NASA Contractor Report 4292, Vol. II

A Three-Dimensional, Compressible, Laminar Boundary-Layer Method for General Fuselages

Volume II—User's Manual

Yong-Sun Wie
High Technology Corporation
Hampton, Virginia

Prepared for
Langley Research Center
under Contract NAS1-18240

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

1990

SUMMARY

This user's manual contains a complete description of the computer programs developed to calculate three-dimensional, compressible, laminar boundary-layers for perfect gas flow on general fuselage shapes. These programs include the 3-D boundary-layer program (3DBLC), the body-oriented coordinate program (BCC), and the streamline coordinate program (SCC). Program BCC reads the inviscid solution from the inviscid code and calculates the nonorthogonal body-oriented coordinates and the boundary-layer edge conditions for the boundary-layer grid. Program SCC reads the inviscid solution from the inviscid code and calculates the orthogonal streamline coordinates and the boundary-layer edge conditions on the streamline boundary-layer grid. Program 3DBLC utilizes the boundary-layer edge conditions obtained from the coordinate programs (BCC or SCC) for the calculation of 3-D boundary-layer. A schematic of the procedure is shown in Fig.1.

Using these programs, the flow of both the subsonic and supersonic speed regimes over any fuselage shape(both the blunted nose and sharp nose fuselages) can be solved in both the body-oriented coordinate and the streamline coordinate systems. The current 3DBLC does not include interaction between the inviscid and viscous flow.

The numerical method is described in volume I. In the present volume, the descriptions of these programs (3DBLC, BCC, and SCC) including subroutine description, input, output, and a sample case are presented. The complete FORTRAN listings of the computer programs are also included.

TABLE OF CONTENTS

NOMENCLATURE

A, B	stagnation point velocity gradients
a, b	major and minor semiaxis length of the ellipsoid of revolution
C	$\rho\mu/\rho_e\mu_e$
C^*	B/A
C_{fx}	skin friction coefficient in the x-direction based on the edge condition, defined in Eqs.(103a) or (104a) of Volume I
C_{fy}	skin friction coefficient in the y-direction based on the edge condition, defined in Eqs.(103b) or (104b) of Volume I
Cp	Pressure coefficient
c_p	specific heat
E	H/H_e, defined in Eq.(48) of Volume I
F	u/u_e, defined in Eq.(48) of Volume I
f_ζ	F, defined in Eq.(48) of Volume I
G	v/V_{ref} *or* v_y/V_{ref}, defined in Eqs.(48) or (56) of Volume I
g_ζ	G, defined in Eqs.(48) or (56) of Volume I
H	total enthalpy
h_1, h_2	metric coefficients in the x and y coordinates, respectively.
i, j, k	numerical index for x, y, and z directions, respectively
$IMAX, JMAX, KMAX$	number of boundary-layer grid points in the x, y, and z directions, respectively
M	Mach number
$m_1, .., m_{13}$	coefficients, defined in Eqs. (54) or (60) of Volume I
p	pressure
Pr	Prandtl number (0.72)
r	radius measured from X axes, Fig.41 in Volume I.
R, Θ, ϕ	spherical polar coordinates, Fig.41 in Volume I.

q_w	heat transfer at the wall, defined in Eq.(108) of Volume I
s	arc length measured along $y = const$ lines.
T	temperature
u, v, w	velocity components in the x, y, and z directions
u_R, u_Θ, u_ϕ	inviscid velocity components in the R, Θ, ϕ directions
$u_{x'}, u_{y'}, u_{z'}$	inviscid velocity components in the x', y' and z' directions
v_y	$\partial v/\partial y$
v_{ye}	$\partial v_e/\partial y$
V	total velocity, defined in Eq. (7) of Volume I
x, y, z	body-oriented coordinates (Fig. 1 in Volume I) or streamline coordinates (Fig. 2 in Volume I)
x', y', z'	rectangular coordinates with the origin at the nose point (Fig.41 in Volume I)
x^*, y^*, z^*	rectangular coordinates with the origin at the stagnation point, Fig.(37) or (38) in Volume I.
X	axial distance measured from the nose, see Fig.1 in Volume I.
α	angle of attack
γ	ratio of specific heat
$\Delta x, \Delta y, \Delta \varsigma$	grid spacing in the x, y, ς directions, respectively.
δ	boundary-layer thickness; $(z)_{V/V_e=0.995}$
δ^*	displacement thickness, defined in Eq.(107) of Volume I
ϵ	small angle to locate the initial streamlines near the stagnation point, Fig.(41) in Volume I.
ς	transformed normal coordinate, defined in Eq.(49) of Volume I
θ	angle between x and y coordinates
θ_r	angle between two rectagular coordinates, (x', y', z') and (x^*, y^*, z^*), Fig.(37) in Volume I.
μ	molecular viscosity

ν	μ/ρ
ρ	density
ϕ	azimuthal angle, 0 and π on the windward and leeward plane of symmetry, respectively, see Fig.1 in Volume I.

<u>subscript</u>

e	edge of the boundary-layer
osp	origin of spherical polar coordinates
s	stagnation point
w	wall
y	partial differentiation with respect to y
ς	partial differentiation with respect to ς
∞	undisturbed free stream

ABBREVIATION

3DBLC	Three-Dimensional Boundary-Layer computer program
BCC	Body-oriented Coordinate computer program
SCC	Streamline Coordinate computer program

PART 1.

THREE-DIMENSIONAL BOUNDARY-LAYER PROGRAM (3DBLC)

1.1 Program Description

Program 3DBLC utilizes the boundary-layer edge conditions obtained from the coordinate programs (BCC or SCC) for the calculation of 3-D boundary-layers on the general fuselage, as shown in Fig. 1. To obtain the 3-D boundary-layer solution on a general fuselage configuration, a geometry program which describes the fuselage shape and an inviscid code are required. A geometry program is needed by BCC or SCC, but not by 3DBLC. The schematic of the procedure shown in Fig. 1 is for the case when the numerical inviscid solution is used. However, this 3DBLC can also use the analytically generated boundary-layer edge conditions, without using the coordinate programs(BCC or SCC), when the analytic inviscid solution exists. The program is coded in the FORTRAN 77 computer language.

The governing boundary-layer equations are in dimensional form; consequently, all inputs to 3DBLC must be consistently dimensional. Either English units (ft, lb, sec, ^oR) or SI (MKS) units (m, kg, sec, ^oK) can be used.

Program 3DBLC is primarily focused on fully three-dimensional, general fuselage type of configurations which have a symmetry plane. This code has been tested intensively for 2-D flows; however, in order to reduce the length of the present user's manual, instructions for calculating 2-D flows are not included. Axisymmetric flow can be solved as a 3-D flow without changing the present 3DBLC.

Using 3DBLC, (1) the flows for both subsonic and supersonic speed can be solved; (2) the flow over both the sharp nose fuselage and blunted nose fuselage can be solved; and (3) the boundary-layer solutions can be obtained both in the body-oriented coordinate system(nonorthogonal) and in the streamline coordinate system.

The code block, COMBLCK, which lists the common blocks, is designed for flexibility in changing the dimensions in the spatial coordinates(x, y, and z) and to avoid listing the common blocks in each subroutine. This COMBLCK is included in the main program and all subroutines (except subroutine SY) by an 'INCLUDE' statement. The dimensions of the common blocks and the local dimensioned variables are controlled by changing the parameters IMAXF, JMAXF, and KMAXF in the COMBLCK.

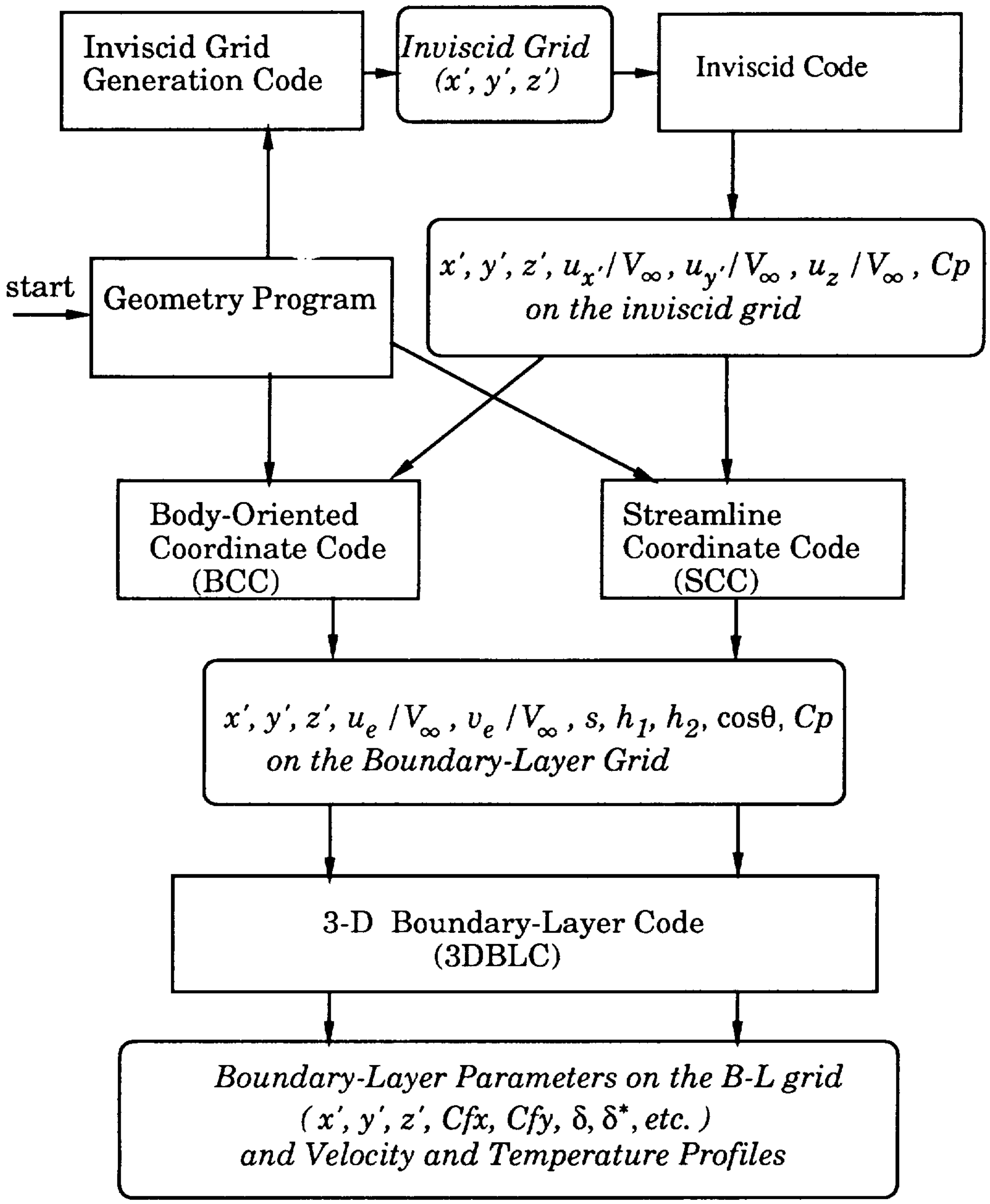


Fig. 1. Program Schematic

1.2 Structure of Main Program

Figure 2 shows the flow chart of the main program BLMAIN. The program first calls subroutine INPUT. The flow conditions are given in subroutine INPUT. Subroutine INPUT also reads in the boundary-layer edge conditions either from BCC or SCC. Wall boundary conditions are also specified in this subroutine. Then, the main program BLMAIN dimensionalizes the velocity components and calculates the temperature, pressure, density and viscosity at the edge of the boundary-layer.

If the nose of the body is blunted, subroutine STAGPT is called to obtain the boundary-layer solution at the stagnation point. For the sharp nose body, subroutine COEFCON is called to calculate m_1 through m_{13} of the boundary-layer equations on the cone, and subroutines CONON and CONOFF are called to obtain the boundary-layer solution based on the body-oriented coordinate system near the nose tip. Then, depending on the shape of the nose and the coordinate system chosen, the initial velocity profiles at i=1 (near the stagnation point or near the nose tip) are calculated by calling subroutine INPOS (for sharp nose body, streamline coordinates), or INBUB (for blunted nose body, body-oriented coordinates), or INBUS (for blunted nose body, streamline coordinates). The notation of i and j on both the blunted and sharp nose fuselage can be found in Figure 3. No Subroutine is required to obtain the initial velocity profiles for the sharp nose body when using the body-oriented coordinate system. The initial profiles, F, G, and E are stored in H(1,J,K), H(2,J,K), and H(3,J,K), respectively.

The boundary-layer calculation starts from i=2. The coefficients $m_1, m_2, .., m_{13}$ are calculated using subroutine COEFBODY when using the body-oriented coordinate system or COEFSTRM when using the streamline coordinate system. Subroutine PREDICT is called to solve the predictor momentum and energy equations. The solutions are stored in HB(1,J,K), HB(2,J,K), and HB(3,J,K). The corrector momentum equations are then solved by calling subroutine CORRECT. The solution of F and G are temporarily stored in HN(1,J,K) and HN(2,J,K), respectively. If $(u/u_e)_{j=JMAX,\,k=KMAX-1}$ is not greater than

UKMAX1, KMAX is increased (ς_e is increased accordingly) and returned back to the predictor step.

Here, the check is also made whether the stepsize taken (Δx_i) satisfies the zone of dependence principle. If the zone of dependence principle is not satisfied, subroutine OUTPUT is called to stop the calculations. If the stepsize satisfies this principle, the solutions of corrector momentum equations are stored in H(1,J,K) and H(2,J,K). Then the corrector energy equation is solved using subroutine CORRENG. The solution of corrector energy equation, the total enthalphy profile, is stored in H(3,J,K).

The boundary-layer parameters are calculated at each step (each i) using subroutine BLPARA. Subroutine PROFILE is called to write the velocity and temperature profiles at the desired i-th and j-th step. If flow separation occurs at any j $((\partial u/\partial \varsigma)_w \leq 0)$, the boundary-layer calculation also terminates. If the stepsize satisfies the zone of dependence principle and separation does not occur, the boundary-layer calculations are continued to the next i-th step. If the boundary-layer calculation is finished or terminated for any reason, subroutine OUTPUT is called to write the input echo, the velocity and temperature profiles at the last i-th step and the boundary-layer parameters on the surface grid.

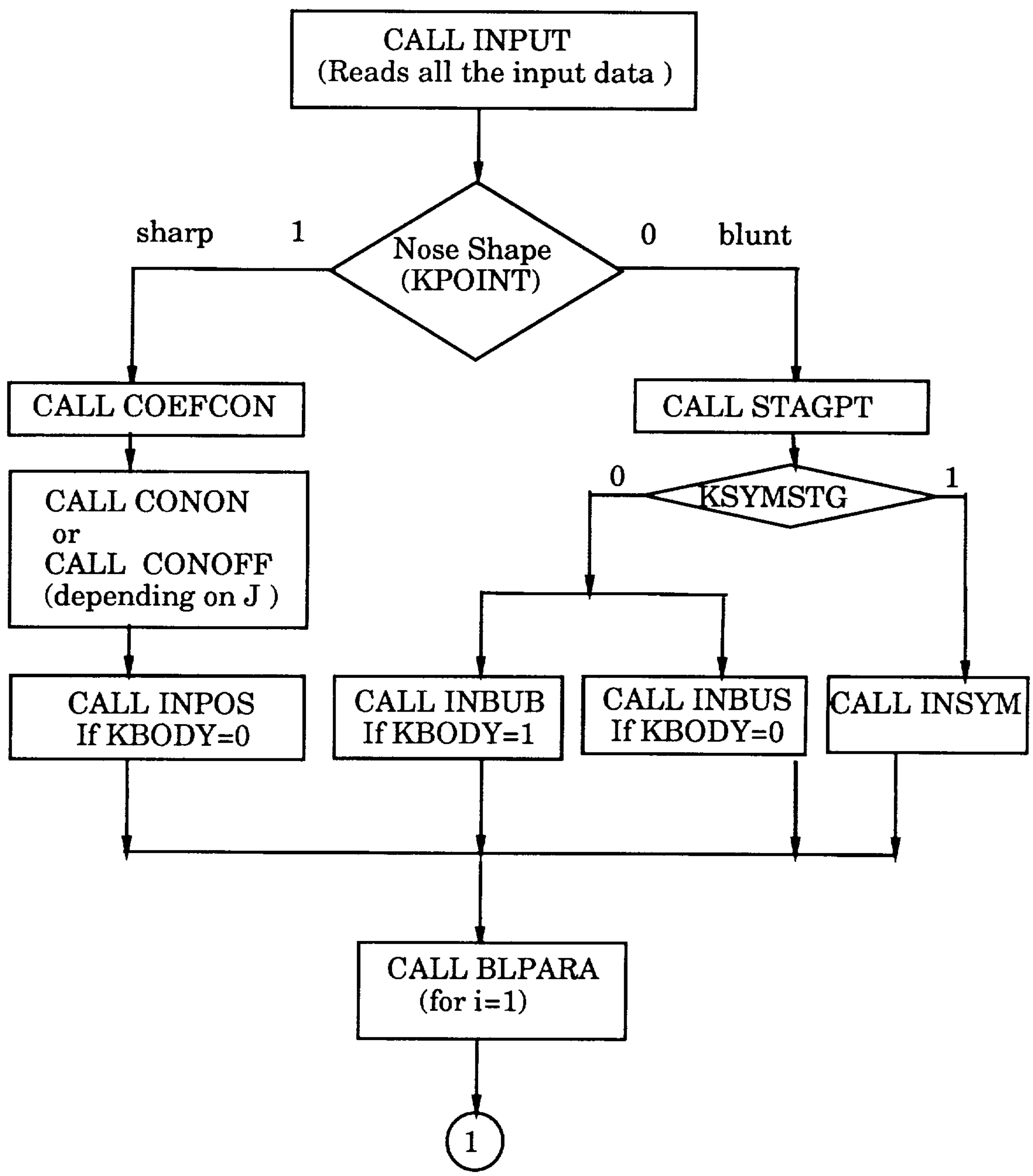


Fig. 2. Flow Chart of the Main Program BLMAIN

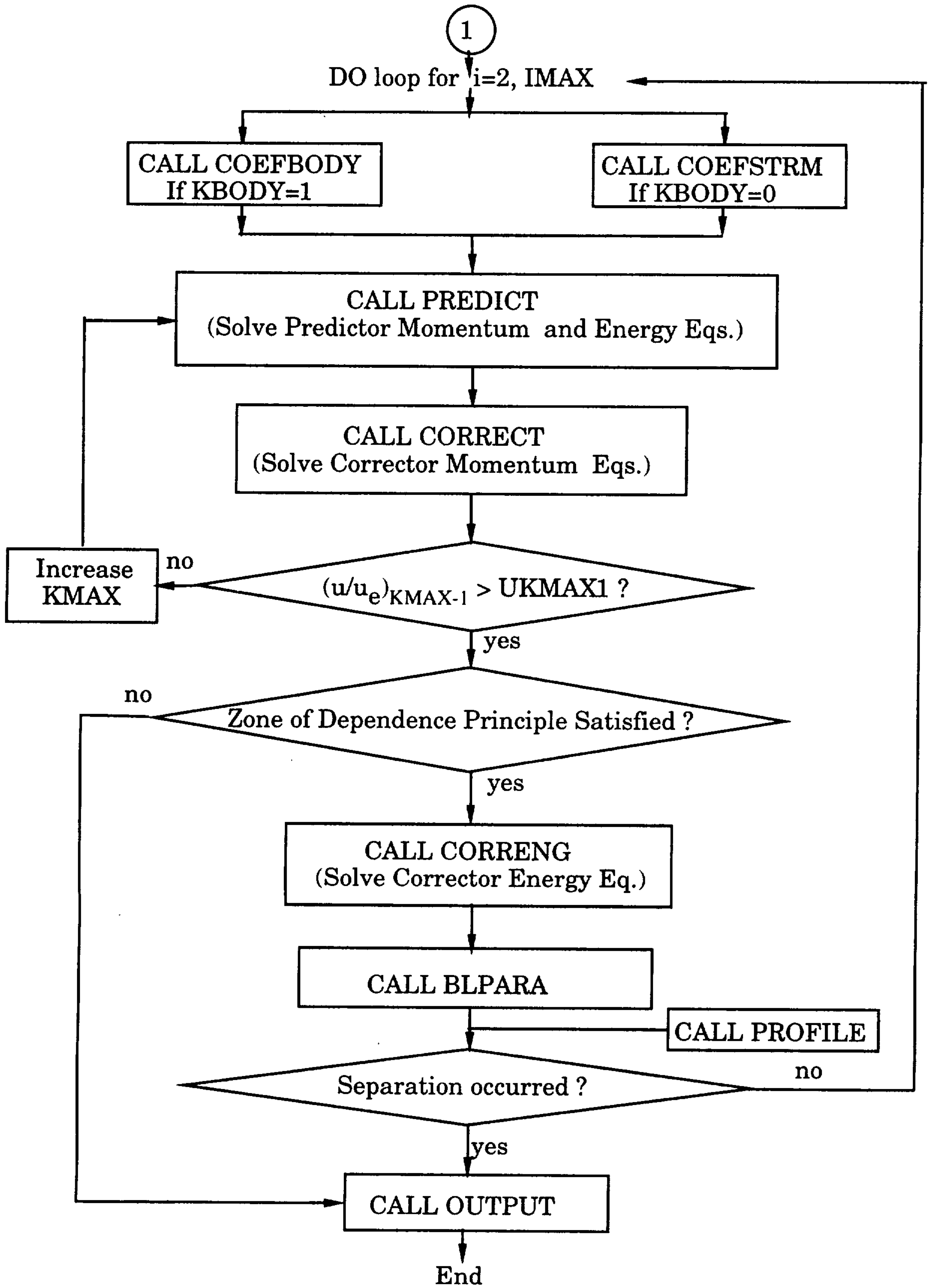
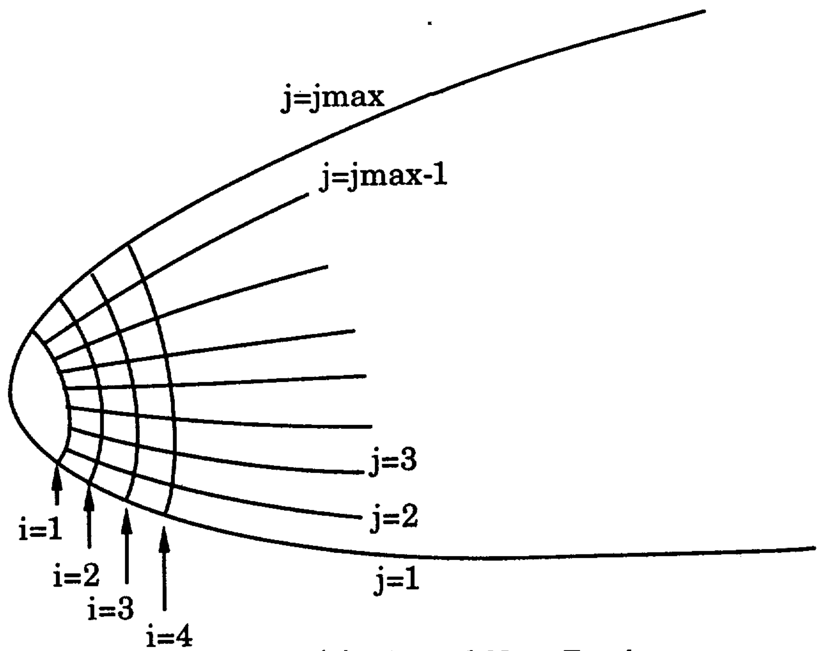
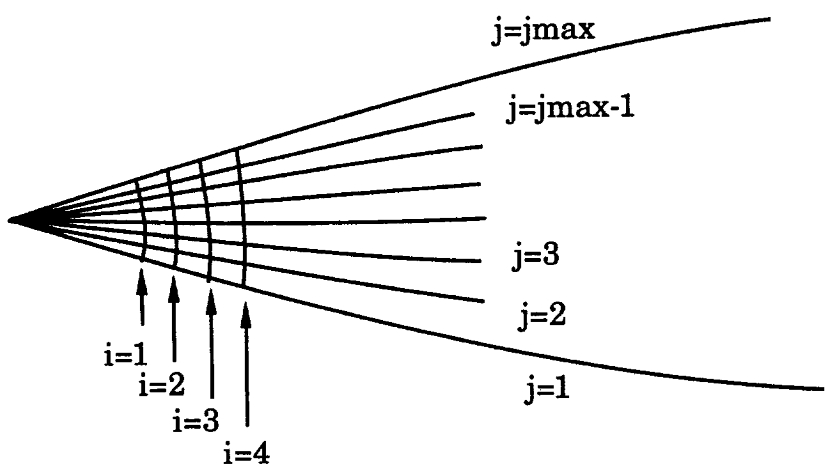


Fig. 2. concluded.



(a) Blunted Nose Fuselage



(b) Sharp Nose Fuselage

Fig. 3. Index notation; i, j

1.3 Subroutine Description

Subroutine BLPARA

- Called by the main program BLMAIN.
- Calculates the boundary-layer parameters $C_{fx}, C_{fy}, \delta, \delta^*, q_w$, or T_w in the physical quantities.

Subroutine COEFBODY

- Called by the main program BLMAIN.
- Calculates the coefficients $m_1, m_2, .., m_{13}$ when using the boody-oriented coordinate system.

Subroutine COEFCON

- Called by the main program BLMAIN.
- Called only for a sharp nosed body when i=1.
- Calculates the coefficients $m_1, m_2, .., m_{13}$ of the transformed equation for the cone.

Subroutine COEFSTRM

- Called by the main program BLMAIN.

- Calculates the coefficients $m_1, m_2, .., m_{13}$ when using the streamline coordinate system.

Subroutine CONOFF

- Called by the main program BLMAIN. Calls subroutines NTRID and SY.
- Called only for the sharp nose body when i=1 and j=2,3,.., JMAX-1.
- Obtains the boundary-layer solution off the line of symmetry near the sharp nose tip using the boundary-layer equations on the cone:
  1. Calculates the coefficients of 2x2 block tridiagonal momentum equations.
  2. Calls subroutine NTRID to solve this block tridiagonal matrix equations.
  3. Calculates the coefficients of tridiagonal energy equations.
  4. Calls subroutine SY to solve this tridiagonal equations.
  5. Repeats steps, 1 through 4, until the converged solution is obtained.
  6. Stores the converged solutions of $F(= u/u_e)$, f, $G(= v/V_\infty)$, g, and $E(= H/H_e)$ in HN(1,J,K), HSN(1,J,K), HN(2,J,K), HSN(2,J,K), and HN(3,J,K), respectively.

Subroutine CONON

- Called by the main program BLMAIN. Calls subroutines NTRID and SY.
- Called only for the sharp nose body when i=1, j=1, or i=1, j=JMAX.
- Obtains the boundary-layer solution on the line of symmetry near the sharp nose tip using the boundary-layer equations on the cone:
  1. Calculates the coefficients of 2x2 block tridiagonal momentum equations.

2. Calls subroutine NTRID to solve this block tridiagonal matrix equations.
3. Calculates the coefficients of tridiagonal energy equations.
4. Calls subroutine SY to solve this tridiagonal equations.
5. Repeats steps, 1 through 4, until the converged solution is obtained.
6. Stores the converged solutions of $F(= u/u_e)$, f, $G(= v_y/V_\infty)$, g, and $E(= H/H_e)$ in HN(1,J,K), HSN(1,J,K), HN(2,J,K), HSN(2,J,K), and HN(3,J,K), respectively.

- The boundary-layer solution on the leeward line of symmetry may or may not exist.

Subroutine CORRECT

- Called by the main program BLMAIN. Calls subroutine NTRID.
- Solves the corrector momentum equations:
  1. Calculates the coefficients of 2x2 block tridiagonal corrector momentum equations.
  2. Calls subroutine NTRID to solve this block tridiagonal equations.
  3. Temporarily store the solutions $F(= u/u_e)$, $G(= v/V_\infty$ or $=v_y/V_\infty)$ in HN(1,J,K), HN(2,J,K), respectively.

Subroutine CORRENG

- Called by the main program BLMAIN. Calls subroutine SY.
- Solves the corrector energy equation:
  1. Calculates the coefficients of tridiagonal corrector energy equation.

2. solves tridiagonal equations by calling subroutine SY.
3. Stores the solution $E(= H/H_e)$ in H(3,J,K); stores C and ρ_e/ρ in BC(J,K) and ROERO(J,K), respectively.

Subroutine INBUB

- Called by the main program BLMAIN.
- Called only when using the body-oriented coordinate system for blunted nose body.
- Obtains the initial velocity and total enthalpy profiles near the stagnation point(at $i = 1$):

1. Assumes the stagnation point solution has been obtained.
2. Assumes the location of the stagnation point (x'_s, z'_s), θ_r, the velocity gradients at the stagnation point (A, B, and C^*), and the coordinates of the initial points near the stagnation point ($x'(1,J)$, $y'(1,J)$, $z'(1,J)$, $J = 1,2,..,JMAX$) are given.
3. Calculates the initial profiles based on the body-oriented coordinate system using the equations presented in Appendix B.3 of Volume I.
4. Stores these profiles, i.e., $F(= u/u_e)$, f, $G(= v/V_\infty$ or $v_y/V_\infty)$, g, $E(= H/H_e)$, in H(1,J,K), HS(1,J,K), H(2,J,K), HS(2,J,K), and H(3,J,K), respectively.

Subroutine INBUS

- Called by the main program BLMAIN.
- Called only when using the streamline coordinate system for blunted nose body.
- Obtains the initial velocity and total enthalpy profiles based on the streamline coordinates near the stagnation point(at $i = 1$):

1. Assumes the stagnation point solution has been obtained.

2. Assumes the location of the stagnation point (x'_s, z'_s), θ_r, the velocity gradients at the stagnation point (A, B, and C^*), and the coordinates of the initial points near the stagnation point ($x'(1,J)$, $y'(1,J)$, $z'(1,J)$, $J = 1,2,..,JMAX$) are given.

3. Calculates the initial profiles using the method presented in Appendix B.2 of Volume I.

4. Stores these profiles, i.e., $F(= u/u_e)$, f, $G(= v/V_\infty$ or $v_y/V_\infty)$, g, $E(= H/H_e)$, in H(1,J,K), HS(1,J,K), H(2,J,K), HS(2,J,K), and H(3,J,K), respectively.

Subroutine INPOS

- Called by the main program BLMAIN.
- Called only when using the streamline coordinate system for sharp nose body.
- Obtains the initial velocity and total enthalpy profiles based on the streamline coordinates near the nose tip(at $i = 1$):

  1. Calculates the initial profiles using the equations presented in Appendix C of Volume I.
  2. Stores these profiles, i.e., $F(= u/u_e)$, f, $G(= v/V_\infty$ or $v_y/V_\infty)$, g, $E(= H/H_e)$, in H(1,J,K), HS(1,J,K), H(2,J,K), HS(2,J,K), and H(3,J,K), respectively.

Subroutine INPUT

- Called by the main program BLMAIN.
- Reads in inputs to 3DBLC:

  1. The flow conditions and other input quantities are given in the code.

2. Reads in the boundary-layer edge conditions either from BCC or SCC when using the numerical inviscid solution.
3. Calculates the ς-distribution.
4. Wall temperature condition(T_w) is specified for fixed wall temperature condition. This is not necessary for adiabatic wall condition.
5. Wall mass injection($(\rho w)_w$) is specified for wall mass injection condition. If there is no mass injection, this is not necessary.

Subroutine INSYM

- Called by the main program BLMAIN.
- Called only for blunted nose body and when KSYMSTG=1.
- Obtains the initial velocity and total enthalpy profiles near the stagnation point(at $i = 1$).
    1. Assumes the axisymmetric stagnation point solution (using $C^* = 1$) has been obtained.
    2. Does not require the location of the stagnation point (x'_s, z'_s), θ_r, and the velocity gradients at the stagnation point (A, B, and C^*).
    3. Uses axisymmetric stagnation point profiles for the initial profiles.
    4. Stores these profiles, i.e., $F(= u/u_e)$, f, $G(= v/V_\infty$ or $v_y/V_\infty)$, g, $E(= H/H_e)$, in H(1,J,K), HS(1,J,K), H(2,J,K), HS(2,J,K), and H(3,J,K), respectively.

Subroutine NTRID

- Called by subroutines STAGPT, CONON, CONOFF, PREDICT and CORRECT.

- Solves 2x2 block tridiagonal matrix equations using Davis Modified Tridiagonal Algorithm (For algorithm, see Appendix A of Volume I).

- Returns the solutions of F, f, G, and g in HN(1,J,K), HSN(1,J,K), HN(2,J,K), and HSN(2,J,K), respectively.

Subroutine PREDICT

- Called by the main program BLMAIN. Calls subroutines NTRID and SY.

- Solves the predictor momentum and energy equations:

  1. Calculates the coefficients of 2x2 block tridiagonal predictor momentum equations.
  2. Solve these block tridiagonal equations using subroutine NTRID.
  3. Calculates the coefficients of tridiagonal predictor energy equations.
  4. Solves these tridiagonal equations using subroutine SY.
  5. Stores the solutions $F(= u/u_e)$, $G(= v/V_\infty$ or $= v_y/V_\infty)$, $E(= H/H_e)$ in HB(1,J,K), HB(2,J,K), HB(3,J,K), respectively.

Subroutine PROFILE

- Called by the main program BLMAIN.

- Writes the velocity and temperature profile at the desired i-th and j-th step on the file unit IW. The desired i-th and j-th step is determined by INI and JNI.

Subroutine STAGPT

- Called by the main program BLMAIN. Calls subroutines NTRID and SY.
- Called only for blunted nose body.
- Obtains the boundary-layer solution at the stagnation point:
    1. Calculates the coefficients of 2x2 block tridiagonal momentum equations.
    2. Solves these block tridiagonal equations using subroutine NTRID.
    3. Calculates the coefficients of tridiagonal energy equations.
    4. Solves these tridiagonal equations using subroutine SY.
    5. Repeats steps, 1 through 4, above until the converged solution is obtained.
    6. Stores the converged solutions of $F(=u/u_e)$, f, $G(=v/v_e)$, g, and $E(=H/H_e)$ in HN(1,1,K), HSN(1,1,K), HN(2,1,K), HSN(2,1,K), and HN(3,1,K), respectively.
- The key parameter for this solution is C^*; the axisymmetric stagnation point solution or 2-D stagnation point solution can be obtained by setting C^*=1 or C^*=0, respectively.

Subroutine SY(IL, IU, B, D, A, C)

- Called by subroutines STAGPT, CONON, CONOFF, PREDICT and CORRENG.
- Does not include COMBLCK.
- Solves the following tridiagonal system of equations using the Thomas algorithm.

  $B_k E_{k-1} + D_k E_k + A_k E_{k+1} = C_k$
- The definitions of IL, IU are:

  IL: subscript k of the first equation in the system

IU: subscript k of the last equation in the system

- Returns the solution vector for $E_K(K = IL, .., IU)$ to the calling program in the C array.

Subroutine OUTPUT

- Called by the main program BLMAIN.

- Writes input echo, the velocity and temperature profiles at the last step, and the boundary-layer parameters on the surface grid.

1.4 Parameter and Variable Directory

ASTAR	A, stagnation point velocity gradient in the x^* direction
BC(J,K)	$C(= \rho\mu/\rho_e\mu_e)$ at $(x_i,\, y_j)$
BCB(J,K)	$C(= \rho\mu/\rho_e\mu_e)$ at the predictor step $(x_{i+1/2},\, y_j)$
BLTH(I,J)	δ, boundary-layer thickness
BSTAR	B, stagnation point velocity gradient in the y^* direction
CAVD(I,J)	V_e, inviscid total velocity
CFX(I,J)	C_{fx}
CFY(I,J)	C_{fy}
COSTH(I,J)	$\cos\theta$
CP	c_p, specific heat
CPD(I,J)	Cp, pressure coefficient
CSTAR	$C^*(= B/A)$
DH2DS	$\partial h_2/\partial s$
DK1DY	$\partial(K_1 \cos\theta)/\partial y$
DSPTH(I,J)	δ^*, displacement thickness
DUEDSD(I,J)	$\partial u_e/\partial s$
DUEDYD(I,J)	$\partial u_e/\partial y$
DX	Δx_i
DXH	$\Delta x_i/2$
DY(J)	Δy_j
DZETA(K)	$\Delta \zeta_k$
GAMMA	γ
H(1,J,K)	u/u_e at the point $(x_i,\, y_j)$
H(2,J,K)	v/V_∞ at the point $(x_i,\, y_j)$
H(3,J,K)	H/H_e at the point $(x_i,\, y_j)$
HB(1,J,K)	u/u_e at the predictor step$(x_{i+1/2},\, y_j)$

HB(2,J,K)	v/V_∞ at the predictor step($x_{i+1/2}$, y_j)
HB(3,J,K)	H/H_e at the predictor step($x_{i+1/2}$, y_j)
HN(1,J,K)	the solution for u/u_e from subroutine NTRID
HN(2,J,K)	the solution for v/V_∞ or v/v_e from subroutine NTRID
HS(1,J,K)	f at the point (x_i, y_j)
HS(2,J,K)	g at the point (x_i, y_j)
HSB(1,J,K)	f at the predictor step ($x_{i+1/2}$, y_j)
HSB(2,J,K)	g at the predictor step ($x_{i+1/2}$, y_j)
HSP(1,J,K)	f at the previous step (x_i, y_j)
H1(I,J)	h_1 at the point (x_i, y_j)
H2(I,J)	h_2 at the point (x_i, y_j)
I	index for the boundary-layer grid in the x direction
IL	i-th step where the boundary-layer calculation stops (not necessarily the same as IMAX)
IMAX	actual number of grid points in the x-direction (IMAX$\leq$IMAXF)
IMAXF	maximum possible number of grid points in the x-direction, given in COMBLCK
INC	=0 when the energy equation is to be solved (for compressible flow) =1 when the energy equation is not to be solved (for incompressible flow)
INI	number of intervals of i-th steps where the velocity and temperature profiles are written on file unit IW
IW	unit for writing the velocity and temperature profiles using subroutine PROFILE
J	index for the boundary-layer grid in the y direction
JMAX	actual number of grid points in the y-direction (JMAX$\leq$JMAXF)
JMAXF	maximum possible number of grid points in the y-direction, given in COMBLCK

JMAX1	J-th station to where the boundary-layer solution exists (JMAX1$\leq$JMAX)
JNI	number of intervals of j-th steps where the velocity and temperature profiles are written on file unit IW
K	index for the boundary-layer grid in the ς direction
KAW	=0 when the wall temperature is given as a boundary condition =1 for adiabatic wall condition
KBODY	=1 when using the body-oriented coordinate system =0 when using the streamline coordinate system
KCPGIVN	=0 when pressure coefficients at the edge of the boundary-layer are not given as input =1 when pressure coefficients at the edge of the boundary-layer are given as input
KMAX	actual number of grid in the z-direction (may be changed as i increases) (KMAX$\leq$KMAXF)
KMAXF	maximum possible number of grid in the z-direction, given in COMBLCK
KPOINT	=1 when the shape of nose is sharp =0 when the shape of nose is blunted
KROW	=1 when wall mass injection exists =0 when there is no wall mass injection
KSYMSTG	=1 to obtain the stagnation point solution using $C^* = 1$ =0 to obtain the stagnation point solution using given C^*
MKS	=0 when using the English units $(ft, lb, sec, {}^oR)$ =1 when using SI(MKS) units $(m, kg, sec, {}^oK)$
M1(J),..,M13(J)	$m_1, .., m_{13}$
PE(I,J)	p, pressure
PI	π

PINF	p_∞, free stream pressure
PR	Pr, Prandtl Number
RMINF	M_∞
RMYUED(I,J)	μ_e
RMYUEH(J)	(RMYUED(I,J)+RMYUED(I+1,J))/2
RNUINF	ν_∞
ROED(I,J)	ρ_e
ROEH(J)	(ROED(I,J)+ROED(I+1,J))/2
ROERO(J,K)	ρ_e/ρ
ROEROB(J,K)	ρ_e/ρ at the predictor step $(x_{i+1/2}, y_j)$
ROINF	ρ_∞
ROWW(I,J)	$(\rho w)_w$
RR	gas constant
SS	speed of sound
S1(I,J)	s
S1H(I,J)	(S1(I,J)+S1(I+1,J))/2
TB(J,K)	temperature inside the boundary-layer at the predictor step$(x_{i+1/2}, y_j)$
TD(J,K)	temperature inside the boundary-layer at (x_i, y_j)
TE(I,J)	boundary-layer edge temperature
THETAR	θ_r
THMOM(I,J)	momentum thickness, defined as $\int_0^\infty \frac{\rho}{\rho_e}\frac{V}{V_e}(1-\frac{V}{V_e})dz$
TINF	T_∞
TWALL(I,J)	T_w
UE(I,J)	u_e
UKMAX1	KMAX and ς_e $(=\varsigma(KMAX))$ are to be increased going downstream so that $(u/u_e)_{k=KMAX-1}$ is greater than this value, typically 0.9995
VE(I,J)	v_e

VINF	V_∞
VMAX(I,J)	maximum crosswise velocity based on the streamline coordinate system
XD(I)	x
XKI(I,J)	cross-flow Reynolds number, defined as $\frac{\rho_e VMAX(I,J)\delta}{\mu_e}$
XPD(I,J)	x'
XPS	x'_s, x' of the stagnation point
XSTAR	x^*
YD(J)	y
YPD(I,J)	y'
YSTAR	y^*
ZACT(J,K)	z
ZETA(K)	ς
ZETAE	ς_e,=ZETA(KMAX)
ZPD(I,J)	z'
ZPS	z'_s, z' of the stagnation point

1.5 Input

All the inputs to 3DBLC are specified or read through subroutine INPUT. The procedure is as follows:

(1) The flow conditions and other input parameters are specified in this subroutine.

MKS	=1 when using the SI (MKS) Units $(m, kg, sec, ^o K)$
	=0 when using the English Units $(ft, lb, sec, ^o R)$
INC	=1 when the energy equation need not be solved (This can be used when the incompressible boundary-layer solution is sought. In this case, the density variation across the boundary-layer is neglected, i.e., $T/T_e = 1$ and $\rho_e/\rho = 1$)
	=0 when the energy equation also need to be solved
KPOINT	=1 when the shape of nose is sharp
	=0 when the shape of nose is blunted
KBODY	=1 when using the body-oriented coordinate system
	=0 when using the streamline coordinate system
KCPGIVN	=1 when the pressure coefficients(Cp) on the BL grid are given
	=0 when the pressure coefficients(Cp) on the BL grid are not given
KMAX	number of grid in the normal direction at i=1 (Note that ς_e is defined as ς(KMAX), where the ς distribution will be given in step (5))
KAW	=1 when adiabatic wall condition is used (T_w is not needed)
	=0 when the wall temperature is given as a boundary condition (T_w must be specified as a function of x and y)
KROW	=1 when wall mass injection exists ($(\rho w)_w$ must be specified)
	=0 when there is no wall mass injection ($(\rho w)_w$ is not needed)
KSYMSTG	=1 when the stagnation point solution and the initial velocity profiles are obtained using $C^* = 1$ (In this case, x'_s, z'_s, θ_r, A, B, and C^* are not required even though the nose is blunted.)

	=0 when the stagnation point solution is obtained using given C^*
	(In this case, x'_s, z'_s, θ_r, A, B, and C^* must be given)
IW	unit for writing the velocity and temperature profiles using subroutine PROFILE
INI	number of intervals of i-th steps where the velocity and temperature profiles are to be written on file unit IW
JNI	number of intervals of j-th steps where the velocity and temperature profiles are to be written on file unit IW
GAMMA(γ)	=1.4 for air
RR(Gas constant)	$=287 m^2/sec^2\,^oK$ (if MKS=1)
	$=1716 ft^2/sec^2\,^oR$ (if MKS=0)
PR(Pr)	=0.72 for air
RMINF(M_∞)	
PINF(P_∞)	in N/m^2 (if MKS=1)
	in lb/ft^2 (if MKS=0)
TINF(T_∞)	in ^oK (if MKS=1)
	in ^oR if (MKS=0)
UKMAX1	KMAX is to be increased in 3DBLC so that $(u/u_e)_{KMAX-1}$ is greater than this value, typically 0.9995

(2) Then, this subroutine reads in the output(boundary-layer edge conditions) either from BCC or SCC when using the numerical inviscid solution. The boundary-layer edge conditions include:

x'_s, z'_s, θ_r, A, B, C^*

(These quantities are required only for the blunted nose body and can be obtained only from SCC. Therefore, if one wants to obtain the solution using the body-oriented coordinates, one should obtain these quantities using SCC first. However, these quantities are not required if we set KSYMSTG=1 even though the nose is blunted. For the sharp nose body, these quantities are not needed.)

$x(i)$ for i=1,2,..,IMAX

$y(j)$ for j=1,2,..,JMAX

x', y', z', u_e/V_∞, v_e/V_∞, s, h_1, h_2, $\cos\theta$, Cp for i=1,2,..,IMAX, j=1,2,..,JMAX

The pressure coefficient Cp is not necessary for subsonic, shock-free flow because the pressure on the body surface can be calculated from the isentropic relationship with the freestream using the inviscid velocity on the body surface. For the supersonic flow, Cp is necessary because the isentropic relationship with undisturbed free stream no longer holds. When using the streamline coordinate system, v_e and $\cos\theta$ are zero throughout the flow field and h_1 is not necessary because h_1 is defined as V_∞/u_e in the boundary-layer code. It is to be noted that the velocity components(u_e/V_∞, v_e/V_∞) based on the body-oriented coordinate system are required for i=1 when using the streamline coordinates on the sharp nose body, and these are obtained from SCC.

(3) The transformed normal coordinate(ς) distribution for k=1,2,..,KMAXF and $\Delta\varsigma$(k) for k=1,2..,KMAXF-1 are either specified or calculated.

(4) Then, T_w is specified in ^oK (if MKS=1) or in ^oR (if MKS=0) for the given wall temperature condition(KAW=0). For adiabatic wall condition(KAW=1), T_w is not needed.

(5) The value of $(\rho w)_w$ is given for the wall mass injection or suction condition(KROW=1). The value of $(\rho w)_w$ is positive for injection and negative for the suction. The value must be in $N\,sec/m^3$ when using SI unit(MKS=1) or in $lb\,sec/ft^3$ when using the English units(MKS=0). When there is no mass injection or suction(KROW=0), this input is not needed.

1.6 Output

The output from 3DBLC is given through subroutine OUTPUT.

(1) The flow condition and other input parameters are echoed:

MKS

INC

KPOINT

KBODY

KCPGIVN

KMAX (This value may be different from KMAX which was given as a input.)

KAW

KROW

KSYMSTG

IW

INI

JNI

GAMMA(γ)

RR(Gas constant)

PR(Pr)

RMINF(M_∞)

PINF(P_∞)

TINF(T_∞)

UKMAX1

The calculated free-stream conditions, CP(c_p, specific heat), ROINF(ρ_∞), RMYUINF (μ_∞), RNUINF(ν_∞), SS(speed of sound), VINF(V_∞) are also printed. IL(the last i-th step) and JMAX1(the last j-th step where the boundary-layer solution exists) are printed.

(2) The velocity and temperature profiles at the last i-th step (i=IL) are printed:

ς, F, G, $T/T_e (= \rho_e/\rho)$ for k=1,2,..,KMAX, j=1,2,..,JMAX1

(3) The boundary-layer parameters are printed:

x', y', z', C_{fx}, C_{fy}, δ, δ^*, momentum thickness, q_w(if KAW=0) or T_w(if KAW=1)

for i=1,2,..,IL, i=1,2,..,JMAX1

The units for δ, δ^*, and momentum thickness are m (if MKS=1) or ft (if MKS=0).

The unit for q_w is $\frac{W}{m^2}$ (if MKS=1) or $\frac{Btu}{sec\,ft^2}$ (if MKS=0).

The unit for T_w is ^oK (if MKS=1) or ^oR (if MKS=0).

1.7 Sample Case

For a sample case, the compressible flow (M_∞=0.3) over a general aviation fuselage at an angle of attack 3 degrees was calculated in the body-oriented coordinate system. The inviscid solution was first obtained using the Hess code [1]. Then, the boundary-layer edge conditions were obtained from BCC(Part 2). The value of KSYMSTG was set equal to 1 not to necessiate the quantities for the stagnation point, i.e., x'_s, z'_s, θ_r, A, B, C^*.

The flow conditions are:

$M_\infty = 0.3$

$P_\infty = 101324\, N/m^2$

$T_\infty = 288^o K$

$\alpha = 3^o$

$T_w = T_{aw}$

$(\rho w)_w = 0$

To reduce the output data, an IMAX=20 by JMAX=31 boundary-layer grid, which was generated by BCC, was used. For the sample case input, subroutine INPUT is presented. The boundary-layer edge conditions on the body-oriented boundary-layer grid obtained from BCC are also used as an input. The outputs generated by subroutine OUTPUT are presented as a sample case output.

1.7.1 Sample Case Input

```
c########################################################################

      subroutine input

c########################################################################
      include 'comblck'

      mks=1

      inc=0

      kpoint=0

      kbody=1

      kcpgivn=1

      kmax=16

      kaw=1

      krow=0

      ksymstg=1

      iw=80
      ini=50
      jni=1

      gamma=1.4

      if(mks.eq.0)rr=1716.
      if(mks.eq.1)rr=287.

      pr=0.72

      rminf=0.3

      pinf=101324

      tinf=288.

      ukmax1=0.9995

c     read the boundary-layer edge conditions from either BCC or SCC

      if(kbody.eq.1)then

      if(kpoint.eq.1.or.ksymstg.eq.1)go to 33

      rewind 25
      read(25,1112)xps,zps,thetar,astar,bstar,cstar
 33   rewind 22
      read(22,463)imax,jmax
      read(22,461)(xd(i),i=1,imax)
```

```
      read(22,461)(yd(j),j=1,jmax)
      do 60 i=1,imax
      do 60 j=1,jmax
      read(22,462)itr,itr,xpd(i,j),ypd(i,j),zpd(i,j),s1(i,j),ue(i,j)
     &,ve(i,j),h1(i,j),h2(i,j),costh(i,j),cpd(i,j)
 60   continue
 461  format(5(1x,e13.6))
 462  format(2i4,5(1x,e13.6)/8x,5(1x,e13.6))
 463  format(2i10)

      go to 1115
      endif

      if(kbody.eq.0)then
      rewind 25
      read(25,1112)xps,zps,thetar,astar,bstar,cstar
      read(25,463)imax,jmax
      read(25,461)(xd(i),i=1,imax)
      read(25,461)(yd(j),j=1,jmax)
      do 160 i=1,imax
      do 160 j=1,jmax
      read(25,464)itr,itr,xpd(i,j),ypd(i,j),zpd(i,j),s1(i,j),ue(i,j)
     &,ve(i,j),h2(i,j),cpd(i,j)
 160  continue
 464  format(2i4,4(1x,e14.7)/8x,4(1x,e14.7))
 1112 format(6e13.6)
      endif

 1115 continue

c     zeta distribution is specified

      dzetas=0.2
      zeta(1)=0.
      dzeta(1)=dzetas
      do 25 k=2,kmaxf
      dzeta(k)=dzetas
c      dzeta(k)=dzeta(k-1)*1.05
      zeta(k)=zeta(k-1)+dzeta(k)
 25   continue

c     wall condition is given if necessary

      if(krow.eq.0.and.kaw.eq.1)return
      if(krow.eq.0)go to 270
      do 176 i=1,imax
      do 176 j=1,jmax
      roww(i,j)=0.001
 176  continue
 270  if(kaw.eq.1)return
      do 276 i=1,imax
      do 276 j=1,jmax
      twall(i,j)=309.7
 276  continue

      return
      end
```

1.7.2 Sample Case Output

```
********** input  echo  *********************

mks=            1
inc=            0
kpoint=            0
kbody=           1
kcpgivn=           1
kmax=           20
kaw=            1
krow=            0
ksymstg=           1
iw=           80
ini=           50
jni=            1
gamma=   1.400000
rr=   287.0000
pr=  0.7200000
rminf=  0.3000000
pinf=   101324.0
tinf=   288.0000
ukmax1=  0.9995000

********** other free-stream conditions **********

cp=   1004.500
roinf=   1.225852
rmyuinf=   1.797080E-05
rnuinf=   1.465984E-05
ss=   340.1741
vinf=   102.0522

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

il=          20
jmax1=          31

the flow is not separated yet

******* velocity profiles **********


i=   20    j=    1   (xp=     0.0390  yp=     0.0000  zp=    -0.0743 )

  k  zeta    u/ue   vy/vinf    t/te    k  zeta    u/ue   vy/vinf    t/te

  1  0.00  0.00000 0.000E+00 1.0071    2  0.20  0.18516-0.416E-01 1.0069
  3  0.40  0.35040-0.745E-01 1.0064    4  0.60  0.49505-0.995E-01 1.0056
  5  0.80  0.61861-0.117E+00 1.0047    6  1.00  0.72104-0.129E+00 1.0039
  7  1.20  0.80305-0.136E+00 1.0030    8  1.40  0.86620-0.140E+00 1.0023
  9  1.60  0.91278-0.141E+00 1.0016   10  1.80  0.94560-0.141E+00 1.0011
 11  2.00  0.96760-0.141E+00 1.0008   12  2.20  0.98163-0.140E+00 1.0005
 13  2.40  0.99010-0.139E+00 1.0003   14  2.60  0.99495-0.139E+00 1.0002
 15  2.80  0.99756-0.138E+00 1.0001   16  3.00  0.99890-0.138E+00 1.0001
 17  3.20  0.99953-0.138E+00 1.0000   18  3.40  0.99982-0.138E+00 1.0000
 19  3.60  0.99995-0.137E+00 1.0000   20  3.80  1.00000-0.137E+00 1.0000
```

i= 20 j= 2 (xp= 0.0390 yp= 0.0078 zp= -0.0741)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0071	2	0.20	0.18559	-0.433E-02	1.0069
3	0.40	0.35118	-0.776E-02	1.0063	4	0.60	0.49609	-0.104E-01	1.0056
5	0.80	0.61981	-0.122E-01	1.0047	6	1.00	0.72228	-0.134E-01	1.0038
7	1.20	0.80423	-0.142E-01	1.0030	8	1.40	0.86725	-0.145E-01	1.0022
9	1.60	0.91365	-0.147E-01	1.0016	10	1.80	0.94626	-0.147E-01	1.0011
11	2.00	0.96808	-0.146E-01	1.0007	12	2.20	0.98195	-0.146E-01	1.0005
13	2.40	0.99031	-0.145E-01	1.0003	14	2.60	0.99507	-0.145E-01	1.0002
15	2.80	0.99763	-0.144E-01	1.0001	16	3.00	0.99893	-0.144E-01	1.0001
17	3.20	0.99955	-0.144E-01	1.0000	18	3.40	0.99983	-0.144E-01	1.0000
19	3.60	0.99995	-0.144E-01	1.0000	20	3.80	1.00000	-0.143E-01	1.0000

i= 20 j= 3 (xp= 0.0390 yp= 0.0156 zp= -0.0732)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0070	2	0.20	0.18636	-0.854E-02	1.0068
3	0.40	0.35257	-0.153E-01	1.0063	4	0.60	0.49794	-0.203E-01	1.0055
5	0.80	0.62191	-0.238E-01	1.0047	6	1.00	0.72445	-0.262E-01	1.0038
7	1.20	0.80631	-0.276E-01	1.0030	8	1.40	0.86909	-0.283E-01	1.0022
9	1.60	0.91517	-0.286E-01	1.0016	10	1.80	0.94745	-0.286E-01	1.0011
11	2.00	0.96895	-0.286E-01	1.0007	12	2.20	0.98254	-0.284E-01	1.0005
13	2.40	0.99068	-0.283E-01	1.0003	14	2.60	0.99529	-0.283E-01	1.0002
15	2.80	0.99776	-0.282E-01	1.0001	16	3.00	0.99900	-0.282E-01	1.0000
17	3.20	0.99958	-0.281E-01	1.0000	18	3.40	0.99984	-0.281E-01	1.0000
19	3.60	0.99995	-0.281E-01	1.0000	20	3.80	1.00000	-0.280E-01	1.0000

i= 20 j= 4 (xp= 0.0390 yp= 0.0233 zp= -0.0717)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0070	2	0.20	0.18737	-0.125E-01	1.0067
3	0.40	0.35442	-0.223E-01	1.0062	4	0.60	0.50040	-0.297E-01	1.0055
5	0.80	0.62475	-0.348E-01	1.0046	6	1.00	0.72741	-0.382E-01	1.0037
7	1.20	0.80914	-0.402E-01	1.0029	8	1.40	0.87161	-0.412E-01	1.0022
9	1.60	0.91726	-0.416E-01	1.0015	10	1.80	0.94907	-0.417E-01	1.0011
11	2.00	0.97013	-0.415E-01	1.0007	12	2.20	0.98334	-0.414E-01	1.0004
13	2.40	0.99119	-0.412E-01	1.0003	14	2.60	0.99560	-0.411E-01	1.0002
15	2.80	0.99792	-0.410E-01	1.0001	16	3.00	0.99908	-0.410E-01	1.0000
17	3.20	0.99962	-0.409E-01	1.0000	18	3.40	0.99986	-0.409E-01	1.0000
19	3.60	0.99996	-0.409E-01	1.0000	20	3.80	1.00000	-0.408E-01	1.0000

i= 20 j= 5 (xp= 0.0390 yp= 0.0310 zp= -0.0696)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0069	2	0.20	0.18889	-0.162E-01	1.0066
3	0.40	0.35717	-0.289E-01	1.0061	4	0.60	0.50403	-0.383E-01	1.0054
5	0.80	0.62888	-0.448E-01	1.0045	6	1.00	0.73167	-0.491E-01	1.0036
7	1.20	0.81318	-0.515E-01	1.0028	8	1.40	0.87517	-0.527E-01	1.0021
9	1.60	0.92020	-0.532E-01	1.0015	10	1.80	0.95132	-0.532E-01	1.0010
11	2.00	0.97175	-0.530E-01	1.0007	12	2.20	0.98444	-0.528E-01	1.0004
13	2.40	0.99188	-0.526E-01	1.0002	14	2.60	0.99600	-0.524E-01	1.0001

```
 15  2.80  0.99815-0.524E-01 1.0001  16  3.00  0.99920-0.523E-01 1.0000
 17  3.20  0.99968-0.523E-01 1.0000  18  3.40  0.99988-0.522E-01 1.0000
 19  3.60  0.99997-0.522E-01 1.0000  20  3.80  1.00000-0.520E-01 1.0000


i=   20     j=    6    (xp=     0.0390   yp=     0.0386  zp=    -0.0669 )

  k  zeta     u/ue     v/vinf     t/te    k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0067   2  0.20  0.19080-0.194E-01 1.0065
  3  0.40  0.36062-0.344E-01 1.0060   4  0.60  0.50861-0.456E-01 1.0052
  5  0.80  0.63409-0.532E-01 1.0044   6  1.00  0.73703-0.581E-01 1.0035
  7  1.20  0.81825-0.608E-01 1.0027   8  1.40  0.87962-0.621E-01 1.0020
  9  1.60  0.92383-0.625E-01 1.0014  10  1.80  0.95409-0.625E-01 1.0009
 11  2.00  0.97372-0.623E-01 1.0006  12  2.20  0.98574-0.620E-01 1.0004
 13  2.40  0.99269-0.618E-01 1.0002  14  2.60  0.99647-0.616E-01 1.0001
 15  2.80  0.99840-0.615E-01 1.0001  16  3.00  0.99932-0.615E-01 1.0000
 17  3.20  0.99973-0.614E-01 1.0000  18  3.40  0.99991-0.614E-01 1.0000
 19  3.60  0.99997-0.613E-01 1.0000  20  3.80  1.00000-0.611E-01 1.0000


i=   20     j=    7    (xp=     0.0390   yp=     0.0461  zp=    -0.0634 )

  k  zeta     u/ue     v/vinf     t/te    k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0066   2  0.20  0.19293-0.218E-01 1.0064
  3  0.40  0.36449-0.386E-01 1.0058   4  0.60  0.51373-0.509E-01 1.0051
  5  0.80  0.63994-0.593E-01 1.0042   6  1.00  0.74304-0.645E-01 1.0034
  7  1.20  0.82393-0.674E-01 1.0026   8  1.40  0.88457-0.687E-01 1.0019
  9  1.60  0.92785-0.690E-01 1.0013  10  1.80  0.95712-0.689E-01 1.0009
 11  2.00  0.97585-0.686E-01 1.0006  12  2.20  0.98714-0.683E-01 1.0003
 13  2.40  0.99354-0.681E-01 1.0002  14  2.60  0.99695-0.679E-01 1.0001
 15  2.80  0.99865-0.678E-01 1.0001  16  3.00  0.99945-0.678E-01 1.0000
 17  3.20  0.99979-0.677E-01 1.0000  18  3.40  0.99993-0.677E-01 1.0000
 19  3.60  0.99998-0.677E-01 1.0000  20  3.80  1.00000-0.674E-01 1.0000


i=   20     j=    8    (xp=     0.0390   yp=     0.0534  zp=    -0.0593 )

  k  zeta     u/ue     v/vinf     t/te    k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0064   2  0.20  0.19510-0.235E-01 1.0062
  3  0.40  0.36851-0.415E-01 1.0056   4  0.60  0.51916-0.545E-01 1.0049
  5  0.80  0.64622-0.631E-01 1.0041   6  1.00  0.74956-0.684E-01 1.0032
  7  1.20  0.83012-0.712E-01 1.0024   8  1.40  0.89000-0.724E-01 1.0018
  9  1.60  0.93224-0.726E-01 1.0012  10  1.80  0.96043-0.724E-01 1.0008
 11  2.00  0.97815-0.721E-01 1.0005  12  2.20  0.98863-0.717E-01 1.0003
 13  2.40  0.99444-0.715E-01 1.0002  14  2.60  0.99745-0.713E-01 1.0001
 15  2.80  0.99891-0.712E-01 1.0000  16  3.00  0.99957-0.712E-01 1.0000
 17  3.20  0.99984-0.712E-01 1.0000  18  3.40  0.99995-0.711E-01 1.0000
 19  3.60  0.99999-0.711E-01 1.0000  20  3.80  1.00000-0.708E-01 1.0000


i=   20     j=    9    (xp=     0.0390   yp=     0.0603  zp=    -0.0543 )

  k  zeta     u/ue     v/vinf     t/te    k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0062   2  0.20  0.19759-0.241E-01 1.0060
  3  0.40  0.37312-0.423E-01 1.0055   4  0.60  0.52536-0.552E-01 1.0047
  5  0.80  0.65337-0.637E-01 1.0039   6  1.00  0.75693-0.686E-01 1.0031
```

```
  7  1.20  0.83705-0.710E-01 1.0023   8  1.40  0.89598-0.719E-01 1.0016
  9  1.60  0.93700-0.719E-01 1.0011  10  1.80  0.96393-0.716E-01 1.0007
 11  2.00  0.98054-0.712E-01 1.0005  12  2.20  0.99013-0.709E-01 1.0003
 13  2.40  0.99531-0.706E-01 1.0001  14  2.60  0.99792-0.705E-01 1.0001
 15  2.80  0.99914-0.704E-01 1.0000  16  3.00  0.99967-0.704E-01 1.0000
 17  3.20  0.99989-0.703E-01 1.0000  18  3.40  0.99996-0.703E-01 1.0000
 19  3.60  0.99999-0.703E-01 1.0000  20  3.80  1.00000-0.700E-01 1.0000


i=   20     j=   10    (xp=     0.0390  yp=     0.0668   zp=    -0.0486 )

  k  zeta     u/ue     v/vinf     t/te   k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0060   2  0.20  0.19970-0.233E-01 1.0058
  3  0.40  0.37713-0.407E-01 1.0053   4  0.60  0.53088-0.526E-01 1.0045
  5  0.80  0.65980-0.602E-01 1.0037   6  1.00  0.76362-0.644E-01 1.0029
  7  1.20  0.84334-0.664E-01 1.0022   8  1.40  0.90140-0.670E-01 1.0015
  9  1.60  0.94129-0.668E-01 1.0010  10  1.80  0.96705-0.664E-01 1.0007
 11  2.00  0.98263-0.660E-01 1.0004  12  2.20  0.99143-0.657E-01 1.0002
 13  2.40  0.99605-0.654E-01 1.0001  14  2.60  0.99831-0.653E-01 1.0001
 15  2.80  0.99933-0.652E-01 1.0000  16  3.00  0.99975-0.652E-01 1.0000
 17  3.20  0.99992-0.652E-01 1.0000  18  3.40  0.99997-0.652E-01 1.0000
 19  3.60  0.99999-0.651E-01 1.0000  20  3.80  1.00000-0.648E-01 1.0000


i=   20     j=   11    (xp=     0.0390  yp=     0.0728   zp=    -0.0420 )

  k  zeta     u/ue     v/vinf     t/te   k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0058   2  0.20  0.20142-0.210E-01 1.0056
  3  0.40  0.38060-0.364E-01 1.0051   4  0.60  0.53584-0.468E-01 1.0044
  5  0.80  0.66578-0.531E-01 1.0036   6  1.00  0.76995-0.564E-01 1.0028
  7  1.20  0.84937-0.577E-01 1.0020   8  1.40  0.90661-0.578E-01 1.0014
  9  1.60  0.94540-0.574E-01 1.0009  10  1.80  0.97001-0.569E-01 1.0006
 11  2.00  0.98459-0.565E-01 1.0004  12  2.20  0.99261-0.561E-01 1.0002
 13  2.40  0.99671-0.559E-01 1.0001  14  2.60  0.99864-0.558E-01 1.0001
 15  2.80  0.99948-0.558E-01 1.0000  16  3.00  0.99982-0.558E-01 1.0000
 17  3.20  0.99994-0.557E-01 1.0000  18  3.40  0.99998-0.557E-01 1.0000
 19  3.60  1.00000-0.557E-01 1.0000  20  3.80  1.00000-0.553E-01 1.0000


i=   20     j=   12    (xp=     0.0390  yp=     0.0780   zp=    -0.0347 )

  k  zeta     u/ue     v/vinf     t/te   k  zeta     u/ue     v/vinf     t/te

  1  0.00  0.00000 0.000E+00 1.0056   2  0.20  0.20283-0.169E-01 1.0054
  3  0.40  0.38355-0.290E-01 1.0049   4  0.60  0.54018-0.369E-01 1.0042
  5  0.80  0.67107-0.414E-01 1.0034   6  1.00  0.77559-0.435E-01 1.0026
  7  1.20  0.85474-0.441E-01 1.0019   8  1.40  0.91121-0.438E-01 1.0013
  9  1.60  0.94898-0.432E-01 1.0009  10  1.80  0.97255-0.426E-01 1.0005
 11  2.00  0.98623-0.422E-01 1.0003  12  2.20  0.99358-0.419E-01 1.0002
 13  2.40  0.99723-0.417E-01 1.0001  14  2.60  0.99890-0.416E-01 1.0000
 15  2.80  0.99960-0.416E-01 1.0000  16  3.00  0.99986-0.416E-01 1.0000
 17  3.20  0.99996-0.416E-01 1.0000  18  3.40  0.99999-0.416E-01 1.0000
 19  3.60  1.00000-0.415E-01 1.0000  20  3.80  1.00000-0.412E-01 1.0000


i=   20     j=   13    (xp=     0.0390  yp=     0.0822   zp=    -0.0267 )

  k  zeta     u/ue     v/vinf     t/te   k  zeta     u/ue     v/vinf     t/te
```

1	0.00	0.00000	0.000E+00	1.0055	2	0.20	0.20356	-0.117E-01	1.0053
3	0.40	0.38526	-0.196E-01	1.0048	4	0.60	0.54287	-0.244E-01	1.0041
5	0.80	0.67451	-0.267E-01	1.0033	6	1.00	0.77937	-0.273E-01	1.0025
7	1.20	0.85840	-0.269E-01	1.0018	8	1.40	0.91438	-0.262E-01	1.0013
9	1.60	0.95145	-0.254E-01	1.0008	10	1.80	0.97429	-0.247E-01	1.0005
11	2.00	0.98734	-0.243E-01	1.0003	12	2.20	0.99423	-0.240E-01	1.0002
13	2.40	0.99757	-0.239E-01	1.0001	14	2.60	0.99906	-0.238E-01	1.0000
15	2.80	0.99967	-0.238E-01	1.0000	16	3.00	0.99989	-0.238E-01	1.0000
17	3.20	0.99997	-0.238E-01	1.0000	18	3.40	0.99999	-0.237E-01	1.0000
19	3.60	1.00000	-0.237E-01	1.0000	20	3.80	1.00000	-0.234E-01	1.0000

i= 20 j= 14 (xp= 0.0390 yp= 0.0854 zp= -0.0182)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0054	2	0.20	0.20360	-0.516E-02	1.0052
3	0.40	0.38568	-0.814E-02	1.0047	4	0.60	0.54384	-0.932E-02	1.0040
5	0.80	0.67600	-0.919E-02	1.0032	6	1.00	0.78119	-0.827E-02	1.0025
7	1.20	0.86028	-0.704E-02	1.0018	8	1.40	0.91608	-0.583E-02	1.0012
9	1.60	0.95282	-0.484E-02	1.0008	10	1.80	0.97528	-0.414E-02	1.0005
11	2.00	0.98798	-0.368E-02	1.0003	12	2.20	0.99460	-0.343E-02	1.0002
13	2.40	0.99777	-0.329E-02	1.0001	14	2.60	0.99915	-0.324E-02	1.0000
15	2.80	0.99971	-0.321E-02	1.0000	16	3.00	0.99991	-0.321E-02	1.0000
17	3.20	0.99997	-0.321E-02	1.0000	18	3.40	0.99999	-0.320E-02	1.0000
19	3.60	1.00000	-0.317E-02	1.0000	20	3.80	1.00000	-0.292E-02	1.0000

i= 20 j= 15 (xp= 0.0390 yp= 0.0874 zp= -0.0092)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0054	2	0.20	0.20250	0.144E-02	1.0052
3	0.40	0.38387	0.360E-02	1.0047	4	0.60	0.54166	0.625E-02	1.0040
5	0.80	0.67381	0.907E-02	1.0032	6	1.00	0.77925	0.117E-01	1.0025
7	1.20	0.85874	0.141E-01	1.0018	8	1.40	0.91498	0.159E-01	1.0012
9	1.60	0.95210	0.172E-01	1.0008	10	1.80	0.97486	0.180E-01	1.0005
11	2.00	0.98775	0.185E-01	1.0003	12	2.20	0.99449	0.188E-01	1.0002
13	2.40	0.99772	0.189E-01	1.0001	14	2.60	0.99913	0.190E-01	1.0000
15	2.80	0.99970	0.190E-01	1.0000	16	3.00	0.99990	0.190E-01	1.0000
17	3.20	0.99997	0.190E-01	1.0000	18	3.40	0.99999	0.190E-01	1.0000
19	3.60	1.00000	0.190E-01	1.0000	20	3.80	1.00000	0.192E-01	1.0000

i= 20 j= 16 (xp= 0.0390 yp= 0.0881 zp= 0.0000)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0054	2	0.20	0.19973	0.696E-02	1.0052
3	0.40	0.37875	0.135E-01	1.0047	4	0.60	0.53483	0.194E-01	1.0041
5	0.80	0.66609	0.246E-01	1.0033	6	1.00	0.77152	0.290E-01	1.0025
7	1.20	0.85178	0.324E-01	1.0019	8	1.40	0.90929	0.349E-01	1.0013
9	1.60	0.94787	0.366E-01	1.0008	10	1.80	0.97198	0.377E-01	1.0005
11	2.00	0.98597	0.383E-01	1.0003	12	2.20	0.99349	0.386E-01	1.0002
13	2.40	0.99720	0.388E-01	1.0001	14	2.60	0.99889	0.388E-01	1.0000
15	2.80	0.99960	0.389E-01	1.0000	16	3.00	0.99987	0.389E-01	1.0000
17	3.20	0.99996	0.389E-01	1.0000	18	3.40	0.99999	0.389E-01	1.0000
19	3.60	1.00000	0.389E-01	1.0000	20	3.80	1.00000	0.391E-01	1.0000

i= 20 j= 17 (xp= 0.0390 yp= 0.0879 zp= 0.0092)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0054	2	0.20	0.19967	0.105E-01	1.0052
3	0.40	0.37889	0.197E-01	1.0047	4	0.60	0.53530	0.278E-01	1.0041
5	0.80	0.66685	0.345E-01	1.0033	6	1.00	0.77245	0.398E-01	1.0026
7	1.20	0.85272	0.438E-01	1.0019	8	1.40	0.91012	0.467E-01	1.0013
9	1.60	0.94851	0.485E-01	1.0008	10	1.80	0.97243	0.497E-01	1.0005
11	2.00	0.98626	0.503E-01	1.0003	12	2.20	0.99365	0.507E-01	1.0002
13	2.40	0.99729	0.508E-01	1.0001	14	2.60	0.99893	0.509E-01	1.0000
15	2.80	0.99961	0.509E-01	1.0000	16	3.00	0.99987	0.509E-01	1.0000
17	3.20	0.99996	0.509E-01	1.0000	18	3.40	0.99999	0.509E-01	1.0000
19	3.60	1.00000	0.509E-01	1.0000	20	3.80	1.00000	0.511E-01	1.0000

i= 20 j= 18 (xp= 0.0390 yp= 0.0869 zp= 0.0185)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0055	2	0.20	0.20120	0.177E-01	1.0053
3	0.40	0.38241	0.326E-01	1.0048	4	0.60	0.54077	0.447E-01	1.0041
5	0.80	0.67376	0.542E-01	1.0033	6	1.00	0.77995	0.612E-01	1.0025
7	1.20	0.85989	0.662E-01	1.0018	8	1.40	0.91623	0.695E-01	1.0012
9	1.60	0.95320	0.715E-01	1.0008	10	1.80	0.97568	0.726E-01	1.0005
11	2.00	0.98830	0.732E-01	1.0003	12	2.20	0.99481	0.735E-01	1.0002
13	2.40	0.99788	0.737E-01	1.0001	14	2.60	0.99921	0.737E-01	1.0000
15	2.80	0.99973	0.737E-01	1.0000	16	3.00	0.99991	0.737E-01	1.0000
17	3.20	0.99997	0.737E-01	1.0000	18	3.40	0.99999	0.737E-01	1.0000
19	3.60	1.00000	0.737E-01	1.0000	20	3.80	1.00000	0.739E-01	1.0000

i= 20 j= 19 (xp= 0.0390 yp= 0.0843 zp= 0.0274)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0057	2	0.20	0.20068	0.262E-01	1.0055
3	0.40	0.38175	0.478E-01	1.0050	4	0.60	0.54023	0.648E-01	1.0042
5	0.80	0.67349	0.776E-01	1.0034	6	1.00	0.77995	0.868E-01	1.0026
7	1.20	0.86009	0.930E-01	1.0019	8	1.40	0.91653	0.969E-01	1.0013
9	1.60	0.95351	0.993E-01	1.0008	10	1.80	0.97594	0.101E+00	1.0005
11	2.00	0.98848	0.101E+00	1.0003	12	2.20	0.99492	0.101E+00	1.0002
13	2.40	0.99795	0.102E+00	1.0001	14	2.60	0.99924	0.102E+00	1.0000
15	2.80	0.99975	0.102E+00	1.0000	16	3.00	0.99992	0.102E+00	1.0000
17	3.20	0.99998	0.102E+00	1.0000	18	3.40	0.99999	0.102E+00	1.0000
19	3.60	1.00000	0.102E+00	1.0000	20	3.80	1.00000	0.102E+00	1.0000

i= 20 j= 20 (xp= 0.0390 yp= 0.0803 zp= 0.0357)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0059	2	0.20	0.19781	0.347E-01	1.0057
3	0.40	0.37645	0.628E-01	1.0052	4	0.60	0.53315	0.846E-01	1.0044
5	0.80	0.66548	0.101E+00	1.0036	6	1.00	0.77195	0.112E+00	1.0028
7	1.20	0.85291	0.120E+00	1.0020	8	1.40	0.91071	0.124E+00	1.0014
9	1.60	0.94922	0.127E+00	1.0009	10	1.80	0.97307	0.128E+00	1.0006
11	2.00	0.98674	0.129E+00	1.0003	12	2.20	0.99396	0.129E+00	1.0002
13	2.40	0.99747	0.130E+00	1.0001	14	2.60	0.99903	0.130E+00	1.0000

15	2.80	0.99966	0.130E+00	1.0000	16	3.00	0.99989	0.130E+00	1.0000
17	3.20	0.99997	0.130E+00	1.0000	18	3.40	0.99999	0.130E+00	1.0000
19	3.60	1.00000	0.130E+00	1.0000	20	3.80	1.00000	0.130E+00	1.0000

i= 20 j= 21 (xp= 0.0390 yp= 0.0748 zp= 0.0432)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0062	2	0.20	0.19335	0.406E-01	1.0060
3	0.40	0.36798	0.736E-01	1.0055	4	0.60	0.52154	0.992E-01	1.0047
5	0.80	0.65204	0.118E+00	1.0039	6	1.00	0.75819	0.132E+00	1.0030
7	1.20	0.84024	0.141E+00	1.0022	8	1.40	0.90012	0.146E+00	1.0016
9	1.60	0.94117	0.150E+00	1.0011	10	1.80	0.96748	0.151E+00	1.0007
11	2.00	0.98320	0.152E+00	1.0004	12	2.20	0.99192	0.153E+00	1.0002
13	2.40	0.99640	0.153E+00	1.0001	14	2.60	0.99852	0.153E+00	1.0001
15	2.80	0.99944	0.153E+00	1.0000	16	3.00	0.99981	0.153E+00	1.0000
17	3.20	0.99994	0.153E+00	1.0000	18	3.40	0.99998	0.153E+00	1.0000
19	3.60	1.00000	0.153E+00	1.0000	20	3.80	1.00000	0.153E+00	1.0000

i= 20 j= 22 (xp= 0.0390 yp= 0.0682 zp= 0.0496)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0065	2	0.20	0.18779	0.437E-01	1.0063
3	0.40	0.35755	0.794E-01	1.0058	4	0.60	0.50728	0.107E+00	1.0050
5	0.80	0.63545	0.128E+00	1.0042	6	1.00	0.74101	0.143E+00	1.0033
7	1.20	0.82411	0.154E+00	1.0025	8	1.40	0.88628	0.160E+00	1.0018
9	1.60	0.93026	0.164E+00	1.0013	10	1.80	0.95959	0.166E+00	1.0008
11	2.00	0.97794	0.167E+00	1.0005	12	2.20	0.98870	0.168E+00	1.0003
13	2.40	0.99458	0.168E+00	1.0002	14	2.60	0.99758	0.168E+00	1.0001
15	2.80	0.99900	0.168E+00	1.0000	16	3.00	0.99962	0.168E+00	1.0000
17	3.20	0.99987	0.168E+00	1.0000	18	3.40	0.99996	0.168E+00	1.0000
19	3.60	0.99999	0.168E+00	1.0000	20	3.80	1.00000	0.168E+00	1.0000

i= 20 j= 23 (xp= 0.0390 yp= 0.0609 zp= 0.0549)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0068	2	0.20	0.18191	0.436E-01	1.0066
3	0.40	0.34657	0.796E-01	1.0061	4	0.60	0.49227	0.108E+00	1.0054
5	0.80	0.61789	0.130E+00	1.0045	6	1.00	0.72260	0.146E+00	1.0037
7	1.20	0.80648	0.157E+00	1.0028	8	1.40	0.87073	0.164E+00	1.0021
9	1.60	0.91760	0.169E+00	1.0015	10	1.80	0.95003	0.171E+00	1.0010
11	2.00	0.97125	0.173E+00	1.0007	12	2.20	0.98435	0.173E+00	1.0004
13	2.40	0.99197	0.174E+00	1.0002	14	2.60	0.99612	0.174E+00	1.0001
15	2.80	0.99825	0.174E+00	1.0001	16	3.00	0.99926	0.174E+00	1.0000
17	3.20	0.99971	0.174E+00	1.0000	18	3.40	0.99990	0.174E+00	1.0000
19	3.60	0.99997	0.174E+00	1.0000	20	3.80	1.00000	0.174E+00	1.0000

i= 20 j= 24 (xp= 0.0390 yp= 0.0532 zp= 0.0591)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0070	2	0.20	0.17613	0.413E-01	1.0068
3	0.40	0.33589	0.755E-01	1.0064	4	0.60	0.47778	0.103E+00	1.0056
5	0.80	0.60096	0.124E+00	1.0048	6	1.00	0.70473	0.140E+00	1.0040

7	1.20	0.78915	0.152E+00	1.0031	8	1.40	0.85516	0.159E+00	1.0024
9	1.60	0.90457	0.164E+00	1.0017	10	1.80	0.93986	0.167E+00	1.0012
11	2.00	0.96383	0.168E+00	1.0008	12	2.20	0.97930	0.169E+00	1.0005
13	2.40	0.98875	0.169E+00	1.0003	14	2.60	0.99421	0.169E+00	1.0002
15	2.80	0.99719	0.169E+00	1.0001	16	3.00	0.99872	0.169E+00	1.0001
17	3.20	0.99946	0.169E+00	1.0000	18	3.40	0.99979	0.169E+00	1.0000
19	3.60	0.99994	0.169E+00	1.0000	20	3.80	1.00000	0.169E+00	1.0000

i= 20 j= 25 (xp= 0.0390 yp= 0.0453 zp= 0.0623)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0073	2	0.20	0.17080	0.368E-01	1.0071
3	0.40	0.32612	0.677E-01	1.0066	4	0.60	0.46456	0.929E-01	1.0059
5	0.80	0.58549	0.113E+00	1.0051	6	1.00	0.68832	0.128E+00	1.0042
7	1.20	0.77306	0.139E+00	1.0034	8	1.40	0.84045	0.146E+00	1.0026
9	1.60	0.89197	0.151E+00	1.0020	10	1.80	0.92974	0.154E+00	1.0014
11	2.00	0.95620	0.156E+00	1.0010	12	2.20	0.97389	0.156E+00	1.0007
13	2.40	0.98515	0.157E+00	1.0004	14	2.60	0.99196	0.157E+00	1.0003
15	2.80	0.99587	0.157E+00	1.0002	16	3.00	0.99800	0.157E+00	1.0001
17	3.20	0.99910	0.157E+00	1.0000	18	3.40	0.99963	0.157E+00	1.0000
19	3.60	0.99988	0.157E+00	1.0000	20	3.80	1.00000	0.157E+00	1.0000

i= 20 j= 26 (xp= 0.0390 yp= 0.0374 zp= 0.0648)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0074	2	0.20	0.16640	0.314E-01	1.0072
3	0.40	0.31802	0.581E-01	1.0068	4	0.60	0.45354	0.799E-01	1.0061
5	0.80	0.57249	0.973E-01	1.0053	6	1.00	0.67437	0.111E+00	1.0045
7	1.20	0.75917	0.120E+00	1.0036	8	1.40	0.82751	0.127E+00	1.0029
9	1.60	0.88066	0.132E+00	1.0022	10	1.80	0.92042	0.135E+00	1.0016
11	2.00	0.94897	0.136E+00	1.0011	12	2.20	0.96859	0.137E+00	1.0008
13	2.40	0.98149	0.138E+00	1.0005	14	2.60	0.98958	0.138E+00	1.0003
15	2.80	0.99441	0.138E+00	1.0002	16	3.00	0.99717	0.138E+00	1.0001
17	3.20	0.99866	0.138E+00	1.0001	18	3.40	0.99943	0.138E+00	1.0000
19	3.60	0.99981	0.138E+00	1.0000	20	3.80	1.00000	0.138E+00	1.0000

i= 20 j= 27 (xp= 0.0390 yp= 0.0297 zp= 0.0666)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0076	2	0.20	0.16294	0.256E-01	1.0074
3	0.40	0.31172	0.473E-01	1.0069	4	0.60	0.44503	0.653E-01	1.0063
5	0.80	0.56249	0.796E-01	1.0055	6	1.00	0.66365	0.908E-01	1.0046
7	1.20	0.74847	0.990E-01	1.0038	8	1.40	0.81748	0.105E+00	1.0030
9	1.60	0.87179	0.109E+00	1.0023	10	1.80	0.91301	0.111E+00	1.0017
11	2.00	0.94310	0.113E+00	1.0012	12	2.20	0.96420	0.114E+00	1.0009
13	2.40	0.97837	0.114E+00	1.0006	14	2.60	0.98749	0.114E+00	1.0004
15	2.80	0.99310	0.114E+00	1.0002	16	3.00	0.99639	0.114E+00	1.0001
17	3.20	0.99823	0.114E+00	1.0001	18	3.40	0.99922	0.114E+00	1.0000
19	3.60	0.99974	0.114E+00	1.0000	20	3.80	1.00000	0.114E+00	1.0000

i= 20 j= 28 (xp= 0.0390 yp= 0.0220 zp= 0.0679)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te

1	0.00	0.00000	0.000E+00	1.0077	2	0.20	0.16010	0.194E-01	1.0075
3	0.40	0.30652	0.359E-01	1.0070	4	0.60	0.43796	0.497E-01	1.0064
5	0.80	0.55413	0.608E-01	1.0056	6	1.00	0.65460	0.695E-01	1.0048
7	1.20	0.73936	0.759E-01	1.0040	8	1.40	0.80886	0.806E-01	1.0032
9	1.60	0.86407	0.837E-01	1.0025	10	1.80	0.90647	0.857E-01	1.0019
11	2.00	0.93785	0.870E-01	1.0013	12	2.20	0.96020	0.876E-01	1.0010
13	2.40	0.97549	0.880E-01	1.0006	14	2.60	0.98552	0.881E-01	1.0004
15	2.80	0.99183	0.882E-01	1.0003	16	3.00	0.99562	0.882E-01	1.0002
17	3.20	0.99780	0.881E-01	1.0001	18	3.40	0.99901	0.881E-01	1.0000
19	3.60	0.99966	0.881E-01	1.0000	20	3.80	1.00000	0.880E-01	1.0000

i= 20 j= 29 (xp= 0.0390 yp= 0.0146 zp= 0.0686)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0077	2	0.20	0.15805	0.130E-01	1.0075
3	0.40	0.30278	0.241E-01	1.0071	4	0.60	0.43290	0.333E-01	1.0065
5	0.80	0.54818	0.408E-01	1.0057	6	1.00	0.64820	0.466E-01	1.0049
7	1.20	0.73293	0.510E-01	1.0041	8	1.40	0.80279	0.541E-01	1.0033
9	1.60	0.85865	0.563E-01	1.0026	10	1.80	0.90187	0.577E-01	1.0019
11	2.00	0.93415	0.586E-01	1.0014	12	2.20	0.95737	0.591E-01	1.0010
13	2.40	0.97344	0.593E-01	1.0007	14	2.60	0.98411	0.594E-01	1.0005
15	2.80	0.99090	0.595E-01	1.0003	16	3.00	0.99505	0.595E-01	1.0002
17	3.20	0.99748	0.595E-01	1.0001	18	3.40	0.99885	0.594E-01	1.0001
19	3.60	0.99959	0.594E-01	1.0000	20	3.80	1.00000	0.594E-01	1.0000

i= 20 j= 30 (xp= 0.0390 yp= 0.0073 zp= 0.0691)

k	zeta	u/ue	v/vinf	t/te	k	zeta	u/ue	v/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0078	2	0.20	0.15664	0.649E-02	1.0076
3	0.40	0.30015	0.121E-01	1.0072	4	0.60	0.42926	0.167E-01	1.0065
5	0.80	0.54377	0.205E-01	1.0058	6	1.00	0.64333	0.235E-01	1.0050
7	1.20	0.72792	0.257E-01	1.0041	8	1.40	0.79792	0.273E-01	1.0033
9	1.60	0.85418	0.285E-01	1.0026	10	1.80	0.89798	0.292E-01	1.0020
11	2.00	0.93094	0.297E-01	1.0015	12	2.20	0.95485	0.299E-01	1.0011
13	2.40	0.97156	0.301E-01	1.0007	14	2.60	0.98278	0.301E-01	1.0005
15	2.80	0.99003	0.302E-01	1.0003	16	3.00	0.99451	0.302E-01	1.0002
17	3.20	0.99717	0.302E-01	1.0001	18	3.40	0.99869	0.301E-01	1.0001
19	3.60	0.99953	0.301E-01	1.0000	20	3.80	1.00000	0.301E-01	1.0000

i= 20 j= 31 (xp= 0.0390 yp= 0.0000 zp= 0.0692)

k	zeta	u/ue	vy/vinf	t/te	k	zeta	u/ue	vy/vinf	t/te
1	0.00	0.00000	0.000E+00	1.0078	2	0.20	0.15777	-0.601E-01	1.0076
3	0.40	0.30243	-0.111E+00	1.0072	4	0.60	0.43267	-0.153E+00	1.0065
5	0.80	0.54821	-0.186E+00	1.0058	6	1.00	0.64856	-0.213E+00	1.0049
7	1.20	0.73363	-0.233E+00	1.0041	8	1.40	0.80375	-0.248E+00	1.0033
9	1.60	0.85976	-0.259E+00	1.0026	10	1.80	0.90300	-0.267E+00	1.0019
11	2.00	0.93519	-0.273E+00	1.0014	12	2.20	0.95825	-0.277E+00	1.0010
13	2.40	0.97412	-0.281E+00	1.0007	14	2.60	0.98460	-0.283E+00	1.0005
15	2.80	0.99124	-0.285E+00	1.0003	16	3.00	0.99526	-0.286E+00	1.0002
17	3.20	0.99760	-0.287E+00	1.0001	18	3.40	0.99890	-0.287E+00	1.0000
19	3.60	0.99961	-0.288E+00	1.0000	20	3.80	1.00000	-0.288E+00	1.0000

******* boundary-layer parameters**********

i	j	xpd blth	ypd dspth	zpd thmom	cfx twall	cfy
1	1	0.400000E-02	-0.148836E-07	-0.232644E-01	0.127639E-01	0.000000E+00
		0.243324E-03	0.656754E-04	0.277774E-04	0.293187E+03	
1	2	0.400000E-02	0.243383E-02	-0.231564E-01	0.127648E-01	-0.711058E-04
		0.243519E-03	0.657272E-04	0.277994E-04	0.293187E+03	
1	3	0.400000E-02	0.485302E-02	-0.228317E-01	0.127702E-01	-0.132842E-03
		0.244129E-03	0.658921E-04	0.278693E-04	0.293187E+03	
1	4	0.400000E-02	0.724221E-02	-0.222893E-01	0.127795E-01	-0.159033E-03
		0.245175E-03	0.661735E-04	0.279888E-04	0.293187E+03	
1	5	0.400000E-02	0.958455E-02	-0.215273E-01	0.127888E-01	-0.153318E-03
		0.246589E-03	0.665546E-04	0.281505E-04	0.293187E+03	
1	6	0.400000E-02	0.118611E-01	-0.205440E-01	0.127959E-01	-0.109954E-03
		0.248352E-03	0.670294E-04	0.283520E-04	0.293187E+03	
1	7	0.400000E-02	0.140503E-01	-0.193386E-01	0.127901E-01	0.785117E-05
		0.250346E-03	0.675655E-04	0.285795E-04	0.293187E+03	
1	8	0.400000E-02	0.161280E-01	-0.179120E-01	0.127732E-01	0.167305E-03
		0.252521E-03	0.681520E-04	0.288282E-04	0.293187E+03	
1	9	0.400000E-02	0.180669E-01	-0.162675E-01	0.127236E-01	0.434733E-03
		0.254681E-03	0.687331E-04	0.290747E-04	0.293187E+03	
1	10	0.400000E-02	0.198374E-01	-0.144127E-01	0.126465E-01	0.740577E-03
		0.256655E-03	0.692654E-04	0.293002E-04	0.293187E+03	
1	11	0.400000E-02	0.214084E-01	-0.123601E-01	0.125320E-01	0.110869E-02
		0.258248E-03	0.696960E-04	0.294823E-04	0.293187E+03	
1	12	0.400000E-02	0.227483E-01	-0.101282E-01	0.123596E-01	0.157034E-02
		0.259278E-03	0.699745E-04	0.295997E-04	0.293187E+03	
1	13	0.400000E-02	0.238273E-01	-0.774197E-02	0.121726E-01	0.197870E-02
		0.259439E-03	0.700190E-04	0.296178E-04	0.293187E+03	
1	14	0.400000E-02	0.246190E-01	-0.523294E-02	0.119242E-01	0.246218E-02
		0.258717E-03	0.698272E-04	0.295351E-04	0.293187E+03	
1	15	0.400000E-02	0.251028E-01	-0.263842E-02	0.116731E-01	0.287860E-02
		0.256801E-03	0.693129E-04	0.293156E-04	0.293187E+03	
1	16	0.400000E-02	0.252644E-01	0.000000E+00	0.113492E-01	0.320311E-02
		0.254517E-03	0.687023E-04	0.290546E-04	0.293187E+03	
1	17	0.400000E-02	0.251809E-01	0.264661E-02	0.110247E-01	0.344518E-02
		0.251526E-03	0.679001E-04	0.287120E-04	0.293187E+03	
1	18	0.400000E-02	0.247446E-01	0.525963E-02	0.108731E-01	0.377420E-02
		0.245833E-03	0.663697E-04	0.280607E-04	0.293187E+03	
1	19	0.400000E-02	0.238921E-01	0.776299E-02	0.108953E-01	0.403497E-02
		0.238236E-03	0.643274E-04	0.271925E-04	0.293187E+03	
1	20	0.400000E-02	0.226377E-01	0.100789E-01	0.110241E-01	0.421305E-02
		0.229758E-03	0.620466E-04	0.262235E-04	0.293187E+03	
1	21	0.400000E-02	0.210391E-01	0.121469E-01	0.111967E-01	0.429182E-02
		0.221211E-03	0.597466E-04	0.252466E-04	0.293187E+03	
1	22	0.400000E-02	0.191770E-01	0.139329E-01	0.113599E-01	0.424265E-02
		0.213374E-03	0.576379E-04	0.243511E-04	0.293187E+03	
1	23	0.400000E-02	0.171360E-01	0.154293E-01	0.114852E-01	0.406942E-02
		0.206555E-03	0.558018E-04	0.235716E-04	0.293187E+03	
1	24	0.400000E-02	0.149911E-01	0.166493E-01	0.115547E-01	0.376950E-02
		0.200909E-03	0.542825E-04	0.229265E-04	0.293187E+03	
1	25	0.400000E-02	0.128007E-01	0.176186E-01	0.115756E-01	0.338650E-02
		0.196336E-03	0.530516E-04	0.224039E-04	0.293187E+03	
1	26	0.400000E-02	0.106047E-01	0.183678E-01	0.115641E-01	0.291033E-02
		0.192768E-03	0.520912E-04	0.219962E-04	0.293187E+03	
1	27	0.400000E-02	0.842718E-02	0.189277E-01	0.115302E-01	0.237901E-02
		0.190066E-03	0.513642E-04	0.216876E-04	0.293187E+03	
1	28	0.400000E-02	0.627944E-02	0.193261E-01	0.114834E-01	0.182863E-02

		0.188058E-03	0.508233E-04	0.214579E-04	0.293187E+03	
1	29	0.400000E-02	0.416331E-02	0.195868E-01	0.114402E-01	0.123912E-02
		0.186722E-03	0.504638E-04	0.213054E-04	0.293187E+03	
1	30	0.400000E-02	0.207367E-02	0.197295E-01	0.114082E-01	0.639749E-03
		0.185972E-03	0.502617E-04	0.212196E-04	0.293187E+03	
1	31	0.400000E-02	0.620667E-08	0.197717E-01	0.113937E-01	0.000000E+00
		0.185755E-03	0.502032E-04	0.211948E-04	0.293187E+03	
2	1	0.450000E-02	-0.158292E-07	-0.247425E-01	0.115377E-01	0.000000E+00
		0.246466E-03	0.672225E-04	0.282609E-04	0.293140E+03	
2	2	0.450000E-02	0.258853E-02	-0.246283E-01	0.115249E-01	-0.223210E-03
		0.247153E-03	0.674008E-04	0.283468E-04	0.293140E+03	
2	3	0.450000E-02	0.516194E-02	-0.242851E-01	0.115233E-01	-0.424816E-03
		0.247643E-03	0.675622E-04	0.284046E-04	0.293140E+03	
2	4	0.450000E-02	0.770432E-02	-0.237115E-01	0.115153E-01	-0.581884E-03
		0.248575E-03	0.678681E-04	0.285154E-04	0.293141E+03	
2	5	0.450000E-02	0.101981E-01	-0.229053E-01	0.114959E-01	-0.727708E-03
		0.249909E-03	0.683129E-04	0.286771E-04	0.293141E+03	
2	6	0.450000E-02	0.126233E-01	-0.218643E-01	0.114684E-01	-0.786488E-03
		0.251440E-03	0.688416E-04	0.288627E-04	0.293142E+03	
2	7	0.450000E-02	0.149575E-01	-0.205872E-01	0.114117E-01	-0.771543E-03
		0.253310E-03	0.695013E-04	0.290928E-04	0.293143E+03	
2	8	0.450000E-02	0.171745E-01	-0.190742E-01	0.113260E-01	-0.707375E-03
		0.255260E-03	0.702413E-04	0.293375E-04	0.293143E+03	
2	9	0.450000E-02	0.192453E-01	-0.173286E-01	0.111874E-01	-0.454452E-03
		0.257302E-03	0.710576E-04	0.295992E-04	0.293145E+03	
2	10	0.450000E-02	0.211382E-01	-0.153578E-01	0.109915E-01	-0.196654E-03
		0.259473E-03	0.719719E-04	0.298863E-04	0.293146E+03	
2	11	0.450000E-02	0.228193E-01	-0.131748E-01	0.107619E-01	0.252666E-03
		0.260875E-03	0.726969E-04	0.300841E-04	0.293145E+03	
2	12	0.450000E-02	0.242544E-01	-0.107988E-01	0.104638E-01	0.795237E-03
		0.262374E-03	0.734375E-04	0.302969E-04	0.293145E+03	
2	13	0.450000E-02	0.254109E-01	-0.825654E-02	0.101431E-01	0.130565E-02
		0.263025E-03	0.739572E-04	0.304159E-04	0.293145E+03	
2	14	0.450000E-02	0.262600E-01	-0.558176E-02	0.978616E-02	0.197000E-02
		0.262736E-03	0.741536E-04	0.304244E-04	0.293144E+03	
2	15	0.450000E-02	0.267791E-01	-0.281461E-02	0.942431E-02	0.249677E-02
		0.262188E-03	0.742173E-04	0.304054E-04	0.293142E+03	
2	16	0.450000E-02	0.269524E-01	0.000000E+00	0.903874E-02	0.290953E-02
		0.261112E-03	0.740491E-04	0.303122E-04	0.293140E+03	
2	17	0.450000E-02	0.268663E-01	0.282374E-02	0.868607E-02	0.333422E-02
		0.258284E-03	0.733430E-04	0.300122E-04	0.293138E+03	
2	18	0.450000E-02	0.264059E-01	0.561275E-02	0.844846E-02	0.383325E-02
		0.253510E-03	0.721114E-04	0.295051E-04	0.293134E+03	
2	19	0.450000E-02	0.255010E-01	0.828576E-02	0.835744E-02	0.407170E-02
		0.248025E-03	0.707116E-04	0.289258E-04	0.293129E+03	
2	20	0.450000E-02	0.241656E-01	0.107592E-01	0.837350E-02	0.419826E-02
		0.241278E-03	0.689546E-04	0.282001E-04	0.293125E+03	
2	21	0.450000E-02	0.224608E-01	0.129677E-01	0.839884E-02	0.418835E-02
		0.234996E-03	0.672027E-04	0.274542E-04	0.293120E+03	
2	22	0.450000E-02	0.204730E-01	0.148745E-01	0.840681E-02	0.399867E-02
		0.229704E-03	0.655810E-04	0.267385E-04	0.293115E+03	
2	23	0.450000E-02	0.182931E-01	0.164712E-01	0.837822E-02	0.375362E-02
		0.224650E-03	0.641372E-04	0.260953E-04	0.293111E+03	
2	24	0.450000E-02	0.160021E-01	0.177721E-01	0.831714E-02	0.336072E-02
		0.220085E-03	0.629207E-04	0.255385E-04	0.293107E+03	
2	25	0.450000E-02	0.136626E-01	0.188049E-01	0.822000E-02	0.296772E-02
		0.216267E-03	0.619448E-04	0.250852E-04	0.293104E+03	
2	26	0.450000E-02	0.113174E-01	0.196023E-01	0.811062E-02	0.249396E-02
		0.213338E-03	0.612228E-04	0.247413E-04	0.293102E+03	
2	27	0.450000E-02	0.899269E-02	0.201979E-01	0.802429E-02	0.201894E-02

		0.210802E-03	0.605854E-04	0.244478E-04	0.293100E+03	
2	28	0.450000E-02	0.670025E-02	0.206212E-01	0.793836E-02	0.153754E-02
		0.208983E-03	0.601366E-04	0.242390E-04	0.293099E+03	
2	29	0.450000E-02	0.444202E-02	0.208980E-01	0.785215E-02	0.981609E-03
		0.207918E-03	0.599088E-04	0.241186E-04	0.293098E+03	
2	30	0.450000E-02	0.221240E-02	0.210494E-01	0.779608E-02	0.468315E-03
		0.207431E-03	0.597999E-04	0.240619E-04	0.293097E+03	
2	31	0.450000E-02	0.662182E-08	0.210942E-01	0.796545E-02	0.000000E+00
		0.200398E-03	0.582322E-04	0.233594E-04	0.293096E+03	

***** For brevity, the results for i=3,4,..,19 are deleted. *****

20	1	0.390000E-01	-0.475658E-07	-0.743496E-01	0.301178E-02	0.000000E+00
		0.348036E-03	0.989335E-04	0.410106E-04	0.292810E+03	
20	2	0.390000E-01	0.778423E-02	-0.740622E-01	0.301940E-02	-0.107584E-03
		0.347447E-03	0.987619E-04	0.409350E-04	0.292810E+03	
20	3	0.390000E-01	0.155578E-01	-0.731938E-01	0.303434E-02	-0.213710E-03
		0.347127E-03	0.985706E-04	0.408478E-04	0.292812E+03	
20	4	0.390000E-01	0.233056E-01	-0.717272E-01	0.305432E-02	-0.313855E-03
		0.346870E-03	0.983939E-04	0.407599E-04	0.292816E+03	
20	5	0.390000E-01	0.310040E-01	-0.696362E-01	0.308457E-02	-0.411398E-03
		0.346332E-03	0.981282E-04	0.406343E-04	0.292820E+03	
20	6	0.390000E-01	0.386171E-01	-0.668870E-01	0.312228E-02	-0.497832E-03
		0.345387E-03	0.978211E-04	0.404831E-04	0.292827E+03	
20	7	0.390000E-01	0.460921E-01	-0.634403E-01	0.316423E-02	-0.568826E-03
		0.344007E-03	0.975555E-04	0.403397E-04	0.292835E+03	
20	8	0.390000E-01	0.533553E-01	-0.592571E-01	0.320640E-02	-0.625028E-03
		0.341547E-03	0.973436E-04	0.401925E-04	0.292844E+03	
20	9	0.390000E-01	0.603084E-01	-0.543020E-01	0.325332E-02	-0.653002E-03
		0.339282E-03	0.971207E-04	0.400332E-04	0.292853E+03	
20	10	0.390000E-01	0.668283E-01	-0.485536E-01	0.329164E-02	-0.647021E-03
		0.339541E-03	0.971503E-04	0.399618E-04	0.292864E+03	
20	11	0.390000E-01	0.727661E-01	-0.420116E-01	0.331888E-02	-0.595260E-03
		0.339173E-03	0.972941E-04	0.399053E-04	0.292874E+03	
20	12	0.390000E-01	0.779553E-01	-0.347080E-01	0.333737E-02	-0.490430E-03
		0.338350E-03	0.975331E-04	0.398831E-04	0.292883E+03	
20	13	0.390000E-01	0.822223E-01	-0.267157E-01	0.334029E-02	-0.348348E-03
		0.337795E-03	0.979613E-04	0.399518E-04	0.292890E+03	
20	14	0.390000E-01	0.854039E-01	-0.181532E-01	0.332901E-02	-0.161793E-03
		0.337668E-03	0.984694E-04	0.400721E-04	0.292894E+03	
20	15	0.390000E-01	0.873683E-01	-0.918281E-02	0.329891E-02	0.278752E-04
		0.340113E-03	0.993108E-04	0.403840E-04	0.292896E+03	
20	16	0.390000E-01	0.880632E-01	0.000000E+00	0.323643E-02	0.185674E-03
		0.348206E-03	0.101100E-03	0.411843E-04	0.292896E+03	
20	17	0.390000E-01	0.879284E-01	0.924159E-02	0.320588E-02	0.282908E-03
		0.347564E-03	0.101018E-03	0.411158E-04	0.292894E+03	
20	18	0.390000E-01	0.868571E-01	0.184620E-01	0.319930E-02	0.482515E-03
		0.337321E-03	0.991544E-04	0.401932E-04	0.292890E+03	
20	19	0.390000E-01	0.843374E-01	0.274028E-01	0.317840E-02	0.708460E-03
		0.332856E-03	0.981473E-04	0.397493E-04	0.292883E+03	
20	20	0.390000E-01	0.802721E-01	0.357394E-01	0.314286E-02	0.920828E-03
		0.335064E-03	0.980269E-04	0.397818E-04	0.292871E+03	
20	21	0.390000E-01	0.748054E-01	0.431888E-01	0.309814E-02	0.105152E-02
		0.339081E-03	0.986131E-04	0.401626E-04	0.292856E+03	
20	22	0.390000E-01	0.682358E-01	0.495762E-01	0.303858E-02	0.110442E-02
		0.343268E-03	0.997929E-04	0.408019E-04	0.292840E+03	
20	23	0.390000E-01	0.609184E-01	0.548512E-01	0.296574E-02	0.107767E-02
		0.351793E-03	0.101412E-03	0.416149E-04	0.292825E+03	
20	24	0.390000E-01	0.531855E-01	0.590685E-01	0.288423E-02	0.100140E-02
		0.358810E-03	0.103224E-03	0.424730E-04	0.292811E+03	

20	25	0.390000E-01	0.452934E-01	0.623409E-01	0.279986E-02	0.878781E-03
		0.366186E-03	0.105112E-03	0.433254E-04	0.292800E+03	
20	26	0.390000E-01	0.374166E-01	0.648073E-01	0.272538E-02	0.743972E-03
		0.372317E-03	0.106840E-03	0.441048E-04	0.292792E+03	
20	27	0.390000E-01	0.296542E-01	0.666043E-01	0.266278E-02	0.600378E-03
		0.377578E-03	0.108157E-03	0.446766E-04	0.292785E+03	
20	28	0.390000E-01	0.220461E-01	0.678509E-01	0.260939E-02	0.452672E-03
		0.380992E-03	0.109387E-03	0.452110E-04	0.292782E+03	
20	29	0.390000E-01	0.145915E-01	0.686478E-01	0.256923E-02	0.302266E-03
		0.382594E-03	0.110258E-03	0.455696E-04	0.292780E+03	
20	30	0.390000E-01	0.726007E-02	0.690744E-01	0.254170E-02	0.151098E-03
		0.386313E-03	0.111069E-03	0.459390E-04	0.292777E+03	
20	31	0.390000E-01	0.217225E-07	0.691985E-01	0.255738E-02	0.000000E+00
		0.379519E-03	0.109577E-03	0.451870E-04	0.292781E+03	

1.8 FORTRAN Listing of 3DBLC

```
c
c       comblck
c
        parameter (imaxf=100,jmaxf=51,kmaxf=51)

        common/pi/pi
        common/writ/iw,ini,jni
        common/ks/mks
        common/term/kterm,jmaxt
        common/jm/jmax1
        common/sep/ksep
        common/stagso/ksymstg
        common/pte/kcpgivn
        common/point/kpoint
        common/bcvel/ue(imaxf,jmaxf),ve(imaxf,jmaxf)
        common/nonorth/costh(imaxf,jmaxf)
        common/rw/roww(imaxf,jmaxf)
        common/body/kbody
        common/end/iend
        common/compr/rminf,vinf,gamma,rr,tinf,ss,pinf,cp,pr
        common/stag/xps,zps
        common/com/inc
        common/compr1/kaw,krow
        common/compr2/acom(kmaxf),bcom(kmaxf),ccom(kmaxf),dcom(kmaxf)
        common/compr3/rmyued(imaxf,jmaxf),roed(imaxf,jmaxf)
        common/compr4/rmyueh(jmaxf),roeh(jmaxf)
        common/compr5/roerob(jmaxf,kmaxf),bcb(jmaxf,kmaxf)
        common/compr6/roero(jmaxf,kmaxf),bc(jmaxf,kmaxf),td(jmaxf,kmaxf)
        common/compr7/pe(imaxf,jmaxf),te(imaxf,jmaxf),twall(imaxf,jmaxf)
        common/star/astar,bstar,cstar,thetar
        common/zetard/zetae,dzetas,zeta(kmaxf),dzeta(kmaxf)
        common/ygrd/yd(jmaxf),dy(jmaxf)
        common/xx/xd(imaxf)
        common/dxx/dx,dxh
        common/nuro/rmyuinf,rnuinf,roinf
        common/ss/s1(imaxf,jmaxf),s1h(imaxf,jmaxf)
        common/mm/m1(jmaxf),m2(jmaxf),m3(jmaxf),m4(jmaxf),m5(jmaxf),
     & m6(jmaxf),m7(jmaxf),m8(jmaxf),m9(jmaxf),m10(jmaxf),
     & m11(jmaxf),m12(jmaxf),m13(jmaxf)
        common/ukmaxm1/ukmax1
        common/str/xpd(imaxf,jmaxf),ypd(imaxf,jmaxf),zpd(imaxf,jmaxf),
     & cavd(imaxf,jmaxf),h1(imaxf,jmaxf),h2(imaxf,jmaxf)
     &,duedsd(imaxf,jmaxf),duedyd(imaxf,jmaxf),cpd(imaxf,jmaxf)
        common/ijk/i,j,k,imax,jmax,kmax
        common/hh/h(3,jmaxf,kmaxf),hs(3,jmaxf,kmaxf)
        common/hhb/hb(3,jmaxf,kmaxf),hsb(3,jmaxf,kmaxf)
        common/hhn/hn(2,jmaxf,kmaxf),hsn(2,jmaxf,kmaxf)
        common/compr4/hsp(2,jmaxf,kmaxf)
        common/abcd/ai(4,kmaxf),bi(4,kmaxf),ci(4,kmaxf),as(4,kmaxf),
     & e(4,kmaxf),es(4,kmaxf),ds(4,kmaxf),di(2,kmaxf)
        common/syt/save(4,kmaxf),b1(jmaxf,kmaxf),b2(jmaxf,kmaxf),
     & b3(jmaxf,kmaxf),b4(jmaxf,kmaxf)
        common/save1/saveh(kmaxf),savehs(kmaxf)
         common/bl/cfx(imaxf,jmaxf),cfy(imaxf,jmaxf),blth(imaxf,jmaxf)
     &,dspth(imaxf,jmaxf),thmom(imaxf,jmaxf),vmax(imaxf,jmaxf)
     &,xki(imaxf,jmaxf),qw(imaxf,jmaxf),zact(jmaxf,kmaxf)
         real m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13
```

```
c#####################################################################

        program blmain

c#####################################################################

       include 'comblck'
       dimension kmaxj(jmaxf)

       pi=acos(-1.)

       call input

       if(imax.gt.imaxf)write(6,*)'imax is greater than imaxf,
     &change imaxf greater or equal to imax'
       if(jmax.gt.jmaxf)write(6,*)'jmax is greater than jmaxf,
     &change jmaxf greater or equal to jmax'
       if(kmax.gt.kmaxf)write(6,*)'kmax is greater than kmaxf,
     &change kmaxf greater or equal to kmax'
       if(imax.gt.imaxf.or.jmax.gt.jmaxf.or.kmax.gt.kmaxf)stop

       if(ksymstg.eq.1)cstar=1.

c      other free-stream conditions are calculated

       cp=gamma*rr/(gamma-1.)

       if(mks.eq.1)rmyuinf=47.88*(2.28d-8*(1.8*tinf)**1.5)
     &/(1.8*tinf+198.6)
       if(mks.eq.0)rmyuinf=(2.28d-8*(tinf)**1.5)/(tinf+198.6)

       roinf=pinf/(rr*tinf)
       rnuinf=rmyuinf/roinf
       ss=sqrt(gamma*rr*tinf)
       vinf=rminf*ss

       rewind 1
       rewind iw

       do 351 i=1,imax
       do 351 j=1,jmax
       cavd(i,j)=sqrt(ue(i,j)**2+ve(i,j)**2+2.*ue(i,j)*ve(i,j)
     &*costh(i,j))
       slh(i,j)=0.5*(sl(i,j)+sl(i-1,j))
 351   continue

c      dimensionalize velocity components

       astar=astar*vinf
       bstar=bstar*vinf

       do 29 i=1,imax
       do 29 j=1,jmax
       cavd(i,j)=cavd(i,j)*vinf
       ue(i,j)=ue(i,j)*vinf
       ve(i,j)=ve(i,j)*vinf
 29    continue

c      other boundary-layer edge conditions  are calculated
```

```
      do 99 i=1,imax
      do 99 j=1,jmax
      te(i,j)=tinf*(1.+0.5*(gamma-1.)*rminf**2*(1.-(cavd(i,j)
     &/vinf)**2))
      if(kcpgivn.eq.1)pe(i,j)=pinf+0.5*cpd(i,j)*roinf*vinf**2
      if(kcpgivn.eq.0)pe(i,j)=pinf*(te(i,j)/tinf)**(gamma/(gamma-1.))
 252  if(mks.eq.1)rmyued(i,j)=47.88*(2.28d-8*(1.8*te(i,j))**1.5)
     & /(1.8*te(i,j)+198.6)
      if(mks.eq.0)rmyued(i,j)=(2.28d-8*(te(i,j))**1.5)/(te(i,j)+198.6)
      roed(i,j)=pe(i,j)/(rr*te(i,j))
 99   continue

c     dy(j) is calculated

      do 4 j=1,jmax-1
      dy(j)=yd(j+1)-yd(j)
 4    continue


      i=1

      if(kpoint.eq.1)go to 45
      call stagpt
      if(ksymstg.eq.1)then
      call insym
      go to 1119
      endif

      if(kbody.eq.1)call inbub
      if(kbody.eq.0)call inbus
      go to 1119

 45   call coefcon
      j=1
      call conon
      do 400 k=1,kmax
      saveh(k)=h(2,1,k)
      savehs(k)=hs(2,1,k)
 400  continue
      do 500 k=1,kmax
      h(2,1,k)=0.
      hs(2,1,k)=0.
 500  continue
      kmaxj(1)=kmax

      do 1156 j=2,jmax-1
      call conoff
      kmaxj(j)=kmax
      if(iend.eq.1)then
      kterm=1
      jmaxt=j-1
      do 410 k=1,kmax
      h(2,1,k)=saveh(k)
      hs(2,1,k)=savehs(k)
 410  continue
      go to 1118
      endif
 1156 continue

      j=jmax
```

```
      call conon
      kmaxj(j)=kmax
      if(iend.eq.1)then
      kterm=1
      jmaxt=j-1
      do 420 k=1,kmax
      h(2,1,k)=saveh(k)
      hs(2,1,k)=savehs(k)
 420  continue
      endif

 1118 do 520 j=1,jmaxt
      do 520 k=kmaxj(j),kmax
      h(1,j,k)=1.0
      h(2,j,k)=h(2,j,kmaxj(j))
      hs(1,j,k)=hs(1,j,k-1)+(h(1,j,k)+h(1,j,k-1))*dzeta(k-1)/2.
      hs(2,j,k)=hs(2,j,k-1)+(h(2,j,k)+h(2,j,k-1))*dzeta(k-1)/2.
      h(3,j,k)=1.0
      roero(j,k)=1.0
      bc(j,k)=1.0
 520  continue
      do 422 k=1,kmaxj(j)
      h(2,1,k)=saveh(k)
      hs(2,1,k)=savehs(k)
 422  continue
      do 423 k=kmaxj(j)+1,kmax
      h(2,1,k)=saveh(kmaxj(j))
      hs(2,j,k)=hs(2,j,k-1)+(h(2,j,k)+h(2,j,k-1))*dzeta(k-1)/2.
 423  continue

 1119 if(kpoint.eq.1.and.kbody.eq.0)then
      call inpos
      do 123 j=1,jmax
      ue(1,j)=sqrt(ue(1,j)**2+ve(1,j)**2)
      ve(1,j)=0.
 123  continue
      endif

      do 2255 j=1,jmax
      do 2256 k=1,kmax
      hsp(1,j,k)=hs(1,j,k)
 2256 continue
 2255 continue


      do 270 j=1,jmax
      call blpara
 270  continue


c*********************************************************************

c     to march away from i=1

c*********************************************************************

      if(kterm.eq.1)jmax1=jmaxt
      if(kterm.eq.0)jmax1=jmax

 1500 do 1000 i=2,imax
```

```
      dx=xd(i)-xd(i-1)
      dxh=dx/2.
      x=xd(i)
      write(6,611)i,x,dx
 611  format('*** i=',i4,5x,' x=',f10.6,' dx=',f10.6)

c-----------------------------------------------------------------------

      if(kbody.eq.1)call coefbody
      if(kbody.eq.0)call coefstrm

 7000 do 60 j=1,jmax1

      call predict

 60   continue

      do 70 j=1,jmax1

      call correct

 70   continue

c-----------------------------------------------------------------------
c
c to increase zetae so that u(kmax-1) is greater than ukmax1(given)
c
c   ( check point is only on the leeward line of symmetry )
c
c----------------------------------------------------------------------

      write(6,*)'jmax1=',jmax1,'hn(1,jmax1,kmax-1)=',hn(1,jmax1,kmax-1)
      if(hn(1,jmax1,kmax-1).gt.ukmax1)go to 7100
      kmax=kmax+1
      write(6,*)' kmax=',kmax

      if(kmax.eq.kmaxf)then
      write(6,*)' kmax was increased to kmaxf'
      go to 2100
      endif

      do 7300 ij=1,jmax1
      h(1,ij,kmax)=1.
      h(2,ij,kmax)=0.
      if(kbody.eq.1)h(2,ij,kmax)=ve(i-1,ij)/vinf
      if(kbody.eq.1.and.ij.eq.1)h(2,ij,kmax)=ve(i-1,ij+1)/(vinf*dy(ij))
      if(kbody.eq.1.and.ij.eq.jmax)h(2,ij,kmax)=-ve(i-1,ij-1)
     &/(vinf*dy(ij-1))
      h(3,ij,kmax)=1.
      hs(1,ij,kmax)=hs(1,ij,kmax-1)+(h(1,ij,kmax)+h(1,ij,kmax-1))
     &  *dzeta(kmax-1)/2.
      hs(2,ij,kmax)=hs(2,ij,kmax-1)+(h(2,ij,kmax)+h(2,ij,kmax-1))
     &  *dzeta(kmax-1)/2.
      hsp(1,ij,kmax)=hs(1,ij,kmax)
      hsp(2,ij,kmax)=hs(2,ij,kmax)
      bc(ij,kmax)=1.
      roero(ij,kmax)=1.
```

```
 7300  continue
        go to 7000

c
c       check whether the zone of dependence principle is satisfied
c

 7100  do 370 j=1,jmax1
       if(j.le.2.or.j.ge.jmax-1)go to 370
       vuwall=abs(vinf*((dzeta(1)+dzeta(2))**2*hn(2,j,2)
     &-dzeta(1)**2*hn(2,j,3)))/(ue(i,j)*((dzeta(1)+dzeta(2))**2
     &*hn(1,j,2)-dzeta(1)**2*hn(1,j,3)))
       vuedge=abs(ve(i,j)/ue(i,j))
       if(vuwall.lt.vuedge)vuwall=vuedge

       if(h(2,j,2).lt.0)then
       ch=h2(i-1,j)*dy(j)/(s1(i,j)-s1(i-1,j))
       if(vuwall.gt.ch)write(6,*)' zone of dependence violated at i=',i
     & ,' j=',j,' vuwall=',vuwall,' ch=',ch
       endif
       if(h(2,j,2).ge.0)then
       ch=h2(i-1,j)*dy(j-1)/(s1(i,j)-s1(i-1,j))
       if(vuwall.gt.ch)write(6,*)' zone of dependence violated at i=',i
     & ,' j=',j,' vuwall=',vuwall,' ch=',ch
       endif

 370   continue

       do 376 j=1,jmax1
       do 5500 k=1,kmax
       do 5550 m=1,2
       h(m,j,k)=hn(m,j,k)
       hs(m,j,k)=hsn(m,j,k)
 5550  continue
 5500  continue
 376   continue

 371   if(inc.eq.1)then
       do 385 j=1,jmax
       do 385 k=1,kmax
       h(3,j,k)=1.0
       roero(j,k)=1.0
       bc(j,k)=1.0
 385   continue
       go to 575
       endif

       do 470 j=1,jmax1

       call correng

 470   continue

 575   do 570 j=1,jmax1

       call blpara

 570   continue

       call profile
```

```
c-----------------------------------------------------------------------
c
c stop computation if as follows
c
c-----------------------------------------------------------------------

       if(((dzeta(1)+dzeta(2))**2*h(1,jmax1,2)-dzeta(1)**2*h(1,jmax1,3))
     &    .lt.0)then
       write(6,*)' dudy is l.t. 0 at j=jmax1 '
       go to 2100
       endif

c
c     to find the first separation point
c

       do 800 j=1,jmax1
       if(((dzeta(1)+dzeta(2))**2*h(1,j,2)-dzeta(1)**2*h(1,j,3)).lt.0)
     &then
       write(6,*)' dudzeta wall is .lt. 0  at i,j=',i,j
       ksep=1
       go to 2100
c
c      if one wants to continue the calculations using the modified
c      procedure, use the following statement instead of 'go to 2100'
c      kterm=1
c      jmax1=j-1

       endif

 800   continue

       write(6,*)' xpd(i,jmax)=',xpd(i,jmax)

       do 9995 k=1,kmax
       do 9995 j=1,jmax1
       hsp(1,j,k)=hs(1,j,k)
       hsp(2,j,k)=hs(2,j,k)
 9995  continue

 1000  continue

c**********************************************************************

 2100  call output

       stop
       end
```

```
c######################################################################

         subroutine blpara

c######################################################################

        include 'comblck'

        bltk=0.995

        f2dotw=((dzeta(1)+dzeta(2))**2*h(1,j,2)-dzeta(1)**2*h(1,j,3))
     &/(dzeta(1)*(dzeta(1)+dzeta(2))**2-(dzeta(1)+dzeta(2))*dzeta(1)**2)
       g2dotw=((dzeta(1)+dzeta(2))**2*h(2,j,2)-dzeta(1)**2*h(2,j,3))
     &/(dzeta(1)*(dzeta(1)+dzeta(2))**2-(dzeta(1)+dzeta(2))*dzeta(1)**2)
       if(j.eq.1.or.j.eq.jmax)g2dotw=0
        if(kaw.eq.1)twall(i,j)=td(j,1)
        if(mks.eq.1)rmyuw=47.88*(2.28d-8*(1.8*td(j,1))**1.5)
     &  /(1.8*td(j,1)+198.6)
        if(mks.eq.0)rmyuw=(2.28d-8*(td(j,1))**1.5)/(td(j,1)+198.6)
        cfx(i,j)=2.*rmyuw*ue(i,j)*sqrt(ue(i,j)*roed(i,j)
     &/(rmyued(i,j)*s1(i,j)))*f2dotw/(roed(i,j)*roero(j,1)*cavd(i,j)**2)
        cfy(i,j)=2.*rmyuw*vinf*sqrt(ue(i,j)*roed(i,j)/(rmyued(i,j)
     &*s1(i,j)))*g2dotw/(roed(i,j)*roero(j,1)*cavd(i,j)**2)
c         a=1.
c         cfx(i,j)=2.*rnuinf*sqrt(ue(i,j)/(rnuinf*s1(i,j)))*f2dotw
c      &*cavd(i,j)*sqrt(vinf*a/rnuinf)/vinf**2
c         cfy(i,j)=2.*rnuinf*sqrt(ue(i,j)/(rnuinf*s1(i,j)))*g2dotw
c      &*sqrt(vinf*a/rnuinf)/vinf

        do 2650 k=1,kmax
        check=sqrt(ue(i,j)**2*h(1,j,k)**2+vinf**2*h(2,j,k)
     &**2+2.*ue(i,j)*vinf*h(1,j,k)*h(2,j,k)*costh(i,j))/cavd(i,j)
        if(j.eq.1.or.j.eq.jmax)check=h(1,j,k)
        if(check.ge.bltk)then
        check1=sqrt(ue(i,j)**2*h(1,j,k-1)**2+vinf**2
     &*h(2,j,k-1)**2+2.*ue(i,j)*vinf*h(1,j,k-1)*h(2,j,k-1)*costh(i,j))
     &/cavd(i,j)
        if(j.eq.1.or.j.eq.jmax)check1=h(1,j,k-1)
        kmm=k
        go to 2655
        endif
 2650   continue
 2655   zact(j,1)=0
        do 2660 k=1,kmax-1
        zact(j,k+1)=zact(j,k)+0.5*(roero(j,k)
     &+roero(j,k+1))*sqrt(rmyued(i,j)*s1(i,j)/(roed(i,j)*ue(i,j)))
     &*dzeta(k)
 2660   continue
        blth(i,j)=zact(j,kmm-1)+(zact(j,kmm)-zact(j,kmm-1))
     &  *(bltk-check1)/(check-check1)

        sinth=sqrt(1.-costh(i,j)**2)
        vedge=sqrt(ue(i,j)**2+ve(i,j)**2+2.*ue(i,j)*ve(i,j)*costh(i,j))
        gammae=asin(ve(i,j)*sinth/vedge)
        vmax(i,j)=0.
        do 2120 k=2,kmax-1
        vins=sqrt((ue(i,j)*h(1,j,k))**2+(vinf*h(2,j,k))**2+2.*ue(i,j)
     &*h(1,j,k)*vinf*h(2,j,k)*costh(i,j))
        gammai=asin(vinf*h(2,j,k)*sinth/vins)
```

```
      if(vmax(i,j).lt.abs(vins*sin(gammai-gammae)))vmax(i,j)=abs(vins
     &*sin(gammai-gammae))
 2120 continue
      if(j.eq.1.or.j.eq.jmax)vmax(i,j)=0.

      vk=sqrt((ue(i,j)*h(1,j,2))**2+(vinf*h(2,j,2))**2+2.*ue(i,j)
     &*h(1,j,2)*vinf*h(2,j,2)*costh(i,j))
      if(j.eq.1.or.j.eq.jmax)vk=ue(i,j)*h(1,j,2)
      dspth(i,j)=0.5*(2.-vk/(cavd(i,j)*roero(j,2)))*zact(j,2)
      thmom(i,j)=0.5*(vk/(cavd(i,j)*roero(j,2)))
     &*(1.-vk/cavd(i,j))*zact(j,2)
      do 2680 k=2,kmax-1
      vk=sqrt((ue(i,j)*h(1,j,k))**2+(vinf*h(2,j,k))**2+2.*ue(i,j)
     &*h(1,j,k)*vinf*h(2,j,k)*costh(i,j))
      if(j.eq.1.or.j.eq.jmax)vk=ue(i,j)*h(1,j,k)
      vkp1=sqrt((ue(i,j)*h(1,j,k+1))**2+(vinf*h(2,j,k+1))**2+2.
     &*ue(i,j)*h(1,j,k+1)*vinf*h(2,j,k+1)*costh(i,j))
      if(j.eq.1.or.j.eq.jmax)vkp1=ue(i,j)*h(1,j,k+1)
      dspth(i,j)=dspth(i,j)+0.5*(2.-vk/(cavd(i,j)
     &*roero(j,k))-vkp1/(cavd(i,j)*roero(j,k+1)))*(zact(j,k+1)
     &-zact(j,k))
      thmom(i,j)=thmom(i,j)+0.5*((vk/(cavd(i,j)
     &*roero(j,k)))*(1.-vk/cavd(i,j))+(vkp1/(cavd(i,j)*roero(j,k+1)))
     &*(1.-vkp1/cavd(i,j)))*(zact(j,k+1)-zact(j,k))

      xki(i,j)=roed(i,j)*vmax(i,j)*blth(i,j)/rmyued(i,j)
      dtdzetw=((dzeta(1)**2-(dzeta(1)+dzeta(2))**2)*td(j,1)
     &+(dzeta(1)+dzeta(2))**2*td(j,2)-dzeta(1)**2*td(j,3))
     &/(dzeta(1)*(dzeta(1)+dzeta(2))**2-(dzeta(1)+dzeta(2))*dzeta(1)**2)
      qw(i,j)=cp*rmyuw*dtdzetw*sqrt(roed(i,j)*ue(i,j)/rmyued(i,j))
     &/(pr*roero(j,1))

 2680 continue
      return
      end
```

```
c#######################################################################

      subroutine coefbody

c#######################################################################

       include 'comblck'

       do 50 j=1,jmax
       teh=0.5*(te(i-1,j)+te(i,j))
       peh=0.5*(pe(i-1,j)+pe(i,j))
      if(mks.eq.1)rmyueh(j)=47.88*(2.28d-8*(1.8*teh)**1.5)
     & /(1.8*teh+198.6)
      if(mks.eq.0)rmyueh(j)=(2.28d-8*(teh)**1.5)/(teh+198.6)
      roeh(j)=peh/(rr*teh)
 50   continue

      do 100 j=1,jmax
      dueds=(ue(i,j)-ue(i-1,j))/(s1(i,j)-s1(i-1,j))
      duedsd(i,j)=dueds
      dh2ds=(h2(i,j)-h2(i-1,j))/(s1(i,j)-s1(i-1,j))
      ups=(ue(i,j)+ue(i-1,j))/2.
      vps=(ve(i,j)+ve(i-1,j))/2.
      h1ps=h1(i,j)
      h2ps=(h2(i-1,j)+h2(i,j))/2.
c       if(i.ge.166)write(1,*)' i=',i,' j=',j,' ups=',ups,' vps=',vps
c    &,'h1ps=',h1ps,' h2ps=',h2ps,' h2(i,j)=',h2(i,j),' h2(i-1,j)=',h2(
c    &i-1,j),' s1(i,j)=',s1(i,j),' s1(i-1,j)=',s1(i-1,j),' ue(i,j)='
c    &,ue(i,j),' ue(i-1,j)=',ue(i-1,j)
c    &,' dueds=',dueds,' dh2ds=',dh2ds,'cavd=',cavd(i,j),'cos='
c    &,costh(i,j)

       if(j.eq.1.or.j.eq.jmax)go to 10
      dh1dy=((dy(j-1)/dy(j))*(h1(i,j+1)-h1(i,j))+(dy(j)/dy(j-1))
     &*(h1(i,j)-h1(i,j-1)))/(dy(j)+dy(j-1))
      dh2dy=((dy(j-1)/dy(j))*(h2(i,j+1)-h2(i,j))+(dy(j)/dy(j-1))
     &*(h2(i,j)-h2(i,j-1)))/(dy(j)+dy(j-1))
      dcosds=(costh(i,j)-costh(i-1,j))/(s1(i,j)-s1(i-1,j))
      dcosdy=((dy(j-1)/dy(j))*(costh(i,j+1)-costh(i,j))+(dy(j)/dy(j-1))
     &*(costh(i,j)-costh(i,j-1)))/(dy(j)+dy(j-1))
      sinth=sqrt(1-costh(i,j)**2)
      cotth=costh(i,j)/sinth
      ck2=(h1(i,j+1)*costh(i,j+1)-h1(i,j-1)*costh(i,j-1))
     &/((dy(j)+dy(j-1))*h1ps*h2ps*sinth)-dh2ds/(h2ps*sinth)
      ck1=(h2(i,j)*costh(i,j)-h2(i-1,j)*costh(i-1,j))/(h2ps*sinth
     &*(s1(i,j)-s1(i-1,j)))-dh1dy/(h1ps*h2ps*sinth)
      ck12=(-ck1+dcosds/sinth+costh(i,j)*(ck2-dcosdy/(h2ps*sinth)))
     &/sinth
      ck21=((1.+costh(i,j)**2)*dh2ds-2.*costh(i,j)*dh1dy/h1ps)
     &/(h2ps*sinth**2)
      sinth1=sqrt(1.-costh(i-1,j)**2)
      m1(j)=0.5*(1.+s1h(i,j)*dueds/ups)+s1h(i,j)*(h2(i,j)*sinth
     & *sqrt(roed(i,j)*rmyued(i,j))-h2(i-1,j)*sinth1*sqrt(roed(i-1,j)
     & *rmyued(i-1,j)))/(dx*h1ps*h2ps*sinth*sqrt(roeh(j)*rmyueh(j)))

      m2(j)=s1h(i,j)*dueds/ups-s1h(i,j)*ck1*cotth
      m3(j)=-s1h(i,j)*cotth*ck2*vinf/ups
      m4(j)=s1h(i,j)*ck21
      duedyd(i,j)=((dy(j-1)/dy(j))*(ue(i,j+1)-ue(i,j))+(dy(j)/dy(j-1))
```

```
     &*(ue(i,j)-ue(i,j-1)))/(dy(j)+dy(j-1))
       m5(j)=s1h(i,j)*vinf*duedyd(i,j)/(h2ps*ups**2)
     &+ck12*s1h(i,j)*vinf/ups
       sinp1=sqrt(1.-costh(i,j+1)**2)
       sinm1=sqrt(1.-costh(i,j-1)**2)
       dp1=sqrt(roeh(j+1)*rmyueh(j+1)*ue(i,j+1)
     & *s1h(i,j+1))*h1(i,j+1)*sinp1*vinf/ue(i,j+1)
       dp0=sqrt(roeh(j)*rmyueh(j)*ue(i,j)
     & *s1h(i,j))*h1(i,j)*sinth*vinf/ue(i,j)
       dm1=sqrt(roeh(j-1)*rmyueh(j-1)*ue(i,j-1)
     & *s1h(i,j-1))*h1(i,j-1)*sinm1*vinf/ue(i,j-1)
       drmdy=((dy(j-1)/dy(j))*(dp1-dp0)+(dy(j)/dy(j-1))
     &*(dp0-dm1))/(dy(j)+dy(j-1))
       m6(j)=drmdy*s1h(i,j)/(h1ps*h2ps*sinth
     & *sqrt(roeh(j)*rmyueh(j)*ups*s1h(i,j)))
       m7(j)=s1h(i,j)*vinf/(h2ps*ups)
       m8(j)=s1h(i,j)*ck2*vinf**2/(ups**2*sinth)
       m9(j)=s1h(i,j)*ck1*ups/(vinf*sinth)
       m10(j)=s1h(i,j)/h1ps
       m11(j)=s1h(i,j)*dueds/ups+s1h(i,j)*vps*duedyd(i,j)
     &/(h2ps*ups**2)
     &-s1h(i,j)*cotth*ck1+s1h(i,j)*ck2*vps**2/(ups**2*sinth)
     &+s1h(i,j)*ck12*vps/ups
       dveds=(ve(i,j)-ve(i-1,j))/(s1(i,j)-s1(i-1,j))
       dvedy=((dy(j-1)/dy(j))*(ve(i,j+1)-ve(i,j))+(dy(j)/dy(j-1))
     &*(ve(i,j)-ve(i,j-1)))/(dy(j)+dy(j-1))
       m12(j)=s1h(i,j)*dveds/vinf+s1h(i,j)*vps*dvedy/(ups
     &*vinf*h2ps)+s1h(i,j)*(-cotth*ck2*vps**2+ck1*ups**2/sinth
     &+ck21*ups*vps)/(ups*vinf)
       m13(j)=roww(i,j)*sqrt(roed(i,j)*ue(i,j)*s1(i,j)/rmyued(i,j))
     &/(roed(i,j)*ue(i,j))
c         if(i.ge.166)write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j)
c     &,m6(j),m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)
 117   format(//,2x,' i=',i3,' j=',i2,' m1=',d9.3,5x,' m2=',d9.3,5x
     &,' m3=',d9.3,5x,' m4=',d11.5,
     &/,5x,'m5=',d9.3,5x,'m6=',d9.3,5x,'m7=',d9.3,5x,'m8=',d9.3,/
     &5x,'m9=',d9.3,5x,'m10=',d9.3,4x,'m11=',d9.3,4x,'m12=',d9.3,//)

        go to 100

 10    m1(j)=0.5*(1.+s1h(i,j)*dueds/ups)+s1h(i,j)*(h2(i,j)
     & *sqrt(roed(i,j)*rmyued(i,j))-h2(i-1,j)*sqrt(roed(i-1,j)
     & *rmyued(i-1,j)))/(dx*h1ps*h2ps*sqrt(roeh(j)*rmyueh(j)))
       m2(j)=s1h(i,j)*dueds/ups
       m3(j)=s1h(i,j)*vinf/(h2ps*ups)
       m4(j)=s1h(i,j)*dh2ds/h2ps
       m5(j)=0
       m6(j)=m3(j)
       m7(j)=0
       m8(j)=0

       if(j.eq.1)then
       dvedy=(ve(i,j+1)-ve(i,j))/dy(j)
       dvedy1=(ve(i-1,j+1)-ve(i-1,j))/dy(j)
       dh1dy=(h1(i,j+1)-h1(i,j))/dy(j)
       dh2dy=(h2(i,j+1)-h2(i,j))/dy(j)
       dcosdy=(costh(i,j+1)-costh(i,j))/dy(j)
       dcosdy1=(costh(i-1,j+1)-costh(i-1,j))/dy(j)
       dcosdys=(dcosdy-dcosdy1)/(s1(i,j)-s1(i-1,j))
       dk1dy=-2.*(h1(i,2)-h1(i,1))/(h1ps*h2ps*dy(1)**2)
```

```
     &+dh2ds*dcosdy/h2ps+dcosdys
       go to 30
       endif

       if(j.eq.jmax)then
       dvedy=(ve(i,j)-ve(i,j-1))/dy(j-1)
       dvedy1=(ve(i-1,j)-ve(i-1,j-1))/dy(j-1)
       dh1dy=(h1(i,j)-h1(i,j-1))/dy(j-1)
       dh2dy=(h2(i,j)-h2(i,j-1))/dy(j-1)
       dcosdy=(costh(i,j)-costh(i,j-1))/dy(j-1)
       dcosdy1=(costh(i-1,j)-costh(i-1,j-1))/dy(j-1)
       dcosdys=(dcosdy-dcosdy1)/(s1(i,j)-s1(i-1,j))
       dk1dy=-2.*(h1(i,jmax-1)-h1(i,jmax))/(h1ps*h2ps*dy(jmax-1)**2)
     &+dh2ds*dcosdy/h2ps+dcosdys
       endif

 30    m9(j)=s1h(i,j)*ups*dk1dy/vinf
       m10(j)=s1h(i,j)/h1ps
       m11(j)=m2(j)
       dveyds=(dvedy-dvedy1)/(s1(i,j)-s1(i-1,j))
       m12(j)=s1h(i,j)*dveyds/vinf+s1h(i,j)*dvedy**2/(vinf*ups*h2ps)
     & +s1h(i,j)*dh2ds*dvedy/(vinf*h2ps)+s1h(i,j)*ups*dk1dy/vinf
       m13(j)=roww(i,j)*sqrt(roed(i,j)*ue(i,j)*s1(i,j)/rmyued(i,j))
     &/(roed(i,j)*ue(i,j))

       if(dy(1).eq.0)m9(j)=0.
       if(dy(1).eq.0)m12(j)=0.

c       if(i.ge.166)write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j),m6(j)
c     &,m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)

 100   continue

       return
       end
```

```
c#######################################################################

        subroutine coefcon

c#######################################################################

        include 'comblck'

        thetaco=atan(-zpd(1,1)/xpd(1,1))
        do 10 j=1,jmax
        if(j.eq.1.or.j.eq.jmax)go to 20
        m1(j)=1.5
        m2(j)=0.
        m3(j)=0.
        dvedy=(ve(1,j+1)-ve(1,j-1))/(2.*dy(j-1))
        duedy=(ue(1,j+1)-ue(1,j-1))/(2.*dy(j-1))
        m4(j)=1.0
        m5(j)=vinf*ve(1,j)/ue(1,j)**2
        dromyu=(roed(1,j+1)*rmyued(1,j+1)-roed(1,j-1)
     &*rmyued(1,j-1))/(2.*dy(j-1))
        m6(j)=-0.5*vinf*ve(1,j)/ue(1,j)**2
     &+0.5*vinf*dromyu/(sin(thetaco)*ue(1,j)*roed(1,j)*rmyued(1,j))
        m7(j)=vinf/(ue(1,j)*sin(thetaco))
        m8(j)=-(vinf/ue(1,j))**2
        m9(j)=0.
        m10(j)=0.
        m11(j)=0.
        m12(j)=ve(1,j)/vinf+ve(1,j)*dvedy
     &/(ue(1,j)*vinf*sin(thetaco))

c         write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j),m6(j)
c      &,m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)
c 117   format(//,2x,' i=',i3,' j=',i2,' m1=',d9.3,5x,' m2=',d9.3,5x
c      &,' m3=',d9.3,5x,' m4=',d11.5,
c      &/,5x,'m5=',d9.3,5x,'m6=',d9.3,5x,'m7=',d9.3,5x,'m8=',d9.3,/
c      &5x,'m9=',d9.3,5x,'m10=',d9.3,4x,'m11=',d9.3,4x,'m12=',d9.3,//)

        go to 10

20      m1(j)=1.5
        m2(j)=0.
        m3(j)=vinf/(ue(1,1)*sin(thetaco))
        if(j.eq.1)dvedy=ve(1,2)/dy(1)
        if(j.eq.jmax)dvedy=-ve(1,jmax-1)/dy(jmax-1)
        m4(j)=1.0
        m5(j)=0.
        m6(j)=m3(j)
        m7(j)=0.
        m8(j)=0.
        m9(j)=0.
        m10(j)=0.
        m11(j)=0.
        m12(j)=dvedy**2/(ue(1,1)*vinf*sin(thetaco))+dvedy/vinf
c         write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j),m6(j)
c      &,m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)
10      continue

        return
        end
```

```
c#########################################################################

      subroutine coefstrm

c#########################################################################

       include 'comblck'

      do 50 j=1,jmax
      teh=0.5*(te(i-1,j)+te(i,j))
      peh=0.5*(pe(i-1,j)+pe(i,j))
      if(mks.eq.1)rmyueh(j)=47.88*(2.28d-8*(1.8*teh)**1.5)
     & /(1.8*teh+198.6)
      if(mks.eq.0)rmyueh(j)=(2.28d-8*(teh)**1.5)/(teh+198.6)
      roeh(j)=peh/(rr*teh)
 50   continue

      do 100 j=1,jmax
      dueds=(cavd(i,j)-cavd(i-1,j))/(s1(i,j)-s1(i-1,j))
      duedsd(i,j)=dueds
      dh2ds=(h2(i,j)-h2(i-1,j))/(s1(i,j)-s1(i-1,j))
      cav1=(cavd(i,j)+cavd(i-1,j))/2.
      h2ps=(h2(i,j)+h2(i-1,j))/2.
      if(j.eq.1.or.j.eq.jmax)go to 40
      duedyd(i,j)=((dy(j-1)
     &/dy(j))*(cavd(i,j+1)-cavd(i,j))+(dy(j)/dy(j-1))
     &*(cavd(i,j)-cavd(i,j-1)))/(dy(j)+dy(j-1))
 40   continue
      if(j.eq.1)then
      dk1dy=2.*(cavd(i,2)-cavd(i,1))/(h2ps*cav1*dy(1)**2)
      if(jmax.ge.90)dk1dy=2.*(cavd(i,6)-cavd(i,1))/(h2ps*cav1
     &*25.*dy(1)**2)
      endif

      if(j.eq.jmax)then
      dk1dy=2.*(cavd(i,jmax-1)-cavd(i,jmax))/(h2ps*cav1
     &*dy(jmax-1)**2)
      if(jmax.ge.90)dk1dy=2.*(cavd(i,jmax-5)-cavd(i,jmax))/(h2ps*cav1
     &*25.*dy(jmax-1)**2)
      endif

 4334  if(j.eq.1.or.j.eq.jmax)go to 8000
      m1(j)=0.5*(1.+s1h(i,j)*dueds/cav1)+s1h(i,j)*cav1*(h2(i,j)
     & *sqrt(roed(i,j)*rmyued(i,j))-h2(i-1,j)*sqrt(roed(i-1,j)
     & *rmyued(i-1,j)))/(dx*vinf*h2ps*sqrt(roeh(j)*rmyueh(j)))
      m2(j)=s1h(i,j)*dueds/cav1
      m3(j)=0
      m4(j)=s1h(i,j)*dh2ds/h2ps
      m5(j)=0
      m6(j)=s1h(i,j)*cav1*((sqrt(roeh(j+1)*rmyueh(j+1)*cavd(i,j+1)
     & *s1h(i,j+1))*(vinf/cavd(i,j+1))**2-sqrt(roeh(j)
     &*rmyueh(j)*cavd(i,j)*s1h(i,j))*(vinf/cavd(i,j))**2)*dy(j-1)/dy(j)
     & +(sqrt(roeh(j)*rmyueh(j)*cavd(i,j)*s1h(i,j))*(vinf/cavd(i,j))**2
     & -sqrt(roeh(j-1)*rmyueh(j-1)*cavd(i,j-1)*s1h(i,j-1))*(vinf
     & /cavd(i,j-1))**2)*dy(j)/dy(j-1))/((dy(j)+dy(j-1))*vinf*h2ps
     & *sqrt(roeh(j)*rmyueh(j)*cav1*s1h(i,j)))
      m7(j)=s1h(i,j)*vinf/(h2ps*cav1)
      m8(j)=-s1h(i,j)*dh2ds*vinf**2/(h2ps*cav1**2)
```

```
      m9(j)=s1h(i,j)*((dy(j-1)/dy(j))
     &*(cavd(i,j+1)-cavd(i,j))+(dy(j)/dy(j-1))
     &*(cavd(i,j)-cavd(i,j-1)))/((dy(j)+dy(j-1))*vinf*h2ps)
      m10(j)=s1h(i,j)*cav1/vinf
      m11(j)=m2(j)
      m12(j)=m9(j)
      m13(j)=roww(i,j)*sqrt(roed(i,j)*cavd(i,j)*s1(i,j)/rmyued(i,j))
     &/(roed(i,j)*cavd(i,j))

c     if(i.ge.99)  write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j),m6(j)
c    &,m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)
 117  format(//,2x,' i=',i3,' j=',i2,' m1=',d9.3,5x,' m2=',d9.3,5x
     &,' m3=',d9.3,5x,' m4=',d11.5,
     &/,5x,'m5=',d9.3,5x,'m6=',d9.3,5x,'m7=',d9.3,5x,'m8=',d9.3,/
     &5x,'m9=',d9.3,5x,'m10=',d9.3,4x,'m11=',d9.3,4x,'m12=',d9.3,//)

      go to 100

 8000 m1(j)=0.5*(1.+s1h(i,j)*dueds/cav1)+s1h(i,j)*cav1*(h2(i,j)
     & *sqrt(roed(i,j)*rmyued(i,j))-h2(i-1,j)*sqrt(roed(i-1,j)
     & *rmyued(i-1,j)))/(dx*vinf*h2ps*sqrt(roeh(j)*rmyueh(j)))
      m2(j)=s1h(i,j)*dueds/cav1
      m3(j)=s1h(i,j)*vinf/(h2ps*cav1)
      m4(j)=s1h(i,j)*dh2ds/h2ps
      m5(j)=0
      m6(j)=m3(j)
      m7(j)=0
      m8(j)=0
      m9(j)=s1h(i,j)*dk1dy*cav1/vinf
      m10(j)=s1h(i,j)*cav1/vinf
      m11(j)=m2(j)
      m12(j)=m9(j)
      m13(j)=roww(i,j)*sqrt(roed(i,j)*cavd(i,j)*s1(i,j)/rmyued(i,j))
     &/(roed(i,j)*cavd(i,j))

c     if(i.ge.99)  write(1,117)i,j,m1(j),m2(j),m3(j),m4(j),m5(j),m6(j)
c    &,m7(j),m8(j),m9(j),m10(j),m11(j),m12(j)


 100  continue
      return
      end
```

```
c######################################################################

      subroutine conoff

c######################################################################

        include 'comblck'
c---------------------------------------------------------------------
c
c       cone off the line of symmetry solution
c
c       (blottner's iterative method is used)
c
c---------------------------------------------------------------------
        he=cp*te(1,j)+0.5*cavd(1,j)**2

4100    write(6,*)' j=',j,' kmax=',kmax,'zeta(kmax)=',zeta(kmax)

        do 1031 k=1,kmax
        h(1,j,k)=h(1,j-1,k)
        h(2,j,k)=h(2,j-1,k)
        hs(1,j,k)=hs(1,j-1,k)
        hs(2,j,k)=hs(2,j-1,k)
 1031   continue

        it=0
        do 1123 k=1,kmax
        bc(j,k)=bc(j-1,k)
 1123   roero(j,k)=roero(j-1,k)

1170    it=it+1
       if(it.gt.30)write(6,*)' iteration for conoff is gt.30',' j=',j
       if(it.gt.30)stop
       do 1110 k=2,kmax-1
       ai(1,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))
     &        *(dzeta(k)/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &     -(bc(j,k)+bc(j,k-1))/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
       ai(2,k)=0
       ai(3,k)=0
       ai(4,k)=ai(1,k)

       ci(1,k)=-(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))
     &        *(dzeta(k-1)/dzeta(k))/(dzeta(k)+dzeta(k-1))
     &     -(bc(j,k)+bc(j,k+1))/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
       ci(2,k)=0
       ci(3,k)=0
       ci(4,k)=ci(1,k)

       bi(1,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &          /dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &          +(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*(dzeta(k)-dzeta(k-1))
     &    /(dzeta(k)*dzeta(k-1))-m5(j)*h(2,j,k)-m7(j)*h(2,j,k)/dy(j-1)

       bi(2,k)=-m4(j)*h(2,j,k)
       bi(3,k)=-m5(j)*h(1,j,k)-2.*m8(j)*h(2,j,k)
       bi(4,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &          /dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &          +(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*(dzeta(k)-dzeta(k-1))
     &    /(dzeta(k)*dzeta(k-1))-m4(j)*h(1,j,k)-m7(j)*h(2,j,k)/dy(j-1)
```

```
        feta=((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
        geta=((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))/(dzeta(k)+dzeta(k-1))

        as(1,k)=m1(j)*feta
        as(2,k)=m1(j)*geta
        as(3,k)=m6(j)*feta+m7(j)*feta/dy(j-1)
        as(4,k)=m6(j)*geta+m7(j)*geta/dy(j-1)

        di(1,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*feta-m5(j)*h(1,j,k)
     &*h(2,j,k)-m8(j)*h(2,j,k)**2-m7(j)*h(2,j,k)*h(1,j-1,k)/dy(j-1)
     &+m7(j)*feta*hs(2,j-1,k)/dy(j-1)
        di(2,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*geta
     &-m4(j)*h(1,j,k)*h(2,j,k)-m12(j)*roero(j,k)
     &-m7(j)*h(2,j,k)*h(2,j-1,k)/dy(j-1)+m7(j)*geta*hs(2,j-1,k)/dy(j-1)

 1110   continue

        do 1120 m=1,4
        e(m,kmax)=0
        es(m,kmax)=0
 1120   continue
        ds(1,kmax)=1.0
        ds(2,kmax)=ve(1,j)/vinf
        do 1140 m=1,2
        hn(m,j,1)=0
        hsn(m,j,1)=0
 1140   continue

        call ntrid

c        do 115 k=1,kmax
c        write(6,*)' it=',it,' k=',k,' hn1=',hn(1,j,k),'hn2=',hn(2,j,k)
c 115    continue

        isat=0

        do 1145 k=2,kmax-1
        er=(h(1,j,k)-hn(1,j,k))/h(1,j,k)
        if(abs(er)-1.d-5)1145,1145,1146
 1145   continue

        isat=1

 1146   do 1150 k=1,kmax
        do 1150 m=1,2
        h(m,j,k)=hn(m,j,k)
        hs(m,j,k)=hsn(m,j,k)
 1150   continue

        do 1151 k=2,kmax-1
        bcom(k)=(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     & -(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k))
     &/dy(j-1))
     &*dzeta(k)/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
        dcom(k)=-(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     & -(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     &+(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k))
```

```
     &/dy(j-1))
     &*dzeta(k)/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
     &-(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k))
     &/dy(j-1))
     &*dzeta(k-1)/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
     &-m7(j)*h(2,j,k)/dy(j-1)
       acom(k)=(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     &+(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k))
     &/dy(j-1))
     &*dzeta(k-1)/(dzeta(k)*(dzeta(k)+dzeta(k-1)))

       feta=((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
       geta=((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))/(dzeta(k)+dzeta(k-1))
       cfeta=h(1,j,k)*feta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*feta**2+bc(j,k)*h(1,j,k)*2.*((h(1,j,k+1)-h(1,j,k))/dzeta
     &(k)-(h(1,j,k)-h(1,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
       cgeta=h(2,j,k)*geta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*geta**2+bc(j,k)*h(2,j,k)*2.*((h(2,j,k+1)-h(2,j,k))/dzeta
     &(k)-(h(2,j,k)-h(2,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
       ccom(k)=-ue(1,j)**2*(1.-1./pr)*(cfeta+cgeta*(vinf/ue(1,j))**2)/he
     & -m7(j)*h(2,j,k)*h(3,j-1,k)/dy(j-1)
1151   continue
       ccom(kmax-1)=ccom(kmax-1)-acom(kmax-1)

       if(kaw.eq.1)dcom(2)=dcom(2)-bcom(2)*(dzeta(1)+dzeta(2))**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
       if(kaw.eq.1)acom(2)=acom(2)+bcom(2)*dzeta(1)**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
       if(kaw.eq.0)ccom(2)=ccom(2)-bcom(2)*cp*twall(1,j)/(cp*te(1,j)
     & +0.5*cavd(1,j)**2)

        call sy(2,kmax-1,bcom,dcom,acom,ccom)

2146    do 1152 k=2,kmax-1
1152   h(3,j,k)=ccom(k)
       h(3,j,kmax)=1.

       if(kaw.eq.1)h(3,j,1)=(dzeta(1)**2*h(3,j,3)-(dzeta(1)+dzeta(2))**2
     & *h(3,j,2))/(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
       if(kaw.eq.0)h(3,j,1)=cp*twall(1,j)/(cp*te(1,j)+0.5*cavd(1,j)**2)


       do 1153 k=1,kmax
       td(j,k)=(he*h(3,j,k)-0.5*(ue(1,j)*h(1,j,k))**2
     &-0.5*(vinf*h(2,j,k))**2)/cp
       if(td(j,k).le.0)then
        write(6,*)' td(j,k) is le. 0. at k=',k,' j=',j
     &,'he=',he,' h3=',h(3,j,k),' ue(1,j)=',ue(1,j),' h1=',h(1,j,k)
        iend=1
        return
        endif

       roero(j,k)=td(j,k)/te(1,j)
       if(mks.eq.1)bc(j,k)=(te(1,j)/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*te(1,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(1,j))**1.5))
       if(mks.eq.0)bc(j,k)=(te(1,j)/td(j,k))*(td(j,k)**1.5)*(te(1,j)
     & +198.6)/((td(j,k)+198.6)*(te(1,j)**1.5))
```

```
1153  continue

      if(isat.eq.1)go to 1160

      go to 1170

1160   if(h(1,j,kmax-1).lt.ukmax1)then
       kmax=kmax+1
       h(1,j,kmax)=1.0
       h(2,j,kmax)=ve(1,j)/vinf
       h(3,j,kmax)=1.0
      hs(1,j,kmax)=hs(1,j,kmax-1)+(h(1,j,kmax)+h(1,j,kmax-1))
     &*dzeta(kmax-1)/2.
      hs(2,j,kmax)=hs(2,j,kmax-1)+(h(2,j,kmax)+h(2,j,kmax-1))
     &*dzeta(kmax-1)/2.
       roero(j,kmax)=1.0
       bc(j,kmax)=1.0
       h(1,j-1,kmax)=1.0
       h(2,j-1,kmax)=ve(1,j-1)/vinf
       h(3,j-1,kmax)=1.0
      hs(1,j-1,kmax)=hs(1,j-1,kmax-1)+(h(1,j-1,kmax)+h(1,j-1,kmax-1))
     &*dzeta(kmax-1)/2.
      hs(2,j-1,kmax)=hs(2,j-1,kmax-1)+(h(2,j-1,kmax)+h(2,j-1,kmax-1))
     &*dzeta(kmax-1)/2.
       roero(j-1,kmax)=1.0
       bc(j-1,kmax)=1.0
       go to 4100
       endif

      return
      end
```

```
c#########################################################################

      subroutine conon

c#########################################################################

       include 'comblck'
c-----------------------------------------------------------------------
c
c      cone on the line of symmetry solution
c
c       (blottner's iterative method is used)
c
c-----------------------------------------------------------------------
       he=cp*te(1,j)+0.5*cavd(1,j)**2

4100   write(6,*)' j=',j,' kmax=',kmax,' zeta(kmax)=',zeta(kmax)

      do 1030 m=1,2
      h(m,j,1)=0
      hs(m,j,1)=0
 1030 continue
      do 1031 k=2,kmax
      h(1,j,k)=1.
      if(j.eq.1)h(2,j,k)=ve(1,2)/(vinf*dy(1))
      if(j.eq.jmax)h(2,j,k)=-ve(1,jmax-1)/(vinf*dy(jmax-1))
      hs(1,j,k)=hs(1,j,k-1)+(h(1,j,k)+h(1,j,k-1))*dzeta(k-1)/2.
      hs(2,j,k)=hs(2,j,k-1)+(h(2,j,k)+h(2,j,k-1))*dzeta(k-1)/2.
 1031 continue

      it=0
      do 1123 k=1,kmax
      bc(j,k)=1.
 1123 roero(j,k)=1.

 1170 it=it+1
      if(it.gt.30)write(6,*)' iteration in conon is gt.30',' j=',j
      if(it.gt.30)stop
      do 1110 k=2,kmax-1
      ai(1,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))
     &        *(dzeta(k)/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &     -(bc(j,k)+bc(j,k-1))/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
      ai(2,k)=0
      ai(3,k)=0
      ai(4,k)=ai(1,k)

      ci(1,k)=-(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))
     &        *(dzeta(k-1)/dzeta(k))/(dzeta(k)+dzeta(k-1))
     &     -(bc(j,k)+bc(j,k+1))/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
      ci(2,k)=0
      ci(3,k)=0
      ci(4,k)=ci(1,k)

      bi(1,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &         /dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &         +(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*(dzeta(k)-dzeta(k-1))
     &         /(dzeta(k)*dzeta(k-1))-m5(j)*h(2,j,k)-m7(j)*h(2,j,k)

      bi(2,k)=-m4(j)*h(2,j,k)
      bi(3,k)=-m5(j)*h(1,j,k)-2.*m8(j)*h(2,j,k)
```

```
      bi(4,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &           /dzeta(k-1))/(dzeta(k)+dzeta(k-1))
     &           +(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*(dzeta(k)-dzeta(k-1))
     &           /(dzeta(k)*dzeta(k-1))-m4(j)*h(1,j,k)-m7(j)*h(2,j,k)
     &           -2.*m3(j)*h(2,j,k)

      feta=((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
      geta=((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))/(dzeta(k)+dzeta(k-1))

      as(1,k)=m1(j)*feta
      as(2,k)=m1(j)*geta
      as(3,k)=m6(j)*feta+m7(j)*feta
      as(4,k)=m6(j)*geta+m7(j)*geta

      di(1,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*feta-m5(j)*h(1,j,k)
     &*h(2,j,k)-m8(j)*h(2,j,k)**2-m7(j)*h(2,j,k)*h(1,j-1,k)
     &+m7(j)*feta*hs(2,j-1,k)
      di(2,k)=(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*geta
     &-m4(j)*h(1,j,k)*h(2,j,k)-m12(j)*roero(j,k)
     &-m7(j)*h(2,j,k)*h(2,j-1,k)+m7(j)*geta*hs(2,j-1,k)
     &-m3(j)*h(2,j,k)**2

1110  continue

      do 1120 m=1,4
      e(m,kmax)=0
      es(m,kmax)=0
1120  continue
      ds(1,kmax)=1.0
      if(j.eq.1)ds(2,kmax)=ve(1,2)/(vinf*dy(1))
      if(j.eq.jmax)ds(2,kmax)=-ve(1,jmax-1)/(vinf*dy(jmax-1))
      do 1140 m=1,2
      hn(m,j,1)=0
      hsn(m,j,1)=0
1140  continue

      call ntrid

      isat=0

      do 1145 k=2,kmax-1
      er=(h(1,j,k)-hn(1,j,k))/h(1,j,k)
      if(abs(er)-1.d-4)1145,1145,1146
1145  continue

      isat=1

1146  do 1150 k=1,kmax
      do 1150 m=1,2
      h(m,j,k)=hn(m,j,k)
      hs(m,j,k)=hsn(m,j,k)
1150  continue

      do 1151 k=2,kmax-1
      bcom(k)=(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     & -(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k)))
     &*dzeta(k)/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
      dcom(k)=-(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
```

```
     & -(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     &+(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k)))
     &*dzeta(k)/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
     &-(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k)))
     &*dzeta(k-1)/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
     &-m7(j)*h(2,j,k)
      acom(k)=(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     &+(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k)+m7(j)*(hs(2,j,k)-hs(2,j-1,k)))
     &*dzeta(k-1)/(dzeta(k)*(dzeta(k)+dzeta(k-1)))

      feta=((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
      geta=((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))/(dzeta(k)+dzeta(k-1))
      cfeta=h(1,j,k)*feta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*feta**2+bc(j,k)*h(1,j,k)*2.*((h(1,j,k+1)-h(1,j,k))/dzeta
     &(k)-(h(1,j,k)-h(1,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cgeta=h(2,j,k)*geta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*geta**2+bc(j,k)*h(2,j,k)*2.*((h(2,j,k+1)-h(2,j,k))/dzeta
     &(k)-(h(2,j,k)-h(2,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      ccom(k)=-ue(1,j)**2*(1.-1./pr)*cfeta/he
     & -m7(j)*h(2,j,k)*h(3,j-1,k)
1151  continue
      ccom(kmax-1)=ccom(kmax-1)-acom(kmax-1)

      if(kaw.eq.1)dcom(2)=dcom(2)-bcom(2)*(dzeta(1)+dzeta(2))**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.1)acom(2)=acom(2)+bcom(2)*dzeta(1)**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)ccom(2)=ccom(2)-bcom(2)*cp*twall(1,j)/(cp*te(1,j)
     & +0.5*ue(1,j)**2)
      call sy(2,kmax-1,bcom,dcom,acom,ccom)

      do 1152 k=2,kmax-1
1152  h(3,j,k)=ccom(k)
      h(3,j,kmax)=1.

      if(kaw.eq.1)h(3,j,1)=(dzeta(1)**2*h(3,j,3)-(dzeta(1)+dzeta(2))**2
     & *h(3,j,2))/(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)h(3,j,1)=cp*twall(1,j)/(cp*te(1,j)+0.5*cavd(1,j)**2)
      do 1153 k=1,kmax
      td(j,k)=(he*h(3,j,k)-0.5*(cavd(1,j)*h(1,j,k))**2)/cp

      if(td(j,k).lt.0)then
      write(6,*)' td(j,k) is lt. 0. at k=',k,' j=',j
     &,'he=',he,' h3=',h(3,j,k),' cavd(1,j)=',cavd(1,j),' h1=',h(1,j,k)
      iend=1
      return
      endif

      roero(j,k)=td(j,k)/te(1,j)
      if(mks.eq.1)bc(j,k)=(te(1,j)/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*te(1,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(1,j))**1.5))
      if(mks.eq.0)bc(j,k)=(te(1,j)/td(j,k))*(td(j,k)**1.5)*(te(1,j)
     & +198.6)/((td(j,k)+198.6)*(te(1,j)**1.5))

1153  continue

      if(isat.eq.1)go to 1160
```

```
      go to 1170

1160  if(h(1,j,kmax-1).lt.ukmax1)then
      kmax=kmax+1
      h(1,j,kmax)=1.0
      h(2,j,kmax)=ve(1,2)/(vinf*dy(1))
      if(j.eq.1)h(2,j,kmax)=ve(1,2)/(vinf*dy(1))
      if(j.eq.jmax)h(2,j,kmax)=-ve(1,jmax-1)/(vinf*dy(jmax-1))
      hs(1,j,kmax)=hs(1,j,kmax-1)+(h(1,j,kmax)+h(1,j,kmax-1))
     &*dzeta(kmax-1)/2.
      hs(2,j,kmax)=hs(2,j,kmax-1)+(h(2,j,kmax)+h(2,j,kmax-1))
     &*dzeta(kmax-1)/2.
      roero(j,kmax)=1.0
      bc(j,kmax)=1.0
      go to 4100
      endif

      return
      end
```

```
c#########################################################################

        subroutine correct

c#########################################################################

        include 'comblck'

       do 1257 k=1,kmax
       b1(j,k)=bcb(j,k)
       b2(j,k)=0.
       b3(j,k)=0.
       b4(j,k)=bcb(j,k)
 1257  continue

c------------------------------------------------------------------

c      to calculate ak, bk, ck, ak, and dk

c-------------------------------------------------------------------

 1256  do 5300 k=2,kmax-1

       de=dzeta(k)+dzeta(k-1)

       ai(1,k)=-0.5*(b1(j,k)+b1(j,k-1))/(de*dzeta(k-1))
       ai(2,k)=-0.5*(b2(j,k)+b2(j,k-1))/(de*dzeta(k-1))
       ai(3,k)=-0.5*(b3(j,k)+b3(j,k-1))/(de*dzeta(k-1))
       ai(4,k)=-0.5*(b4(j,k)+b4(j,k-1))/(de*dzeta(k-1))

       ci(1,k)=-0.5*(b1(j,k)+b1(j,k+1))/(de*dzeta(k))
       ci(2,k)=-0.5*(b2(j,k)+b2(j,k+1))/(de*dzeta(k))
       ci(3,k)=-0.5*(b3(j,k)+b3(j,k+1))/(de*dzeta(k))
       ci(4,k)=-0.5*(b4(j,k)+b4(j,k+1))/(de*dzeta(k))

       bi(1,k)=-0.5*((b1(j,k)+b1(j,k+1))/dzeta(k)+(b1(j,k)+b1(j,k-1))
     &/dzeta(k-1))/de-m10(j)*hb(1,j,k)/dx
       bi(2,k)=-0.5*((b2(j,k)+b2(j,k+1))/dzeta(k)+(b2(j,k)+b2(j,k-1))
     &/dzeta(k-1))/de
       bi(3,k)=-0.5*((b3(j,k)+b3(j,k+1))/dzeta(k)+(b3(j,k)+b3(j,k-1))
     &/dzeta(k-1))/de
       bi(4,k)=-0.5*((b4(j,k)+b4(j,k+1))/dzeta(k)+(b4(j,k)+b4(j,k-1))
     &/dzeta(k-1))/de-m10(j)*hb(1,j,k)/dx

       fbeta=(hb(1,j,k+1)-hb(1,j,k-1))/(dzeta(k)+dzeta(k-1))
       gbeta=(hb(2,j,k+1)-hb(2,j,k-1))/(dzeta(k)+dzeta(k-1))

       as(1,k)=m10(j)*fbeta/dx
       as(2,k)=m10(j)*gbeta/dx
       as(3,k)=0
       as(4,k)=0

       di(1,k)=(0.5*(b1(j,k)+b1(j,k+1))*(h(1,j,k)-h(1,j,k+1))
     &/dzeta(k)+0.5*(b1(j,k)+b1(j,k-1))*(h(1,j,k)-h(1,j,k-1))
     &/dzeta(k-1))/de+(0.5*(b3(j,k)+b3(j,k+1))*(h(2,j,k)-h(2,j,k+1))
     &/dzeta(k)+0.5*(b3(j,k)+b3(j,k-1))*(h(2,j,k)-h(2,j,k-1))
     &/dzeta(k-1))/de-m1(j)*hsb(1,j,k)*fbeta+m2(j)*hb(1,j,k)**2
     &+m5(j)*hb(1,j,k)*hb(2,j,k)-m6(j)*hsb(2,j,k)*fbeta
     &+m8(j)*hb(2,j,k)**2-m11(j)*roerob(j,k)
     &+m10(j)*hb(1,j,k)*(-h(1,j,k))/dx+m10(j)*hs(1,j,k)*fbeta/dx
```

```
     &+m13(j)*fbeta
       di(2,k)=(0.5*(b4(j,k)+b4(j,k+1))*(h(2,j,k)-h(2,j,k+1))
     &/dzeta(k)+0.5*(b4(j,k)+b4(j,k-1))*(h(2,j,k)-h(2,j,k-1))
     &/dzeta(k-1))/de+(0.5*(b2(j,k)+b2(j,k+1))*(h(1,j,k)-h(1,j,k+1))
     &/dzeta(k)+0.5*(b2(j,k)+b2(j,k-1))*(h(1,j,k)-h(1,j,k-1))
     &/dzeta(k-1))/de-m1(j)*hsb(1,j,k)*gbeta+m4(j)*hb(1,j,k)*hb(2,j,k)
     &+m3(j)*hb(2,j,k)**2-m6(j)*hsb(2,j,k)*gbeta
     &+m9(j)*hb(1,j,k)**2-m12(j)*roerob(j,k)+m10(j)*hb(1,j,k)*
     &(-h(2,j,k))/dx+m10(j)*hs(1,j,k)*gbeta/dx+m13(j)*gbeta
       if(j.eq.1.or.j.eq.jmax)go to 5300
       db=dy(j)+dy(j-1)

       if(j.eq.2)save(1,k)=hb(2,j-1,k)
       if(j.eq.2)save(2,k)=hsb(2,j-1,k)
       if(j.eq.jmax-1)save(3,k)=hb(2,j+1,k)
       if(j.eq.jmax-1)save(4,k)=hsb(2,j+1,k)

       if(j.eq.2)hb(2,j-1,k)=0
       if(j.eq.2)hsb(2,j-1,k)=0
       if(j.eq.jmax-1)hb(2,j+1,k)=0
       if(j.eq.jmax-1)hsb(2,j+1,k)=0

       fby=((dy(j-1)/dy(j))*(hb(1,j+1,k)-hb(1,j,k))
     &+(dy(j)/dy(j-1))*(hb(1,j,k)-hb(1,j-1,k)))/db
       sgby=((dy(j-1)/dy(j))*(hsb(2,j+1,k)-hsb(2,j,k))
     &+(dy(j)/dy(j-1))*(hsb(2,j,k)-hsb(2,j-1,k)))/db
       gby=((dy(j-1)/dy(j))*(hb(2,j+1,k)-hb(2,j,k))
     &+(dy(j)/dy(j-1))*(hb(2,j,k)-hb(2,j-1,k)))/db

       if(kterm.eq.1.and.j.eq.jmaxt)then
       fby=(3.*hb(1,j,k)-4.*hb(1,j-1,k)+hb(1,j-2,k))/(2.*dy(j-1))
       gby=(3.*hb(2,j,k)-4.*hb(2,j-1,k)+hb(2,j-2,k))/(2.*dy(j-1))
       sgby=(3.*hsb(2,j,k)-4.*hsb(2,j-1,k)+hsb(2,j-2,k))/(2.*dy(j-1))
       endif

       di(1,k)=di(1,k)
     &+m7(j)*(hb(2,j,k)*fby-fbeta*sgby)
       di(2,k)=di(2,k)
     &+m7(j)*(hb(2,j,k)*gby-gbeta*sgby)

       if(j.eq.2)hb(2,j-1,k)=save(1,k)
       if(j.eq.2)hsb(2,j-1,k)=save(2,k)
       if(j.eq.jmax-1)hb(2,j+1,k)=save(3,k)
       if(j.eq.jmax-1)hsb(2,j+1,k)=save(4,k)

5300   continue

       do 5320 m=1,4
       e(m,kmax)=0
       es(m,kmax)=0
5320   continue
       ds(1,kmax)=1.0
       ds(2,kmax)=ve(i,j)/vinf
       if(j.eq.1)ds(2,kmax)=ve(i,2)/(vinf*dy(1))
       if(j.eq.jmax)ds(2,kmax)=-ve(i,jmax-1)/(vinf*dy(jmax-1))

       do 5340 m=1,2
       hn(m,j,1)=0
       hsn(m,j,1)=0
5340   continue
```

```
c-----------------------------------------------------------------------------

c       To solve the block tridiagonal matrix equation, call ntrid

c-----------------------------------------------------------------------------

        call ntrid

c        do 33 k=1,kmax
c 33     write(6,*)'corr i,j,k=',i,j,k,'h1=',hn(1,j,k),'h2=',hn(2,j,k)

        return
        end
```

```
c#####################################################################

      subroutine correng

c#####################################################################

      include 'comblck'

      he=cp*te(i,j)+0.5*cavd(i,j)**2

      do 7151 k=2,kmax-1
      bcom(k)=0.5*(bcb(j,k)+bcb(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))
     & *dzeta(k-1))
      dcom(k)=-0.5*(bcb(j,k)+bcb(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))
     & *dzeta(k))
     & -0.5*(bcb(j,k)+bcb(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     & -m10(j)*hb(1,j,k)/dx
      acom(k)=0.5*(bcb(j,k)+bcb(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))
     &*dzeta(k))
      fbeta=((dzeta(k-1)/dzeta(k))*(hb(1,j,k+1)-hb(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(hb(1,j,k)-hb(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
      gbeta=((dzeta(k-1)/dzeta(k))*(hb(2,j,k+1)-hb(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(hb(2,j,k)-hb(2,j,k-1)))/(dzeta(k)+dzeta(k-1))
      ebeta=((dzeta(k-1)/dzeta(k))*(hb(3,j,k+1)-hb(3,j,k))+(dzeta(k)
     &/dzeta(k-1))*(hb(3,j,k)-hb(3,j,k-1)))/(dzeta(k)+dzeta(k-1))

      if(j.eq.1.or.j.eq.jmax)then
      sgby=0.
      eby=0.
      go to 322
      endif

      sgby=((dy(j-1)/dy(j))*(hsb(2,j+1,k)-hsb(2,j,k))
     &+(dy(j)/dy(j-1))*(hsb(2,j,k)-hsb(2,j-1,k)))/(dy(j)+dy(j-1))
      if(j.eq.2)sgby=((dy(j-1)/dy(j))*(hsb(2,j+1,k)-hsb(2,j,k))
     &+(dy(j)/dy(j-1))*hsb(2,j,k))/(dy(j)+dy(j-1))
      if(j.eq.jmax-1)sgby=((dy(j-1)/dy(j))*(-hsb(2,j,k))
     &+(dy(j)/dy(j-1))*(hsb(2,j,k)-hsb(2,j-1,k)))/(dy(j)+dy(j-1))
      eby=((dy(j-1)/dy(j))*(hb(3,j+1,k)-hb(3,j,k))
     &+(dy(j)/dy(j-1))*(hb(3,j,k)-hb(3,j-1,k)))/(dy(j)+dy(j-1))

 322  if(kterm.eq.1.and.j.eq.jmaxt)then
      sgby=(3.*hsb(2,j,k)-4.*hsb(2,j-1,k)+hsb(2,j-2,k))/(2.*dy(j-1))
      eby=(3.*hb(3,j,k)-4.*hb(3,j-1,k)+hb(3,j-2,k))/(2.*dy(j-1))
      endif

      cfbeta=hb(1,j,k)*fbeta*(bcb(j,k+1)-bcb(j,k-1))/(dzeta(k)
     & +dzeta(k-1))+bcb(j,k)*fbeta**2+bcb(j,k)*hb(1,j,k)*2.
     &*((hb(1,j,k+1)-hb(1,j,k))/dzeta(k)-(hb(1,j,k)-hb(1,j,k-1))
     &/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cgbeta=hb(2,j,k)*gbeta*(bcb(j,k+1)-bcb(j,k-1))/(dzeta(k)
     & +dzeta(k-1))+bcb(j,k)*gbeta**2+bcb(j,k)*hb(2,j,k)*2.
     &*((hb(2,j,k+1)-hb(2,j,k))/dzeta(k)-(hb(2,j,k)-hb(2,j,k-1))
     &/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cfgb=hb(1,j,k)*gbeta*(bcb(j,k+1)-bcb(j,k-1))/(dzeta(k)
     &+dzeta(k-1))+bcb(j,k)*fbeta*gbeta+bcb(j,k)*hb(1,j,k)*2.
     &*((hb(2,j,k+1)-hb(2,j,k))/dzeta(k)-(hb(2,j,k)-hb(2,j,k-1))
     &/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cgfb=hb(2,j,k)*fbeta*(bcb(j,k+1)-bcb(j,k-1))/(dzeta(k)
```

```
     & +dzeta(k-1))+bcb(j,k)*fbeta*gbeta+bcb(j,k)*hb(2,j,k)*2.
     &*((hb(1,j,k+1)-hb(1,j,k))/dzeta(k)-(hb(1,j,k)-hb(1,j,k-1))
     &/dzeta(k-1))/(dzeta(k)+dzeta(k-1))

       if(j.eq.1.or.j.eq.jmax)cgbeta=0.
       if(j.eq.1.or.j.eq.jmax)cfgb=0.
       if(j.eq.1.or.j.eq.jmax)cgfb=0.
       ccom(k)=-0.5*(bcb(j,k)+bcb(j,k+1))*(h(3,j,k+1)
     &-h(3,j,k))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     & +0.5*(bcb(j,k)+bcb(j,k-1))*(h(3,j,k)-h(3,j,k-1))
     & /(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     &-(m1(j)*hsb(1,j,k)+m6(j)*hsb(2,j,k))*ebeta
     & -ue(i,j)**2*(1.-1./pr)*(cfbeta+(vinf/ue(i,j))**2*cgbeta+vinf
     &*costh(i,j)*(cfgb+cgfb)/ue(i,j))/he
     & -m10(j)*(hb(1,j,k)*h(3,j,k)-ebeta*(hs(1,j,k)-hsp(1,j,k)))/dx
     & +m7(j)*(hb(2,j,k)*eby-ebeta*sgby)+m13(j)*ebeta
c        write(1,*)'i,k,j=',i,k,j,'corr',cfbeta,cgbeta,cfgb,cgfb,ccom(k)
 7151  continue
       ccom(kmax-1)=ccom(kmax-1)-acom(kmax-1)
       if(kaw.eq.1)dcom(2)=dcom(2)-bcom(2)*(dzeta(1)+dzeta(2))**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)

c      for the exact adiabatic wall b.c.
       if(kaw.eq.1)ccom(2)=ccom(2)-0.5*acom(2)*((dzeta(1)+dzeta(2))**2
     &*(hb(1,j,2)**2*cavd(i,j)**2)-dzeta(1)**2*hb(1,j,3)**2
     &*cavd(i,j)**2)/(he*(dzeta(1)**2-(dzeta(1)+dzeta(2))**2))

       if(kaw.eq.1)acom(2)=acom(2)+bcom(2)*dzeta(1)**2/(dzeta(1)**2
     & -(dzeta(1)+dzeta(2))**2)
       if(kaw.eq.0)ccom(2)=ccom(2)-bcom(2)*cp*twall(i,j)/he

       call sy(2,kmax-1,bcom,dcom,acom,ccom)

       do 7152 k=2,kmax-1
       h(3,j,k)=ccom(k)
 7152  continue
       h(3,j,kmax)=1.

       if(kaw.eq.1)h(3,j,1)=(dzeta(1)**2*h(3,j,3)-(dzeta(1)+dzeta(2))**2
     & *h(3,j,2))/(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
       if(kaw.eq.0)h(3,j,1)=cp*twall(i,j)/he

c        do 33 k=1,kmax
c 33     write(6,*)'corr i,j,k=',i,j,k,'h3=',h(3,j,k)

       do 7153 k=1,kmax
       td(j,k)=(he*h(3,j,k)-0.5*((ue(i,j)
     &*h(1,j,k))**2+(vinf*h(2,j,k))**2+2.*ue(i,j)*vinf*h(1,j,k)
     &*h(2,j,k)*costh(i,j)))/cp
       if(j.eq.1.or.j.eq.jmax)td(j,k)=(he*h(3,j,k)-0.5*(ue(i,j)
     &*h(1,j,k))**2)/cp

       if(td(j,k).lt.0)then
       write(6,*)' td(j,k) is lt.0 at i=',i,' j=',j,' k=',k
       iend=1
       return
       endif

 7153  continue
```

```
      do 7154 k=1,kmax
      roero(j,k)=td(j,k)/te(i,j)
      if(mks.eq.1)bc(j,k)=(te(i,j)/td(j,k))*((1.8*td(j,k))**1.5)
     &*(1.8*te(i,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(i,j))**1.5))
      if(mks.eq.0)bc(j,k)=(te(i,j)/td(j,k))*(td(j,k)**1.5)*(te(i,j)
     &+198.6)/((td(j,k)+198.6)*(te(i,j)**1.5))
 7154 continue

      return
      end
```

```
c#########################################################################

      subroutine inbub

c#########################################################################

      include 'comblck'

c     to calculate initial velocity profile at i=1

      do 35 j=1,jmax
      xp=xpd(1,j)
      yp=ypd(1,j)
      zp=zpd(1,j)

      ystar=yp
      xstar=cos(thetar)*(zp-zps)-sin(thetar)*(xp-xps)

      if(j.eq.1.or.j.eq.jmax)go to 81
      do 8 k=1,kmax
      h(1,j,k)=(-astar*xstar*hn(1,1,k)*cos(yd(j))+bstar*ystar*hn(2,1,k)
     &*sin(yd(j)))/(-astar*xstar*cos(yd(j))+bstar*ystar*sin(yd(j)))
      h(2,j,k)=(astar*xstar*hn(1,1,k)*sin(yd(j))+bstar*ystar
     &*hn(2,1,k)*cos(yd(j)))/vinf
      hs(1,j,k)=(-astar*xstar*hsn(1,1,k)*cos(yd(j))+bstar*ystar
     &*hsn(2,1,k)*sin(yd(j)))
     &/(-astar*xstar*cos(yd(j))+bstar*ystar*sin(yd(j)))
      hs(2,j,k)=(astar*xstar*hsn(1,1,k)*sin(yd(j))+bstar*ystar
     &*hsn(2,1,k)*cos(yd(j)))/vinf
 8    continue
      go to 135

 81   do 82 k=1,kmax
      h(1,j,k)=hn(1,1,k)
      hs(1,j,k)=hsn(1,1,k)
      h(2,j,k)=(-astar*abs(xstar)*hn(1,1,k)+bstar*h2(1,j)*hn(2,1,k))
     &/vinf
      hs(2,j,k)=(-astar*abs(xstar)*hsn(1,1,k)+bstar*h2(1,j)*hsn(2,1,k))
     &/vinf
 82   continue

 135  do 140 k=1,kmax
      h(3,j,k)=h(3,1,k)
      he=cp*te(1,j)+0.5*cavd(1,j)**2
      td(j,k)=(he*h(3,j,k)-0.5*(cavd(1,j)*h(1,j,k))**2)/cp
      roero(j,k)=td(j,k)/te(1,j)
      if(mks.eq.1)bc(j,k)=(te(1,j)/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*te(1,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(1,j))**1.5))
      if(mks.eq.0)bc(j,k)=(te(1,j)/td(j,k))*(td(j,k)**1.5)*(te(1,j)
     & +198.6)/((td(j,k)+198.6)*(te(1,j)**1.5))

 140  continue
 35   continue
      return
      end
```

```
c#######################################################################

       subroutine inbus

c#######################################################################

       include 'comblck'

c      to calculate initial velocity profile at i=1

       do 35 j=1,jmax
       xp=xpd(1,j)
       yp=ypd(1,j)
       zp=zpd(1,j)

       ystar=yp
       xstar=cos(thetar)*(zp-zps)-sin(thetar)*(xp-xps)

       if(j.eq.1.or.j.eq.jmax)go to 81
       do 8 k=1,kmax
       h(1,j,k)=(hn(1,1,k)+hn(2,1,k)*cstar**2*(ystar/xstar)**2)
     &           /(1.+cstar**2*(ystar/xstar)**2)
       hs(1,j,k)=(hsn(1,1,k)+hsn(2,1,k)*cstar**2*(ystar/xstar)**2)
     &            /(1.+cstar**2*(ystar/xstar)**2)

       h(2,j,k)=bstar*xstar*ystar*(hn(1,1,k)-hn(2,1,k))/(vinf
     &           *sqrt(xstar**2+cstar**2*ystar**2))
       hs(2,j,k)=bstar*xstar*ystar*(hsn(1,1,k)-hsn(2,1,k))
     &           /(vinf*sqrt(xstar**2+cstar**2*ystar**2))
 8     continue
       go to 135

 81    do 82 k=1,kmax
       h(1,j,k)=hn(1,1,k)
       hs(1,j,k)=hsn(1,1,k)
       h(2,j,k)=h2(1,j)*bstar*(hn(2,1,k)-hn(1,1,k))/vinf
       hs(2,j,k)=h2(1,j)*bstar*(hsn(2,1,k)-hsn(1,1,k))/vinf
 82    continue

 135   do 140 k=1,kmax
       h(3,j,k)=h(3,1,k)
       he=cp*te(1,j)+0.5*cavd(1,j)**2
       td(j,k)=(he*h(3,j,k)-0.5*(cavd(1,j)*h(1,j,k))**2)/cp
       roero(j,k)=td(j,k)/te(1,j)
       if(mks.eq.1)bc(j,k)=(te(1,j)/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*te(1,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(1,j))**1.5))
       if(mks.eq.0)bc(j,k)=(te(1,j)/td(j,k))*(td(j,k)**1.5)*(te(1,j)
     & +198.6)/((td(j,k)+198.6)*(te(1,j)**1.5))

 140   continue
 35    continue
       return
       end
```

```
c#########################################################################

      subroutine inpos

c#########################################################################

      include 'comblck'

c     to calculate initial velocity profile at i=1 based on the
c     streamline coordinate system

      do 35 j=2,jmax-1

      do 8 k=1,kmax
      h1t=h(1,j,k)
      hs1t=hs(1,j,k)
      h2t=h(2,j,k)
      hs2t=hs(2,j,k)
      h(1,j,k)=(ue(1,j)**2*h1t+ve(1,j)*vinf*h2t)
     &/(ue(1,j)**2+ve(1,j)**2)
      hs(1,j,k)=(ue(1,j)**2*hs1t+ve(1,j)*vinf*hs2t)
     &/(ue(1,j)**2+ve(1,j)**2)
      h(2,j,k)=(ue(1,j)*vinf*h2t-ue(1,j)*ve(1,j)*h1t)
     &/(vinf*sqrt(ue(1,j)**2+ve(1,j)**2))
      hs(2,j,k)=(ue(1,j)*vinf*hs2t-ue(1,j)*ve(1,j)*hs1t)
     &/(vinf*sqrt(ue(1,j)**2+ve(1,j)**2))
c      write(6,*)'inpos, j,k=',j,k,'h1=',h(1,j,k)
 8    continue
 35   continue

      do 9 k=1,kmax
      h(2,1,k)=h(2,2,k)/dy(1)
      hs(2,1,k)=hs(2,2,k)/dy(1)
      h(2,jmax,k)=h(2,jmax-1,k)/dy(jmax-1)
      hs(2,jmax,k)=hs(2,jmax-1,k)/dy(jmax-1)
 9    continue
      return
      end
```

```
c###########################################################################

      subroutine input

c###########################################################################
      include 'comblck'

      mks=1

      inc=0

      kpoint=0

      kbody=1

      kcpgivn=1

      kmax=16

      kaw=1

      krow=0

      ksymstg=1

      iw=80
      ini=50
      jni=1

      gamma=1.4

      if(mks.eq.0)rr=1716.
      if(mks.eq.1)rr=287.

      pr=0.72

      rminf=0.3

      pinf=101324

      tinf=288.

      ukmax1=0.9995

c     read the boundary-layer edge conditions from either BCC or SCC

      if(kbody.eq.1)then

      if(kpoint.eq.1.or.ksymstg.eq.1)go to 33

      rewind 25
      read(25,1112)xps,zps,thetar,astar,bstar,cstar
 33   rewind 22
      read(22,463)imax,jmax
      read(22,461)(xd(i),i=1,imax)
```

```
      read(22,461)(yd(j),j=1,jmax)
      do 60 i=1,imax
      do 60 j=1,jmax
      read(22,462)itr,itr,xpd(i,j),ypd(i,j),zpd(i,j),s1(i,j),ue(i,j)
     &,ve(i,j),h1(i,j),h2(i,j),costh(i,j),cpd(i,j)
 60   continue
 461  format(5(1x,e13.6))
 462  format(2i4,5(1x,e13.6)/8x,5(1x,e13.6))
 463  format(2i10)

      go to 1115
      endif

      if(kbody.eq.0)then
      rewind 25
      read(25,1112)xps,zps,thetar,astar,bstar,cstar
      read(25,463)imax,jmax
      read(25,461)(xd(i),i=1,imax)
      read(25,461)(yd(j),j=1,jmax)
      do 160 i=1,imax
      do 160 j=1,jmax
      read(25,464)itr,itr,xpd(i,j),ypd(i,j),zpd(i,j),s1(i,j),ue(i,j)
     &,ve(i,j),h2(i,j),cpd(i,j)
 160  continue
 464  format(2i4,4(1x,e14.7)/8x,4(1x,e14.7))
 1112 format(6e13.6)
      endif

 1115 continue

c     zeta distribution is specified

      dzetas=0.2
      zeta(1)=0.
      dzeta(1)=dzetas
      do 25 k=2,kmaxf
      dzeta(k)=dzetas
c      dzeta(k)=dzeta(k-1)*1.05
      zeta(k)=zeta(k-1)+dzeta(k)
 25   continue

c     wall condition is given if necessary

      if(krow.eq.0.and.kaw.eq.1)return
      if(krow.eq.0)go to 270
      do 176 i=1,imax
      do 176 j=1,jmax
      roww(i,j)=0.001
 176  continue
 270  if(kaw.eq.1)return
      do 276 i=1,imax
      do 276 j=1,jmax
      twall(i,j)=309.7
 276  continue

      return
      end
```

```
c#######################################################################

       subroutine insym

c#######################################################################

        include 'comblck'

c      to calculate initial velocity profile at i=1

       do 35 j=1,jmax

       do 8 k=1,kmax
       h(1,j,k)=hn(1,1,k)
       h(2,j,k)=hn(2,1,k)*ve(i,j)/vinf
       hs(1,j,k)=hsn(1,1,k)
       hs(2,j,k)=hsn(2,1,k)*ve(i,j)/vinf
       h(3,j,k)=h(3,1,k)
       he=cp*te(1,j)+0.5*cavd(1,j)**2
       td(j,k)=(he*h(3,j,k)-0.5*(cavd(1,j)*h(1,j,k))**2)/cp
       roero(j,k)=td(j,k)/te(1,j)
       if(mks.eq.1)bc(j,k)=(te(1,j)/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*te(1,j)+198.6)/((1.8*td(j,k)+198.6)*((1.8*te(1,j))**1.5))
       if(mks.eq.0)bc(j,k)=(te(1,j)/td(j,k))*(td(j,k)**1.5)*(te(1,j)
     & +198.6)/((td(j,k)+198.6)*(te(1,j)**1.5))
 8     continue

 35    continue
       return
       end
```

```
c###########################################################################

      subroutine ntrid

c     (block tridiagonal matrix eqn. solver)

c###########################################################################

      include 'comblck'
      dimension r(4,kmaxf),pil(4,kmaxf),p(4,kmaxf),den(kmaxf)
     &,cds(2,kmaxf)

      do 10 k=kmax-1,2,-1
      do 20 m=1,2
      do 20 lj=1,2
      l=m+2*(lj-1)
      r(l,k)=as(l,k)-ci(m,k)*es(2*lj-1,k+1)-ci(m+2,k)*es(2*lj,k+1)
      p(l,k)=bi(l,k)-ci(m,k)*e(2*lj-1,k+1)-ci(m+2,k)*e(2*lj,k+1)
     &+r(l,k)*dzeta(k-1)/2.
 20      continue

c     invert matrix p
      den(k)=p(1,k)*p(4,k)-p(2,k)*p(3,k)
      pil(1,k)=p(4,k)/den(k)
      pil(2,k)=-p(2,k)/den(k)
      pil(3,k)=-p(3,k)/den(k)
      pil(4,k)=p(1,k)/den(k)

      do 30 m=1,2
      cds(m,k)=ci(m,k)*ds(1,k+1)+ci(m+2,k)*ds(2,k+1)+di(m,k)
 30   continue

      do 40 m=1,2
      ds(m,k)=pil(m,k)*cds(1,k)+pil(m+2,k)*cds(2,k)
 40   continue

      do 50 m=1,2
      do 50 lj=1,2
      l=m+2.*(lj-1)
      es(l,k)=-pil(m,k)*r(2*lj-1,k)-pil(m+2,k)*r(2*lj,k)
      e(l,k)=pil(m,k)*ai(2*lj-1,k)+pil(m+2,k)*ai(2*lj,k)+es(l,k)
     &       *dzeta(k-1)/2.
 50   continue
 10   continue

      do 60 k=2,kmax
      do 60 m=1,2
      hn(m,j,k)=e(m,k)*hn(1,j,k-1)+e(m+2,k)*hn(2,j,k-1)
     &           +es(m,k)*hsn(1,j,k-1)+es(m+2,k)*hsn(2,j,k-1)+ds(m,k)
      hsn(m,j,k)=hsn(m,j,k-1)+(hn(m,j,k)+hn(m,j,k-1))*dzeta(k-1)/2.
 60   continue
      return
      end
```

```
c########################################################################
        subroutine output
c########################################################################
       include 'comblck'

       il=i
       if(il.gt.imax)il=imax

       rewind 30
       write(30,*)'    '
       write(30,*)'    '
       write(30,*)'    '
       write(30,*)'    '
       write(30,*)'********** input  echo  *********************'
       write(30,*)'    '
       write(30,*)'mks=',mks
       write(30,*)'inc=',inc
       write(30,*)'kpoint=',kpoint
       write(30,*)'kbody=',kbody
       write(30,*)'kcpgivn=',kcpgivn
       write(30,*)'kmax=',kmax
       write(30,*)'kaw=',kaw
       write(30,*)'krow=',krow
       write(30,*)'ksymstg=',ksymstg
       write(30,*)'iw=',iw
       write(30,*)'ini=',ini
       write(30,*)'jni=',jni
       write(30,*)'gamma=',gamma
       write(30,*)'rr=',rr
       write(30,*)'pr=',pr
       write(30,*)'rminf=',rminf
       write(30,*)'pinf=',pinf
       write(30,*)'tinf=',tinf
       write(30,*)'ukmax1=',ukmax1

       write(30,*)'    '
       write(30,*)'********** other free-stream conditions **********'
       write(30,*)'    '
       write(30,*)'cp=',cp
       write(30,*)'roinf=',roinf
       write(30,*)'rmyuinf=',rmyuinf
       write(30,*)'rnuinf=',rnuinf
       write(30,*)'ss=',ss
       write(30,*)'vinf=',vinf

       write(30,*)'    '
       write(30,*)'********************'
       write(30,*)'    '

       write(30,*)'il=',il
       write(30,*)'jmax1=',jmax1

       write(30,*)'    '
       if(ksep.eq.0)write(30,*)'the flow is not separated yet '
       if(ksep.eq.1)write(30,*)'the flow is separated at i=',il
```

```
      write(30,*)'    '
      write(30,*)'******* velocity profiles ********** '

      do 400 j=1,jmax

      write(30,*)'   '
      write(30,*)'    '
      write(30,401)il,j,xpd(il,j),ypd(il,j),zpd(il,j)
 401  format(' i=',i5,3x,' j=',i5,3x,'(xp=',f10.4,2x,'yp=',f10.4,
     &2x,'zp=',f10.4,' )')
      write(30,*)'   '
      if(j.eq.1.or.j.eq.jmax)then
      write(30,402)
 402  format(2(3x,'k',2x,'zeta',4x,'u/ue',3x,'vy/vinf',4x,'t/te'))
      else
      write(30,403)
 403  format(2(3x,'k',2x,'zeta',4x,'u/ue',4x,'v/vinf',4x,'t/te'))
       endif

      write(30,*)'   '
      write(30,404)(k,zeta(k),h(1,j,k),h(2,j,k),roero(j,k),k=1,kmax)
 400     continue

 404   format(2(1x,i3,f6.2,f9.5,e10.3,f7.4))

      write(30,*)'   '
      write(30,*)'******* boundary-layer parameters********** '
      write(30,*)' '
      if(kaw.eq.0)write(30,405)
 405  format(3x,'i',1x,2x,'j',6x,'xpd',11x,'ypd',11x,'zpd',11x,'cfx'
     &,11x,'cfy',/14x,'blth',10x,'dspth',9x,'thmom',10x,'qw')
      if(kaw.eq.1)write(30,406)
 406  format(3x,'i',1x,2x,'j',6x,'xpd',11x,'ypd',11x,'zpd',11x,'cfx'
     &,11x,'cfy',/14x,'blth',10x,'dspth',9x,'thmom',8x,'twall')
      write(30,*)' '
       do 560 i=1,il
       do 560 j=1,jmax
       if(kaw.eq.0)write(30,561)i,j,xpd(i,j),ypd(i,j),zpd(i,j),cfx(i,j)
     &,cfy(i,j),blth(i,j),dspth(i,j),thmom(i,j),qw(i,j)
       if(kaw.eq.1)write(30,561)i,j,xpd(i,j),ypd(i,j),zpd(i,j),cfx(i,j)
     &,cfy(i,j),blth(i,j),dspth(i,j),thmom(i,j),twall(i,j)
 560  continue
 561  format(2i4,5(1x,e13.6)/8x,4(1x,e13.6))

c      rewind 40
c      do 41 i=1,il
c      do 41 j=1,jmax
c      write(40,*)twall(i,j)
c 41   continue

      return
      end
```

```
c######################################################################

      subroutine predict

c######################################################################

       include 'comblck'
       dimension tb(jmaxf,kmaxf)

       he=cp*te(i,j)+0.5*cavd(i,j)**2

 255  do 257 k=1,kmax
      b1(j,k)=bc(j,k)
      b2(j,k)=0.
      b3(j,k)=0.
      b4(j,k)=bc(j,k)

 257  continue

c--------------------------------------------------------------------

c     to calculate ai, bi, ci, ai, and di

c---------------------------------------------------------------------

 256  do 5110 k=2,kmax-1

      de=dzeta(k)+dzeta(k-1)

      ai(1,k)=-(b1(j,k)+b1(j,k-1))/(de*dzeta(k-1))
      ai(2,k)=-(b2(j,k)+b2(j,k-1))/(de*dzeta(k-1))
      ai(3,k)=-(b3(j,k)+b3(j,k-1))/(de*dzeta(k-1))
      ai(4,k)=-(b4(j,k)+b4(j,k-1))/(de*dzeta(k-1))

      ci(1,k)=-(b1(j,k)+b1(j,k+1))/(de*dzeta(k))
      ci(2,k)=-(b2(j,k)+b2(j,k+1))/(de*dzeta(k))
      ci(3,k)=-(b3(j,k)+b3(j,k+1))/(de*dzeta(k))
      ci(4,k)=-(b4(j,k)+b4(j,k+1))/(de*dzeta(k))

      bi(1,k)=-((b1(j,k)+b1(j,k+1))/dzeta(k)+(b1(j,k)+b1(j,k-1))
     &/dzeta(k-1))/de-m10(j)*h(1,j,k)/dxh
      bi(2,k)=-((b2(j,k)+b2(j,k+1))/dzeta(k)+(b2(j,k)+b2(j,k-1))
     &/dzeta(k-1))/de
      bi(3,k)=-((b3(j,k)+b3(j,k+1))/dzeta(k)+(b3(j,k)+b3(j,k-1))
     &/dzeta(k-1))/de
      bi(4,k)=-((b4(j,k)+b4(j,k+1))/dzeta(k)+(b4(j,k)+b4(j,k-1))
     &/dzeta(k-1))/de-m10(j)*h(1,j,k)/dxh

      feta=(h(1,j,k+1)-h(1,j,k-1))/(dzeta(k)+dzeta(k-1))
      geta=(h(2,j,k+1)-h(2,j,k-1))/(dzeta(k)+dzeta(k-1))

      as(1,k)=m10(j)*feta/dxh
      as(2,k)=m10(j)*geta/dxh
      as(3,k)=0
      as(4,k)=0

      di(1,k)=-m1(j)*hs(1,j,k)*feta+m2(j)*h(1,j,k)**2+m5(j)*h(1,j,k)
     &*h(2,j,k)-m6(j)*hs(2,j,k)*feta+m8(j)*h(2,j,k)**2-m11(j)*roero(j,k)
     &+m13(j)*feta
      di(2,k)=-m1(j)*hs(1,j,k)*geta+m4(j)*h(1,j,k)*h(2,j,k)
```

```
     &+m3(j)*h(2,j,k)**2-m6(j)*hs(2,j,k)*geta+m9(j)*h(1,j,k)**2
     &-m12(j)*roero(j,k)+m13(j)*geta

       if(j.eq.1.or.j.eq.jmax)go to 9500

       db=dy(j)+dy(j-1)

       if(j.eq.2)save(1,k)=h(2,j-1,k)
       if(j.eq.2)save(2,k)=hs(2,j-1,k)
       if(j.eq.jmax-1)save(3,k)=h(2,j+1,k)
       if(j.eq.jmax-1)save(4,k)=hs(2,j+1,k)

       if(j.eq.2)h(2,j-1,k)=0
       if(j.eq.2)hs(2,j-1,k)=0
       if(j.eq.jmax-1)h(2,j+1,k)=0
       if(j.eq.jmax-1)hs(2,j+1,k)=0

       fcap=h(1,j,k)
       sfcap=hs(1,j,k)
       gcap=h(2,j,k)

       fy=((dy(j-1)/dy(j))*(h(1,j+1,k)-h(1,j,k))
     &+(dy(j)/dy(j-1))*(h(1,j,k)-h(1,j-1,k)))/db
       sgy=((dy(j-1)/dy(j))*(hs(2,j+1,k)-hs(2,j,k))+(dy(j)
     &/dy(j-1))*(hs(2,j,k)-hs(2,j-1,k)))/db
       gy=((dy(j-1)/dy(j))*(h(2,j+1,k)-h(2,j,k))
     &+(dy(j)/dy(j-1))*(h(2,j,k)-h(2,j-1,k)))/db

       if(kterm.eq.1.and.j.eq.jmaxt)then
       fy=(3.*h(1,j,k)-4.*h(1,j-1,k)+h(1,j-2,k))/(2.*dy(j-1))
       gy=(3.*h(2,j,k)-4.*h(2,j-1,k)+h(2,j-2,k))/(2.*dy(j-1))
       sgy=(3.*hs(2,j,k)-4.*hs(2,j-1,k)+hs(2,j-2,k))/(2.*dy(j-1))
       endif

       di(1,k)=di(1,k)+m10(j)*h(1,j,k)*(-fcap)/dxh
     &+m10(j)*feta*sfcap/dxh+m7(j)*(h(2,j,k)*fy-feta*sgy)

       di(2,k)=di(2,k)+m10(j)*h(1,j,k)*(-gcap)/dxh
     &+m10(j)*geta*sfcap/dxh+m7(j)*(h(2,j,k)*gy-geta*sgy)

       if(j.eq.2)h(2,j-1,k)=save(1,k)
       if(j.eq.2)hs(2,j-1,k)=save(2,k)
       if(j.eq.jmax-1)h(2,j+1,k)=save(3,k)
       if(j.eq.jmax-1)hs(2,j+1,k)=save(4,k)

       go to 5110

9500   di(1,k)=di(1,k)+m10(j)*h(1,j,k)*(-h(1,j,k))/dxh
     &+m10(j)*hs(1,j,k)*feta/dxh
       di(2,k)=di(2,k)+m10(j)*h(1,j,k)*(-h(2,j,k))/dxh
     &+m10(j)*hs(1,j,k)*geta/dxh

5110   continue

       do 5140 m=1,4
       e(m,kmax)=0
       es(m,kmax)=0
5140   continue
       ds(1,kmax)=1.0
       ds(2,kmax)=ve(i,j)/vinf
```

```
      if(j.eq.1)ds(2,kmax)=ve(i,2)/(vinf*dy(1))
      if(j.eq.jmax)ds(2,kmax)=-ve(i,jmax-1)/(vinf*dy(jmax-1))
      do 5160 m=1,2
      hn(m,j,1)=0
      hsn(m,j,1)=0
 5160 continue

c-------------------------------------------------------------------

c     To solve the block tridiagonal matrix equation, call ntrid

c-------------------------------------------------------------------

      call ntrid

      do 5200 k=1,kmax
      do 5200 m=1,2
      hb(m,j,k)=hn(m,j,k)
      hsb(m,j,k)=hsn(m,j,k)
 5200 continue

      if(inc.eq.1)then
      do 5250 k=1,kmax
      hb(3,j,k)=1.0
      roerob(j,k)=1.0
      bcb(j,k)=1.0
 5250 continue
      return
      endif

      do 6151 k=2,kmax-1
      bcom(k)=(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
      dcom(k)=-(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     & -(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     & -m10(j)*h(1,j,k)/dxh
      acom(k)=(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
      feta=((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))/(dzeta(k)+dzeta(k-1))
      geta=((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))/(dzeta(k)+dzeta(k-1))
      eeta=((dzeta(k-1)/dzeta(k))*(h(3,j,k+1)-h(3,j,k))+(dzeta(k)
     &/dzeta(k-1))*(h(3,j,k)-h(3,j,k-1)))/(dzeta(k)+dzeta(k-1))

      if(j.eq.1.or.j.eq.jmax)then
      sgy=0.
      ey=0.
      go to 222
      endif

      sgy=((dy(j-1)/dy(j))*(hs(2,j+1,k)-hs(2,j,k))+(dy(j)
     &/dy(j-1))*(hs(2,j,k)-hs(2,j-1,k)))/(dy(j)+dy(j-1))
      if(j.eq.2)sgy=((dy(j-1)/dy(j))*(hs(2,j+1,k)-hs(2,j,k))+(dy(j)
     &/dy(j-1))*hs(2,j,k))/(dy(j)+dy(j-1))
      if(j.eq.jmax-1)sgy=((dy(j-1)/dy(j))*(-hs(2,j,k))+(dy(j)
     &/dy(j-1))*(hs(2,j,k)-hs(2,j-1,k)))/(dy(j)+dy(j-1))
      ey=((dy(j-1)/dy(j))*(h(3,j+1,k)-h(3,j,k))
     &+(dy(j)/dy(j-1))*(h(3,j,k)-h(3,j-1,k)))/(dy(j)+dy(j-1))

 222  if(kterm.eq.1.and.j.eq.jmaxt)then
      sgy=(3.*hs(2,j,k)-4.*hs(2,j-1,k)+hs(2,j-2,k))/(2.*dy(j-1))
```

```
      ey=(3.*h(3,j,k)-4.*h(3,j-1,k)+h(3,j-2,k))/(2.*dy(j-1))
      endif

      cfeta=h(1,j,k)*feta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*feta**2+bc(j,k)*h(1,j,k)*2.*((h(1,j,k+1)-h(1,j,k))/dzeta
     &(k)-(h(1,j,k)-h(1,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cgeta=h(2,j,k)*geta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*geta**2+bc(j,k)*h(2,j,k)*2.*((h(2,j,k+1)-h(2,j,k))/dzeta
     &(k)-(h(2,j,k)-h(2,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cfg=h(1,j,k)*geta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*feta*geta+bc(j,k)*h(1,j,k)*2.*((h(2,j,k+1)-h(2,j,k))
     &/dzeta(k)-(h(2,j,k)-h(2,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))
      cgf=h(2,j,k)*feta*(bc(j,k+1)-bc(j,k-1))/(dzeta(k)+dzeta(k-1))
     & +bc(j,k)*feta*geta+bc(j,k)*h(2,j,k)*2.*((h(1,j,k+1)-h(1,j,k))
     &/dzeta(k)-(h(1,j,k)-h(1,j,k-1))/dzeta(k-1))/(dzeta(k)+dzeta(k-1))

      if(j.eq.1.or.j.eq.jmax)cgeta=0.
      if(j.eq.1.or.j.eq.jmax)cfg=0.
      if(j.eq.1.or.j.eq.jmax)cgf=0.
      ecap=h(3,j,k)
      sfcap=hs(1,j,k)
      sgcap=hs(2,j,k)
      ccom(k)=-(m1(j)*hs(1,j,k)+m6(j)*hs(2,j,k))*eeta
     & -ue(i,j)**2*(1.-1./pr)*(cfeta+cgeta*(vinf/ue(i,j))**2+costh(i,j)
     &*vinf*(cfg+cgf)/ue(i,j))/he
     & -m10(j)*(h(1,j,k)*ecap-eeta*(hsb(1,j,k)-sfcap))/dxh
     & +m7(j)*(h(2,j,k)*ey-eeta*sgy)+m13(j)*eeta
C       write(1,*)'i,k,j=',i,k,j,'pred',cfeta,cgeta,cfg,cgf,ccom(k)
 6151 continue
      ccom(kmax-1)=ccom(kmax-1)-acom(kmax-1)
      if(kaw.eq.1)dcom(2)=dcom(2)-bcom(2)*(dzeta(1)+dzeta(2))**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)

c     for the exact adiabatic wall b.c.
      if(kaw.eq.1)ccom(2)=ccom(2)-0.5*acom(2)*((dzeta(1)+dzeta(2))**2
     &*(h(1,j,2)**2*cavd(i,j)**2)-dzeta(1)**2*h(1,j,3)**2*cavd(i,j)**2)
     &/(he*(dzeta(1)**2-(dzeta(1)+dzeta(2))**2))

      if(kaw.eq.1)acom(2)=acom(2)+bcom(2)*dzeta(1)**2/(dzeta(1)**2
     & -(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)ccom(2)=ccom(2)-bcom(2)*cp*twall(i,j)/he

      call sy(2,kmax-1,bcom,dcom,acom,ccom)

      do 6152 k=2,kmax-1
      hb(3,j,k)=ccom(k)
 6152 continue
      hb(3,j,kmax)=1.

      if(kaw.eq.1)hb(3,j,1)=(dzeta(1)**2*hb(3,j,3)-(dzeta(1)+dzeta(2))
     &**2*hb(3,j,2))/(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)hb(3,j,1)=cp*twall(i,j)/he

c       do 6159 k=1,kmax
c       write(6,*)'i,j,k=',i,j,k,'hb3=',hb(3,j,k)
c 6159  continue

      do 6153 k=1,kmax
      tb(j,k)=(he*hb(3,j,k)-0.5*((ue(i,j)
     &*hb(1,j,k))**2+(vinf*hb(2,j,k))**2+2.*ue(i,j)*vinf*hb(1,j,k)
```

```
     &*hb(2,j,k)*costh(i,j)))/cp
      if(j.eq.1.or.j.eq.jmax)tb(j,k)=(he*hb(3,j,k)-0.5*(ue(i,j)
     &*hb(1,j,k))**2)/cp
      roerob(j,k)=tb(j,k)/te(i,j)

      if(roerob(j,k).lt.0)then
      write(6,*)' roerob(j,k) is lt.0 at i=',i,' j=',j,' k=',k
      iend=1
      return
      endif

6153  continue

      do 6157 k=1,kmax
      if(mks.eq.1)bcb(j,k)=(te(i,j)/tb(j,k))*((1.8*tb(j,k))**1.5)
     & *(1.8*te(i,j)+198.6)/((1.8*tb(j,k)+198.6)*((1.8*te(i,j))**1.5))
      if(mks.eq.0)bcb(j,k)=(te(i,j)/tb(j,k))*(tb(j,k)**1.5)*(te(i,j)
     &+198.6)/((tb(j,k)+198.6)*(te(i,j)**1.5))
6157  continue

      return
      end
```

```
c#########################################################################

      subroutine profile

c#########################################################################

      include 'comblck'

      icheck=(i/ini)*ini-i
      if(icheck.eq.0)then
      write(iw,*)' '
      write(iw,*)' profiles (i, j, k, u/ue, v/vinf (vy/vinf if j=1 or
     & j=jmax), t/te, z (ft))'
      write(iw,*)' '

      do 71 j=1,jmaxt,jni
      write(iw,*)kmax
      write(iw,72)(i,j,k,h(1,j,k),h(2,j,k),roero(j,k),zact(j,k)
     &,k=1,kmax)
 71   continue
      endif
 72   format(3i5,4e13.6)
      return
      end
```

```
c#########################################################################

      subroutine stagpt

c#########################################################################

      include 'comblck'

c-----------------------------------------------------------------------
c
c      the stagnation point solution
c
c      (blottner's iterative method is used)
c
c-----------------------------------------------------------------------
       write(6,*)' cstar=',cstar

       j=1

       testag=tinf*(1.+0.5*(gamma-1.)*rminf**2)
       he=cp*testag

 4100  kmax=kmax+1
       write(6,*)' kmax=',kmax,' zeta(kmax)=',zeta(kmax)

      do 1030 m=1,2
      h(m,j,1)=0
      hs(m,j,1)=0
      do 1030 k=2,kmax
      h(m,j,k)=1.
      hs(m,j,k)=hs(m,j,k-1)+(h(m,j,k)+h(m,j,k-1))*dzeta(k-1)/2.
 1030 continue

      it=0
      do 1123 k=1,kmax
      bc(j,k)=1.
 1123 roero(j,k)=1.

 1170 it=it+1
      if(it.gt.10)write(6,*)' iteration for stag. pt. soln is gt.10'
      if(it.gt.10)stop
      do 1110 k=2,kmax-1
      ai(1,k)=(hs(1,j,k)+cstar*hs(2,j,k))/(dzeta(k)+dzeta(k-1))
     &        *(dzeta(k)/dzeta(k-1))
     &      -(bc(j,k)+bc(j,k-1))/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
      ai(2,k)=0
      ai(3,k)=0
      ai(4,k)=ai(1,k)

      ci(1,k)=-(hs(1,j,k)+cstar*hs(2,j,k))/(dzeta(k)+dzeta(k-1))
     &        *(dzeta(k-1)/dzeta(k))
     &      -(bc(j,k)+bc(j,k+1))/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
      ci(2,k)=0
      ci(3,k)=0
      ci(4,k)=ci(1,k)

      bi(1,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &          /dzeta(k-1))/(dzeta(k)+dzeta(k-1))-2.*h(1,j,k)
     &          -(cstar*hs(2,j,k)+hs(1,j,k))*(dzeta(k-1)/dzeta(k))
```

```
     &           /(dzeta(k)+dzeta(k-1))
     &           +(cstar*hs(2,j,k)+hs(1,j,k))*(dzeta(k)/dzeta(k-1))
     &           /(dzeta(k)+dzeta(k-1))

      bi(4,k)=-((bc(j,k)+bc(j,k+1))/dzeta(k)+(bc(j,k)+bc(j,k-1))
     &           /dzeta(k-1))/(dzeta(k)+dzeta(k-1))-2.*cstar*h(2,j,k)
     &           -(cstar*hs(2,j,k)+hs(1,j,k))*(dzeta(k-1)/dzeta(k))
     &           /(dzeta(k)+dzeta(k-1))
     &           +(cstar*hs(2,j,k)+hs(1,j,k))*(dzeta(k)/dzeta(k-1))
     &           /(dzeta(k)+dzeta(k-1))

      bi(2,k)=0
      bi(3,k)=0

      do 1115 m=1,2

      as(m,k)=((dzeta(k-1)/dzeta(k))*(h(m,j,k+1)-h(m,j,k))
     &            +(dzeta(k)/dzeta(k-1))*(h(m,j,k)-h(m,j,k-1)))
     &            /(dzeta(k)+dzeta(k-1))

      as(m+2,k)=cstar*((dzeta(k-1)/dzeta(k))*(h(m,j,k+1)-h(m,j,k))
     &            +(dzeta(k)/dzeta(k-1))*(h(m,j,k)-h(m,j,k-1)))
     &            /(dzeta(k)+dzeta(k-1))

1115       continue
      di(1,k)=-roero(j,k)-h(1,j,k)**2+(hs(1,j,k)+cstar*hs(2,j,k))*
     &            ((dzeta(k-1)/dzeta(k))*(h(1,j,k+1)-h(1,j,k))
     &            +(dzeta(k)/dzeta(k-1))*(h(1,j,k)-h(1,j,k-1)))
     &            /(dzeta(k)+dzeta(k-1))

      di(2,k)=-cstar*(roero(j,k)+h(2,j,k)**2)+(cstar*hs(2,j,k)
     &+hs(1,j,k))*((dzeta(k-1)/dzeta(k))*(h(2,j,k+1)-h(2,j,k))
     &            +(dzeta(k)/dzeta(k-1))*(h(2,j,k)-h(2,j,k-1)))
     &            /(dzeta(k)+dzeta(k-1))

1110       continue

      do 1120 m=1,4
      e(m,kmax)=0
      es(m,kmax)=0
1120  continue
      do 1130 m=1,2
      ds(m,kmax)=1.0
1130  continue
      do 1140 m=1,2
      hn(m,j,1)=0
      hsn(m,j,1)=0
1140  continue

      call ntrid

      do 1145 k=2,kmax-1
      er=(h(1,j,k)-hn(1,j,k))/h(1,j,k)
      if(abs(er)-1.d-4)1145,1145,1146
1145  continue

      go to 1160

1146  do 1150 m=1,2
      do 1150 k=1,kmax
```


[redacted] C-2

```
      h(m,j,k)=hn(m,j,k)
      hs(m,j,k)=hsn(m,j,k)
 1150 continue
 1100 continue

      do 1151 k=2,kmax-1
      bcom(k)=(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     & -(hs(1,j,k)+cstar*hs(2,j,k))*dzeta(k)
     & /(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
      dcom(k)=-(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     & -(bc(j,k)+bc(j,k-1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k-1))
     &+(hs(1,j,k)+cstar*hs(2,j,k))*dzeta(k)
     &/(dzeta(k-1)*(dzeta(k)+dzeta(k-1)))
     &-(hs(1,j,k)+cstar*hs(2,j,k))*dzeta(k-1)
     &/(dzeta(k)*(dzeta(k)+dzeta(k-1)))

      acom(k)=(bc(j,k)+bc(j,k+1))/(pr*(dzeta(k)+dzeta(k-1))*dzeta(k))
     &+(hs(1,j,k)+cstar*hs(2,j,k))*dzeta(k-1)
     &/(dzeta(k)*(dzeta(k)+dzeta(k-1)))
      ccom(k)=0.
 1151 continue
      ccom(kmax-1)=-acom(kmax-1)

      if(kaw.eq.1)dcom(2)=dcom(2)-bcom(2)*(dzeta(1)+dzeta(2))**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.1)acom(2)=acom(2)+bcom(2)*dzeta(1)**2
     & /(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)ccom(2)=-bcom(2)*twall(1,j)/testag

      call sy(2,kmax-1,bcom,dcom,acom,ccom)

      do 1152 k=2,kmax-1
 1152 h(3,j,k)=ccom(k)
      h(3,j,kmax)=1.

      if(kaw.eq.1)h(3,j,1)=(dzeta(1)**2*h(3,j,3)-(dzeta(1)+dzeta(2))**2
     & *h(3,j,2))/(dzeta(1)**2-(dzeta(1)+dzeta(2))**2)
      if(kaw.eq.0)h(3,j,1)=twall(1,j)/testag

      do 1153 k=1,kmax
      td(j,k)=testag*h(3,j,k)
      roero(j,k)=td(j,k)/testag
      if(mks.eq.1)bc(j,k)=(testag/td(j,k))*((1.8*td(j,k))**1.5)
     & *(1.8*testag+198.6)/((1.8*td(j,k)+198.6)*((1.8*testag)**1.5))
      if(mks.eq.0)bc(j,k)=(testag/td(j,k))*(td(j,k)**1.5)*(testag
     &+198.6)/((td(j,k)+198.6)*(testag**1.5))
 1153 continue
      go to 1170

 1160 if(h(1,j,kmax-1).lt.ukmax1)go to 4100
      write(6,*)' it=',it
      return
      end
```

```
c#########################################################################

      subroutine sy(il,iu,bb,dd,aa,cc)

c     (tridiagonal matrix eqn. solver)

c#########################################################################

      dimension aa(1),bb(1),cc(1),dd(1)
      lp=il+1
      do 10 i=lp,iu
      r=bb(i)/dd(i-1)
      dd(i)=dd(i)-r*aa(i-1)
      cc(i)=cc(i)-r*cc(i-1)
 10   continue
      cc(iu)=cc(iu)/dd(iu)
      do 20 i=lp,iu
      k=iu-i+il
 20   cc(k)=(cc(k)-aa(k)*cc(k+1))/dd(k)
      return
      end
```

PART 2.

BODY-ORIENTED COORDINATE PROGRAM (BCC)

2.1 Program Description

Program BCC is used for the generation of the boundary-layer edge conditions based on the body-oriented coordinates for the general fuselage. This code reads the numerical inviscid solution based on the Cartesian coordinates (x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, Cp) on the inviscid grid and calculates the boundary-layer edge conditions (x', y', z', u_e/V_∞, v_e/V_∞, s, h_1, h_2, $\cos\theta$, Cp) on the body-oriented boundary-layer grid.

A geometry program which defines the fuselage is required to run the BCC code. This code is written to be generally applied, so any geometry routine, which returns the body radius r for given axial coordinate X and angle ϕ, can be used. Because the raw data defining the sample case general aviation fuselage were nonsmooth, a semi-analytic geometry program specially made for this fuselage by Raymond L. Barger at the NASA Langley Research Center is used. It should be noted that the angle ϕ must be defined as $-\pi/2$ and $\pi/2$ on the windward and leeward lines of symmetry, respectively, in this geometry routine.

To obtain the boundary-layer edge conditions on the boundary-layer grid, BCC uses a bidirectional cubic spline-under-tension interpolation subroutine. There is no subroutine other than those related to the geometry and the interpolation. All the input and output are given or written by the main program.

2.2 Structure of Main Program BCMAIN

The flow chart for the main program BCMAIN is presented in Figure 4. The x and y distribution for the boundary-layer grid are given in the main program BCMAIN. The numerical inviscid solution based on the Cartesian coordinates (x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, Cp) on the inviscid grid are read. Then, $\cos\theta$ on the inviscid grid is calculated using the geometry programs. The calculation of u_e/V_∞ and v_e/V_∞ on the inviscid grid follows. The inviscid properties (u_e/V_∞, Cp) along the lines of symmetry are extrapolated. Then u_e/V_∞, v_e/V_∞, $\cos\theta$, Cp on the boundary-layer grid are interpolated using a bidirectional cubic spline-under-tension interpolation subroutine. The calculations of x', y', z', s, h_1, and h_2 on the boundary-layer grid follow next. Finally, the the boundary-layer edge conditions (x', y', z', u_e/V_∞, v_e/V_∞, s, h_1, h_2, $\cos\theta$, Cp) on the body-oriented boundary-layer grid are written in the file fort.22.

Parameters IM and JM provide the flexibility of changing the dimensions of the inviscid grid to be read. The parameter IM must be the same as the number of the inviscid grid points in the streamwise direction, i.e., IM=NT. The parameter JM should be the number of the inviscid grid points in the crosswise direction plus 2, i.e., JM=NP+2. Parameters IMAXD and JMAXD provide the flexibility of changing the dimensions of the boundary-layer grid in the streamwise and crosswise directions. IMAXD may be different from IMAX, but must be greater or equal to IMAX. Also, JMAXD may be different from JMAX, but must be greater or equal to JMAX. The dimensions of the variable arrays are controlled by changing these parameters, IM, JM, IMAXD, and JMAXD.

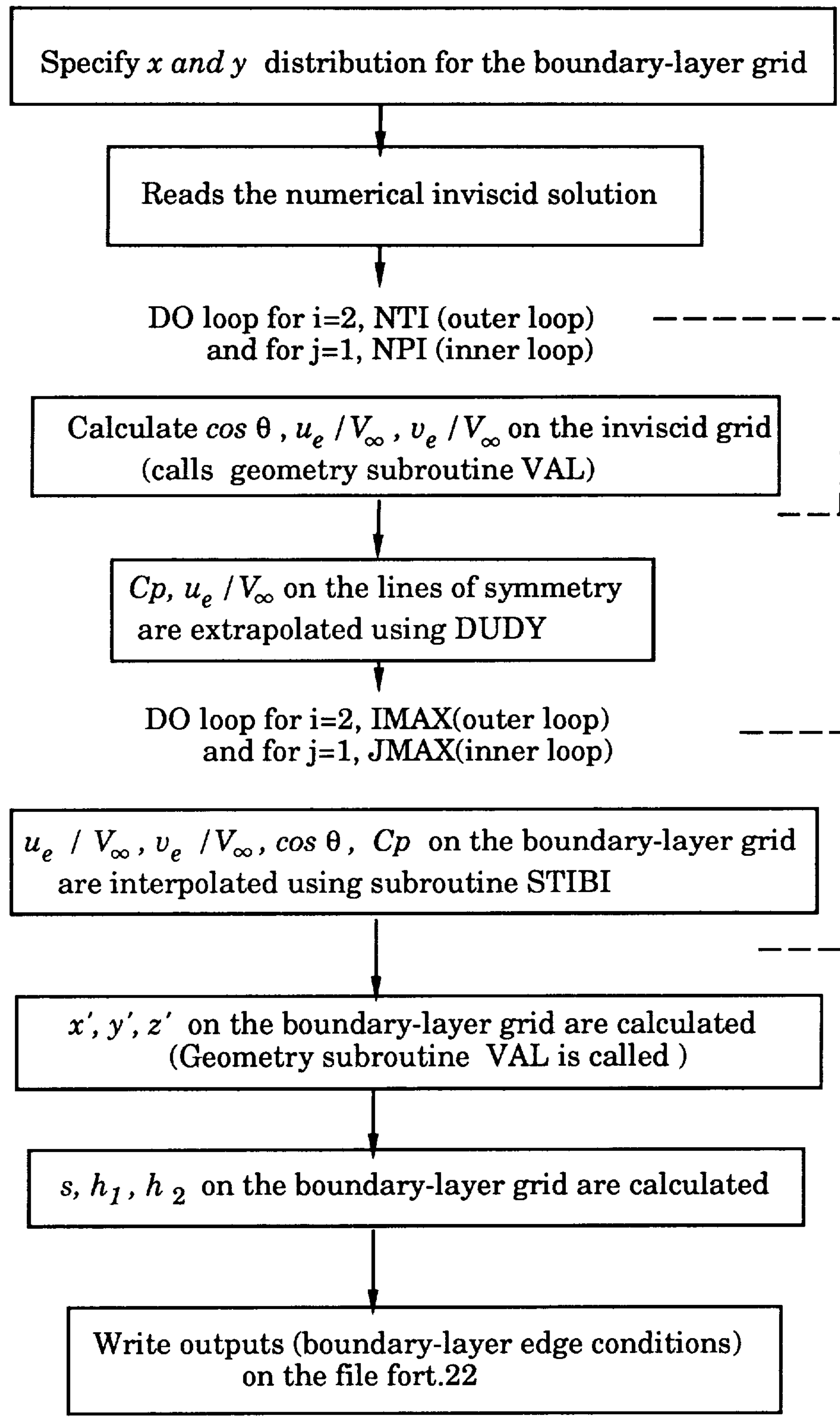


Fig. 4. Flow Chart for the Main Program BCMAIN.

2.3 Subroutine Description

Subroutine DUDY(X1, X2, X3, Y1, Y2, Y3)

- Called by main program BCMAIN.
- Used to obtain the inviscid properties (u_e/V_∞ and Cp) on the lines of symmetry.
- Calculates Y3 at X3 by the second order Lagrangian extrapolation, utilizing the symmetry condition at X3 and with given (X1,Y1) and (X2, Y2).

Subroutine STIBI

- Called by the main program BCMAIN.
- Avaiable as a mathematical library routine at NASA Langley Research Center.
- Interpolates the spline-under-tension approximation to one function of two independent variables. Input values of the function are specified at all nodes of a rectangular grid. Output values may be requested at one or more individual points or at all nodes of a second rectangular grid. In BCC, the two independent variables for the interpolation are X and ϕ.

Subroutine VAL(X, ϕ, r, rx, N)

- Called by main program BCMAIN.
- This is a geometry subroutine to interrogate the radius(r) for a given X and ϕ.
- Output: r only if N=0; r and $rx(= \partial r/\partial x)$ if N=1
- Must be supplied by the user.

2.4 Parameter and Variable Directory

CAVT(I,J)	V_e/V_∞, inviscid total velocity on the inviscid grid
COSTH(I,J)	$\cos\theta$ on the boundary-layer grid
COSTHT(I,J)	$\cos\theta$ on the inviscid grid
CPD(I,J)	Cp, pressure coefficient on the boundary-layer grid
DXPDY	$\partial x'/\partial y$
DY(J)	Δy
DYPDY	$\partial y'/\partial y$
DZPDY	$\partial z'/\partial y$
ENDX1,ENDXN,ENDY1,ENDYN,ENDXY	Arguments of the interpolation subroutine STIBI
H1(I,J)	h_1 on the boundary-layer grid
H2(I,J)	h_2 on the boundary-layer grid
IM	number of inviscid grid points in the x-direction, including the nose point
IMAX	actual number of boundary-layer grid points in the x-direction
IMAXD	maximum possible number of boundary-layer grid points in the x-direction (given as a parameter, IMAXD$\geq$IMAX)
IENDSW, IERR, IOPT, IW, LINOUT	Arguments of the interpolation subroutine STIBI
JM	number of inviscid grid points in the y-direction, including the lines of symmetry
JMAX	actual number of boundary-layer grid points in the y-direction
JMAXD	maximum possible number of boundary-layer grid points in the y-direction (given as a parameter, JMAXD$\geq$JMAX)
NP	number of inviscid grid points in the y-direction in the numerical inviscid data
NPI	number of inviscid grid points in the y-direction, including the lines of

	symmetry (NPI=NP+2)
NT	number of inviscid grid points in the x-direction in the numerical inviscid data
NTI	number of inviscid grid points in the x-direction (NTI=NT)
PI	π
PCOEF(I,J)	Cp, pressure coefficient on the inviscid grid
PHIT(J)	ϕ on the inviscid grid
RX,RP	$\partial r/\partial X$, $\partial r/\partial \phi$
SIGMA	argument of the interpolation subroutine STIBI
S1(I,J)	s on the boundary-layer grid
UE(I,J)	u_e/V_∞ on the boundary-layer grid
UET(I,J)	u_e/V_∞ on the inviscid grid
VE(I,J)	v_e/V_∞ on the boundary-layer grid
VET(I,J)	v_e/V_∞ on the inviscid grid
VX(I,J)	$u_{x'}/V_\infty$ on the inviscid grid
VY(I,J)	$u_{y'}/V_\infty$ on the inviscid grid
VZ(I,J)	$u_{z'}/V_\infty$ on the inviscid grid
WK	argument of the interpolation subroutine STIBI
X(I)	x_i for the boundary-layer grid
XO(I,J)	x' on the inviscid grid
XPD(I,J)	x' on the boundary-layer grid
XX(I)	X on the inviscid grid
Y(J)	y_j for the boundary-layer grid
YO(I,J)	y' on the inviscid grid
YPD(I,J)	y' on the boundary-layer grid
ZO(I,J)	z' on the inviscid grid
ZPD(I,J)	z' on the boundary-layer grid

2.5 Input

The inputs to BCC are given or read in the main program BCMAIN, as follows:

(1) In the main program BCMAIN, the x and y distributions for the body-oriented boundary -layer grid are specified.

First, IMAX and $x(i)$ for i=1,2,..,IMAX are set.

In this (body-oriented) coordinate system, x has the same value as X. For the blunted nose body, $x_{i=1}$ must be at least greater than x'_s which can be obtained from SCC. If the inviscid solution near the nose is not accurate, $x_{i=1}$ must not be too small. In the sample case, using the inviscid solution from the Hess code(which is not accurate near the nose), $x_{i=1} = 0.004$ was found to be adequate. The x distribution can be given arbitrarily. However, the stepsizes(Δx) near the nose must be small to obtain nonoscillating boundary-layer parameters.

Next, JMAX and $y(j)$ for j=1,2,..,JMAX are set.

In this coordinate system, y has the same value as ϕ. Therefore, $y(1)$=0 on the windward line of symmetry, and y(JMAX)=π on the leeward line of symmetry. The y-distribution can be given arbitrarily. However, a uniform distribution of grid points in the y direction is recommended to obtain a nearly uniform grid distribution downstream.

(2) The numerical inviscid solution based on the Cartesian coordinates is read also by the main program BCMAIN. This sets the values of

x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, Cp for i=1,2,..,NT, j=1,2,..,NP.

It is to be noted that j is increasing from the windward line of symmetry to the leeward line of symmetry.

2.6 Output

The output from BCC, which is to be used as input for 3DBLC, is written by the main program BCMAIN on file fort.22. The output lists the boundary-layer edge conditions including the following.

$x(i)$ for i=1,2,..,IMAX

$y(j)$ for j=1,2,..,JMAX

x', y', z', u_e/V_∞, v_e/V_∞, s, h_1, h_2, $\cos\theta$, Cp for i=1,2,..,IMAX, j=1,2,..,JMAX

The quantities, x'_s, z'_s, θ_r, A, B, and C^*, which are needed only for the blunted nose body, are not obtained using BCC.

2.7 Sample Case

For a sample case, the boundary-layer edge conditions on the body-oriented boundary-layer grid over a general aviation fuselage an angle of attack 3^o are calculated. The inviscid solution was obtained using 53x36(IxJ) inviscid grid from the Hess code [1] for the compressible flow($M_\infty = 0.3$). To reduce the input data, only the first 15x36(IxJ) inviscid grid solution is used for this sample case. Also, to reduce the size of the output data, only a 20x31(IxJ) body-oriented boundary-layer grid is generated. For this case, parameters are given as IM=15, JM=38 (=36+2), IMAXD=100 ($\geq$ 20), JMAXD=51 ($\geq$ 31).

For the sample case input, the FORTRAN statement (in the BCMAIN) for generating x and y distributions for the boundary-layer grid and the inviscid solution (for first 15x36 inviscid grid) from the Hess code are presented. The output written on file fort.22 is presented for a sample case output, which is input for 3DBLC.

2.7.1 Sample Case Input

```
      imax=20
      jmax=31

      x(1)=0.004
      do 5000 i=2,imax
      if(i.le.3)dx=0.0005
      if(i.gt.3.and.i.le.20)dx=0.002
      x(i)=x(i-1)+dx
      write(6,*)' i=',i,' x=',x(i)
5000  continue

      pi=acos(-1.)
      pio2=pi/2.

      do 5200 j=1,jmax
      y(j)=pi*(j-1.)/(jmax-1.)
5200  continue
```

ORIGINAL PAGE IS
OF POOR QUALITY

Numerical Inviscid Solution

```
    1     1    36
0.100000E-020.403333E-03-.923000E-02-.920655E-010.390607E-02-.562156E-010.101052E+01
0.100000E-020.120667E-02-.916667E-02-.920748E-010.121826E-01-.552482E-010.101049E+01
0.100000E-020.200333E-02-.904333E-02-.920683E-010.208146E-01-.534518E-010.101040E+01
0.100000E-020.279333E-02-.885667E-02-.920530E-010.286157E-01-.508697E-010.101028E+01
0.100000E-020.356667E-02-.860667E-02-.920420E-010.364627E-01-.472657E-010.101012E+01
0.100000E-020.432000E-02-.829667E-02-.920288E-010.443194E-01-.427402E-010.100989E+01
0.100000E-020.505000E-02-.792333E-02-.919519E-010.524400E-01-.370829E-010.100955E+01
0.100000E-020.574667E-02-.749000E-02-.918834E-010.594662E-01-.310275E-010.100918E+01
0.100000E-020.641000E-02-.699667E-02-.917573E-010.664914E-01-.237627E-010.100869E+01
0.100000E-020.703333E-02-.644333E-02-.916123E-010.728345E-01-.157926E-010.100812E+01
0.100000E-020.761000E-02-.583667E-02-.914207E-010.786035E-01-.726155E-020.100745E+01
0.100000E-020.813000E-02-.517667E-02-.911079E-010.841406E-010.270953E-020.100662E+01
0.100000E-020.859000E-02-.447000E-02-.908611E-010.883522E-010.123918E-010.100575E+01
0.100000E-020.898667E-02-.372000E-02-.904194E-010.924065E-010.236300E-010.100465E+01
0.100000E-020.931000E-02-.293333E-02-.899712E-010.953665E-010.347393E-010.100348E+01
0.100000E-020.955667E-02-.211667E-02-.893958E-010.975739E-010.468261E-010.100211E+01
0.100000E-020.972333E-02-.127667E-02-.887937E-010.987836E-010.589367E-010.100063E+01
0.100000E-020.980667E-02-.426667E-03-.880488E-010.992010E-010.718757E-010.998917E+00
0.100000E-020.981333E-020.430000E-03-.874377E-010.986932E-010.839554E-010.997168E+00
0.100000E-020.974000E-020.128333E-02-.864698E-010.972453E-010.971707E-010.995140E+00
0.100000E-020.956667E-020.212000E-02-.852386E-010.947582E-010.109988E+000.993087E+00
0.100000E-020.928333E-020.292333E-02-.837142E-010.911711E-010.122583E+000.990993E+00
0.100000E-020.889000E-020.367667E-02-.820928E-010.866046E-010.134322E+000.988973E+00
0.100000E-020.840333E-020.437000E-02-.805616E-010.813468E-010.144899E+000.987071E+00
0.100000E-020.784333E-020.499333E-02-.790106E-010.754187E-010.154609E+000.985264E+00
0.100000E-020.722333E-020.554000E-02-.774365E-010.688190E-010.163457E+000.983577E+00
0.100000E-020.656333E-020.601000E-02-.761332E-010.623940E-010.170617E+000.982169E+00
0.100000E-020.587667E-020.640667E-02-.747962E-010.551793E-010.177287E+000.980843E+00
0.100000E-020.517333E-020.673667E-02-.737804E-010.484672E-010.182527E+000.979758E+00
0.100000E-020.446667E-020.700667E-02-.728406E-010.414593E-010.187007E+000.978832E+00
0.100000E-020.376333E-020.722000E-02-.720702E-010.346765E-010.190526E+000.978101E+00
0.100000E-020.306333E-020.738667E-02-.715116E-010.283443E-010.193177E+000.977540E+00
0.100000E-020.237000E-020.751333E-02-.710670E-010.220083E-010.195240E+000.977103E+00
0.100000E-020.168667E-020.760333E-02-.707659E-010.159155E-010.196649E+000.976812E+00
0.100000E-020.101000E-020.765667E-02-.705070E-010.897701E-020.197735E+000.976583E+00
0.100000E-020.336667E-030.768000E-02-.704315E-010.334137E-020.198127E+000.976505E+00
    1     2    36
0.262936E-020.796024E-03-.182172E-01-.442675E-010.857746E-02-.191989E+000.982071E+00
0.262976E-020.238310E-02-.180996E-01-.444561E-010.253185E-01-.190245E+000.982157E+00
0.263029E-020.396044E-02-.178628E-01-.447416E-010.422311E-01-.186802E+000.982293E+00
0.262859E-020.552038E-02-.174957E-01-.451721E-010.588075E-01-.181597E+000.982506E+00
0.263053E-020.705411E-02-.170202E-01-.457258E-010.754291E-01-.174515E+000.982757E+00
0.262961E-020.854850E-02-.164147E-01-.463699E-010.908698E-01-.165967E+000.983053E+00
0.263043E-020.100031E-01-.156932E-01-.471133E-010.106806E+00-.155113E+000.983330E+00
0.263075E-020.113960E-01-.148505E-01-.478856E-010.121312E+00-.142977E+000.983575E+00
0.263059E-020.127245E-01-.138865E-01-.486663E-010.135334E+00-.128880E+000.983740E+00
0.263112E-020.139787E-01-.128049E-01-.494071E-010.148454E+00-.113056E+000.983774E+00
0.263122E-020.151374E-01-.116098E-01-.499856E-010.160477E+00-.957663E-010.983606E+00
0.263183E-020.161918E-01-.103084E-01-.504723E-010.171399E+00-.766432E-010.983213E+00
0.263162E-020.171260E-01-.891138E-02-.508752E-010.180371E+00-.569296E-010.982624E+00
0.263187E-020.179295E-01-.742311E-02-.508694E-010.188675E+00-.343559E-010.981577E+00
0.263232E-020.185888E-01-.585826E-02-.507072E-010.194721E+00-.123930E-010.980246E+00
0.263197E-020.190920E-01-.423128E-02-.501877E-010.199283E+000.118164E-010.978440E+00
0.263237E-020.194353E-01-.255653E-02-.494072E-010.201818E+000.359161E-010.976259E+00
```

```
0.263334E-020.196104E-01-.855847E-03-.481336E-010.202576E+000.617052E-010.973442E+00
0.263196E-020.196252E-010.858192E-03-.472247E-010.201449E+000.843724E-010.970553E+00
0.263359E-020.194979E-010.256762E-02-.454013E-010.198167E+000.110841E+000.966707E+00
0.263273E-020.191560E-010.424542E-02-.426374E-010.192438E+000.137681E+000.962337E+00
0.263354E-020.185994E-010.585882E-02-.390092E-010.184288E+000.163926E+000.957593E+00
0.263308E-020.178213E-010.737407E-02-.348968E-010.174017E+000.188435E+000.952743E+00
0.263328E-020.168548E-010.876763E-02-.303815E-010.162173E+000.210910E+000.947845E+00
0.263281E-020.157299E-010.100145E-01-.255367E-010.149166E+000.231139E+000.943028E+00
0.263267E-020.144865E-010.111066E-01-.207009E-010.135267E+000.248847E+000.938524E+00
0.263406E-020.131636E-010.120499E-01-.162116E-010.121365E+000.263738E+000.934462E+00
0.263306E-020.117798E-010.128420E-01-.120357E-010.107090E+000.276463E+000.930821E+00
0.263273E-020.103682E-010.134977E-01-.821140E-020.928806E-010.287086E+000.927626E+00
0.263289E-020.894964E-020.140327E-01-.509204E-020.796782E-010.295358E+000.925025E+00
0.263429E-020.753828E-020.144624E-01-.230959E-020.658518E-010.302343E+000.922794E+00
0.263160E-020.613079E-020.147846E-01-.365310E-030.535113E-010.307299E+000.921187E+00
0.263364E-020.474544E-020.150407E-010.135399E-020.405353E-010.311386E+000.919824E+00
0.263221E-020.337396E-020.152087E-010.268183E-020.280762E-010.314260E+000.918837E+00
0.263332E-020.201950E-020.153167E-010.354129E-020.156276E-010.316128E+000.918171E+00
0.263219E-020.672157E-030.153613E-010.378156E-020.533098E-020.316774E+000.917968E+00
```

***** For brevity, inviscid data for i=3,4,..,14 are deleted. *****

```
    1   15   36
0.321581E+000.872099E-02-.199535E+000.927619E+000.544486E-02-.257310E+000.734057E-01
0.321584E+000.262031E-01-.198821E+000.927110E+000.164166E-01-.257759E+000.738801E-01
0.321585E+000.438005E-01-.197358E+000.926009E+000.279428E-01-.258777E+000.748860E-01
0.321591E+000.615761E-01-.195087E+000.924247E+000.404108E-01-.260299E+000.765103E-01
0.321593E+000.795842E-01-.191927E+000.921834E+000.541335E-01-.262069E+000.787498E-01
0.321603E+000.978590E-01-.187778E+000.918702E+000.695334E-01-.263877E+000.816696E-01
0.321607E+000.116399E+00-.182503E+000.914710E+000.870987E-01-.265530E+000.853770E-01
0.321613E+000.135164E+00-.175943E+000.909732E+000.107335E+00-.266465E+000.900447E-01
0.321621E+000.154062E+00-.167922E+000.903628E+000.130682E+00-.266091E+000.957803E-01
0.321629E+000.172927E+00-.158255E+000.896291E+000.157627E+00-.263429E+000.102657E+00
0.321633E+000.191502E+00-.146749E+000.887744E+000.188471E+00-.257000E+000.110614E+00
0.321638E+000.209427E+00-.133236E+000.878105E+000.222843E+00-.245150E+000.119494E+00
0.321638E+000.226242E+00-.117611E+000.867727E+000.259765E+00-.226038E+000.128850E+00
0.321640E+000.241408E+00-.998548E-010.857238E+000.297378E+00-.197882E+000.137978E+00
0.321641E+000.254333E+00-.800799E-010.847732E+000.332383E+00-.159817E+000.145806E+00
0.321639E+000.264471E+00-.585512E-010.840390E+000.360853E+00-.112937E+000.151288E+00
0.321638E+000.271412E+00-.356817E-010.835964E+000.379623E+00-.606036E-010.153908E+00
0.321638E+000.275039E+00-.119861E-010.834246E+000.387878E+00-.760883E-020.154058E+00
0.321638E+000.276037E+000.120769E-010.833639E+000.388666E+000.358414E-010.153225E+00
0.321642E+000.275727E+000.363597E-010.831833E+000.387891E+000.685396E-010.153423E+00
0.321643E+000.274072E+000.608228E-010.828364E+000.383992E+000.112419E+000.154258E+00
0.321648E+000.269850E+000.851295E-010.824694E+000.373235E+000.162692E+000.154642E+00
0.321650E+000.262265E+000.108643E+000.822758E+000.352006E+000.215510E+000.153242E+00
0.321651E+000.251013E+000.130626E+000.824259E+000.319198E+000.264835E+000.149070E+00
0.321653E+000.236261E+000.150419E+000.829531E+000.277831E+000.304725E+000.142284E+00
0.321655E+000.218570E+000.167567E+000.837334E+000.233827E+000.332443E+000.134080E+00
0.321660E+000.198724E+000.181905E+000.846150E+000.192099E+000.348978E+000.125696E+00
0.321663E+000.177532E+000.193514E+000.854545E+000.155561E+000.357398E+000.118133E+00
0.321666E+000.155695E+000.202650E+000.861858E+000.124717E+000.360777E+000.111766E+00
0.321665E+000.133738E+000.209657E+000.867935E+000.991650E-010.361277E+000.106589E+00
0.321667E+000.112014E+000.214896E+000.872626E+000.779487E-010.360661E+000.102607E+00
0.321667E+000.907100E-010.218696E+000.876202E+000.599057E-010.359654E+000.995533E-01
0.321666E+000.698886E-010.221345E+000.878848E+000.443014E-010.358623E+000.972642E-01
0.321664E+000.495275E-010.223077E+000.880700E+000.303237E-010.357783E+000.956443E-01
0.321664E+000.295469E-010.224085E+000.881882E+000.175828E-010.357222E+000.945684E-01
0.321663E+000.981894E-020.224528E+000.882450E+000.563722E-020.356955E+000.940319E-01
```

2.7.2 Sample Case Output

```
      20          31
0.400000E-02  0.450000E-02  0.500000E-02  0.700000E-02  0.900000E-02
0.110000E-01  0.130000E-01  0.150000E-01  0.170000E-01  0.190000E-01
0.210000E-01  0.230000E-01  0.250000E-01  0.270000E-01  0.290000E-01
0.310000E-01  0.330000E-01  0.350000E-01  0.370000E-01  0.390000E-01
0.000000E+00  0.104720E+00  0.209440E+00  0.314159E+00  0.418879E+00
0.523599E+00  0.628319E+00  0.733038E+00  0.837758E+00  0.942478E+00
0.104720E+01  0.115192E+01  0.125664E+01  0.136136E+01  0.146608E+01
0.157080E+01  0.167552E+01  0.178024E+01  0.188496E+01  0.198968E+01
0.209439E+01  0.219911E+01  0.230383E+01  0.240856E+01  0.251327E+01
0.261799E+01  0.272271E+01  0.282743E+01  0.293215E+01  0.303687E+01
0.314159E+01
 1    1  0.400000E-02 -0.148836E-07 -0.232644E-01  0.236058E-01  0.250007E+00
         0.000000E+00  0.590144E+01  0.232644E-01  0.000000E+00  0.964495E+00
 1    2  0.400000E-02  0.243383E-02 -0.231564E-01  0.236250E-01  0.249811E+00
        -0.140179E-02  0.590625E+01  0.232634E-01  0.102811E-01  0.964641E+00
 1    3  0.400000E-02  0.485302E-02 -0.228317E-01  0.236821E-01  0.249152E+00
        -0.253219E-02  0.592052E+01  0.233286E-01  0.246516E-01  0.964944E+00
 1    4  0.400000E-02  0.724221E-02 -0.222893E-01  0.237752E-01  0.248002E+00
        -0.305573E-02  0.594381E+01  0.234343E-01  0.376655E-01  0.965482E+00
 1    5  0.400000E-02  0.958455E-02 -0.215273E-01  0.239016E-01  0.246458E+00
        -0.293748E-02  0.597540E+01  0.235753E-01  0.492020E-01  0.966214E+00
 1    6  0.400000E-02  0.118611E-01 -0.205440E-01  0.240570E-01  0.244544E+00
        -0.210847E-02  0.601426E+01  0.237455E-01  0.587947E-01  0.967030E+00
 1    7  0.400000E-02  0.140503E-01 -0.193386E-01  0.242362E-01  0.242450E+00
         0.182996E-03  0.605906E+01  0.239373E-01  0.657850E-01  0.967897E+00
 1    8  0.400000E-02  0.161280E-01 -0.179120E-01  0.244326E-01  0.240201E+00
         0.323252E-02  0.610815E+01  0.241413E-01  0.701368E-01  0.968749E+00
 1    9  0.400000E-02  0.180669E-01 -0.162675E-01  0.246382E-01  0.238143E+00
         0.815048E-02  0.615956E+01  0.243481E-01  0.714157E-01  0.969436E+00
 1   10  0.400000E-02  0.198374E-01 -0.144127E-01  0.248445E-01  0.236453E+00
         0.138459E-01  0.621112E+01  0.245479E-01  0.694549E-01  0.969848E+00
 1   11  0.400000E-02  0.214084E-01 -0.123601E-01  0.250418E-01  0.235392E+00
         0.208390E-01  0.626044E+01  0.247318E-01  0.635639E-01  0.969890E+00
 1   12  0.400000E-02  0.227483E-01 -0.101282E-01  0.252204E-01  0.235199E+00
         0.298814E-01  0.630509E+01  0.248918E-01  0.547637E-01  0.969390E+00
 1   13  0.400000E-02  0.238273E-01 -0.774197E-02  0.253708E-01  0.236326E+00
         0.383833E-01  0.634269E+01  0.250217E-01  0.426212E-01  0.968436E+00
 1   14  0.400000E-02  0.246190E-01 -0.523294E-02  0.254848E-01  0.238730E+00
         0.492255E-01  0.637121E+01  0.251171E-01  0.281493E-01  0.966547E+00
 1   15  0.400000E-02  0.251028E-01 -0.263842E-02  0.255560E-01  0.242999E+00
         0.598276E-01  0.638900E+01  0.251747E-01  0.106124E-01  0.963918E+00
 1   16  0.400000E-02  0.252644E-01  0.000000E+00  0.255791E-01  0.247625E+00
         0.698607E-01  0.639476E+01  0.252369E-01  0.107769E-01  0.960271E+00
 1   17  0.400000E-02  0.251809E-01  0.264661E-02  0.256337E-01  0.254166E+00
         0.793903E-01  0.640841E+01  0.252352E-01  0.688476E-02  0.955619E+00
 1   18  0.400000E-02  0.247446E-01  0.525963E-02  0.256117E-01  0.265902E+00
         0.923270E-01  0.640293E+01  0.251921E-01 -0.397059E-01  0.950264E+00
 1   19  0.400000E-02  0.238921E-01  0.776299E-02  0.254381E-01  0.281280E+00
         0.104250E+00  0.635952E+01  0.251134E-01 -0.985603E-01  0.943952E+00
```

1	20	0.400000E-02	0.226377E-01	0.100789E-01	0.251008E-01	0.298495E+00
		0.114168E+00	0.627520E+01	0.249739E-01	-0.154656E+00	0.937161E+00
1	21	0.400000E-02	0.210391E-01	0.121469E-01	0.246210E-01	0.315926E+00
		0.121089E+00	0.615525E+01	0.247314E-01	-0.200431E+00	0.930387E+00
1	22	0.400000E-02	0.191770E-01	0.139329E-01	0.240392E-01	0.331608E+00
		0.123761E+00	0.600979E+01	0.243500E-01	-0.230828E+00	0.923777E+00
1	23	0.400000E-02	0.171360E-01	0.154293E-01	0.234031E-01	0.344599E+00
		0.122036E+00	0.585077E+01	0.238257E-01	-0.246230E+00	0.917558E+00
1	24	0.400000E-02	0.149911E-01	0.166493E-01	0.227581E-01	0.354280E+00
		0.115570E+00	0.568952E+01	0.231892E-01	-0.246601E+00	0.912179E+00
1	25	0.400000E-02	0.128007E-01	0.176186E-01	0.221421E-01	0.360999E+00
		0.105573E+00	0.553553E+01	0.224937E-01	-0.235283E+00	0.907764E+00
1	26	0.400000E-02	0.106047E-01	0.183678E-01	0.215832E-01	0.365092E+00
		0.918703E-01	0.539579E+01	0.217974E-01	-0.213841E+00	0.904107E+00
1	27	0.400000E-02	0.842718E-02	0.189277E-01	0.211016E-01	0.367212E+00
		0.757737E-01	0.527539E+01	0.211522E-01	-0.184084E+00	0.901169E+00
1	28	0.400000E-02	0.627944E-02	0.193261E-01	0.207106E-01	0.368181E+00
		0.585943E-01	0.517765E+01	0.206002E-01	-0.147437E+00	0.898929E+00
1	29	0.400000E-02	0.416331E-02	0.195868E-01	0.204200E-01	0.368245E+00
		0.398796E-01	0.510500E+01	0.201732E-01	-0.104913E+00	0.897520E+00
1	30	0.400000E-02	0.207367E-02	0.197295E-01	0.202374E-01	0.367920E+00
		0.206593E-01	0.505935E+01	0.198979E-01	-0.576419E-01	0.896751E+00
1	31	0.400000E-02	0.620667E-08	0.197717E-01	0.201723E-01	0.367597E+00
		0.000000E+00	0.504307E+01	0.198060E-01	0.000000E+00	0.896493E+00
2	1	0.450000E-02	-0.158292E-07	-0.247425E-01	0.251661E-01	0.265181E+00
		0.000000E+00	0.312070E+01	0.247428E-01	0.000000E+00	0.954012E+00
2	2	0.450000E-02	0.258853E-02	-0.246283E-01	0.251872E-01	0.264981E+00
		-0.186515E-02	0.312440E+01	0.247431E-01	0.107155E-01	0.954188E+00
2	3	0.450000E-02	0.516194E-02	-0.242851E-01	0.252498E-01	0.264246E+00
		-0.346922E-02	0.313547E+01	0.248152E-01	0.254391E-01	0.954573E+00
2	4	0.450000E-02	0.770432E-02	-0.237115E-01	0.253520E-01	0.262961E+00
		-0.439319E-02	0.315358E+01	0.249321E-01	0.387918E-01	0.955257E+00
2	5	0.450000E-02	0.101981E-01	-0.229053E-01	0.254908E-01	0.261226E+00
		-0.473907E-02	0.317830E+01	0.250880E-01	0.506322E-01	0.956192E+00
2	6	0.450000E-02	0.126233E-01	-0.218643E-01	0.256615E-01	0.259074E+00
		-0.420289E-02	0.320892E+01	0.252764E-01	0.605659E-01	0.957254E+00
2	7	0.450000E-02	0.149575E-01	-0.205872E-01	0.258585E-01	0.256696E+00
		-0.216411E-02	0.324454E+01	0.254885E-01	0.678072E-01	0.958404E+00
2	8	0.450000E-02	0.171745E-01	-0.190742E-01	0.260745E-01	0.254083E+00
		0.652706E-03	0.328387E+01	0.257142E-01	0.722809E-01	0.959560E+00
2	9	0.450000E-02	0.192453E-01	-0.173286E-01	0.263010E-01	0.251635E+00
		0.557891E-02	0.332547E+01	0.259429E-01	0.736809E-01	0.960551E+00
2	10	0.450000E-02	0.211382E-01	-0.153578E-01	0.265283E-01	0.249488E+00
		0.111430E-01	0.336765E+01	0.261640E-01	0.717078E-01	0.961260E+00
2	11	0.450000E-02	0.228193E-01	-0.131748E-01	0.267460E-01	0.247978E+00
		0.183400E-01	0.340843E+01	0.263670E-01	0.657245E-01	0.961552E+00
2	12	0.450000E-02	0.242544E-01	-0.107988E-01	0.269432E-01	0.247376E+00
		0.275728E-01	0.344563E+01	0.265435E-01	0.566467E-01	0.961261E+00
2	13	0.450000E-02	0.254109E-01	-0.825654E-02	0.271094E-01	0.248064E+00
		0.362584E-01	0.347720E+01	0.266867E-01	0.441718E-01	0.960441E+00
2	14	0.450000E-02	0.262600E-01	-0.558176E-02	0.272355E-01	0.250112E+00
		0.474849E-01	0.350126E+01	0.267918E-01	0.292633E-01	0.958595E+00
2	15	0.450000E-02	0.267791E-01	-0.281461E-02	0.273142E-01	0.254107E+00
		0.584786E-01	0.351631E+01	0.268551E-01	0.111631E-01	0.955900E+00
2	16	0.450000E-02	0.269524E-01	0.000000E+00	0.273396E-01	0.258499E+00
		0.688537E-01	0.352116E+01	0.269244E-01	0.112628E-01	0.952126E+00
2	17	0.450000E-02	0.268663E-01	0.282374E-02	0.274005E-01	0.264792E+00
		0.786456E-01	0.353364E+01	0.269257E-01	0.828914E-02	0.947242E+00
2	18	0.450000E-02	0.264059E-01	0.561275E-02	0.273822E-01	0.276313E+00
		0.922146E-01	0.354095E+01	0.268816E-01	-0.377897E-01	0.941458E+00

```
 2  19  0.450000E-02  0.255010E-01  0.828576E-02  0.272022E-01  0.291668E+00
        0.104683E+00  0.352813E+01  0.267998E-01 -0.967838E-01  0.934577E+00
 2  20  0.450000E-02  0.241656E-01  0.107592E-01  0.268464E-01  0.309073E+00
        0.115255E+00  0.349121E+01  0.266543E-01 -0.153230E+00  0.927090E+00
 2  21  0.450000E-02  0.224608E-01  0.129677E-01  0.263370E-01  0.326773E+00
        0.122805E+00  0.343209E+01  0.263995E-01 -0.199220E+00  0.919559E+00
 2  22  0.450000E-02  0.204730E-01  0.148745E-01  0.257173E-01  0.342777E+00
        0.125813E+00  0.335630E+01  0.259960E-01 -0.229987E+00  0.912225E+00
 2  23  0.450000E-02  0.182931E-01  0.164712E-01  0.250385E-01  0.356062E+00
        0.124464E+00  0.327089E+01  0.254382E-01 -0.245476E+00  0.905306E+00
 2  24  0.450000E-02  0.160021E-01  0.177721E-01  0.243496E-01  0.366009E+00
        0.117946E+00  0.318295E+01  0.247584E-01 -0.246159E+00  0.899347E+00
 2  25  0.450000E-02  0.136626E-01  0.188049E-01  0.236913E-01  0.372876E+00
        0.107911E+00  0.309838E+01  0.240141E-01 -0.234915E+00  0.894447E+00
 2  26  0.450000E-02  0.113174E-01  0.196023E-01  0.230939E-01  0.377037E+00
        0.939484E-01  0.302137E+01  0.232679E-01 -0.213509E+00  0.890411E+00
 2  27  0.450000E-02  0.899269E-02  0.201979E-01  0.225791E-01  0.379217E+00
        0.775102E-01  0.295504E+01  0.225757E-01 -0.183789E+00  0.887182E+00
 2  28  0.450000E-02  0.670025E-02  0.206212E-01  0.221613E-01  0.380223E+00
        0.599501E-01  0.290132E+01  0.219835E-01 -0.147171E+00  0.884720E+00
 2  29  0.450000E-02  0.444202E-02  0.208980E-01  0.218507E-01  0.380218E+00
        0.405319E-01  0.286143E+01  0.215252E-01 -0.104712E+00  0.883190E+00
 2  30  0.450000E-02  0.221240E-02  0.210494E-01  0.216557E-01  0.379829E+00
        0.208467E-01  0.283653E+01  0.212297E-01 -0.574984E-01  0.882371E+00
 2  31  0.450000E-02  0.662182E-08  0.210942E-01  0.215861E-01  0.379516E+00
        0.000000E+00  0.282770E+01  0.211310E-01  0.000000E+00  0.882095E+00

***** For brevity, output for i=3,4,..,19 is deleted. *****

20   1  0.390000E-01 -0.475658E-07 -0.743496E-01  0.861731E-01  0.685748E+00
        0.000000E+00  0.138152E+01  0.743850E-01  0.000000E+00  0.543562E+00
20   2  0.390000E-01  0.778423E-02 -0.740622E-01  0.862742E-01  0.685165E+00
       -0.143568E-01  0.138320E+01  0.744878E-01  0.178137E-01  0.544749E+00
20   3  0.390000E-01  0.155578E-01 -0.731938E-01  0.865754E-01  0.683438E+00
       -0.279416E-01  0.138820E+01  0.749428E-01  0.379004E-01  0.547554E+00
20   4  0.390000E-01  0.233056E-01 -0.717272E-01  0.870711E-01  0.680559E+00
       -0.407296E-01  0.139651E+01  0.756813E-01  0.567094E-01  0.552411E+00
20   5  0.390000E-01  0.310040E-01 -0.696362E-01  0.877522E-01  0.676345E+00
       -0.520173E-01  0.140806E+01  0.766733E-01  0.740519E-01  0.559298E+00
20   6  0.390000E-01  0.386171E-01 -0.668870E-01  0.886049E-01  0.670756E+00
       -0.611592E-01  0.142273E+01  0.778776E-01  0.893371E-01  0.567953E+00
20   7  0.390000E-01  0.460921E-01 -0.634403E-01  0.896087E-01  0.663836E+00
       -0.673612E-01  0.144038E+01  0.792400E-01  0.101946E+00  0.578215E+00
20   8  0.390000E-01  0.533553E-01 -0.592571E-01  0.907358E-01  0.655563E+00
       -0.706555E-01  0.146061E+01  0.806923E-01  0.111085E+00  0.589841E+00
20   9  0.390000E-01  0.603084E-01 -0.543020E-01  0.919476E-01  0.645979E+00
       -0.699325E-01  0.148289E+01  0.821580E-01  0.115855E+00  0.602586E+00
20  10  0.390000E-01  0.668283E-01 -0.485536E-01  0.931964E-01  0.635488E+00
       -0.647821E-01  0.150637E+01  0.835560E-01  0.115706E+00  0.615796E+00
20  11  0.390000E-01  0.727661E-01 -0.420116E-01  0.944228E-01  0.624662E+00
       -0.553089E-01  0.152993E+01  0.848106E-01  0.109097E+00  0.628571E+00
20  12  0.390000E-01  0.779553E-01 -0.347080E-01  0.955596E-01  0.614301E+00
       -0.411998E-01  0.155222E+01  0.858620E-01  0.966681E-01  0.640082E+00
20  13  0.390000E-01  0.822223E-01 -0.267157E-01  0.965362E-01  0.605464E+00
       -0.234702E-01  0.157164E+01  0.866756E-01  0.775878E-01  0.649431E+00
20  14  0.390000E-01  0.854039E-01 -0.181532E-01  0.972862E-01  0.599182E+00
       -0.306474E-02  0.158670E+01  0.872446E-01  0.532809E-01  0.655450E+00
20  15  0.390000E-01  0.873683E-01 -0.918281E-02  0.977571E-01  0.596398E+00
        0.190340E-01  0.159619E+01  0.876000E-01  0.221946E-01  0.657675E+00
20  16  0.390000E-01  0.880632E-01  0.000000E+00  0.979440E-01  0.595588E+00
```

		0.389856E-01	0.160004E+01	0.880107E-01	0.220268E-01	0.656822E+00
20	17	0.390000E-01	0.879284E-01	0.924159E-02	0.982531E-01	0.595936E+00
		0.510731E-01	0.160534E+01	0.883377E-01	0.387339E-01	0.653686E+00
20	18	0.390000E-01	0.868571E-01	0.184620E-01	0.985790E-01	0.599896E+00
		0.739315E-01	0.161506E+01	0.883924E-01	0.932750E-02	0.647450E+00
20	19	0.390000E-01	0.843374E-01	0.274028E-01	0.984452E-01	0.609807E+00
		0.101843E+00	0.162015E+01	0.882820E-01	-0.438659E-01	0.636418E+00
20	20	0.390000E-01	0.802721E-01	0.357394E-01	0.976922E-01	0.626105E+00
		0.129797E+00	0.161657E+01	0.880476E-01	-0.102442E+00	0.620516E+00
20	21	0.390000E-01	0.748054E-01	0.431888E-01	0.963326E-01	0.646648E+00
		0.152814E+00	0.160334E+01	0.875636E-01	-0.154077E+00	0.601819E+00
20	22	0.390000E-01	0.682358E-01	0.495762E-01	0.944974E-01	0.667418E+00
		0.167856E+00	0.158183E+01	0.865853E-01	-0.190672E+00	0.581962E+00
20	23	0.390000E-01	0.609184E-01	0.548512E-01	0.923767E-01	0.685324E+00
		0.173449E+00	0.155496E+01	0.849584E-01	-0.209656E+00	0.562584E+00
20	24	0.390000E-01	0.531855E-01	0.590685E-01	0.901667E-01	0.699112E+00
		0.169254E+00	0.152592E+01	0.827323E-01	-0.213841E+00	0.545546E+00
20	25	0.390000E-01	0.452934E-01	0.623409E-01	0.880286E-01	0.708247E+00
		0.156728E+00	0.149736E+01	0.801225E-01	-0.204416E+00	0.531567E+00
20	26	0.390000E-01	0.374166E-01	0.648073E-01	0.860827E-01	0.713664E+00
		0.137704E+00	0.147137E+01	0.773966E-01	-0.185507E+00	0.520345E+00
20	27	0.390000E-01	0.296542E-01	0.666043E-01	0.844094E-01	0.716398E+00
		0.114226E+00	0.144903E+01	0.748133E-01	-0.158965E+00	0.511634E+00
20	28	0.390000E-01	0.220461E-01	0.678509E-01	0.830575E-01	0.717387E+00
		0.879869E-01	0.143113E+01	0.725774E-01	-0.126559E+00	0.505200E+00
20	29	0.390000E-01	0.145915E-01	0.686478E-01	0.820585E-01	0.717351E+00
		0.593864E-01	0.141796E+01	0.708396E-01	-0.893690E-01	0.501022E+00
20	30	0.390000E-01	0.726007E-02	0.690744E-01	0.814354E-01	0.716944E+00
		0.301647E-01	0.140982E+01	0.697191E-01	-0.483743E-01	0.498686E+00
20	31	0.390000E-01	0.217225E-07	0.691985E-01	0.812158E-01	0.716659E+00
		0.000000E+00	0.140699E+01	0.693382E-01	0.000000E+00	0.497821E+00

2.8 FORTRAN Listing of BCC

Subroutines STIBI and VAL are not presented here. Subroutine DUDY can be found in Section 3.8.

```
##########################################################################
      program bcmain
##########################################################################
c**    obtain the boundary-layer edge conditions
c**    on the body-oriented boundary-layer grids.

      parameter(im=15,jm=38,imaxd=100,jmaxd=51)
      dimension xo(im,jm),yo(im,jm),zo(im,jm)
     &,vx(im,jm),vy(im,jm),vz(im,jm),pcoef(im,jm)
      dimension phit(jm),cavt(im,jm),xx(im)
      dimension cavl(im),cavw(im),costht(im,jm),uet(im,jm)
     &,vet(im,jm),cpl(im),cpw(im)
      integer iendsw(8),ierr,iopt(3),iw,mx,my,mz
      real endy1(im),endyn(im),sigma
      real wk(5*im*jm)
      dimension x(imaxd),y(jmaxd),ue(imaxd,jmaxd),ve(imaxd,jmaxd)
     &,costh(imaxd,jmaxd),cpd(imaxd,jmaxd)
      dimension xpd(imaxd,jmaxd),ypd(imaxd,jmaxd),zpd(imaxd,jmaxd)
      dimension h1(imaxd,jmaxd),s1(imaxd,jmaxd),h2(imaxd,jmaxd)
     &,dy(jmaxd)

      imax=20
      jmax=31

      x(1)=0.004
      do 5000 i=2,imax
      if(i.le.3)dx=0.0005
      if(i.gt.3.and.i.le.20)dx=0.002
      x(i)=x(i-1)+dx
      write(6,*)' i=',i,' x=',x(i)
5000  continue

      pi=acos(-1.)
      pio2=pi/2.

      do 5200 j=1,jmax
      y(j)=pi*(j-1.)/(jmax-1.)
5200  continue

      if(imax.gt.imaxd)write(6,*)'change parameter imaxd greater or
     &equal to',imax
      if(jmax.gt.jmaxd)write(6,*)'change parameter jmaxd greater or
     &equal to',jmax
      if(imax.gt.imaxd.or.jmax.gt.jmaxd)stop

c
c     read inviscid data
```

```
c
      rewind 2
 100  read(2,410,end=1000)is,lk,ksorce

      do 800 k=1,ksorce
      read(2,411)xo(lk,k),yo(lk,k),zo(lk,k),vx(lk,k),
     &vy(lk,k),vz(lk,k),pcoef(lk,k)
      cavt(lk,k)=sqrt(vx(lk,k)**2+vy(lk,k)**2+vz(lk,k)**2)
      if(yo(lk,k).lt.0)yo(lk,k)=1.e-7
      if(lk.eq.3)phit(k)=atan(zo(lk,k)/yo(lk,k))+pio2
 800  continue
      xx(lk)=xo(lk,1)
 410  format(3i5)
 411  format(7e12.6)
      go to 100

c     to give values on  the lines of symmetry

 1000 nt=lk
      np=ksorce

      if(nt.ne.im)then
      write(6,*)'im is not same as nt, change parameter im to',nt
      stop
      endif

      if(np+2.ne.jm)then
      write(6,*)'jm is not same as np+2 , change parameter jm to', np+2
      stop
      endif

c     to obtain costht

       do 3000 i=1,nt
       do 3100 k=1,np
        ph1=phit(k)-pio2
        call val(xo(i,k),ph1,r1,rx,0)
        x1=xo(i,k)
        y1=-r1*(-cos(ph1))
        z1=r1*sin(ph1)

        x2=xo(i,k)+0.01
        call val(x2,ph1,r2,rx,0)
        y2=-r2*(-cos(ph1))
        z2=r2*sin(ph1)

        ph3=ph1+0.01
        call val(xo(i,k),ph3,r3,rx,0)
        x3=xo(i,k)
        y3=-r3*(-cos(ph3))
        z3=r3*sin(ph3)

        costht(i,k)=((y2-y1)*(y3-y1)+(z2-z1)*(z3-z1)+(x2-x1)*(x3-x1))
     &  /(sqrt((y3-y1)**2+(z3-z1)**2+(x3-x1)**2)
     &  *sqrt((y2-y1)**2+(z2-z1)**2+(x2-x1)**2))

       delta=acos(((x3-x1)*vx(i,k)+(y3-y1)*vy(i,k)+(z3-z1)*vz(i,k))
     &/(sqrt((x3-x1)**2+(y3-y1)**2+(z3-z1)**2)*cavt(i,k)))
       gamma=acos(costht(i,k))-delta
```

```
      vet(i,k)=cavt(i,k)*sin(gamma)/sqrt(1.-costht(i,k)**2)
      uet(i,k)=-vet(i,k)*costht(i,k)+cavt(i,k)*cos(gamma)
 3100 continue

 3000 continue

      do 2600 lk=1,nt
      call dudy(phit(np-1),phit(np),pi,uet(lk,np-1)
     &,uet(lk,np),cavl(lk))
      call dudy(phit(np-1),phit(np),pi,pcoef(lk,np-1)
     &,pcoef(lk,np),cpl(lk))

      call dudy(phit(2),phit(1),0.,uet(lk,2),uet(lk,1),cavw(lk))
      call dudy(phit(2),phit(1),0.,pcoef(lk,2),pcoef(lk,1),cpw(lk))
 2600 continue

      do 2100 lk=1,nt
      do 1200 k=ksorce,1,-1
      uet(lk,k+1)=uet(lk,k)
      vet(lk,k+1)=vet(lk,k)
      costht(lk,k+1)=costht(lk,k)
      pcoef(lk,k+1)=pcoef(lk,k)
 1200 continue
 2100 continue


      do 1300 k=ksorce,1,-1
      phit(k+1)=phit(k)
 1300 continue

      phit(1)=0.
      phit(np+2)=pi

      npi=np+2
      nti=nt

      do 3300 lk=1,nti
      uet(lk,1)=cavw(lk)
      uet(lk,npi)=cavl(lk)
      vet(lk,1)=0.
      vet(lk,npi)=0.
      costht(lk,1)=0.
      costht(lk,npi)=0.
      pcoef(lk,1)=cpw(lk)
      pcoef(lk,npi)=cpl(lk)
 3300 continue

c     biviarate spline under tension

      iendsw(1)=2
      iendsw(2)=2
      iendsw(3)=0
      iendsw(4)=0

      do 50 ii=1,nti
      endy1(ii)=0.
   50 endyn(ii)=0.
      sigma=2.0

      iopt(1)=3
```

```
      iopt(2)=3
      mx=1
      my=1
      mz=1
      iw=0

       do 7000 i=1,imax
       do 7000 j=1,jmax
      call stibi(iopt,im,jm,xx,phit,im,uet,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,x(i),y(j),iw,mz,ues,
     &linout,wk,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi) ierr=',ierr
      ue(i,j)=ues
7000  continue

       do 7500 i=1,imax
       do 7500 j=1,jmax
      call stibi(iopt,im,jm,xx,phit,im,pcoef,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,x(i),y(j),iw,mz,cps,
     &linout,wk,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi) ierr=',ierr
      cpd(i,j)=cps
7500  continue

      iw=0
      iendsw(3)=2
      iendsw(4)=2

       do 8000 i=1,imax
       do 8000 j=1,jmax
      call stibi(iopt,im,jm,xx,phit,im,vet,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,x(i),y(j),iw,mz,ves,
     &linout,wk,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi) ierr=',ierr
      ve(i,j)=ves
8000  continue

      iw=0

       do 9000 i=1,imax
       do 9000 j=1,jmax
      call stibi(iopt,im,jm,xx,phit,im,costht,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,x(i),y(j),iw,mz,cosths,
     &linout,wk,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi) ierr=',ierr
      costh(i,j)=cosths
9000  continue

      do 6000 i=1,imax
      do 6000 j=1,jmax
        ph5=y(j)-pio2
        call val(x(i),ph5,r5,rx,0)
        xpd(i,j)=x(i)
        ypd(i,j)=-r5*(-cos(ph5))
        zpd(i,j)=r5*sin(ph5)
6000  continue

c      obtain h1 and s1

        do 600 j=1,jmax
```

```
      s1(1,j)=sqrt(xpd(1,j)**2+ypd(1,j)**2+zpd(1,j)**2)
      h1(1,j)=s1(1,j)/x(1)
      do 610 i=2,imax
      ds1=sqrt((xpd(i,j)-xpd(i-1,j))**2+(ypd(i,j)-ypd(i-1,j))**2
     & +(zpd(i,j)-zpd(i-1,j))**2)
      s1(i,j)=s1(i-1,j)+ds1
      h1(i,j)=ds1/(x(i)-x(i-1))
 610  continue
 600  continue

c     obtain h2 by f-d

      do 790 n=1,jmax-1
      dy(n)=y(n+1)-y(n)
 790  continue

      do 2200 l=1,imax
      do 840 n=1,jmax
      dxpdy=(dy(n-1)**2*xpd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *xpd(l,n)-dy(n)**2*xpd(l,n-1))/(dy(n)*dy(n-1)*(dy(n)
     & +dy(n-1)))
      dypdy=(dy(n-1)**2*ypd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *ypd(l,n)-dy(n)**2*ypd(l,n-1))/(dy(n)*dy(n-1)*(dy(n)
     & +dy(n-1)))
      dzpdy=(dy(n-1)**2*zpd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *zpd(l,n)-dy(n)**2*zpd(l,n-1))/(dy(n)*dy(n-1)*(dy(n)
     & +dy(n-1)))


      if(n.eq.1)then
      dxpdy=(xpd(l,2)-xpd(l,1))/dy(1)
      dypdy=(ypd(l,2)-ypd(l,1))/dy(1)
      dzpdy=(zpd(l,2)-zpd(l,1))/dy(1)
      endif

      if(n.eq.jmax)then
      dxpdy=(xpd(l,jmax)-xpd(l,jmax-1))/dy(jmax-1)
      dypdy=(ypd(l,jmax)-ypd(l,jmax-1))/dy(jmax-1)
      dzpdy=(zpd(l,jmax)-zpd(l,jmax-1))/dy(jmax-1)
      endif

      h2(l,n)=sqrt(dxpdy**2+dypdy**2+dzpdy**2)
 840  continue
 2200 continue

      rewind 22
      write(22,463)imax,jmax
      write(22,461)(x(i),i=1,imax)
      write(22,461)(y(j),j=1,jmax)
      do 460 i=1,imax
      do 460 j=1,jmax
      write(22,462)i,j,xpd(i,j),ypd(i,j),zpd(i,j),s1(i,j),ue(i,j)
     &,ve(i,j),h1(i,j),h2(i,j),costh(i,j),cpd(i,j)
 460  continue
 461  format(5(1x,e13.6))
 462  format(2i4,5(1x,e13.6)/8x,5(1x,e13.6))
 463  format(2i10)

      stop
      end
```

PART 3.

STREAMLINE COORDINATE PROGRAM (SCC)

3.1 Program Description

Program SCC is used for the generation of the boundary-layer edge conditions based on the streamline coordinates for the general fuselage. This code reads the numerical inviscid solution based on the Cartesian coordinates (x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, Cp) on the inviscid grid and calculates the boundary-layer edge conditions (x', y', z', u_e/V_∞, v_e/V_∞, s, h_2, Cp) on the streamline boundary-layer grid.

A geometry program which defines the fuselage is required to run the SCC code. This code is written to be generally applied, so any geometry routine, which returns the body radius r for given axial coordinate X and angle ϕ, can be used. Because the raw data defining the sample case general aviation fuselage were nonsmooth, a semi-analytic geometry program specially made for this fuselage by Raymond L. Barger at the NASA Langley Research Center is used. It should be noted that the angle ϕ must be defined as $-\pi/2$ and $\pi/2$ on the windward and leeward lines of symmetry, respectively, in this geometry routine.

To calculate the streamline coordinates, a method developed by Hamilton et al. [3] is used (for detail, see Appendix D.2 of Volume I). Program SCC is modified from the code developed by Hamilton et al.. A fourth order Runge-Kutta method is used for the integration, and a bidirectional cubic spline-under-tension subroutine is used for the interpolation.

3.2 Structure of Main Program SCMAIN

The flow chart for the main program SCMAIN is in Fig. 5. The program first calls subroutine INPUT. The x and y distributions for the boundary-layer grid are given in subroutine INPUT. Subroutine INVDAT is called to read the inviscid solution from the inviscid code. The calculation of the inviscid velocity components based on the spherical coordinates from those based on the Cartesian coordinates is done in this subroutine. Subroutine INVDAT also includes the extrapolation of the inviscid properties (like u_e, Cp, u_R, u_Θ) on the lines of symmetry.

If the nose of the body is blunted, subroutine STAGLO is called to locate the stagnation point. The values of x'_s and z'_s are obtained in this subroutine; subroutine ECON is then called to locate the initial locations of the streamlines; the angle θ_r and the velocity gradients at the stagnation point (A, B, C^*) are calculated in the main program. However, if the nose of the body is sharp, no subroutine is required to locate the initial streamline locations, and x'_s, z'_s, θ_r, A, B, and C^* are not calculated.

To generate the orthogonal streamline coordinates, the initial locations of the streamlines are readjusted by integration along the streamlines to an $x = const$ plane. Then, the integration along the streamlines for each streamline is performed using the fourth order Runge-Kutta method. For the integration, subprogram KRUNGE, subroutine FCN, and geometry subroutine VAL (CSGEOM when using QUICK geometry program [2]) are used. Each step (for each i-th and j-th step), x', y', z', u_e/V_∞, s, h_1 ($= V_\infty/u_e$), and Cp are saved. After the integrations are finished, the metric coefficient h_2 is calculated. For a sharp nose body, the velocity components (u_e/V_∞, v_e/V_∞) based on the body-oriented coordinate system are calculated for i=1. Finally, the outputs, which are to be used as inputs to the boundary-layer code, are written in the file fort.25.

Parameters IM and JM provide the flexibility of changing the dimensions of the inviscid grid to be read. These parameters are given in the main program SCMAIN and subroutines INVDAT, FCN, STAGLO. The parameter IM should be the number of inviscid grid points in the streamwise direction plus 1, i.e., IM=NT+1. The parameter JM should be the number of inviscid grid points in the crosswise direction plus 2, i.e., JM=NP+2. The parameters IM and JM must be same for all subroutines which use these parameters.

Parameters IMAXD and JMAXD provide the flexibility of changing the dimensions of the boundary-layer grid in the streamwise and crosswise directions. These parameters are given in the main program SCMAIN and subroutine INPUT. IMAXD may be different from IMAX, but should be greater or equal to IMAX. Also, JMAXD may be different from JMAX, but should be greater or equal to JMAX. The dimensions of the common blocks and the local variable arrays are controlled by changing these parameters, IM, JM, IMAXD, and JMAXD.

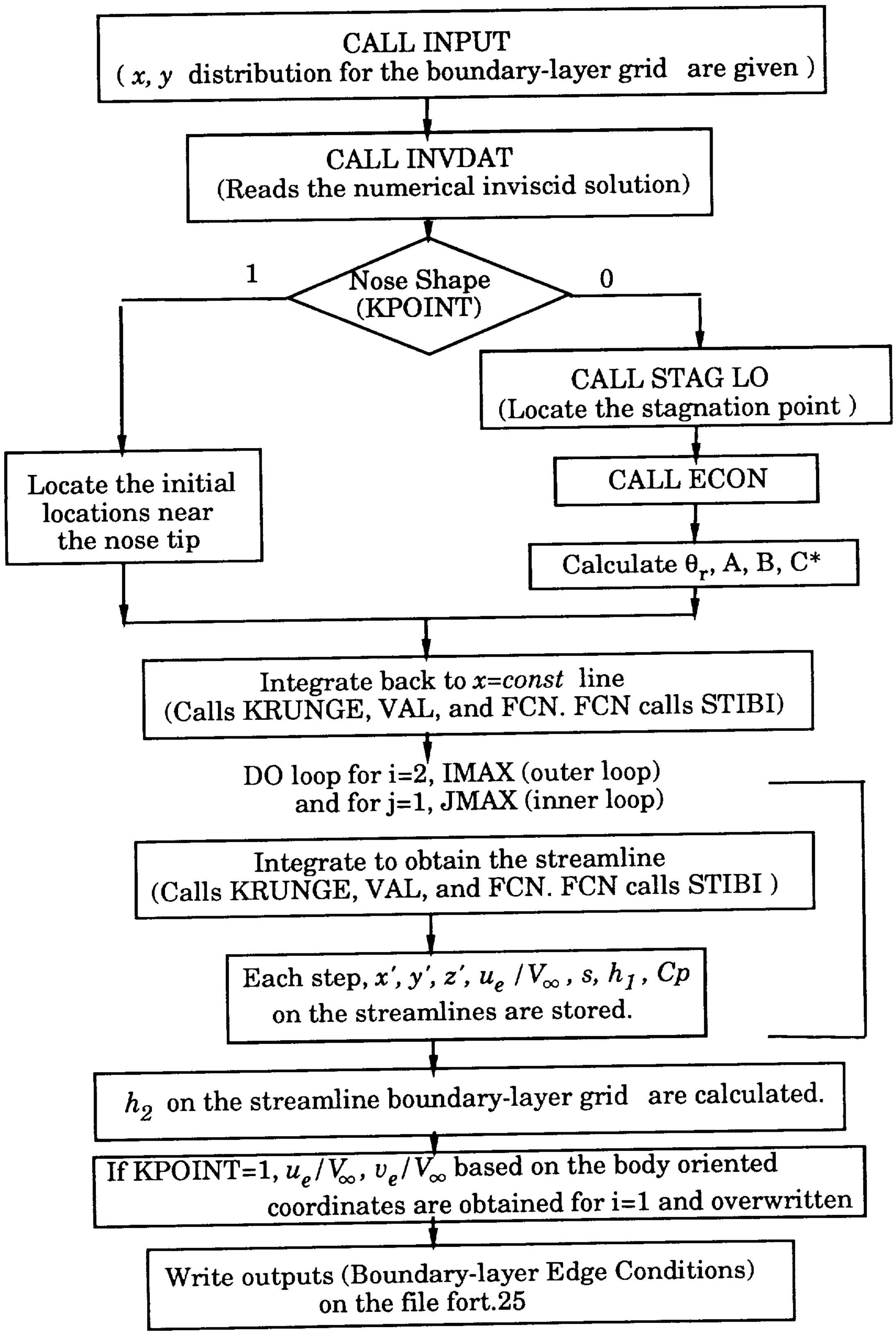


Fig. 5. Flow Chart for the Main Program SCMAIN.

3.3 Subroutine Description

Subroutine DUDY(X1, X2, X3, Y1, Y2, Y3)

- Called by subroutine INVDAT.
- Used to obtain the inviscid properties (u_e/V_∞, Cp, R, u_R/V_∞, u_Θ/V_∞) on the lines of symmetry.
- Calculates Y3 at X3 by the second order Lagrangian extrapolation, utilizing the symmetry condition at X3 and with given (X1,Y1) and (X2, Y2).

Subroutine ECON(TH, XOSP, PHI, X)

- Called by the main program SCMAIN and calls subroutine CSGEOM.
- Used to obtain X from given X_{osp}, Θ and ϕ on the ϵ-cone using Newton's method.

Subroutine FCN

- Called by the main program SCMAIN and calls subroutine STIBI.
- Parameters IM and JM are given.
- Calculates u_e/V_∞, u_R/V_∞, u_Θ/V_∞, u_ϕ/V_∞ and Cp on the surface for given X and ϕ and follows with the calculation of the derivatives of X, ϕ and s with respect to x, i.e., F(1), F(2), and F(4).

Subroutine IUNI

- Called by subroutine STAGLO.
- Avaiable as a mathematical library routine at NASA Langley Research Center.
- Interpolates one function of one independent variable at a single value of each of the independent variables at each call. Conventional first- or second-order Lagrangian interpolation is used. In SCC, the independent variable is $\pi - \Theta$.

Subroutine INPUT

- Called by the main program SCMAIN.
- Parameters IMAXD and JMAXD are given.
- IMAX and JMAX are given.
- The x-distributions are given for i=1,2,..,IMAX.
- The y-distribution is given for j=1,2,..,JMAX.

Subroutine INVDAT

- Called by the main program SCMAIN and calls subroutine DUDY.
- Parameters IM and JM are given.
- Reads the numerical inviscid solution based on the Cartesian coordinates (x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, and Cp) on the inviscid grid.
- Then calculates the inviscid velocity components in the spherical coordinates (u_R/V_∞, u_Θ/V_∞, u_ϕ/V_∞) and spherical coordinates (R, Θ, ϕ) on the inviscid grid.

- Extrapolates the inviscid velocity in the spherical coordinates (u_R/V_∞, u_Θ/V_∞, and u_e/V_∞), Cp, and R on the lines of symmetry, utilizing the symmetry condition along these lines.

- Adds one array in the x-direction and two arrays in the y-direction for the nose point and for the two (windward and leeward) lines of symmetry.

Function KRUNGE (Y, F, X, H, N, MR)

- Called by the main program SCMAIN.

- Integrates along the streamlines using 4-th order Runge-Kutta method.

- Returns KRUNGE =1 while the integration is being done.
  KRUNGE=2 when the integration is finished.

- Arguments are
  Y: Array of N dependent variables
  F: Array of the N derivatives of the variable Y
  X: Independent variable
  H: Stepsize ΔX
  N: Number of differential equations to be solved (N=4 for SCC)
  MR: index for the following options
  MR=0; variable stepsize 4-th order Runge-Kutta method. Using this option, the stepsize H is automatically determined for accurate integration.
  (not currently used in this code.)
  MR=2; original predetermined stepsize 4-th order Runge-Kutta method.

Subroutine LAGEXT(X1, X2, X3, Y1, Y2, Y3, C1)

- Called by subroutine INVDAT.
- Used to obtain the inviscid properties (u_e/V_∞, Cp, R, u_R/V_∞, u_Θ/V_∞) at nose point.
- Calculates C1 at X=0 by second order Lagrangian extrapolation with given (X1,Y1), (X2, Y2) and (X3, Y3).

Subroutine STAGLO

- Called by the main program SCMAIN and calls subroutine IUNI.
- Parameters IM and JM are given.
- Locates the stagnation point from the inviscid data. The location of the stagnation point is assumed to be where u_Θ is zero.
- Interpolates R at the stagnation point and calculates x'_s and z'_s.

Subroutine STIBI

- Called by subroutine FCN.
- Avaiable as a mathematical library routine at NASA Langley Research Center.
- Interpolates the spline under tension approximation to one function of two independent variables. Input values of the function are specified at all nodes of a rectangular grid. Output values may be requested at one or more individual points or at all nodes of a second rectangular grid. In SCC, the two independent variables for the interpolation are X and ϕ.

Subroutine VAL(X, ϕ, r, rx, N)

- Called by main program SCMAIN and subroutine ECON.
- This is a geometry subroutine to interrogate the radius(r) for a given X and ϕ.
- Output: r only if N=0; r and $rx(= \partial r/\partial x)$ if N=1
- Must be supplied by the user.

3.4 Parameter and Variable Directory

ASTAR	A, stagnation point velocity gradient in the x^* direction
BSTAR	B, stagnation point velocity gradient in the y^* direction
CAVT(I,J)	V_e/V_∞, inviscid total velocity on the inviscid grid
CPP	Cp, pressure coefficient
CPD(I,J)	Cp, pressure coefficient on the boundary-layer grid
CSTAR	$C^*(= B/A)$
CTH	$\cos\Theta$
DXPDY	$\partial x'/\partial y$
DY(J)	Δy
DYPDY	$\partial y'/\partial y$
DZPDY	$\partial z'/\partial y$
ENDX1,ENDXN,ENDY1,ENDYN,ENDXY	Arguments of the interpolation subroutine STIBI
EOR	ϵ, small angle to locate the initial streamlines near the stagnation point
F(1)	DX/Dx
F(2)	$D\phi/Dx$
F(4)	Ds/Dx
H1(I,J)	h_1 on the boundary-layer grid
H2(I,J)	h_2 on the boundary-layer grid
IM	number of inviscid grid points in the x-direction including the nose point
IMAX	actual number of boundary-layer grid points in the x-direction
IMAXD	maximum possible number of boundary-layer grid points in the x-direction (IMAXD$\geq$IMAX)
IENDSW, IERR, IOPT, IW	Arguments of the interpolation subroutine STIBI

IORDER, IPT, IERR	
	Arguments of the interpolation subroutine IUNI
JM	number of inviscid grid points in the y-direction, including the lines of symmetry
JMAX	actual number of boundary-layer grid points in the y-direction
JMAXD	maximum possible number of boundary-layer grid points in the y-direction (JMAXD$\geq$JMAX)
KPOINT	=1 when the shape of nose is sharp
	=0 when the shape of nose is blunted
LINOUT	Argument of the interpolation subroutine STIBI
NP	number of grid points in the y-direction in the numerical inviscid data
NPI	number of inviscid grid points in the y-direction, including the lines of symmetry (NPI=NP+2)
NT	number of grid points in the x-direction in the numerical inviscid data
NTI	number of inviscid grid points in the x-direction, including the nose point (NTI=NT+1)
PI	π
RT(I,J)	R on the inviscid grid
PCOEF(I,J)	Cp on the inviscid grid
PHIT(J)	ϕ on the inviscid grid
RADIUS(I,J)	r on the boundary-layer grid
RXSTAG	$\partial r/\partial X$ at the stagnation point
RX,RP	$\partial r/\partial X$, $\partial r/\partial \phi$
SIGMA	Arguments of the interpolation subroutine STIBI
SR	r
STH	$\sin\Theta$
THETAR	θ_r

THSTAG	θ_s
UE(I,J)	u_e/V_∞ on the boundary-layer grid
UPT(I,J)	u_ϕ/V_∞ on the inviscid grid
URT(I,J)	u_R/V_∞ on the inviscid grid
U3T(I,J)	u_Θ/V_∞ on the inviscid grid
V	u_e/V_∞
VE(1,J)	v_e/V_∞ based on the body-oriented coordinate system at i=1
VP	u_ϕ/V_∞
VR	u_R/V_∞
VT	u_Θ/V_∞
VX(I,J)	$u_{x'}/V_\infty$ on the inviscid grid
VY(I,J)	$u_{y'}/V_\infty$ on the inviscid grid
VZ(I,J)	$u_{z'}/V_\infty$ on the inviscid grid
WK	Arguments of the interpolation subroutine STIBI
X(I)	x_i
XIN(I)	X on the inviscid grid
XO(I,J)	x' on the inviscid grid
XOSP	X_{osp}, X of the origin of the spherical coordinates ($X_{osp} = 1$ is used in this code)
XPD(I,J)	x' on the boundary-layer grid
XPS	x'_s, x' of the stagnation point
X3TP(I)	$\pi - \Theta$ on the inviscid grid along the windward line of symmetry
Y(J)	y_j for the boundary-layer grid
YY(1)	X
YY(2)	ϕ
YY(4)	s
YO(I,J)	y' on the inviscid grid

YPD(I,J)	y' on the boundary-layer grid
ZO(I,J)	z' on the inviscid grid
ZPD(I,J)	z' on the boundary-layer grid
ZPS	z'_s, z' of the stagnation point

3.5 Input

The input to SCC is given or read through subroutine INPUT and INVDAT, as follows:
(1) In subroutine INPUT, the following quantities are given instead of being read from a file.

KPOINT	=1 when the shape of nose is sharp.
	=0 when the shape of nose is blunted.
EOR	$=\epsilon$, small angle to locate the initial streamlines near the stagnation point, typically 0.01. It should be noted that if the inviscid solution near the stagnation point (or near the nose) is not accurate, this value should be increased. EOR is needed only if KPOINT is 0, i.e., for the blunted nose body.

The x and y distributions for the streamline boundary-layer grid are specified.

First, IMAX and $x(i)$ for i=1,2,..,IMAX are set.
For the blunted nose body, $x_{i=1}$ is not used and does not affect the boundary-layer solution. Therefore, $x_{i=1}$ can be given as zero or another small value like 0.01. Initial locations of the streamlines are determined by EOR. For the sharp nose body, $x_{i=1}$ is nearly the same as $X_{i=1}$, and $x_{i=1}$ should not be so small that it restricts the next step sizes (Δx). The x distribution can be given arbitrarily. However, the stepsizes(Δx) near the nose must be small to obtain nonoscillating boundary-layer parameters.

Next, JMAX and $y(j)$ for j=1,2,..,JMAX are set,
where Y(1)=0 on the windward line of symmetry, and Y(JMAX)=π on the leeward line of symmetry. In this coordinate system, the y distribution is used to locate the initial streamlines near the stagnation point or near the nose tip. This y-distribution can be given arbitrarily. Even when a uniform grid distribution in the y direction is given, the downstream grid spacings will generally become nonuniform.

(2) The numerical inviscid solution based on the Cartesian coordinates are read through subroutine INVDAT. This sets the values of

x', y', z', $u_{x'}/V_\infty$, $u_{y'}/V_\infty$, $u_{z'}/V_\infty$, Cp for i=1,2,..,NT, j=1,2,..,NP.

It is to be noted that j is increasing from the windward line of symmetry to the leeward line of symmetry.

3.6 Output

The output from SCC, which can be used as input for 3DBLC, is written by the main program SCMAIN on file fort.25. The output lists the boundary-layer edge conditions including the following:

x'_s, z'_s, θ_r, A, B, C^*

(These quantities are given only for the blunted nose body; for the sharp nose body, they are not given.)

$x(i)$ for i=1,2,..,IMAX

$y(j)$ for j=1,2,..,JMAX

x', y', z', u_e/V_∞, v_e/V_∞, s, h_2, Cp for i=1,2,..,IMAX, j=1,2,..,JMAX

Here, h_1 is not necessary; it will be defined as V_∞/u_e in 3DBLC. Because the streamline coordinates system is used, v_e and $\cos\theta$ are zero throughout the field. However, it should be noted that the velocity components based on the body-oriented coordinate system are calculated for i=1 when using the streamline coordinates on the sharp nose body; therefore v_e for i=1 may not be zero in this case.

3.7 Sample Case

For a sample case, the boundary-layer edge conditions on the streamline boundary-layer grid on a general aviation fuselage at an angle of attack 3^o are calculated. The inviscid solution was obtained using a 53x36(IxJ) inviscid grid from the Hess code [1] for a compressible flow ($M_\infty = 0.3$). To reduce the input data, only the first 15x36(IxJ) inviscid grid solution is used for this sample case. The input data for the inviscid solution is the same as that used for the calculation of the body-oriented boundary-layer grid in Part 2. Also, to reduce the size of the output data, only a 20x31(IxJ) streamline boundary-layer grid is generated, i.e., 31 streamlines are integrated for 20 steps. For this case, parameters are given as IM=16(=15+1), JM=38(=36+2), IMAXD=100 ($\geq$ 20), JMAXD=51 ($\geq$ 31).

For the sample case input, the subroutine INPUT program is presented. Since the inviscid solution used for this code is the same as that used for BCC, the inviscid solution is not listed here. The output written on file fort.25 is presented for a sample case output.

3.7.1 Sample Case Input

```
c###########################################################################

      subroutine input

c###########################################################################
c**                                                                      **
c**   subroutine to read input data                                      **
c**                                                                      **

      parameter(im=16,jm=38,imaxd=100,jmaxd=51)
      common/com1/pi,pio2,dtr,rtd
      common/com2/ir,iw
      common/point/kpoint
      common/com5/imax,jmax
      common/com6/eor
      common/sl1/x(imaxd),y(jmaxd)

      ir=10
      iw=6

c**************************************************************************
c
c     description of inputs
c     imax= no.of steps in the streamline direction
c     jmax=number of streamlines to be computed
c     eor=ratio of epsilon to r on starting circle
c         (approx.value=.01, but, if the inviscid solution near the
c          stagnation point is not accurate, this value should be
c          increased up to 0.05)
c
c**************************************************************************

      imax=20
      jmax=31

      kpoint=0
      eor=0.05

c
c     x-distribution is given
c
      x(1)=0.001
      do 250 i=2,imax
      if(i.le.5)dx=0.0005
      if(i.gt.5.and.l.le.20)dx=0.002
      if(i.gt.20.and.l.le.80)dx=0.01
      if(i.gt.80)dx=0.04
      x(i)=x(i-1)+dx
      write(6,*)'i=',i,'x=',x(i)
 250  continue

c
c     y-distribution is given
c
      pi=acos(-1.)
```

```
      do 270 i=1,jmax
      y(i)=pi*(1.-(jmax-i)/(jmax-1.))
270   continue

      if(imax.gt.imaxd)write(6,*)'change imaxd to',imax
      if(jmax.gt.jmaxd)write(6,*)'change jmaxd to',jmax

      pio2=pi/2.
      dtr=pi/180.
      rtd=180/pi

      return
      end
```

3.7.2 Sample Case Output

```
0.588477E-03-0.613555E-02 0.168517E+00 0.110475E+02 0.801237E+01 0.725267E+00
      20          31
 0.100000E-02  0.150000E-02  0.200000E-02  0.250000E-02  0.300000E-02
 0.500000E-02  0.700000E-02  0.900000E-02  0.110000E-01  0.130000E-01
 0.150000E-01  0.170000E-01  0.190000E-01  0.210000E-01  0.230000E-01
 0.250000E-01  0.270000E-01  0.290000E-01  0.310000E-01  0.330000E-01
 0.000000E+00  0.104720E+00  0.209440E+00  0.314159E+00  0.418879E+00
 0.523599E+00  0.628319E+00  0.733038E+00  0.837758E+00  0.942478E+00
 0.104720E+01  0.115192E+01  0.125664E+01  0.136136E+01  0.146608E+01
 0.157080E+01  0.167552E+01  0.178024E+01  0.188496E+01  0.198968E+01
 0.209440E+01  0.219911E+01  0.230383E+01  0.240856E+01  0.251327E+01
 0.261799E+01  0.272271E+01  0.282743E+01  0.293215E+01  0.303687E+01
 0.314159E+01
  1    1   0.2131660E-01   0.1727211E-07  -0.5502135E-01   0.5309875E-01
           0.5565680E+00   0.0000000E+00   0.4906141E-01   0.7011493E+00
  1    2   0.2140342E-01   0.5136549E-02  -0.5495409E-01   0.5331881E-01
           0.5568949E+00   0.0000000E+00   0.4909295E-01   0.7007861E+00
  1    3   0.2150218E-01   0.1026911E-01  -0.5454088E-01   0.5372070E-01
           0.5562661E+00   0.0000000E+00   0.4925086E-01   0.7015188E+00
  1    4   0.2161471E-01   0.1538245E-01  -0.5378032E-01   0.5430236E-01
           0.5546848E+00   0.0000000E+00   0.4949043E-01   0.7033401E+00
  1    5   0.2173857E-01   0.2046011E-01  -0.5266307E-01   0.5505227E-01
           0.5522384E+00   0.0000000E+00   0.4986230E-01   0.7061423E+00
  1    6   0.2187476E-01   0.2549301E-01  -0.5117871E-01   0.5596325E-01
           0.5490109E+00   0.0000000E+00   0.5038423E-01   0.7098241E+00
  1    7   0.2201824E-01   0.3046072E-01  -0.4930693E-01   0.5701629E-01
           0.5448820E+00   0.0000000E+00   0.5114413E-01   0.7144900E+00
  1    8   0.2217211E-01   0.3536104E-01  -0.4702277E-01   0.5820671E-01
           0.5400218E+00   0.0000000E+00   0.5223080E-01   0.7199321E+00
  1    9   0.2233158E-01   0.4017415E-01  -0.4428529E-01   0.5951577E-01
           0.5344703E+00   0.0000000E+00   0.5378800E-01   0.7260934E+00
  1   10   0.2249721E-01   0.4489562E-01  -0.4103163E-01   0.6093723E-01
           0.5283826E+00   0.0000000E+00   0.5586878E-01   0.7327759E+00
  1   11   0.2265898E-01   0.4947086E-01  -0.3718742E-01   0.6243951E-01
           0.5218853E+00   0.0000000E+00   0.5859625E-01   0.7398231E+00
  1   12   0.2281635E-01   0.5385179E-01  -0.3264734E-01   0.6400758E-01
           0.5152732E+00   0.0000000E+00   0.6222019E-01   0.7469040E+00
  1   13   0.2296127E-01   0.5793228E-01  -0.2728133E-01   0.6560361E-01
           0.5088928E+00   0.0000000E+00   0.6659305E-01   0.7536401E+00
  1   14   0.2307175E-01   0.6149529E-01  -0.2098388E-01   0.6713897E-01
           0.5036086E+00   0.0000000E+00   0.7128835E-01   0.7591636E+00
  1   15   0.2312814E-01   0.6423390E-01  -0.1374677E-01   0.6849790E-01
           0.5003346E+00   0.0000000E+00   0.7593352E-01   0.7625557E+00
  1   16   0.2309866E-01   0.6578106E-01  -0.5668722E-02   0.6952745E-01
           0.4999020E+00   0.0000000E+00   0.7778996E-01   0.7629762E+00
  1   17   0.2290910E-01   0.6580716E-01   0.2467883E-02   0.7002008E-01
           0.5015913E+00   0.0000000E+00   0.7722396E-01   0.7612140E+00
  1   18   0.2243826E-01   0.6459934E-01   0.1044825E-01   0.7018203E-01
           0.5043224E+00   0.0000000E+00   0.8050317E-01   0.7583557E+00
  1   19   0.2193052E-01   0.6172913E-01   0.1879850E-01   0.6991190E-01
           0.5133672E+00   0.0000000E+00   0.8181310E-01   0.7488213E+00
  1   20   0.2140341E-01   0.5727113E-01   0.2590239E-01   0.6884533E-01
           0.5262801E+00   0.0000000E+00   0.7560533E-01   0.7349193E+00
  1   21   0.2091940E-01   0.5207658E-01   0.3131036E-01   0.6728679E-01
           0.5397345E+00   0.0000000E+00   0.6744111E-01   0.7200801E+00
```

1	22	0.2049985E-01	0.4676031E-01	0.3529478E-01	0.6557029E-01
		0.5519031E+00	0.0000000E+00	0.6052845E-01	0.7063442E+00
1	23	0.2015039E-01	0.4152218E-01	0.3829034E-01	0.6387818E-01
		0.5621388E+00	0.0000000E+00	0.5584310E-01	0.6945565E+00
1	24	0.1986676E-01	0.3636179E-01	0.4061095E-01	0.6228216E-01
		0.5708042E+00	0.0000000E+00	0.5299279E-01	0.6844117E+00
1	25	0.1964075E-01	0.3123998E-01	0.4243757E-01	0.6081343E-01
		0.5781351E+00	0.0000000E+00	0.5137400E-01	0.6757129E+00
1	26	0.1946770E-01	0.2611776E-01	0.4387778E-01	0.5949702E-01
		0.5842409E+00	0.0000000E+00	0.5058524E-01	0.6683841E+00
1	27	0.1934543E-01	0.2096435E-01	0.4500046E-01	0.5836282E-01
		0.5891426E+00	0.0000000E+00	0.5028610E-01	0.6624448E+00
1	28	0.1927176E-01	0.1577314E-01	0.4584524E-01	0.5744440E-01
		0.5931574E+00	0.0000000E+00	0.5025856E-01	0.6575512E+00
1	29	0.1924171E-01	0.1053746E-01	0.4643896E-01	0.5677200E-01
		0.5961660E+00	0.0000000E+00	0.5033484E-01	0.6538659E+00
1	30	0.1924910E-01	0.5274452E-02	0.4680067E-01	0.5637625E-01
		0.5980836E+00	0.0000000E+00	0.5037227E-01	0.6515038E+00
1	31	0.1928287E-01	0.1473854E-07	0.4695051E-01	0.5628153E-01
		0.5989781E+00	0.0000000E+00	0.5038835E-01	0.6504023E+00
2	1	0.2179582E-01	0.1746645E-07	-0.5564045E-01	0.5399316E-01
		0.5614840E+00	0.0000000E+00	0.4944862E-01	0.6955535E+00
2	2	0.2188341E-01	0.5177084E-02	-0.5557247E-01	0.5421271E-01
		0.5617936E+00	0.0000000E+00	0.4947965E-01	0.6952044E+00
2	3	0.2198177E-01	0.1035001E-01	-0.5515593E-01	0.5461563E-01
		0.5611313E+00	0.0000000E+00	0.4964374E-01	0.6959769E+00
2	4	0.2209350E-01	0.1550486E-01	-0.5439013E-01	0.5519986E-01
		0.5595186E+00	0.0000000E+00	0.4989476E-01	0.6978442E+00
2	5	0.2221626E-01	0.2062491E-01	-0.5326578E-01	0.5595375E-01
		0.5570349E+00	0.0000000E+00	0.5028460E-01	0.7007080E+00
2	6	0.2235098E-01	0.2570241E-01	-0.5177131E-01	0.5687005E-01
		0.5537605E+00	0.0000000E+00	0.5082739E-01	0.7044712E+00
2	7	0.2249260E-01	0.3071575E-01	-0.4988643E-01	0.5792999E-01
		0.5495713E+00	0.0000000E+00	0.5161476E-01	0.7092392E+00
2	8	0.2264402E-01	0.3566447E-01	-0.4758431E-01	0.5912866E-01
		0.5446277E+00	0.0000000E+00	0.5274067E-01	0.7148145E+00
2	9	0.2280058E-01	0.4052821E-01	-0.4482339E-01	0.6044734E-01
		0.5389879E+00	0.0000000E+00	0.5435222E-01	0.7211205E+00
2	10	0.2296276E-01	0.4530410E-01	-0.4153778E-01	0.6187959E-01
		0.5327720E+00	0.0000000E+00	0.5649966E-01	0.7279949E+00
2	11	0.2312056E-01	0.4993540E-01	-0.3765205E-01	0.6339371E-01
		0.5261195E+00	0.0000000E+00	0.5930579E-01	0.7352641E+00
2	12	0.2327308E-01	0.5437367E-01	-0.3305676E-01	0.6497413E-01
		0.5193416E+00	0.0000000E+00	0.6304157E-01	0.7425752E+00
2	13	0.2341231E-01	0.5851100E-01	-0.2761688E-01	0.6658244E-01
		0.5127575E+00	0.0000000E+00	0.6754267E-01	0.7495749E+00
2	14	0.2351658E-01	0.6212427E-01	-0.2122543E-01	0.6812823E-01
		0.5072691E+00	0.0000000E+00	0.7237005E-01	0.7553512E+00
2	15	0.2356654E-01	0.6489927E-01	-0.1387259E-01	0.6949377E-01
		0.5038256E+00	0.0000000E+00	0.7714403E-01	0.7589430E+00
2	16	0.2353099E-01	0.6645918E-01	-0.5660761E-02	0.7052428E-01
		0.5032874E+00	0.0000000E+00	0.7898599E-01	0.7594743E+00
2	17	0.2333421E-01	0.6647235E-01	0.2593581E-02	0.7101363E-01
		0.5049234E+00	0.0000000E+00	0.7835972E-01	0.7577574E+00
2	18	0.2285100E-01	0.6523132E-01	0.1069070E-01	0.7117021E-01
		0.5076409E+00	0.0000000E+00	0.8180487E-01	0.7548980E+00
2	19	0.2233290E-01	0.6228468E-01	0.1917689E-01	0.7088262E-01
		0.5167842E+00	0.0000000E+00	0.8317882E-01	0.7452024E+00
2	20	0.2180382E-01	0.5773133E-01	0.2637959E-01	0.6979209E-01
		0.5299179E+00	0.0000000E+00	0.7673556E-01	0.7309731E+00

2	21	0.2132332E-01	0.5244737E-01	0.3184580E-01	0.6820989E-01
		0.5435712E+00	0.0000000E+00	0.6829768E-01	0.7158163E+00
2	22	0.2090963E-01	0.4705848E-01	0.3586122E-01	0.6647301E-01
		0.5558765E+00	0.0000000E+00	0.6116609E-01	0.7018342E+00
2	23	0.2056615E-01	0.4176251E-01	0.3887227E-01	0.6476444E-01
		0.5661590E+00	0.0000000E+00	0.5633678E-01	0.6899129E+00
2	24	0.2028803E-01	0.3655468E-01	0.4120019E-01	0.6315500E-01
		0.5748472E+00	0.0000000E+00	0.5339361E-01	0.6796736E+00
2	25	0.2006708E-01	0.3139344E-01	0.4302998E-01	0.6167526E-01
		0.5821734E+00	0.0000000E+00	0.5170918E-01	0.6709223E+00
2	26	0.1989845E-01	0.2623774E-01	0.4447069E-01	0.6034990E-01
		0.5882506E+00	0.0000000E+00	0.5087377E-01	0.6635808E+00
2	27	0.1977963E-01	0.2105517E-01	0.4559236E-01	0.5920864E-01
		0.5931367E+00	0.0000000E+00	0.5054139E-01	0.6576238E+00
2	28	0.1970879E-01	0.1583813E-01	0.4643583E-01	0.5828453E-01
		0.5971373E+00	0.0000000E+00	0.5049000E-01	0.6527157E+00
2	29	0.1968082E-01	0.1057891E-01	0.4702801E-01	0.5760792E-01
		0.6001134E+00	0.0000000E+00	0.5054935E-01	0.6490469E+00
2	30	0.1968937E-01	0.5294103E-02	0.4738881E-01	0.5720951E-01
		0.6020176E+00	0.0000000E+00	0.5056964E-01	0.6466888E+00
2	31	0.1972364E-01	0.1492296E-07	0.4753799E-01	0.5711355E-01
		0.6029072E+00	0.0000000E+00	0.5057577E-01	0.6455870E+00

***** For brevity, output for i=3,4,..,19 is deleted. *****

20	1	0.5339842E-01	0.2722279E-07	-0.8671987E-01	0.1013265E+00
		0.7477883E+00	0.0000000E+00	0.6669593E-01	0.4452463E+00
20	2	0.5348459E-01	0.6982815E-02	-0.8659786E-01	0.1015430E+00
		0.7476993E+00	0.0000000E+00	0.6683928E-01	0.4453836E+00
20	3	0.5356924E-01	0.1398360E-01	-0.8609015E-01	0.1020083E+00
		0.7468434E+00	0.0000000E+00	0.6734782E-01	0.4466891E+00
20	4	0.5365556E-01	0.2101623E-01	-0.8518595E-01	0.1027191E+00
		0.7451553E+00	0.0000000E+00	0.6815946E-01	0.4492517E+00
20	5	0.5374004E-01	0.2808297E-01	-0.8386260E-01	0.1036662E+00
		0.7426076E+00	0.0000000E+00	0.6940788E-01	0.4531115E+00
20	6	0.5382244E-01	0.3521694E-01	-0.8208281E-01	0.1048481E+00
		0.7392169E+00	0.0000000E+00	0.7109463E-01	0.4582351E+00
20	7	0.5389548E-01	0.4240713E-01	-0.7979965E-01	0.1062497E+00
		0.7348646E+00	0.0000000E+00	0.7349938E-01	0.4647825E+00
20	8	0.5396207E-01	0.4972142E-01	-0.7692862E-01	0.1078682E+00
		0.7294322E+00	0.0000000E+00	0.7683021E-01	0.4728999E+00
20	9	0.5401718E-01	0.5715732E-01	-0.7336956E-01	0.1096888E+00
		0.7228225E+00	0.0000000E+00	0.8137351E-01	0.4827014E+00
20	10	0.5404890E-01	0.6477350E-01	-0.6893575E-01	0.1117224E+00
		0.7148196E+00	0.0000000E+00	0.8759296E-01	0.4944515E+00
20	11	0.5404298E-01	0.7255226E-01	-0.6339210E-01	0.1139398E+00
		0.7053228E+00	0.0000000E+00	0.9571612E-01	0.5082423E+00
20	12	0.5399594E-01	0.8043021E-01	-0.5641630E-01	0.1163088E+00
		0.6942089E+00	0.0000000E+00	0.1068476E+00	0.5241536E+00
20	13	0.5387398E-01	0.8826607E-01	-0.4746002E-01	0.1187830E+00
		0.6816097E+00	0.0000000E+00	0.1209916E+00	0.5418864E+00
20	14	0.5366346E-01	0.9550011E-01	-0.3604664E-01	0.1211705E+00
		0.6688458E+00	0.0000000E+00	0.1365166E+00	0.5595481E+00
20	15	0.5336437E-01	0.1011360E+00	-0.2193347E-01	0.1231819E+00
		0.6585218E+00	0.0000000E+00	0.1514878E+00	0.5736009E+00
20	16	0.5300516E-01	0.1038876E+00	-0.5454816E-02	0.1244504E+00
		0.6540886E+00	0.0000000E+00	0.1503893E+00	0.5795627E+00
20	17	0.5240468E-01	0.1034006E+00	0.9467792E-02	0.1249033E+00
		0.6534757E+00	0.0000000E+00	0.1472242E+00	0.5803867E+00
20	18	0.5117634E-01	0.1000908E+00	0.2509039E-01	0.1248460E+00

		0.6554263E+00	0.0000000E+00	0.1653035E+00	0.5777726E+00
20	19	0.5007504E-01	0.9172904E-01	0.4197872E-01	0.1235133E+00
		0.6696919E+00	0.0000000E+00	0.1673374E+00	0.5583991E+00
20	20	0.4954826E-01	0.8077651E-01	0.5428970E-01	0.1209157E+00
		0.6904761E+00	0.0000000E+00	0.1403251E+00	0.5294422E+00
20	21	0.4933331E-01	0.7029593E-01	0.6207415E-01	0.1179985E+00
		0.7076862E+00	0.0000000E+00	0.1118721E+00	0.5048262E+00
20	22	0.4921802E-01	0.6111583E-01	0.6703079E-01	0.1152506E+00
		0.7199948E+00	0.0000000E+00	0.9166652E-01	0.4868659E+00
20	23	0.4913859E-01	0.5299253E-01	0.7038915E-01	0.1127868E+00
		0.7288219E+00	0.0000000E+00	0.7919842E-01	0.4738010E+00
20	24	0.4908637E-01	0.4556711E-01	0.7280654E-01	0.1105964E+00
		0.7354037E+00	0.0000000E+00	0.7165581E-01	0.4639657E+00
20	25	0.4905048E-01	0.3858873E-01	0.7460219E-01	0.1086677E+00
		0.7404506E+00	0.0000000E+00	0.6698173E-01	0.4563653E+00
20	26	0.4903155E-01	0.3189647E-01	0.7595491E-01	0.1069883E+00
		0.7443505E+00	0.0000000E+00	0.6415582E-01	0.4504582E+00
20	27	0.4903331E-01	0.2536306E-01	0.7697457E-01	0.1055679E+00
		0.7474102E+00	0.0000000E+00	0.6245471E-01	0.4458036E+00
20	28	0.4905703E-01	0.1893588E-01	0.7772154E-01	0.1044276E+00
		0.7496761E+00	0.0000000E+00	0.6130892E-01	0.4423398E+00
20	29	0.4909214E-01	0.1258406E-01	0.7822889E-01	0.1035968E+00
		0.7513192E+00	0.0000000E+00	0.6072512E-01	0.4398260E+00
20	30	0.4913681E-01	0.6243661E-02	0.7853049E-01	0.1031012E+00
		0.7523556E+00	0.0000000E+00	0.6011952E-01	0.4382388E+00
20	31	0.4918521E-01	0.2468921E-07	0.7864898E-01	0.1029614E+00
		0.7527975E+00	0.0000000E+00	0.5963464E-01	0.4375606E+00

3.8 FORTRAN Listing of SCC

Subroutine IUNI, STIBI, and VAL are not presented here.

```
c#######################################################################

      program scmain

c#######################################################################
c**
c**  program for calculating streamline coordinates
c**  with inviscid solution obtained from the numerical inviscid code
c**

      parameter(im=16,jm=38,imaxd=100,jmaxd=51)
      common/cpcom/cpp
      common/com1/pi,pio2,dtr,rtd
      common/com2/ir,iw
      common/com3/iordr(2),iptb(2),ider
      common/com5/imax,jmax
      common/com6/eor
      common/point/kpoint
      common/sl1/x(imaxd),y(jmaxd)
      common/invtab/phit(jm),rt(im,jm),urt(im,jm)
     1,upt(im,jm),u3t(im,jm),cavt(im,jm),pcoef(im,jm)
     2,x3tp(im),xin(im)
      common/invcon/npi,nti,xosp
      common/stgpt/thstag,xps,zps
      common/nn/n
      common/rr/r,rx,rp,rs,sth,cth
      common/vv/v

      dimension yy(4),f(4),
     &xpd(imaxd,jmaxd),zpd(imaxd,jmaxd),ypd(imaxd,jmaxd)
      dimension yd(4,imaxd,jmaxd),fd(4,jmaxd),ue(imaxd,jmaxd)
     &,ve(imaxd,jmaxd),h2(imaxd,jmaxd),h1(imaxd,jmaxd),cpd(imaxd,jmaxd)
      dimension dy(jmaxd)

      iordr(1)=2
      iordr(2)=2
      iptb(1)=-1

      call input

c     start calculation for streamlines

      call invdat

      if(kpoint.eq.1)then
      do 41 n=1,jmax
      yd(1,1,n)=x(1)
      yd(2,1,n)=y(n)
      yy(1)=x(1)
      yy(2)=y(n)
      yy2n=yy(2)-pio2
c      call csgeom(1,x(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(x(1),yy2n,r,rx,0)
      xs=yy(1)-xosp
      rs=sqrt(xs**2+r**2)
```

```
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)
      do 36 nc=1,4
      fd(nc,n)=f(nc)
 36   continue

      xpd(1,n)=yy(1)
      ypd(1,n)=r*cos(yy(2)-pio2)
      zpd(1,n)=r*sin(yy(2)-pio2)
      yy(4)=sqrt(xpd(1,n)**2+ypd(1,n)**2+zpd(1,n)**2)
      yd(4,1,n)=yy(4)

 41   continue
      go to 330
      endif

      call staglo
c      call csgeom(1,xps,-pio2,r,rxstag,rp,rxx,rxp,rpp)
      if(xps.eq.0)then
      thetar=0.
      go to 15
      endif
      call val(xps,-pio2,r,rxstag,1)
      thetar=atan(1/rxstag)
 15     write(6,*)'xps=',xps,' zps=',zps,'zps(val)=',-r,'thetar=',thetar
c*********************************************************************
c**                                                                 **
c**   independent variable of integration is x                      **
c**   v=velocity and r=cylindrical radius                           **
c**   s=distance along streamline                                   **
c**   yy(1)=x  yy(2)=phi  yy(4)=s                                   **
c**   f(i)=d(yy(i))/dx,  i=1,2,3,4                                  **
c**                                                                 **
c*********************************************************************
      calp=-cos(thstag)
      salp=sin(thstag)
      sqe=sqrt(1.-eor**2)

      ysm1=y(1)-pio2
      do 450 n=1,jmax
      ys=y(n)-pio2
      if(ys*ysm1.le.0)then
      jbstar=n
      go to 460
      endif
      ysm1=ys
 450  continue
 460  l=1

c     calculate properties on epsilon cone

      do 500 n=1,jmax
      sb=sin(y(n)-pio2)
      cb1=cos(y(n)-pio2)
      cth=-sqe*calp-eor*sb*salp
      th=acos(cth)
      tn=-sqe*salp+eor*sb*calp
      yy(2)=asin(tn/sqrt(tn**2+(eor*cb1)**2))+pio2
      yy2n=yy(2)-pio2
```

```
      call econ(th,xosp,yy2n,yy(1))
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      rs=(yy(1)-xosp)/cth
      xs=yy(1)-xosp
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)

      xpd(1,n)=yy(1)
      ypd(1,n)=r*cos(yy(2)-pio2)
      zpd(1,n)=r*sin(yy(2)-pio2)
      yy(4)=sqrt((xpd(1,n)-xps)**2+ypd(1,n)**2+(zpd(1,n)-zps)**2)

      if(n.eq.jbstar)then
      bstar=v/yy(4)
      endif
      if(n.eq.jmax)then
      astar=v/yy(4)
      cstar=bstar/astar
      write(6,*)' astar=',astar,' bstar=',bstar,' cstar=',cstar
      endif

      do 51 nc=1,4
      fd(nc,n)=f(nc)
 51   yd(nc,1,n)=yy(nc)
 500  continue

c     integrating forward or back to x=constant


330   l=1
      do 550 n=1,jmax
      it=0
      do 68 nc=1,4
      f(nc)=fd(nc,n)
 68   yy(nc)=yd(nc,1,n)

      if(n.eq.1)go to 100

200   it=it+1
      cost=((xpd2-xpd(1,n-1))*(xpd(1,n)-xpd(1,n-1))
     &+(ypd2-ypd(1,n-1))*(ypd(1,n)-ypd(1,n-1))
     &+ (zpd2-zpd(1,n-1))*(zpd(1,n)-zpd(1,n-1)))
     &/(sqrt((xpd2-xpd(1,n-1))**2+(ypd2-ypd(1,n-1))**2+(zpd2-zpd(1,n-1))
     &**2)*sqrt((xpd(1,n)-xpd(1,n-1))**2+(ypd(1,n)-ypd(1,n-1))**2
     &+(zpd(1,n)-zpd(1,n-1))**2))
c      write(6,*)' n=',n,' it=',it,' dt=',dt,' cost=',cost

      if(it.ge.50)then
      write(6,*)'iteration fails for n=',n
      if(kpoint.eq.1)write(6,*)'increase x(1)'
      if(kpoint.eq.0)write(6,*)'increase EOR'
      stop
      endif

      err=0.0001
      dtt=0.001
      if(it.eq.1.and.cost.gt.err)dt=-dtt
      if(it.eq.1.and.cost.lt.-err)dt=dtt
```

```
      if(cost.le.err.and.cost.ge.-err)go to 400
      if(abs(dt).lt.1.e-8)go to 400
      if(it.gt.1.and.costo*cost.lt.0)then
      dt=-dt/2.
      endif

 33   costo=cost

 45   continue
      k2=krunge(yy,f,t,dt,4,2)
      yy2n=yy(2)-pio2
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      xs=yy(1)-xosp
      rs=sqrt(xs**2+r**2)
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)
      if(k2.eq.1)go to 45

      xpd(1,n)=yy(1)
      ypd(1,n)=r*cos(yy(2)-pio2)
      zpd(1,n)=r*sin(yy(2)-pio2)
      go to 200

 400  continue
      xpd(1,n)=yy(1)
      ypd(1,n)=r*cos(yy(2)-pio2)
      zpd(1,n)=r*sin(yy(2)-pio2)
      do 71 nc=1,4
      fd(nc,n)=f(nc)
 71   yd(nc,1,n)=yy(nc)
      if(n.eq.jmax) go to 550

 100  dt=0.0005
 65   continue
      k2=krunge(yy,f,t,dt,4,2)
      yy2n=yy(2)-pio2
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      xs=yy(1)-xosp
      rs=sqrt(xs**2+r**2)
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)
      if(k2.eq.1)go to 65

      xpd2=yy(1)
      ypd2=r*cos(yy(2)-pio2)
      zpd2=r*sin(yy(2)-pio2)
 550  continue

c
c     calculates the edge conditions at i=1 (at x=const)
c

      do 600 n=1,jmax

      do 88 nc=1,4
```

```
      f(nc)=fd(nc,n)
 88   yy(nc)=yd(nc,1,n)
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      xs=yy(1)-xosp
      rs=sqrt(xs**2+r**2)
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)
      ue(1,n)=v
      h1(1,n)=1./v
      yy2n=yy(2)-pio2
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      xpd(1,n)=yy(1)
      ypd(1,n)=r*cos(yy(2)-pio2)
      zpd(1,n)=r*sin(yy(2)-pio2)
      yy(4)=sqrt((xpd(1,n)-xps)**2+ypd(1,n)**2+(zpd(1,n)-zps)**2)
      do 152 nc=1,4
      fd(nc,n)=f(nc)
 152   yd(nc,1,n)=yy(nc)
      cpd(1,n)=cpp
 600  continue


c*************************************************************************
c
c      start integration along the streamlines
c
c*************************************************************************

      do 2000 l=2,imax

      write(6,*)'******* l=',l,'************'
      do 700 n=1,jmax
      do 58 nc=1,4
      f(nc)=fd(nc,n)
 58   yy(nc)=yd(nc,l-1,n)
      t=x(l-1)
      dx=x(l)-x(l-1)
 25   continue
      k2=krunge(yy,f,t,dx,4,2)
      if(yy(1).lt.0.)yy(1)=1.e-10
      yy2n=yy(2)-pio2
c      call csgeom(1,yy(1),yy2n,r,rx,rp,rxx,rxp,rpp)
      call val(yy(1),yy2n,r,rx,0)
      xs=yy(1)-xosp
      rs=sqrt(xs**2+r**2)
      cth=xs/rs
      sth=r/rs
      call fcn(yy,f)
      if(k2.eq.1)go to 25
      if(abs(yy(1)).gt.1.e+10.or.abs(yy(2)).gt.1.e+10)stop
      do 52 nc=1,4
      fd(nc,n)=f(nc)
 52   yd(nc,l,n)=yy(nc)
      ue(l,n)=v
      h1(l,n)=1./v
      xpd(l,n)=yy(1)
      ypd(l,n)=r*cos(yy(2)-pio2)
```

```
      zpd(l,n)=r*sin(yy(2)-pio2)
      cpd(l,n)=cpp

 700  continue
 2000 continue

c
c     calculate the metric coefficient h2
c
       do 690 n=1,jmax-1
       dy(n)=y(n+1)-y(n)
 690   continue

       do 2100 l=1,imax
       do 2100 n=1,jmax
       dxpdy=(dy(n-1)**2*xpd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *xpd(l,n)-dy(n)**2*xpd(l,n-1))/(dy(n)*dy(n-1)*(dy(n)
     & +dy(n-1)))
       dypdy=(dy(n-1)**2*ypd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *ypd(l,n)-dy(n)**2*ypd(l,n-1))/(dy(n)*dy(n-1)*(dy(n)
     & +dy(n-1)))
       dzpdy=(dy(n-1)**2*zpd(l,n+1)-(dy(n-1)**2-dy(n)**2)
     & *zpd(l,n)-dy(n)**2*zpd(l,n-1))/(dy(n)*dy(n-1)
     &*(dy(n)+dy(n-1)))


       if(n.eq.1)then
       dxpdy=(xpd(l,2)-xpd(l,1))/dy(1)
       dypdy=(ypd(l,2)-ypd(l,1))/dy(1)
       dzpdy=(zpd(l,2)-zpd(l,1))/dy(1)
       endif

       if(n.eq.jmax)then
       dxpdy=(xpd(l,jmax)-xpd(l,jmax-1))/dy(jmax-1)
       dypdy=(ypd(l,jmax)-ypd(l,jmax-1))/dy(jmax-1)
       dzpdy=(zpd(l,jmax)-zpd(l,jmax-1))/dy(jmax-1)
       endif

       h2(l,n)=sqrt(dxpdy**2+dypdy**2+dzpdy**2)

 2100 continue
       if(kpoint.eq.0)go to 2400
c
c      for the sharp nose body, the velocity components
c      based on the body-oriented coordinate system are required at i=1
c
       do 3100 j=2,jmax-1
        ph1=atan(zpd(1,j)/ypd(1,j))
        x1=xpd(1,j)
        y1=ypd(1,j)
        z1=zpd(1,j)
        x2=xpd(2,j)
        y2=ypd(2,j)
        z2=zpd(2,j)
        x3=xpd(2,j)
c      call csgeom(1,x3,ph1,r3,rx,rp,rxx,rxp,rpp)
        call val(x3,ph1,r3,rx,0)
        y3=-r3*(-cos(ph1))
        z3=r3*sin(ph1)
```

```
        ph4=ph1+0.01
c      call csgeom(1,xpd(1,j),ph4,r4,rx,rp,rxx,rxp,rpp)
        call val(xpd(1,j),ph4,r4,rx,0)
        x4=xpd(1,j)
        y4=-r4*(-cos(ph4))
        z4=r4*sin(ph4)

        costh=((y2-y1)*(y3-y1)+(z2-z1)*(z3-z1)+(x2-x1)*(x3-x1))
     &  /(sqrt((y3-y1)**2+(z3-z1)**2+(x3-x1)**2)
     &  *sqrt((y2-y1)**2+(z2-z1)**2+(x2-x1)**2))

        costh1=((y2-y1)*(y4-y1)+(z2-z1)*(z4-z1)+(x2-x1)*(x4-x1))
     &  /(sqrt((y4-y1)**2+(z4-z1)**2+(x4-x1)**2)
     &  *sqrt((y2-y1)**2+(z2-z1)**2+(x2-x1)**2))

c        write(6,*)'n=',n,'costh=',costh,'costh1=',costh1

        uesave=ue(1,j)
        ue(1,j)= uesave*costh
        ve(1,j)=uesave*sqrt(1.-costh**2)
        if(costh1.lt.0.)ve(1,j)=-uesave*sqrt(1.-costh**2)
c       ve(1,j)=ue(1,j)*costh1
c       ue(1,j)=sqrt(ue(1,j)**2-ve(1,j)**2)

 3100  continue

 2400  continue

       rewind 25
       write(25,465)xps,zps,thetar,astar,bstar,cstar
       write(25,463)imax,jmax
       write(25,461)(x(i),i=1,imax)
       write(25,461)(y(j),j=1,jmax)
       do 160 i=1,imax
       do 160 j=1,jmax
       write(25,462)i,j,xpd(i,j),ypd(i,j),zpd(i,j),yd(4,i,j),ue(i,j)
     &,ve(i,j),h2(i,j),cpd(i,j)
 160   continue
 463   format(2i10)
 462   format(2i4,4(1x,e14.7)/8x,4(1x,e14.7))
 461   format(5(1x,e13.6))
 465   format(6e13.6)
      stop
      end
```

```
c###########################################################################

      subroutine dudy(x1,x2,x3,y1,y2,y)

c###########################################################################
      a=(y1-y2)/(x1**2-x2**2-2.*x1*x3+2.*x2*x3)
      b=-2.*a*x3
      c=y1-a*x1**2-b*x1
      y=a*x3**2+b*x3+c
      return
      end
```

```
c########################################################################

      subroutine econ(th,xosp,phi,x)

c########################################################################
c**
c**   routine to calculate x from theta(th) and phi on epsilon cone    **
c**   using newtons method, drive tan(th)*(x-xosp)-r to zero           **
c**   r is cylindrical radius, initial guess is x=xosp+xosp*cos(th)    **
c
      common/com2/ir,iw
      pi=acos(-1.)
      tth=tan(th)
      x=xosp+xosp*cos(th)
      if(x.le.1.e-05)x=.01*xosp
      it=0
   10 continue
      it=it+1
      if(it.ge.51)write(iw,1000)
      if(it.ge.51)stop
c      call csgeom(1,x,phi,r,rx,rp,rxx,rxp,rpp)
      call val(x,phi,r,rx,1)
      dx=(r-tth*(x-xosp))/(tth-rx)
      x=x+dx
      if(x.le.0)x=1.0e-5
      if(abs(dx).lt.1.0e-6)go to 20
      go to 10
   20 return
 1000 format(////5x,45h*****  iteration fail in econ  -  stop  *****/)
      end
```

```
c#######################################################################

      subroutine fcn(yy,f)

c#######################################################################

      parameter(im=16,jm=38)
      common/cpcom/cpp
      common/com1/pi,pio2,dtr,rtd
      common/com3/iordr(2),iptb(2),ider
      common/invtab/phit(jm),rt(im,jm),urt(im,jm)
     1,upt(im,jm),u3t(im,jm),cavt(im,jm),pcoef(im,jm)
     2,x3tp(im),xin(im)
      common/invcon/npi,nti,xosp
      common/stgpt/thstag,xps,zps
      common/nn/n
      common/rr/r,rx,rp,rs,sth,cth
      common/vv/v
      dimension yy(4),f(4)
      integer iendsw(8),ierr,iopt(3),iwi,iw1,iw2,iw3,iw4,mx,my,mz
      real endy1(im),endyn(im),sigma
      real wk(5*im*jm)
      real wk1(5*im*jm)
      real wk2(5*im*jm)
      real wk3(5*im*jm)
      real wk4(5*im*jm)

c     biviarate spline under tension

      iendsw(1)=2
      iendsw(2)=2
      iendsw(3)=0
      iendsw(4)=0
      iopt(1)=3
      iopt(2)=3

      do 50 ii=1,nti
      endy1(ii)=0.
 50   endyn(ii)=0.
      sigma=2.0
      if(iwi.eq.1)go to 22
      iwi=0
 22   continue
      mx=1
      my=1
      mz=1
      call stibi(iopt,nti,npi,xin,phit,nti,pcoef,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,yy(1),yy(2),iwi,mz,cpp,
     &linout,wk,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi cp) ierr=',ierr
      if(iw1.eq.1)go to 33
      iw1=0
 33   continue

      call stibi(iopt,nti,npi,xin,phit,nti,cavt,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,yy(1),yy(2),iw1,mz,v,
     &linout,wk1,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi v) ierr=',ierr

      if(iw2.eq.1)go to 44
```

```
      iw2=0
44    continue

      call stibi(iopt,nti,npi,xin,phit,nti,u3t,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,yy(1),yy(2),iw2,mz,vt,
     &linout,wk2,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi vt) ierr=',ierr

      if(iw3.eq.1)go to 55
      iw3=0
55    continue

      call stibi(iopt,nti,npi,xin,phit,nti,urt,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,yy(1),yy(2),iw3,mz,vr,
     &linout,wk3,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi vr) ierr=',ierr

      iendsw(3)=2
      iendsw(4)=2

      if(iw4.eq.1)go to 66
      iw4=0
66    continue

       call stibi(iopt,nti,npi,xin,phit,nti,upt,iendsw,endx1
     &,endxn,endy1,endyn,endxy,sigma,mx,my,yy(1),yy(2),iw4,mz,vp,
     &linout,wk4,ierr)
      if(ierr.gt.0)write(6,*)' ***** ierr is gt.0 (stibi vp) ierr=',ierr
      iwi=1
      iw1=1
      iw2=1
      iw3=1
      iw4=1

      f(1)=(cth*vr-sth*vt)/v**2
      f(2)=vp/(r*v**2)
      f(4)=1./v
      return
      end
```

```
c###########################################################################

      subroutine input

c###########################################################################
c**                                                                      **
c**   subroutine to read input data                                      **
c**                                                                      **

      parameter(im=16,jm=38,imaxd=100,jmaxd=51)
      common/com1/pi,pio2,dtr,rtd
      common/com2/ir,iw
      common/point/kpoint
      common/com5/imax,jmax
      common/com6/eor
      common/sl1/x(imaxd),y(jmaxd)

      ir=10
      iw=6

c*************************************************************************
c
c     description of inputs
c     imax= no.of steps in the streamline direction
c     jmax=number of streamlines to be computed
c     eor=ratio of epsilon to r on starting circle
c         (approx.value=.01, but, if the inviscid solution near the
c          stagnation point is not accurate, this value should be
c          increased up to 0.05)
c
c*************************************************************************

      imax=20
      jmax=31

      kpoint=0
      eor=0.05

c
c     x-distribution is given
c
      x(1)=0.001
      do 250 i=2,imax
      if(i.le.5)dx=0.0005
      if(i.gt.5.and.1.le.20)dx=0.002
      if(i.gt.20.and.1.le.80)dx=0.01
      if(i.gt.80)dx=0.04
      x(i)=x(i-1)+dx
      write(6,*)'i=',i,'x=',x(i)
 250  continue

c
c     y-distribution is given
c
      pi=acos(-1.)
```

```
      do 270 i=1,jmax
      y(i)=pi*(1.-(jmax-i)/(jmax-1.))
270   continue

      if(imax.gt.imaxd)write(6,*)'change imaxd to',imax
      if(jmax.gt.jmaxd)write(6,*)'change jmaxd to',jmax

      pio2=pi/2.
      dtr=pi/180.
      rtd=180/pi

      return
      end
```

```
c#######################################################################

      subroutine invdat

c#######################################################################
c**                                                                   **
c**    read inviscid data
c**                                                                   **

      parameter(im=16,jm=38)
      common/hess/xo(im,jm),yo(im,jm),zo(im,jm)
     &,vx(im,jm),vy(im,jm),vz(im,jm)
      common/com1/pi,pio2,dtr,rtd
      common/invtab/phit(jm),rt(im,jm),urt(im,jm)
     1,upt(im,jm),u3t(im,jm),cavt(im,jm),pcoef(im,jm)
     2,x3tp(im),xin(im)
      common/invcon/npi,nti,xosp
      common/point/kpoint
      common/com3/iordr(2),iptb(2),ider
      common/wind/u3tw(im),urtw(im),rtw(im),cavw(im),cpw(im)
      common/lee/u3tl(im),urtl(im),rtl(im),cavl(im),cpl(im)
      dimension cavnd(jm),cpnd(jm),urnd(jm),u3tn(jm),upnd(jm)

c
c     x3t=theta
c     u3t=table of (u)theta
c
c     read inviscid data
c
      xosp=1.

      rewind 2
 100  read(2,410,end=1000)is,lk,ksorce

      do 800 k=1,ksorce
      read(2,411)xo(lk,k),yo(lk,k),zo(lk,k),vx(lk,k),
     &vy(lk,k),vz(lk,k),pcoef(lk,k)
      if(yo(lk,k).lt.0)yo(lk,k)=1.e-7
      sr=(yo(lk,k)**2+zo(lk,k)**2)**0.5
      rt(lk,k)=(sr**2+(xo(lk,k)-xosp)**2)**0.5
      cavt(lk,k)=sqrt(vx(lk,k)**2+vy(lk,k)**2+vz(lk,k)**2)
      urt(lk,k)=(yo(lk,k)*vy(lk,k)+zo(lk,k)*vz(lk,k)
     &+(xo(lk,k)-xosp)*vx(lk,k))/rt(lk,k)
      u3t(lk,k)=(yo(lk,k)*(xo(lk,k)-xosp)*vy(lk,k)+zo(lk,k)*(xo(lk,k)
     &-xosp)*vz(lk,k)-sr**2*vx(lk,k))/(sr*rt(lk,k))
      upt(lk,k)=-(zo(lk,k)*vy(lk,k)-yo(lk,k)*vz(lk,k))/sr
      if(lk.eq.3)phit(k)=atan(zo(lk,k)/yo(lk,k))+pio2
 800  continue
      x3t=asin((yo(lk,1)**2+zo(lk,1)**2)**0.5/rt(lk,1))
      if((xo(lk,1)-xosp).lt.0)x3t=pi-x3t
      x3tp(lk)=pi-x3t
      xin(lk)=xo(lk,1)
c      write(6,*)' lk=',lk,' xin(lk)=',xin(lk),' x3tp(lk)=',x3tp(lk)
 410  format(3i5)
 411  format(7e12.6)
      go to 100
c
c     to give values on  the lines of symmetry and the nose
c
 1000 nt=lk
```

```
      np=ksorce

      if(im.lt.nt+1)then
      write(6,*)' change parameter im to ', nt+1
      write(6,*)' parameter im are given in subroutines fcn, input,
     &invdat, staglo, and main program scmain.'
      stop
      endif
      if(jm.ne.np+2)then
      write(6,*)' change parameter jm to ', np+2
      write(6,*)' parameter jm are given in subroutines fcn, input,
     &invdat, staglo, and main program scmain.'
      stop
      endif
      iptb(1)=-1
      do 2000 lk=1,nt
      xins=xin(lk)
      call dudy(phit(np-1),phit(np),pi,cavt(lk,np-1)
     &,cavt(lk,np),cavl(lk))
      call dudy(phit(2),phit(1),0.,cavt(lk,2),cavt(lk,1),cavw(lk))
      call dudy(phit(np-1),phit(np),pi,pcoef(lk,np-1)
     &,pcoef(lk,np),cpl(lk))
      call dudy(phit(2),phit(1),0.,pcoef(lk,2),pcoef(lk,1),cpw(lk))
      call dudy(phit(np-1),phit(np),pi,urt(lk,np-1)
     &,urt(lk,np),urtl(lk))
      call dudy(phit(2),phit(1),0.,urt(lk,2),urt(lk,1),urtw(lk))
      call dudy(phit(np-1),phit(np),pi,rt(lk,np-1)
     &,rt(lk,np),rtl(lk))
      call dudy(phit(2),phit(1),0.,rt(lk,2),rt(lk,1),rtw(lk))
      call dudy(phit(np-1),phit(np),pi,u3t(lk,np-1)
     &,u3t(lk,np),u3tl(lk))
      call dudy(phit(2),phit(1),0.,u3t(lk,2),u3t(lk,1),u3tw(lk))
 2000 continue

      do 2100 lk=1,nt
      do 2200 k=ksorce,1,-1
      cavt(lk,k+1)=cavt(lk,k)
      pcoef(lk,k+1)=pcoef(lk,k)
      urt(lk,k+1)=urt(lk,k)
      rt(lk,k+1)=rt(lk,k)
      u3t(lk,k+1)=u3t(lk,k)
      upt(lk,k+1)=upt(lk,k)
 2200 continue
      cavt(lk,1)=cavw(lk)
      cavt(lk,np+2)=cavl(lk)
      pcoef(lk,1)=cpw(lk)
      pcoef(lk,np+2)=cpl(lk)
      urt(lk,1)=urtw(lk)
      urt(lk,np+2)=urtl(lk)
      rt(lk,1)=rtw(lk)
      rt(lk,np+2)=rtl(lk)
      u3t(lk,1)=u3tw(lk)
      u3t(lk,np+2)=u3tl(lk)
      upt(lk,1)=0
      upt(lk,np+2)=0
 2100 continue

      do 2300 k=ksorce,1,-1
      phit(k+1)=phit(k)
 2300 continue
```

```
      phit(1)=0.
      phit(np+2)=pi

      if(kpoint.eq.1)go to 3000
c
c     calculate the velocity at the nose point for the blunted nose body
c
      call lagext(xin(1),xin(2),xin(3),cavt(1,np+2),cavt(2,np+2)
     &,cavt(3,np+2),cavn)

      call lagext(xin(1),xin(2),xin(3),pcoef(1,np+2),pcoef(2,np+2)
     &,pcoef(3,np+2),cpn)

      do 2401 k=1,ksorce+2
      u3tn(k)=cavn
      if(phit(k).gt.0)u3tn(k)=-abs(cavn)
 2401 continue

      do 2500 k=1,ksorce+2
      do 2600 lk=nt,1,-1
      rt(lk+1,k)=rt(lk,k)
      cavt(lk+1,k)=cavt(lk,k)
      pcoef(lk+1,k)=pcoef(lk,k)
      urt(lk+1,k)=urt(lk,k)
      u3t(lk+1,k)=u3t(lk,k)
      upt(lk+1,k)=upt(lk,k)
 2600 continue
      rt(1,k)=xosp
      cavt(1,k)=cavn
      pcoef(1,k)=cpn
      urt(1,k)=0
      u3t(1,k)=u3tn(k)
      upt(1,k)=0
 2500 continue
      go to 3100

c
c     calculate the velocity at the nose point for the sharp nose body
c
 3000 x1=xin(1)
      x2=xin(2)
      x3=xin(3)

      do 2440 k=1,ksorce+2
      call lagext(xin(1),xin(2),xin(3),cavt(1,k),cavt(2,k)
     &,cavt(3,k),cavnd(k))

      call lagext(xin(1),xin(2),xin(3),pcoef(1,k),pcoef(2,k)
     &,pcoef(3,k),cpnd(k))

      call lagext(xin(1),xin(2),xin(3),urt(1,k),urt(2,k)
     &,urt(3,k),urnd(k))

      call lagext(xin(1),xin(2),xin(3),u3t(1,k),u3t(2,k)
     &,u3t(3,k),u3tn(k))

      call lagext(xin(1),xin(2),xin(3),upt(1,k),upt(2,k)
     &,upt(3,k),upnd(k))

 2440 continue
```

```
      do 2510 k=1,ksorce+2
      do 2610 lk=nt,1,-1
      rt(lk+1,k)=rt(lk,k)
      cavt(lk+1,k)=cavt(lk,k)
      pcoef(lk+1,k)=pcoef(lk,k)
      urt(lk+1,k)=urt(lk,k)
      u3t(lk+1,k)=u3t(lk,k)
      upt(lk+1,k)=upt(lk,k)
 2610 continue
      rt(1,k)=xosp
      cavt(1,k)=cavnd(k)
      pcoef(1,k)=cpnd(k)
      urt(1,k)=urnd(k)
      u3t(1,k)=u3tn(k)
      upt(1,k)=upnd(k)
 2510 continue

 3100 continue
      do 2700 lk=nt,1,-1
      xin(lk+1)=xin(lk)
      x3tp(lk+1)=x3tp(lk)
      rtw(lk+1)=rtw(lk)
      u3tw(lk+1)=u3tw(lk)
      urtw(lk+1)=urtw(lk)
 2700 continue
      xin(1)=0
      x3tp(1)=0
      rtw(1)=xosp
      u3tw(1)=cavn
      urtw(1)=0.

      npi=np+2
      nti=nt+1
      return
      end
```

```
c######################################################################

      function krunge(y,f,x,h,n,mr)

c######################################################################
      dimension phi(6),savey(6),y1(6),y2(6),ykp(6),fkp(6),y(n),f(n)
      data m,loop,reb/0,0,5.e-4/
      m=m+1
      go to (5,45,65,85),m
 5    if(loop.gt.0)go to 25
      if(mr.eq.1)go to 205
      xo=x
      do 15 j=1,n
      ykp(j)=y(j)
 15   fkp(j)=f(j)
 25   do 35 j=1,n
      savey(j)=y(j)
      phi(j)=f(j)
 35   y(j)=savey(j)+0.5*h*f(j)
      x=x+0.5*h
      krunge=1
      return
 45   do 55 j=1,n
      phi(j)=phi(j)+2.0*f(j)
 55   y(j)=savey(j)+0.5*h*f(j)
      krunge=1
      return
 65   do 75 j=1,n
      phi(j)=phi(j)+2.0*f(j)
 75   y(j)=savey(j)+h*f(j)
      x=x+0.5*h
      krunge=1
      return
 85   do 95 j=1,n
 95   y(j)=savey(j)+(phi(j)+f(j))*h/6.0
      if(mr.eq.1)go to 165

      if(mr.eq.2)then
       krunge=0
       loop=0
       m=0
       return
       endif

       if(loop-1)105,125,145
105    do 115 j=1,n
      y2(j)=y(j)
      f(j)=fkp(j)
115    y(j)=ykp(j)
      x=xo
      h=h/2.
      m=1
      loop=1
      go to 25
125    do 135 j=1,n
135    y1(j)=y(j)
      xh=x
       loop=2
      m=0
      krunge=1
```

```
      return
145   if(mr.eq.3)go to 165
151   do 155 j=1,n
      if(abs(y(j)).lt.1.d-5)go to 155
      er=(y(j)-y2(j))/y(j)
      if(abs(er)-reb)155,155,175
155   continue
165    h=4.*h
      if(mr.eq.3)go to 170
      mr=0
170   loop=0
      krunge=0
      m=0
      return
175   do 185 j=1,n
       y(j)=ykp(j)
       f(j)=fkp(j)
185   y2(j)=y1(j)
      x=xo
      h=h/2.
      loop=1
      m=1
      krunge=1
      go to 25
195   krunge=2
      m=0
      loop=0
      return
205   do 215 j=1,n
      y(j)=ykp(j)
215   f(j)=fkp(j)
      x=xo
      h=h/2.
      go to 25
      end
```

```
c######################################################################

       subroutine lagext(x1,x2,x3,y1,y2,y3,c1)

c######################################################################

      a1=((y1-y2)*(x1-x3)-(y1-y3)*(x1-x2))
     &/((x1**2-x2**2)*(x1-x3)-(x1**2-x3**2)*(x1-x2))
      b1=(y1-y2-a1*(x1**2-x2**2))/(x1-x2)
      c1=y1-a1*x1**2-b1*x1
      return
      end
```

```
c#######################################################################
      subroutine staglo
c#######################################################################
c**   subroutine to locate stagnation point from inviscid velocity    **
c**   components                                                       **
c**                                                                    **
      parameter(im=16,jm=38)
      common/invtab/phit(jm),rt(im,jm),urt(im,jm)
     1,upt(im,jm),u3t(im,jm),cavt(im,jm),pcoef(im,jm)
     2,x3tp(im),xin(im)
      common/invcon/npi,nti,xosp
      common/stgpt/thstag,xps,zps
      common/com1/pi,pio2,dtr,rtd
      common/com2/ir,iw
      common/com3/iordr(2),iptb(2),ider
      common/wind/u3tw(im),urtw(im),rtw(im),cavw(im),cpw(im)
c
      iorder=2
      ipt=-1
      call iuni(nti,nti,x3tp,1,u3tw,iorder,0,vt,ipt,ierr)
      test1=vt
      epsi=0.000001
      the=0.
      if(abs(test1).lt.epsi)go to 200
      if(test1.lt.0.)write(iw,1000)
 1000 format(////5x,63h***** inconsistent velocities in stagnation regio
     1n - stop *****)
      if(test1.lt.0.)stop
      dthe=pi/10.
      n=0
  100 continue
      n=n+1
      if(n.gt.100)write(iw,1020)n
 1020 format(////5x,50h***** too many iterations - stop 1 in staglo ****
     1*, 2hn=,i4)
      call iuni(nti,nti,x3tp,1,u3tw,iorder,the,uthe,ipt,ierr)
      if(ierr.ne.0)write(iw,1030)n,etaw,the,uthe,ierr
 1030 format(/5x,i3,3e14.5,i3)
      if(abs(uthe).lt.epsi)go to 140
      if(uthe.lt.0.)go to 120
      the=the+dthe
      go to 100
  120 continue
      the=the-dthe
      dthe=dthe/10.
      go to 100
  140 continue
      thstag=pi-the
      go to 250
  200 continue
      thstag=pi
  250 continue
      call iuni(nti,nti,x3tp,1,rtw,iorder,the,rstag,ipt,ierr)
      call iuni(nti,nti,x3tp,1,urtw,iorder,the,urstag,ipt,ierr)
      xps=-rstag*cos(the)+xosp
      zps=-rstag*sin(the)
      return
      end
```

References

[1] Hess, J. L., "Calculation of Potential Flow About Arbitrary 3-D Lifting Bodies," Rep. No. MDC J5679-01 (Contract N00019-C-71-0524), Douglas Aircraft Co., Oct. 1972. (Available from DTIC as AD 755 480.)

[2] Vachris, A. F., Jr., and Yaeger, L. S., "QUICK-GEOMETRY - A Rapid Responce Method for Mathematically Modeling Configuration Geometry." Applications of Computer Graphics in Engineering, NASA SP-390, Oct. 1975, pp. 49-73.

[3] Hamilton, H. H., DeJarnette, F. R., and Weilmuenster, K. J., "Application of Axisymmetric Analogue for Calculating Heating in Three-Dimensional Flows," AIAA Paper No. 85-0245, 1987.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-4292, Volume II		

4. Title and Subtitle	5. Report Date
A Three-Dimensional, Compressible, Laminar Boundary-Layer Method for General Fuselages Volume II - User's Manual	May 1990
	6. Performing Organization Code

7. Author(s)

Yong-Sun Wie

8. Performing Organization Report No.

10. Work Unit No.

505-60-31

9. Performing Organization Name and Address

High Technology Corporation
28 Research Drive
Hampton, VA 23665

11. Contract or Grant No.

NAS1-18240

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

13. Type of Report and Period Covered

Contractor Report

14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Julius E. Harris
Final Report - Task No. 10
Volume I - Numerical Method

16. Abstract

This user's manual contains a complete description of the computer programs developed to calculate three-dimensional, compressible, laminar boundary-layers for perfect gas flow on general fuselage shapes. These programs include the 3-D boundary-layer program (3DBLC), the body-oriented coordinate program (BCC), and the streamline coordinate program (SCC). In the present volume, the descriptions of these computer programs including subroutine description, input, output, and a sample case are presented. The complete FORTRAN listings of the computer programs are also included. The numerical method is described in Volume I.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Boundary-Layer Compressible Fuselage Three-Dimensional Laminar	[redacted] REVIEW for general release May 31, 1992 Subject Category 34

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	168	

NASA FORM 1626 OCT 86

NASA-Langley, 1990