTAR)
0.0000	0.0000
-2.5239	.7805
5.0478	1.3972
7.5717	1.8571
10.0956	2.1652
12.6195	2.3248
15.1434	2.3500
17.6673	2.3500
20.1912	2.3500
22.7151	2.3500
-25.2390	-2.3500
27.7629	2.3500
30.2868	2.3500
-32.8107	-2.3500
35.3346	2.3500
37.8585	2.3500
-40.3824	-2.3500
42.9063	2.3500
45.4302	2.3500
-47.9541	-2.3500

~~RMAX=4.7000~~
 AW= 2.3500
 F=10.7400
~~S=17.3494~~

ANGLE-OF-ATTACK= 15.0 DEGREES
 3DS REYNOLDS NO.= 4725600.00
 2DLS REYNOLDS NO.= 56940.19

2DLS PARAMETERS

DELT= .125
 RC=.050
 SIGMA= .600

PROGRAM CONTRL

KFINAL= 39
 IFINAL= 90.0
 ZFINAL=1.000
 LR= 3
 LW= 4
 LEVEL= 5
 KPLN= 5

BOUNDARY LAYER VELOCITY DISTRIBUTION

K=39

T= 5.048

ZHAT= 2.686

AK= 1.000

AKDOT= 0.000

NCYCLE	0	3	3	3	3	3	3	4	4	4	4	4
R+ (RAD)	0.00000	.08727	.17453	.26180	.34907	.43633	.52360	.61087	.69813	.78540	.87266	.95993
(DEG)	0.00000	5.00000	10.00000	15.00000	20.00000	25.00000	30.00000	35.00000	40.00000	45.00000	50.00000	55.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
.14000	0.00000	.03785	.07501	.11081	.14459	.17570	.20353	.22752	.24710	.26176	.27101	.27436
.28000	0.00000	.06936	.13755	.20341	.26581	.32363	.37579	.42128	.45910	.48829	.50792	.51704
.42000	0.00000	.09493	.18847	.27901	.36514	.44543	.51848	.58293	.63745	.68074	.71147	.72627
.56000	0.00000	.11531	.22896	.33929	.44466	.54346	.63407	.71492	.78442	.84101	.88307	.90887
.70000	0.00000	.13105	.26037	.38621	.50684	.62055	.72562	.82033	.90296	.97175	1.02486	1.06030
.84000	0.00000	.14295	.28415	.42185	.55430	.67975	.79644	.90261	.99645	1.07609	1.13957	1.18472
.98000	0.00000	.15171	.30170	.44824	.58962	.72410	.84993	.96533	1.06850	1.15752	1.23037	1.28478
1.12000	0.00000	.15799	.31431	.46729	.61524	.75649	.88931	1.01199	1.12271	1.21960	1.30063	1.36351
1.26000	0.00000	.16237	.32313	.48066	.63333	.77953	.91758	1.04583	1.16251	1.26582	1.35375	1.42408
1.40000	0.00000	.16534	.32913	.48979	.64576	.79548	.93734	1.06974	1.19101	1.29938	1.39236	1.46961
1.54000	0.00000	.16730	.33309	.49585	.65406	.80621	.95078	1.08620	1.21088	1.32315	1.42122	1.50304
1.68000	0.00000	.16855	.33563	.49975	.65944	.81324	.95966	1.09721	1.22438	1.33397	1.44109	1.52732
1.82000	0.00000	.16933	.33720	.50218	.66283	.81770	.96536	1.10438	1.23330	1.35060	1.45469	1.54380
1.96000	0.00000	.16973	.33815	.50366	.66489	.82045	.96892	1.10892	1.23903	1.35782	1.46373	1.55526
2.10000	0.00000	.17006	.33870	.50452	.66511	.82209	.97107	1.11170	1.24261	1.36242	1.46970	1.56289
2.24000	0.00000	.17021	.33902	.50501	.66581	.82304	.97234	1.11336	1.24479	1.36527	1.47345	1.56735
2.38000	0.00000	.17029	.33918	.50528	.66620	.82357	.97305	1.11432	1.24607	1.36659	1.47576	1.57099
2.52000	0.00000	.17034	.33927	.50542	.66640	.82386	.97345	1.11486	1.24680	1.36699	1.47714	1.57292
2.66000	0.00000	.17036	.33932	.50549	.66651	.82401	.97366	1.11515	1.24721	1.36688	1.47795	1.57403
2.80000	0.00000	.17037	.33934	.50553	.66656	.82408	.97376	1.11530	1.24742	1.36686	1.47841	1.57478
2.94000	0.00000	.17037	.33935	.50555	.66659	.82412	.97382	1.11538	1.24754	1.36694	1.47867	1.57515
3.08000	0.00000	.17037	.33936	.50555	.66660	.82414	.97384	1.11542	1.24760	1.36691	1.47880	1.57537
3.22000	0.00000	.17037	.33936	.50555	.66660	.82415	.97386	1.11544	1.24762	1.36691	1.47887	1.57546
3.36000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24764	1.36692	1.47891	1.57554
3.50000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24764	1.36692	1.47891	1.57554
3.64000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47893	1.57559
3.78000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
3.92000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.06000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.20000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.34000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.48000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.62000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.76000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
4.90000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.04000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.18000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.32000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.46000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.60000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.74000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
5.88000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.02000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.16000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.30000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.44000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.58000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.72000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
6.86000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560
7.00000	0.00000	.17038	.33936	.50556	.66661	.82415	.97386	1.11545	1.24765	1.36692	1.47894	1.57560

142	RCYCLE	4	5	5	5	5	5	6	6	6	6	6	
	(RAD)▲	1.04720	1.13446	1.22173	1.23918	1.25664	1.27409	1.29154	1.30900	1.32645	1.34390	1.36136	1.37881
	(DEG)▲	60.00000	65.00000	70.00000	71.00000	72.00000	73.00000	74.00000	75.00000	76.00000	77.00000	78.00000	79.00000
	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	.14000	.27127	.26105	.24270	.23763	.23241	.22664	.22046	.21372	.20643	.19848	.18983	.18033
	.28000	.51457	.49919	.46890	.46040	.45140	.44152	.43081	.41911	.40637	.39243	.37716	.36036
	.42000	.72960	.71351	.67713	.66662	.65528	.64278	.62911	.61409	.59763	.57953	.55960	.53755
	.56000	.91644	.90330	.86581	.85451	.84211	.82831	.81308	.79621	.77759	.75699	.73416	.70877
	.70000	1.07578	1.06835	1.03376	1.02265	1.01026	.99628	.98067	.96373	.94380	.92213	.89795	.87087
	.84000	1.20849	1.20913	1.18043	1.17028	1.15971	1.14544	1.13042	1.11342	1.09428	1.07274	1.04850	1.02114
	.98000	1.31888	1.32681	1.30597	1.29737	1.28717	1.27527	1.26155	1.24578	1.22779	1.20731	1.18403	1.15752
	1.12000	1.40552	1.42315	1.41128	1.40045	1.39010	1.38060	1.37406	1.36006	1.34384	1.32510	1.30355	1.27874
	1.26000	1.47411	1.50037	1.49779	1.49295	1.48660	1.47853	1.46865	1.45675	1.44266	1.42611	1.40679	1.38428
	1.40000	1.52673	1.56096	1.56741	1.56456	1.56025	1.55428	1.54657	1.53691	1.52514	1.51100	1.49419	1.47432
	1.54000	1.56618	1.60747	1.62227	1.62131	1.61896	1.61504	1.60945	1.60202	1.59259	1.58091	1.56671	1.54961
	1.68000	1.59511	1.64240	1.66460	1.66537	1.66481	1.66278	1.65918	1.65386	1.64665	1.63736	1.62570	1.61134
	1.82000	1.61583	1.66806	1.69658	1.69886	1.69988	1.69953	1.69772	1.69430	1.68914	1.68204	1.67275	1.66098
	1.96000	1.63032	1.68650	1.72025	1.72379	1.72616	1.72726	1.72699	1.72524	1.72187	1.71672	1.70957	1.70014
	2.10000	1.64022	1.69945	1.73738	1.74197	1.74545	1.74774	1.74877	1.74842	1.74559	1.74312	1.73781	1.73043
	2.24000	1.64683	1.70835	1.74954	1.75495	1.75932	1.76258	1.76466	1.76547	1.76490	1.76282	1.75917	1.75344
	2.38000	1.65113	1.71432	1.75797	1.76402	1.76908	1.77310	1.77601	1.77774	1.77819	1.77725	1.77478	1.77058
	2.52000	1.65387	1.71824	1.76371	1.77023	1.77582	1.78041	1.78397	1.78641	1.78766	1.78762	1.78616	1.78312
	2.66000	1.65556	1.72076	1.76752	1.77439	1.78036	1.78539	1.78943	1.79241	1.79427	1.79492	1.79425	1.79212
	2.80000	1.65659	1.72233	1.77000	1.77712	1.78337	1.78872	1.79310	1.79648	1.79879	1.79996	1.79989	1.79846
	2.94000	1.65719	1.72323	1.77158	1.77889	1.78532	1.79089	1.79552	1.79919	1.80183	1.80338	1.80375	1.80284
	3.08000	1.65754	1.72387	1.77256	1.77998	1.78656	1.79227	1.79709	1.80096	1.80383	1.80565	1.80635	1.80582
	3.22000	1.65773	1.72421	1.77317	1.78066	1.78733	1.79315	1.79808	1.80209	1.80512	1.80714	1.80806	1.80780
	3.36000	1.65784	1.72440	1.77352	1.78107	1.78779	1.79368	1.79869	1.80279	1.80594	1.80809	1.80916	1.80910
	3.50000	1.65789	1.72451	1.77373	1.78131	1.78807	1.79400	1.79907	1.80323	1.80645	1.80868	1.80961	1.80993
	3.64000	1.65792	1.72456	1.77385	1.78145	1.78823	1.79419	1.79929	1.80349	1.80676	1.80905	1.81030	1.81046
	3.78000	1.65794	1.72459	1.77392	1.78153	1.78833	1.79430	1.79942	1.80364	1.80694	1.80927	1.81057	1.81078
	3.92000	1.65795	1.72461	1.77396	1.78157	1.78838	1.79436	1.79949	1.80373	1.80705	1.80940	1.81073	1.81098
	4.06000	1.65795	1.72462	1.77398	1.78159	1.78841	1.79440	1.79953	1.80378	1.80711	1.80948	1.81082	1.81110
	4.20000	1.65795	1.72462	1.77399	1.78160	1.78842	1.79442	1.79956	1.80381	1.80715	1.80952	1.81088	1.81117
	4.34000	1.65795	1.72462	1.77399	1.78161	1.78843	1.79443	1.79957	1.80383	1.80717	1.80954	1.81091	1.81121
	4.48000	1.65795	1.72462	1.77399	1.78161	1.78843	1.79443	1.79958	1.80384	1.80718	1.80956	1.81093	1.81123
	4.62000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79443	1.79958	1.80384	1.80718	1.80956	1.81094	1.81124
	4.76000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79443	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
	4.90000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
	5.04000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
	5.18000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81125
	5.32000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	5.46000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	5.60000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	5.74000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	5.88000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.02000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.16000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.30000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.44000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.58000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.72000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	6.86000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126
	7.00000	1.65795	1.72462	1.77400	1.78162	1.78844	1.79444	1.79958	1.80384	1.80719	1.80957	1.81095	1.81126

NCYCLE	7	7	7	8	8	9	10	12
R+ (RAD)*	1.39626	1.41372	1.43117	1.44862	1.46608	1.48353	1.50098	1.51844
(DEG)*	80.80700	81.00000	82.00000	83.00000	84.00000	85.00000	86.00000	87.00000
0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
.14000	.16987	.15823	.14516	.13024	.11286	.09190	.06510	.02541
.29000	.34176	.32109	.29758	.27073	.23926	.20107	.15174	.07719
.42000	.51704	.48554	.45437	.41846	.37613	.32438	.25681	.15256
.56000	.68076	.64835	.61186	.56958	.51943	.45766	.37613	.24779
.70000	.84819	.80582	.76619	.72000	.66485	.59639	.50507	.35858
.84000	.99017	.95471	.91384	.86590	.80926	.73615	.63897	.48030
.98000	1.12723	1.09237	1.05186	1.00400	.94604	.87295	.77348	.60838
1.12000	1.25013	1.21693	1.17803	1.13173	1.07525	1.00344	.90477	.73850
1.26000	1.35803	1.32728	1.29093	1.24732	1.19370	1.12498	1.02967	.86679
1.40000	1.45095	1.42386	1.38989	1.34976	1.30000	1.23572	1.14576	.99000
1.54000	1.52912	1.50454	1.47491	1.43871	1.39345	1.33451	1.25132	1.10556
1.68000	1.59383	1.57252	1.54653	1.51445	1.47400	1.42090	1.34536	1.21158
1.82000	1.64631	1.62815	1.60572	1.57773	1.54212	1.49501	1.42751	1.30687
1.96000	1.68806	1.67283	1.65372	1.62961	1.59865	1.55741	1.49793	1.39066
2.10000	1.72056	1.70804	1.69194	1.67137	1.64473	1.60900	1.55722	1.46353
2.24000	1.74564	1.73528	1.72181	1.70439	1.68162	1.65090	1.60624	1.52527
2.38000	1.76443	1.75597	1.74475	1.73003	1.71063	1.68434	1.64608	1.57683
2.52000	1.77830	1.77141	1.76204	1.74959	1.73305	1.71058	1.67792	1.61915
2.66000	1.78836	1.78272	1.77485	1.76426	1.75009	1.73082	1.70293	1.65332
2.80000	1.79552	1.79085	1.78418	1.77506	1.76281	1.74617	1.72227	1.68046
2.94000	1.80052	1.79660	1.79085	1.78289	1.77216	1.75762	1.73697	1.70167
3.08000	1.80395	1.80060	1.79554	1.78847	1.77891	1.76603	1.74797	1.71799
3.22000	1.80627	1.80332	1.79877	1.79237	1.78370	1.77210	1.75607	1.73036
3.36000	1.80780	1.80514	1.80097	1.79505	1.78705	1.77641	1.76195	1.73958
3.50000	1.80880	1.80635	1.80244	1.79687	1.78935	1.77942	1.76614	1.74635
3.64000	1.80943	1.80712	1.80340	1.79888	1.79091	1.78150	1.76908	1.75125
3.78000	1.80983	1.80762	1.80402	1.79887	1.79194	1.78290	1.77112	1.75475
3.92000	1.81008	1.80792	1.80441	1.79938	1.79262	1.78384	1.77251	1.75720
4.06000	1.81022	1.80811	1.80465	1.79970	1.79306	1.78445	1.77344	1.75890
4.20000	1.81031	1.80823	1.80480	1.79990	1.79333	1.78485	1.77405	1.76006
4.34000	1.81036	1.80829	1.80489	1.80002	1.79350	1.78510	1.77445	1.76084
4.48000	1.81039	1.80833	1.80494	1.80009	1.79361	1.78526	1.77471	1.76136
4.62000	1.81041	1.80836	1.80498	1.80014	1.79367	1.78536	1.77487	1.76169
4.76000	1.81042	1.80837	1.80499	1.80016	1.79371	1.78541	1.77497	1.76191
4.90000	1.81042	1.80837	1.80500	1.80018	1.79373	1.78545	1.77504	1.76205
5.04000	1.81043	1.80839	1.80501	1.80019	1.79374	1.78547	1.77507	1.76214
5.18000	1.81043	1.80839	1.80501	1.80019	1.79375	1.78548	1.77510	1.76219
5.32000	1.81043	1.80839	1.80501	1.80019	1.79376	1.78549	1.77511	1.76222
5.46000	1.81043	1.80839	1.80501	1.80019	1.79376	1.78549	1.77512	1.76224
5.60000	1.81043	1.80839	1.80501	1.80019	1.79376	1.78550	1.77512	1.76225
5.74000	1.81043	1.80839	1.80501	1.80020	1.79376	1.78550	1.77512	1.76226
5.88000	1.81043	1.80839	1.80501	1.80020	1.79376	1.78550	1.77512	1.76226
6.02000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.16000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.30000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.44000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.58000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.72000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
6.86000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227
7.00000	1.81043	1.80838	1.80501	1.80020	1.79376	1.78550	1.77513	1.76227

143

OGAMMA= 1.552795 SMALLM= .035059

POINT VORTEX LOCATIONS

TOP BOUNDARY LAYER

BOTTOM BOUNDARY LAYER

A	K	X(A)	Y(A)	XDOT(A)	YDOT(A)	GAMMA(A)	XB(A)	YB(A)	XDOTB(A)	YDOTB(A)	GMA3(A)
1	39	-1.470798	.387896	-.084505	-.001953	.017547	-1.470798	-.387896	-.084505	.001953	-.017547
2	39	-1.478492	.562271	-.400360	-.037539	.037135	-1.478492	-.562271	-.400360	.037539	-.037135
3	39	-1.374651	.513777	-.274386	-.037525	.061020	-1.374651	-.513777	-.274386	.037525	-.061020
4	39	-1.204024	.592420	-.283578	.054769	.072364	-1.204024	-.592420	-.283578	.054769	-.072364
5	39	-1.250174	.423280	-.092518	.084677	.084417	-1.250174	-.423280	-.092518	.084677	-.084417
6	39	-1.049625	.658744	-.317442	.186665	.080703	-1.049625	-.658744	-.317442	.186665	-.080703
7	39	-.897825	.456571	.231030	.491555	.095220	-.897825	-.456571	.231030	.491555	-.095220
8	39	-1.173107	.302748	.129747	.263879	.106676	-1.173107	-.302748	.129747	.263879	-.106676
9	39	-1.011672	.435365	-.028465	.268545	.102046	-1.011672	-.435365	-.028465	.268545	-.102046
10	39	-.915831	.436608	.237372	.536720	.115301	-.915831	-.436608	.237372	.536720	-.115301
11	39	-1.007367	.135213	.115211	.471943	.124202	-1.007367	-.135213	.115211	.471943	-.124202
12	39	-1.673170	.311740	-.007193	-.031816	.131124	-1.673170	-.311740	-.007193	-.031816	-.131124
13	39	-1.094772	.049451	.405832	.028246	.126061	-1.094772	-.049451	.405832	-.028246	-.126061
14	39	-1.479694	.161685	.392592	.037173	.134868	-1.479694	-.161685	.392592	.037173	-.134868
15	39	-1.319320	.198246	.317274	.082329	.129898	-1.319320	-.198246	.317274	.082329	-.129898
16	39	-1.607210	.030250	1.050349	-.005364	.127849	-1.607210	-.030250	1.050349	.005364	-.127849
17	39	-2.076149	.255191	-.185664	-.474283	.135813	-2.076149	-.255191	-.185664	.474283	-.135813
18	39	-1.710810	.148758	.450861	-.114611	.131037	-1.710810	-.148758	.450861	-.114611	-.131037
19	39	-1.983040	.186633	.276527	-.098828	.129110	-1.983040	-.186633	.276527	.098828	-.129110
20	39	-2.204532	.125398	.014800	-.275954	.127702	-2.204532	-.125398	.014800	.275954	-.127702
21	39	-1.624535	.627607	-.481717	-.082781	.134314	-1.624535	-.627607	.481717	.082781	-.134314
22	39	-1.795804	.462980	-.345550	-.150479	.129791	-1.795804	-.462980	.345550	.150479	-.129791
23	39	-1.940624	.367292	-.170558	-.338341	.128055	-1.940624	-.367292	.170558	.338341	-.128055
24	39	-1.961811	.431517	-.641739	-.599320	.126807	-1.961811	-.431517	.641739	.599320	-.126807
25	39	-1.864390	.601270	-.847009	-.472699	.125738	-1.864390	-.601270	.847009	.472699	-.125738
26	39	-1.666043	.723831	-.902404	-.217859	.124786	-1.666043	-.723831	.902404	.217859	-.124786
27	39	-1.728547	.794835	-1.210554	-.499095	.123862	-1.728547	-.794835	1.210554	.499095	-.123862
28	39	-1.294580	.767692	-.569317	-.130324	.124797	-1.294580	-.767692	.569317	.130324	-.124797
29	39	-1.393426	.819677	-.948762	-.372662	.123000	-1.393426	-.819677	.948762	.372662	-.123000
30	39	-1.221233	.865369	-.945012	-.188080	.121990	-1.221233	-.865369	.945012	.188080	-.121990
31	39	-1.127894	.909917	-1.034864	-.050010	.121092	-1.127894	-.909917	1.034864	.050010	-.121092
32	39	-.970191	.899408	-.728856	-.174231	.122041	-.970191	-.899408	.728856	.174231	-.122041
33	39	-.880058	.931829	-.855026	-.078367	.120286	-.880058	-.931829	.855026	.078367	-.120286
34	39	-.769794	.949858	-.836941	-.104686	.119314	-.769794	-.949858	.836941	.104686	-.119314
35	39	-.654144	.970624	-.894004	-.135379	.118474	-.654144	-.970624	.894004	.135379	-.118474
36	39	-.523922	.997195	-1.025793	-.180170	.117752	-.523922	-.997195	1.025793	.180170	-.117752
37	39	-.353272	1.030128	-1.257200	-.240873	.118833	-.353272	-1.030128	1.257200	.240873	-.118833
38	39	-.159547	1.048199	-1.529307	-.134673	.117236	-.159547	-1.048199	1.529307	.134673	-.117236
39	39	.054171	1.033641	-1.704354	.115828	.116460	.054171	-1.033641	-1.704354	-.115828	-.116460

TOP REAR SHEAR LAYER

BOTTOM REAR SHEAR LAYER

A	K	XRT(A)	YRT(A)	XROOT(A)	YROOT(A)	GMRT(A)	XRB(A)	YRB(A)	XROOTB(A)	YROOTB(A)	GMA3(A)
1	39	-1.391255	.428310	-.060685	-.034127	-.001701	-1.391255	-.428310	-.060685	.034127	.001701
2	39	-1.366517	.602144	-.457523	-.084011	-.009489	-1.366517	-.602144	-.457523	.084011	.009489
3	39	-1.445651	.649619	-.541731	-.076439	-.020866	-1.445651	-.649619	-.541731	.076439	.020866
4	39	-.960804	.712634	-.271053	.221530	-.012567	-.960804	-.712634	-.271053	.221530	.012567
5	39	-.850305	.590171	.030460	.210316	-.064597	-.850305	-.590171	.030460	-.210316	.064597

PRESSURE DISTRIBUTION

T= 5.0477

7FAT= .8770

AK= 1.0000

AKOOT= 0.0000

THETAS= 87.00

CEG	PHI1	PHI2	Z(PHIT)	PSIKSQD	CPK	PDHAG	UTAN
0.00	.007121	0.000000	.014242	-.000000	.014242	.014242	-.000000
10.00	-.007059	0.000000	.014115	-.115164	-.101046	-.099511	.339359
20.00	.006851	0.000000	.013701	-.445701	-.432000	-.405947	.667608
30.00	-.006413	0.000000	.012826	-.948413	-.935586	-.810241	.973865
40.00	.005566	0.000000	.011131	-1.556424	-1.545493	-1.183916	1.247647
50.00	-.003913	0.000000	.007826	-2.187262	-2.179436	-1.400915	1.476939
60.00	.000524	0.000000	.001048	-2.748803	-2.747755	-1.373878	1.657952
65.00	-.002488	0.000000	-.004975	-2.974329	-2.974304	-1.259108	1.724624
70.00	-.007181	0.000000	-.014366	-3.147067	-3.161433	-1.081274	1.773997
75.00	-.014934	0.000000	-.029868	-3.253856	-3.283724	-.849890	1.803845
80.00	-.028889	0.000000	-.057779	-3.277452	-3.335431	-.579192	1.810429
85.00	-.057824	0.000000	-.115648	-3.188001	-3.303649	-.287932	1.785498
90.00	-.135008	0.000000	-.270016	-2.898292	-3.168308	-.000000	1.702437
95.00	-.457438	0.000000	-.914875	-2.016878	-2.931754	.255519	1.420168
100.00	-.954259	0.000000	-1.988499	-1.043036	-2.951535	.512529	1.021291
105.00	-.595697	0.000000	-1.101379	-1.270289	-2.461668	.637127	1.127071
110.00	-.780847	0.000000	-1.561682	-.704720	-2.266402	.775155	.839476
115.00	-.673573	0.000000	-1.347147	-.574824	-1.921971	.812260	.758172
120.00	-.730796	0.000000	-1.441592	-.302666	-1.764258	.882129	.550151
125.00	-.745331	0.000000	-1.400663	-.164166	-1.654829	.949171	.405174
130.00	-.758687	0.000000	-1.517375	-.077875	-1.595244	1.025406	.279060
135.00	-.766323	0.000000	-1.542646	-.027253	-1.559899	1.103015	.165885
140.00	-.775455	0.000000	-1.540910	-.005173	-1.556083	1.192029	.071922
POTENTIAL VORTEX APPROX INVALID			ALPHA= 5	CORE RADIUS=	.035297		
145.00	-.715488	0.000000	-1.430977	-.037499	-1.468676	1.203069	-.194163
150.00	-.543223	0.000000	-1.086447	-.234987	-1.321433	1.144395	-.484754
POTENTIAL VORTEX APPROX INVALID			ALPHA= 7	CORE RADIUS=	.034997		
POTENTIAL VORTEX APPROX INVALID			ALPHA= 10	CORE RADIUS=	.016923		
155.00	-.752162	0.000000	-1.504324	-.359233	-1.863557	1.688956	-.599360
160.00	-.709888	0.000000	-1.419772	-.424160	-1.843937	1.732734	-.651276
165.00	-.686461	0.000000	-1.372923	-.515227	-1.888150	1.823813	-.717793
POTENTIAL VORTEX APPROX INVALID			ALPHA= 11	CORE RADIUS=	.031208		
170.00	-.756222	0.000000	-1.512445	-.233699	-1.746144	1.719616	-.483425
175.00	-.652105	0.000000	-1.304209	-.253229	-1.557438	1.551512	-.503219
180.00	-.551320	0.000000	-1.102640	-.000000	-1.102640	1.102640	-.000000

K= 39
 CDPK= .503871
 CDSK= .012166
 GEM= .516037
 CDN= .034568

MAX VELOCITY= 1.811 AT THETA= 79.0 DEGREES
MAX BACKFLOW VEL= -.718 AT THETA=165.0 DEGREES
MIN TANGENTIAL VELOCITY BETWEEN MAX VELOCITY AND MAX BACKFLOW VELOCITY= .072 AT THETA=140.0 DEGREES
REAR SEPARATION ANGLE=161.7 DEGREES

147

RVIA SUBROUTINE N= 5 THETA(S(N))= 161.721311 U00(N)= -.635553
GAMA(N)= -.025248

GAMA CHECK SUM= -.058117 N= 5 FLAG(N=1)= 1

BEFORE VFIX(X(ALPHA),Y(ALPHA))= (-.85786, .50642)

VMFIX ALPHA= 10 ANGLE= 75.488734
X= -.903573 Y= .430764 XDOT= .304025 YDOT= .637725

AFTER VFIX(X(ALPHA),Y(ALPHA))= (-.86557, .51048)
ELAPSED TIME= 44.988000

ADYNF SAMPLE INPUT
CASE ... OGIVE CYLINDER ... ANGLE OF ATTACK (5 DEG)

*****		ADYNF	SAMPLE	CASE	*****		
5.0		0.0	50.478	10.74	2.35	2.35	CARD 1
4	15						CARD 2
0.0		0.0					CARD 3
		.096516	9.036431	.181378	1		CARD 4
		.163031	8.165342	.306378	2		CARD 5
		.229547	4.086353	.431378	3		CARD 6
		.296063	.047975	.556378	4		CARD 7
		.362578	.041644	.681378	5		CARD 8
		.429094	.043186	.806378	6		CARD 9
		.495610	.055931	.931378	7		CARD 10
		.562125	.069124	1.056378	8		CARD 11
		.628641	.087352	1.181378	9		CARD 12
		.695157	.105378	1.306378	10		CARD 13
		.761672	.132163	1.431378	11		CARD 14
		.828188	.154608	1.556378	12		CARD 15
		.894704	.187945	.681378	13		CARD 16
		.961215	.222631	.806378	14		CARD 17
	1.0	.249663	1.931378	15			CARD 18
0.0		.125	.3	1.0			CARD 19
							CARD 20

***** ADYNF SAMPLE OUTPUT *****

GIVEN DATA

1	0.0000000	0.0000000
2	.09651600	.06864185
3	.16303100	.06202495
4	.22954700	.03104044
5	.29606300	.00036442
6	.36257800	.00031633
7	.42909400	.00032805
8	.49561000	.00042486
9	.56212500	.00052507
10	.62864100	.00066354
11	.69515700	.00080046
12	.76167200	.00100393
13	.82818800	.00117442
14	.89470400	.00142765
15	.96121900	.00169113
16	1.0000000	.00189647

NO. OF INITIAL KNOTS = 4
 ITER = 8
 KNOTS PRIOR TO OPTIMIZATION
 0.000000 .125000 .300000 1.000000

*** FINAL OUTPUT ***

KNOTS	CUBIC COEFFICIENTS
XI(1) = -0.000000	C(1) = -2.892801E-14 C(2) = -4.355484E+00 C(3) = 1.185283E+02 C(4) = -6.991980E+02
XI(2) = .071209	C(1) = 3.840772E-02 C(2) = 1.888742E+00 C(3) = -3.083995E+01 C(4) = 1.349466E+02
XI(3) = .149811	C(1) = 6.186244E-02 C(2) = -4.582100E-01 C(3) = 9.812087E-01 C(4) = -6.308512E-01
XI(4) = 1.000000	

LEAST SQUARE ERROR = 4.933780E-03
 AVERAGE ERROR = 3.812333E-03
 MAXIMUM ERROR = 1.349815E-02 AT .296063

APPROXIMATION AND SCALED ERROR CURVE

	DATA POINT	APPROXIMATION	DEVIATION X 10E+3
1	0.00000000	-.00000000	.000000
2	.09551500	.06864185	.000000
3	.16303100	.05597500	6.049947
4	.22954700	.03124520	-.204763
5	.29506300	.01386257	-13.496148
6	.36257800	.00271330	-2.396967
7	.42909400	-.00331686	3.644905
8	.49561000	-.00534162	5.766478
9	.56212500	-.00447494	5.000012
10	.62864100	-.00183070	2.494236
11	.69515700	.00147717	-.676710
12	.76167200	.00433472	3.330796
13	.82818800	.00562812	-4.453697
14	.89470400	.00424339	-2.815733
15	.96121900	-.00093329	2.624420
16	1.00000000	-.00614404	8.040511

Z	CON(Z)
0.000000	-.000000
.050000	-.008853
.100000	.070443
.150000	.061776
.200000	.041257
.250000	.025170
.300000	.013040
.350000	.004395
.400000	-.001278
.450000	-.004332
.500000	-.005361
.550000	-.004798
.600000	-.003116
.650000	-.000787
.700000	.0001714
.750000	.003915
.800000	.005342
.850000	.005523
.900000	.003985
.950000	.000253
1.000000	-.006144

BEG,STEP 0. 1.00000000E+00 1
 BEG,STEP 5.00000000E-01 5.00000000E-01 1
 H2 CONVRGENCE AT ROW 3
 INTEGRAL IS 3.46173213E-04, ERROR 2.01407345E-04 FROM T(3,2)
 BEG,STEP 0. 5.00000000E-01 1
 BEG,STEP 2.50000000E-01 2.50000000E-01 1
 H2 CONVRGENCE AT ROW 3
 INTEGRAL IS 1.03065782E-03, ERROR 9.03363896E-05 FROM T(3,2)
 BEG,STEP 0. 2.50000000E-01 1
 H2 CONVRGENCE AT ROW 5
 INTEGRAL IS 8.09074826E-03, ERROR 8.07258925E-05 FROM T(5,2)

147

..... CN= .129465
 CNI= .009468 ERROR= .00037247 IFLAG= 1

GIVEN DATA

150

1	0.00000000	0.00000000
2	.09651600	.00707654
3	.16303100	.01047993
4	.22954700	.00770532
5	.29606300	.00010789
6	.36257800	.00011470
7	.42909400	.00014076
8	.49561000	.00021056
9	.56212500	.00029516
10	.62864100	.00041713
11	.69515700	.00055645
12	.76167200	.00076466
13	.82818800	.00097264
14	.89470400	.00127733
15	.96121900	.00162555
16	1.00000000	.00189647

NO. OF INITIAL KNOTS = 6
ITER = 0

KNOTS PRIOR TO OPTIMIZATION

0.000000 .071209 .149811 1.000000

*** FINAL OUTPUT ***

KNOTS

CUBIC COEFFICIENTS

XI(1) = 0.000000

C(1) = -1.284775E-14
C(2) = -9.314970E+00
C(3) = 2.089747E+02
C(4) = -1.183289E+03

XI(2) = .071209

C(1) = -3.091969E+02
C(2) = 2.446370E+00
C(3) = -4.380895E+01
C(4) = 2.525566E+02

XI(3) = .129259

C(1) = 1.286898E-02
C(2) = -8.665235E-02
C(3) = 1.733354E-01
C(4) = -1.033547E-01

XI(4) = 1.000000

LEAST SQUARE ERROR = 9.302876E-04
AVERAGE ERROR = 6.924194E-04
MAXIMUM ERROR = 2.650252E-03 AT .296063

APPROXIMATION AND SCALED ERROR CURVE

DATA POINT	APPROXIMATION	DEVIATION X 10E+4
1	0.00000000	-.00000000
2	.09651600	.00702654
3	.16303100	.01013625
4	.22954700	.00581788
5	.29606300	.00275814
6	.36257800	.00077458
7	.42909400	-.00031538
8	.49561000	-.00069418
9	.56212500	-.00054434
10	.62864100	-.00004834
11	.69515700	.00061131
12	.76167200	.00125211
13	.82818800	.00169159
14	.89470400	.00174722
15	.96121900	.00123654
16	1.00000000	.00060493

(.5=Z)CON

0.000000	-.000000
.050000	-.091223
.100000	.009227
.150000	.011145
.200000	.007570
.250000	.004752
.300000	.002613
.350000	.001076
.400000	.000063
.450000	-.000502
.500000	-.000699
.550000	-.000603
.600000	-.000293
.650000	.000154
.700000	.000661
.750000	.001149
.800000	.001542
.850000	.001761
.900000	.001730
.950000	.001370
1.000000	.000605

BEG,STEP 0. 1.00000000E+00 1
 BEG,STEP 5.00000000E-01 5.00000000E-01 1
 H2 CONVERGENCE AT ROW 3
 INTEGRAL IS 3.75214479E-04, ERROR 2.49143177E-05 FROM T(3,2)
 BEG,STEP 0. 5.00000000E-01 1
 BEG,STEP 2.50000000E-01 2.50000000E-01 1
 H2 CONVERGENCE AT ROW 3
 INTEGRAL IS 2.53646229E-04, ERROR 1.58105198E-05 FROM T(3,2)
 BEG,STEP 0. 2.50000000E-01 1
 H2 CONVERGENCE AT ROW 5
 INTEGRAL IS -4.86222290E-03, ERROR 1.84561813E-04 FROM T(5,2)

151

..... CM= .057890
 CMI= -.004233 ERROR= .00022529 IFLAG= 1

LIST OF REFERENCES

1. Hall, M. G., "A Numerical Method for Calculating the Unsteady Two Dimensional Laminar Boundary Layer," Ing-Arch 38 97 (1969).
2. Wundt, H., "Wachstum der laminaren Grenzschicht an schräg angeströmten Zylinder bei Anfahrt aus der Ruhe," Ingen-Arch 23 212 (1955).
3. Carl de Boor and John R. Rice, "Least Squares Cubic Spline Approximation I - Fixed Knots," Purdue University Computer Sciences Department Report CSD TR20, (1968).
4. Carl de Boor and John R. Rice, "Least Squares Cubic Spline Approximation II - Variable Knots," Purdue University Computer Sciences Department Report CSD TR21, (1968).

☆ U.S. GOVERNMENT PRINTING OFFICE: 1974-739-161/138