NASA Contractor Report 4269

Computation of Three-Dimensional Compressible Boundary Layers to Fourth-Order Accuracy on Wings and Fuselages

Venkit Iyer

CONTRACT NAS1-18585
JANUARY 1990

Review for general release January 31, 1994

NASA

NASA Contractor Report 4269

Computation of Three-Dimensional Compressible Boundary Layers to Fourth-Order Accuracy on Wings and Fuselages

Venkit Iyer
Vigyan, Inc.
Hampton, Virginia

Prepared for
Langley Research Center
under Contract NAS1-18585

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

1990

Table of Contents

1. INTRODUCTION

Although Navier-Stokes solutions of aerodynamic bodies have become fairly routine, their use is somewhat restrictive when highly accurate solutions within the boundary layer are being sought. The grid point distributions used here have to be relatively coarse due to limitations in computer memory and speed. In addition, the necessity of having to add enthalpy damping and other smoothing procedures further reduce the accuracy of the solution. In contrast, the accuracy requirement of a three-dimensional stability analysis indicates a finer grid point distribution and an algorithm that is reasonably fast and efficient. Solution procedures for the three-dimensional boundary layer equations are important in this context.

Numerical methods to solve the two-dimensional boundary layer flow have been in wide use for design and analysis for the last 30 years. However, a corresponding development of three-dimensional boundary layer analysis methods has not occurred on a large scale. The few 3D boundary layer programs in existence are either proprietary or not used widely due to their restrictive application. With the current emphasis on laminar flow control for wings and fuselages, there is a need for an accurate, efficient and well-documented three-dimensional boundary layer procedure. The present effort is an attempt at fulfilling this need.

The theory and numerical procedures used in 2D flow cannot readily be extended to 3D flow due to the mathematical character of the governing equations and their initial, boundary and geometric requirements. The hyperbolic character of the 3D boundary layer equations implies that there are regions of influence and dependence that must be correctly treated when solving on each stream surface plane. The initial data plane to start the calculation is obtained by solving a special subset of the 3D boundary layer equations applicable at the stagnation point or attachment line, from where the boundary layer originates. In addition, a side boundary such as a symmetry plane or a plane of zero spanwise variation has to be provided. In the boundary layer normal direction, the solution is obtained implicitly similar to the 2D boundary layer solution.

The 3D boundary layer equations are computationally much less expensive to solve than the Navier-Stokes equations due to a number of reasons. One reason is that the first-order boundary layer equations employ a number of simplifying assumptions, the chief of them being that the pressure is constant normal to the boundary surface. Another reason is that the boundary

layer grid has less storage requirements and is easier to generate. The boundary layer grid is essentially an algebraically generated non-orthogonal surface grid. The grid in the normal direction ζ is orthogonal to the (ξ, η) surface grid. The grid storage requirements are thus limited to approximately 6.im.jm compared to 13.im.jm.km in a 3D Navier-Stokes computation. The boundary layer computation however, also involves calculation of the inviscid flow which requires only a coarse grid in the surface region.

The 3D boundary layer equations proportionally involve more effort in formulation. The equations are usually transformed to new flow variables and coordinates. These transformations are necessary to reduce the boundary layer growth in the computational plane. For laminar flow, the transformation provides a grid of nearly constant normal distance at which the inviscid conditions are applied. Transformations are also necessary to remove singularities in the boundary layer equations present at the stagnation location.

When used in a design procedure, repetitive runs of a Navier-Stokes code will be very expensive. In the boundary layer approach, once the inviscid results are generated, repetitive runs of the boundary layer solver can be made rather inexpensively. For example, if one is interested in optimizing wall suction rates, it can be done interactively using a boundary layer code. Since the boundary layer solution uses a marching method, calculations can be confined to zones of interest as in, for example, the study of boundary layer stability near a wing leading edge or a fuselage forebody.

The present boundary layer solution method employs a compact differencing scheme in the wall-normal direction to achieve fourth-order accuracy. Solution profiles are fourth-order accurate in the ζ direction since the two components of velocity and their normal gradients are solved for simultaneously and implicitly. The solution in the stream surface directions ξ and η is accurate to the second-order. The algorithm used is such that wall and edge boundary conditions are easy to apply and do not compromise the fourth-order accuracy.

The present report also addresses the issue of accurately providing the interpolation interface between the inviscid results and specification of the boundary layer edge conditions. In this interface program, attention is focussed on correctly interpolating quantities from the inviscid grid to the boundary layer grid. Errors in interpolation or in the calculation of the metric terms will degrade the accuracy of the solution and could be very detrimental when applied to boundary layer stability analysis. The interface provides

the means by which these sources of error can be carefully monitored and minimized.

The program in its present version is limited to attached laminar flows with perfect gas assumption. The formulation includes a provision to incorporate basic eddy viscosity and transition parameters. Turbulence modelling is however not included. The energy equation is non-dimensionalized using an arbitrary reference total enthalpy value which should facilitate the inclusion of real gas effects externally. The program incorporates a number of wall boundary conditions such as normal mass injection, wall heat transfer or temperature. The method is applicable to supersonic flow with leading edge shocks if conditions downstream of the shock are used as free-stream conditions. Alternately, if conditions upstream of the shock are used, the boundary layer edge temperature or density values need to be specified in addition to the pressure, by means of an inviscid solver. Variable entropy effects are not considered. Boundary layer interaction with the external inviscid flow is also not included in the present work.

The present program is applicable only to flow cases where the boundary layer assumptions are valid. Solving laminar attached flow on wings and fuselages at moderate angles at attack is the main objective of the program. With an appropriate turbulence model, the method is valid for turbulent flows as well. Separated flows and vortex-dominated flows in which viscous-inviscid interaction dominates, cannot be solved by the present program. Flows near wing-fuselage junction and wing tip are not of the boundary layer type and are excluded in the present calculation. Such regions must be solved by a subset of the Navier-Stokes equations of higher order than the boundary layer set.

This report includes an Appendix providing details of the computer programs. A number of test and validation cases have been presented and explained in the report. The organization of the different programs have been presented based on the UNIX† operating system. However, there are no system-dependent features in any of the source programs and they should be readily adaptable to any system equipped with a FORTRAN compiler. Option exists to use either the SI or the US system of units. Graphics programs are necessary to monitor the results at intermediate and final stages. The information provided in this report should be adequate to write graphics programs as required.

† UNIX is a trademark of AT&T Bell Laboratories

2. SYMBOLS AND ABBREVIATIONS

$\tilde{A}_{i,\ i=1,4}$	Coefficients in transformed continuity equation
$a_{i,\ i=1,3}$	ξ differencing coefficients
a_{ij}^{k}	Coefficients of discretized system
$B_{i,\ i=1,6}$	Coefficients in transformed ξ momentum equation
$b_{i,\ i=1,7}$	η differencing coefficients
b_{ij}^{k}	Coefficients of discretized system
C_f	Skin friction coefficient
$C_{i,\ i=1,6}$	Coefficients in transformed η momentum equation
$C_{ij,\ i=1,3\ ,j=1,8}$	Metric coefficient terms, see Table 1
C_p	Pressure coefficient
$C_{r,s}$	Stagnation point flow parameter, $\frac{h_1 g_e}{h_2 f_e}$
$c_{i,\ i=1,4}$	ζ differencing coefficients
$D_{i,\ i=1,5}$	Coefficients in transformed energy equation
F	Ratio, u / u_e
f	Velocity gradient, $\frac{\partial u}{\partial x}$
$\overline{f}$	Any arbitrary function
G	Ratio, v / v_r
g	Velocity gradient, $\frac{\partial v}{\partial y}$
$\overline{g}$	Metrics function, $\overline{g} = h_1^2 h_2^2 - g_{12}^2$, see Table 1
g_{12}	Metric coefficient
H	Total enthalpy nondimensionalized by $\frac{u_\infty^{*2}}{(\gamma-1)M_\infty^2}$
H_{ref}^{*}	Reference total enthalpy
h_1	Metric coefficient in x direction
h_2	Metric coefficient in y direction
I	Normal derivative of H
i	Index in ξ
im	Total number of grid points in ξ
j	Index in η
jm	Total number of grid points in η
k	Index in ζ
k_e	Stretching constant for ζ direction grid distribution
km	Total number of grid points in ζ

L	Normal (ζ) derivative of F
L^*	Reference length, dimensional
l	Ratio, $\frac{\rho\mu}{\rho_e\mu_e}$
l_1	Ratio, $l\,(1+\frac{\varepsilon_1}{\mu}\Gamma)$
l_2	Ratio, $l\,(1+\frac{\varepsilon_2}{\mu}\Gamma)$
l_3	Ratio, $\frac{l}{\sigma}\,(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma)$
M	Normal (ζ) derivative of G
M_∞	Free stream Mach number
p	Non-dimensional pressure
p_∞^*	Free stream static pressure
$\vec{Q}$	Solution vector
q	Total velocity, nondimensional
$\dot{q}$	Heat flux
$\dot{q}_w^*$	Wall heat flux, dimensional
r_i	RHS terms of discretized system
R^*	Gas constant
Re_{ref}	Reference Reynolds number based on L^*
Re_∞	Free-stream Reynolds number based on L^*
S	Reference temperature for viscosity calculation (eq. 18)
$\vec{S}_k$	Vector of unknowns at k (eq. 119)
s_1	Non-dimensional length along x, $\int_0^\xi h_1\,d\xi$
s_2	Non-dimensional length along y, $\int_0^\eta h_2\,d\eta$
T	Non-dimensional temperature
T_∞^*	Free stream static temperature
T_{ref}^*	Reference temperature, u_∞^{*2}/R^*
t_x^*	Boundary layer thickness in the x direction, dimensional
t_y^*	Boundary layer thickness in the y direction, dimensional
u	Streamwise (ξ) direction velocity
u'_i	Cartesian component of inviscid flow in x′ direction
u_∞^*	Free stream velocity
v	Crossflow (η) direction velocity
v'_i	Cartesian component of inviscid flow in y′ direction

w	Transformed normal velocity
w'_i	Cartesian component of inviscid flow in w′ direction
w^*_w	Wall injection velocity, dimensional
$\tilde{w}$	Nondimensional physical normal velocity multiplied by $\sqrt{Re_{ref}}$
x	Nondimensional streamwise coordinate
x'	Cartesian coordinate
y	Nondimensional crossflow direction coordinate
y'	Cartesian coordinate
z	Nondimensional normal coordinate
z'	Cartesian coordinate
$\tilde{z}$	z multiplied by $\sqrt{Re_{ref}}$

Greek Symbols

α	Angle of attack
α_c	Surface tangent angle
α_k	Block-tridiagonal element in sub-diagonal location
β	Angle between x and y coordinate lines
β_k	Block-tridiagonal element in diagonal location
γ	Ratio of specific heats
γ_k	Block-tridiagonal element in super-diagonal location
$\Delta\xi$, $\Delta\eta$, $\Delta\zeta$	Step sizes in the ξ, η, ζ directions
δ	Solution change at each iteration
δ^*	Displacement thickness, dimensional
ε	η differencing switch variable
ε_1,ε_2	Turbulent eddy viscosity in x,y directions
ε_H	Turbulent viscosity term in the energy equation
ε_F	Convergence limit for the momentum equations
ε_I	Convergence limit for the energy equation
ζ	Transformed normal coordinate
η	Transformed crossflow direction coordinate
θ	Density ratio, $\frac{\rho_e}{\rho}$
Λ	Sweep angle
μ	Nondimensional viscosity
μ^*_s	Viscosity constant (eq. 17)
μ^*_{ref}	Reference viscosity at T^*_{ref}
ξ	Transformed streamwise coordinate
ρ	Nondimensional density
ρ^*_∞	Free stream density

σ	Prandtl number, laminar
σ_t	Prandtl number, turbulent
τ	Shear stress
ϕ	ζ transformation parameter, $\sqrt{(\rho_e \mu_e s_1 u_e)}$
ψ	ζ transformation parameter, $\sqrt{\left\{\frac{u_e}{\rho_e \mu_e s_1}\right\}}$
ω	Blending constant for ξ differencing

Superscripts

*	Dimensional quantity
'	Derivative in the ζ direction

Subscripts

e	Boundary layer edge
i	Index in ξ direction
j	Index in η direction
k	Index in ζ direction
r	Reference value
ref	Reference value
t	Turbulent value
w	Wall quantity
x	x direction derivative
y	y direction derivative
,x	along x direction
,y	along y direction
1	x direction
2	y direction
∞	Free-stream quantity

Abbreviations

ATT BL	Attachment Line Boundary Layer
BL	Boundary Layer
CFL	Courant-Friedrichs-Lewy
INF BL	Infinite Swept Wing Boundary Layer
LHS	Left Hand Side
LINF	Locally Infinite Swept Wing
N-S	Navier-Stokes
RHS	Right Hand Side
S.E.	Surface Euler

S/R	Subroutine
SIM BL	Similarity Boundary Layer
STG BL	Stagnation Boundary Layer
SYM BL	Symmetry Line Boundary Layer
2D	Two-Dimensional
3D	Three-Dimensional
3D BL	Three-Dimensional Boundary Layer

3. THE 3D BOUNDARY LAYER EQUATIONS

We start with the three-dimensional compressible first-order boundary layer (3D BL) equations in a curvilinear non-orthogonal system. The Cartesian coordinate system is referred to by (x', y', z') and the curvilinear system by (x, y, z) (see Fig. 1). The coordinate lines x and y are on the body surface and, in general, need not be orthogonal. For example, the boundary layer grid on a wing may consist of x coordinate lines defined in the chordwise direction over the wing surface and the y coordinate lines defined in the span direction along constant percent chord lines over the wing surface. The third curvilinear coordinate z is chosen to be mutually orthogonal to both x and y. The superscript * refers to the fact that the body is defined in terms of physical dimensional quantities. At a later step, the length quantities will be nondimensionaiized by a reference length L^*.

The boundary layer grid is basically a surface grid. In the body-normal direction, it is assumed that the grid consists of planes parallel to the surface grid. This assumption yields a great simplification in that the 3D grid is completely definable in terms of a surface grid.

The 3D BL equations in physical units consist of the continuity equation, the momentum equations in the x^* and y^* directions and the energy equation. These have been presented by several authors (Ref. 1 and 2, for example) and are given below:

$$\frac{\partial}{\partial x^*}(C_{11}\rho^* u^*) + \frac{\partial}{\partial y^*}(C_{12}\rho^* v^*) + C_{13}\frac{\partial}{\partial z^*}(\rho^* w^*) = 0 \tag{1}$$

$$\frac{u^*}{h_1}\frac{\partial u^*}{\partial x^*} + \frac{v^*}{h_2}\frac{\partial u^*}{\partial y^*} + w^*\frac{\partial u^*}{\partial z^*} + C_{24}u^{*2} + C_{25}u^*v^* + C_{26}v^{*2} =$$

$$\frac{1}{\rho^*}\left[C_{27}\frac{\partial p^*}{\partial x^*} + C_{28}\frac{\partial p^*}{\partial y^*}\right] + \frac{1}{\rho^*}\frac{\partial}{\partial z^*}\left\{\mu^*\left(1+\frac{\varepsilon_1^*}{\mu^*}\Gamma\right)\frac{\partial u^*}{\partial z^*}\right\} \tag{2}$$

$$\frac{u^*}{h_1}\frac{\partial v^*}{\partial x^*} + \frac{v^*}{h_2}\frac{\partial v^*}{\partial y^*} + w^*\frac{\partial v^*}{\partial z^*} + C_{34}u^{*2} + C_{35}u^*v^* + C_{36}v^{*2} =$$

$$\frac{1}{\rho^*}\left[C_{37}\frac{\partial p^*}{\partial x^*} + C_{38}\frac{\partial p^*}{\partial y^*}\right] + \frac{1}{\rho^*}\frac{\partial}{\partial z^*}\left\{\mu^*\left(1+\frac{\varepsilon_2^*}{\mu^*}\Gamma\right)\frac{\partial v^*}{\partial z^*}\right\} \tag{3}$$

$$\frac{u^*}{h_1}\frac{\partial H^*}{\partial x^*} + \frac{v^*}{h_2}\frac{\partial H^*}{\partial y^*} + w^*\frac{\partial H^*}{\partial z^*} =$$

$$\frac{1}{\rho^*}\frac{\partial}{\partial z^*}\left\{\frac{\partial H^*}{\partial z^*} - \mu^*\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial z^*}\left(\frac{q^{*2}}{2}\right)\right\} \qquad (4)$$

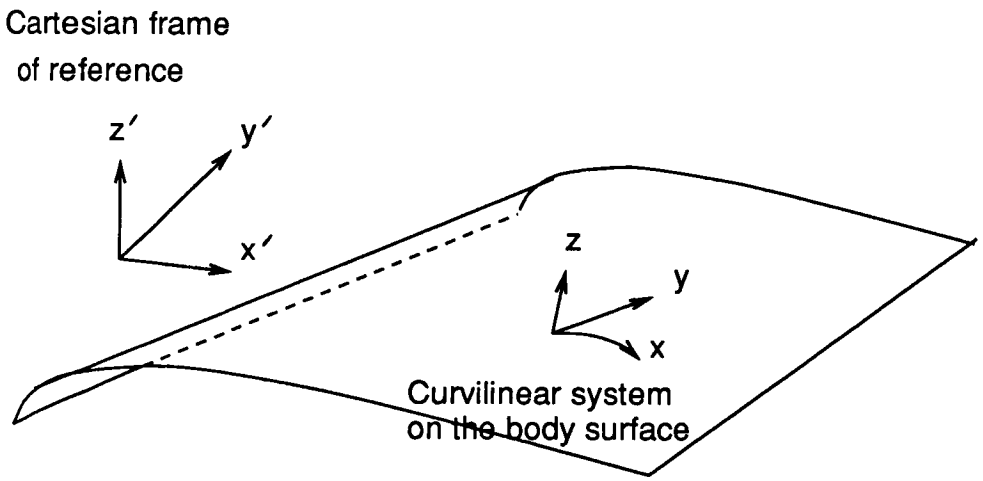


Fig. 1 3D BL Coordinate System Definition

The coefficients h_1, h_2 and C_{ij} relate to the metrics of the transformation from (x′, y′, z′) to (x, y, z). The symbol q^* refers to the absolute velocity. The quantities ε_1^*, ε_2^* and ε_H^* are explained in the next section.

Non-Dimensionalization of 3D BL Equations

The equations (1)-(4) are made non-dimensional by dividing the dimensional quantities by corresponding reference quantities. The reference quantities are as given below:

x^*, y^*, z^*	by a reference length L^*
u^*, v^*, w^*	by free-stream velocity $u^*_{ref} = u^*_\infty$
ρ^*	by free-stream density $\rho^*_{ref} = \rho^*_\infty$
p^*	by $\rho^*_\infty u^{*2}_\infty$
T^*	by $T^*_{ref} = u^{*2}_\infty / R^*$
H^*	by H^*_{ref}; choice of H^*_{ref} is arbitrary
μ^*	by μ^*_{ref}, where μ^*_{ref} is the absolute viscosity at T^*_{ref}

The reference Reynolds number is then defined as,

$$Re_{ref} = \rho^*_\infty u^*_\infty L^* / \mu^*_{ref} \tag{5}$$

Re_{ref} is different from the free-stream Reynolds number Re_∞,

$$Re_\infty = \rho^*_\infty u^*_\infty L^* / \mu^*_\infty = Re_{ref} / \mu_\infty \tag{6}$$

The non-dimensionalization results in a Re_{ref} factor in the viscous terms. A new stretched coordinate $\tilde{z}$ and a new normal velocity $\tilde{w}$ are defined to absorb this factor.

$$\tilde{z} = z\sqrt{Re_{ref}} \;\; ; \;\; \tilde{w} = w\sqrt{Re_{ref}} \tag{7}$$

The resulting non-dimensionalized equations are given below (the superscript * has been dropped to denote non-dimensional quantities):

$$\frac{\partial}{\partial x}(C_{11}\rho u) + \frac{\partial}{\partial y}(C_{12}\rho v) + C_{13}\frac{\partial}{\partial \tilde{z}}(\rho \tilde{w}) = 0 \tag{8}$$

$$\frac{u}{h_1}\frac{\partial u}{\partial x} + \frac{v}{h_2}\frac{\partial u}{\partial y} + \tilde{w}\frac{\partial u}{\partial \tilde{z}} + C_{24}u^2 + C_{25}uv + C_{26}v^2 =$$

$$\frac{1}{\rho}\left[C_{27}\frac{\partial p}{\partial x} + C_{28}\frac{\partial p}{\partial y}\right] + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_1}{\mu}\Gamma\right)\frac{\partial u}{\partial \tilde{z}}\right\} \tag{9}$$

$$\frac{u}{h_1}\frac{\partial v}{\partial x} + \frac{v}{h_2}\frac{\partial v}{\partial y} + \tilde{w}\frac{\partial v}{\partial \tilde{z}} + C_{34}u^2 + C_{35}uv + C_{36}v^2 =$$

$$\frac{1}{\rho}\left[C_{37}\frac{\partial p}{\partial x} + C_{38}\frac{\partial p}{\partial y}\right] + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_2}{\mu}\Gamma\right)\frac{\partial v}{\partial \tilde{z}}\right\} \tag{10}$$

$$\frac{u}{h_1}\frac{\partial H}{\partial x} + \frac{v}{h_2}\frac{\partial H}{\partial y} + \tilde{w}\frac{\partial H}{\partial \tilde{z}} =$$

$$\frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\frac{\mu}{\sigma}\left(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma\right)\frac{\partial H}{\partial \tilde{z}} - \frac{u_\infty^{*2}}{H_{ref}^*}\mu\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial \tilde{z}}\left(\frac{q^2}{2}\right)\right\} \qquad (11)$$

The equation of state, $p^* = \rho^* R^* T^*$ reduces to the non-dimensional form,

$$p = \rho T \qquad (12)$$

The equation for total enthalpy, $H^* = \frac{R^*\gamma}{\gamma-1}T^* + \frac{1}{2}q^{*2}$ reduces to,

$$H = \frac{u_\infty^{*2}}{H_{ref}^*}\left[\frac{\gamma T}{\gamma-1} + \frac{1}{2}q^2\right] \qquad (13)$$

The choice of H_{ref}^* used here is,

$$H_{ref}^* = \frac{u_\infty^{*2}}{(\gamma-1)M_\infty^2} = \frac{a_\infty^{*2}}{\gamma-1} \qquad (14)$$

With this choice of H_{ref}^*, the total enthalpy at free-stream conditions H_∞ can be obtained by substituting free-stream values in Eq. (13). Also, since total enthalpy at the boundary layer edge is assumed to be a constant, $H_e = H_\infty$. Hence,

$$H_\infty = H_e = 1 + \frac{\gamma-1}{2}M_\infty^2 \qquad (15)$$

The non-dimensional temperature at any point in the boundary layer edge can be obtained as,

$$T_e = \frac{H_e}{\gamma M_\infty^2} - \frac{\gamma-1}{2\gamma}q^2 \qquad (16)$$

It can be seen that as $M_\infty \to 0$, H_∞ and $H_e \to 1$. However, $T_{ref}^* \to 0$ as well, resulting in large values of $T = T^*/T_{ref}^*$. Since in the present formulation, H is used as the solution variable in the energy equation, there are no numerical inaccuracies even when solving extremely low Mach number flows.

The Sutherland law of viscosity for air (γ=1.4) is given by,

$$\mu^* = \frac{\mu_s^* T^{*1.5}}{T^*+S^*} \tag{17}$$

$$\mu_s^* = 2.27\text{x}10^{-8}\ \frac{\text{lb sec}}{\text{ft}^2\ {}^\circ\text{R}^{1/2}} = 1.458\text{x}10^{-6}\ \frac{\text{Pa sec}}{\text{m}^2\ {}^\circ\text{K}^{1/2}}$$

$$S^* = 198.6\ {}^\circ\text{R} = 110.33\ {}^\circ\text{K}$$

In non-dimensional form, the viscosity law becomes,

$$\mu = T\,\frac{1+S}{T+S}\ ;\ S = \frac{S^*}{T^*_{ref}} \tag{18}$$

Viscous Terms

The present analysis is for laminar flow only. However, in order to provide for basic turbulent flow closure, the equations include an eddy viscosity and a transition parameter. The laminar term in the x momentum equation $\frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\mu\frac{\partial u}{\partial \check{z}}\right\}$ is modified by a turbulent viscosity term ε_1 and a transition parameter Γ such that the viscous term is represented by, $\frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\mu(1+\frac{\varepsilon_1}{\mu}\Gamma)\frac{\partial u}{\partial \check{z}}\right\}$. Modelling of ε_1 and Γ is not dealt with here, but is however, provided for in the formulation. Similarly, the viscous term in the y momentum equation includes an ε_2 as in $\frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\mu(1+\frac{\varepsilon_2}{\mu}\Gamma)\frac{\partial v}{\partial \check{z}}\right\}$. The energy equation is modified to include the turbulent Prandtl number σ_t and a turbulent viscosity term ε_H resulting in the term, $\frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\frac{\mu}{\sigma}(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma)\frac{\partial H}{\partial \check{z}}\right\}$. The transitional parameter Γ and the eddy viscosity terms ε_1 , ε_2 and ε_H are set to zero for laminar flow.

Metric Coefficients

The components of the covariant metric tensor associated with the transformation

$$x^* = x^*\,(\,x'^*, y'^*, z'^*\,)\ ,\ y^* = y^*\,(\,x'^*, y'^*, z'^*\,)\ ,\ z^* = z^*\,(\,x'^*, y'^*, z'^*\,) \tag{19}$$

are given by the following expressions (the superscript * can be dropped since the non-dimensionalization by L^* does not affect the transformation).

$$g_{11} = \left[\frac{\partial x'}{\partial x}\right]^2 + \left[\frac{\partial y'}{\partial x}\right]^2 + \left[\frac{\partial z'}{\partial x}\right]^2$$

$$g_{21} = g_{12} = \frac{\partial x'}{\partial x}\cdot\frac{\partial x'}{\partial y} + \frac{\partial y'}{\partial x}\cdot\frac{\partial y'}{\partial y} + \frac{\partial z'}{\partial x}\cdot\frac{\partial z'}{\partial y}$$

$$g_{22} = \left[\frac{\partial x'}{\partial y}\right]^2 + \left[\frac{\partial y'}{\partial y}\right]^2 + \left[\frac{\partial z'}{\partial y}\right]^2$$

$$g_{33} = 1 \quad , \quad g_{23} = g_{32} = g_{13} = g_{31} = 0$$

The metric scale coefficients h_1, h_2 and h_3 are defined as,

$$h_1 = \sqrt{g_{11}} = \sqrt{x_x'^2 + y_x'^2 + z_x'^2} \tag{20}$$

$$h_2 = \sqrt{g_{22}} = \sqrt{x_y'^2 + y_y'^2 + z_y'^2} \tag{21}$$

$$h_3 = 1 \tag{22}$$

The component g_{12} is a measure of the angle β between the two surface coordinate lines x and y.

$$g_{12} = h_1 h_2 \cos(\beta) \tag{23}$$

The coefficients h_1 and h_2 are measures of the arc lengths in the x and y direction.

$$ds_1 = \text{Incremental surface distance in x direction} = h_1 dx \tag{24}$$

$$ds_2 = \text{Incremental surface distance in y direction} = h_2 dy \tag{25}$$

The non-dimensional absolute velocity q at any location is given by,

$$q^2 = u^2 + v^2 + 2uv\cos(\beta) = u^2 + v^2 + 2uv\frac{g_{12}}{h_1 h_2} \tag{26}$$

Although x and y are defined on the surface, it is not necessary that they be measured along the surface (see Ref. 2, page 10). For example, in the case of a boundary layer grid on a conically shaped body, x can be measured along the axis. The metric coefficient h_1 will be dependent on the way x is measured. If x is measured exactly along x, then $ds_1 = dx$ and h_1 will be equal to unity.

The coefficients C_{ij} in the equations (8)-(10) are functions of the metric scale coefficients h_1, h_2, g_{12} and their derivatives in x and y directions. The expressions for these coefficients are given in Table 1.

Table 1. Coefficients Based on the Metrics h_1, h_2 and g_{12}

$$C_{11} = \frac{\sqrt{g}}{h_1} \quad ; \quad g = h_1^2 h_2^2 - g_{12}^2$$

$$C_{12} = \frac{\sqrt{g}}{h_2}$$

$$C_{13} = \sqrt{g}$$

$$C_{24} = \frac{g_{12}}{g}\left\{\frac{g_{12}}{h_1^2}\frac{\partial h_1}{\partial x} + \frac{\partial h_1}{\partial y} - \frac{1}{h_1}\frac{\partial g_{12}}{\partial x}\right\}$$

$$C_{25} = \frac{1}{g}\left\{h_1 h_2\left[1+\left[\frac{g_{12}}{h_1 h_2}\right]^2\right]\frac{\partial h_1}{\partial y} - 2g_{12}\frac{\partial h_2}{\partial x}\right\}$$

$$C_{26} = \frac{h_1}{g}\left\{\frac{\partial g_{12}}{\partial y} - h_2\frac{\partial h_2}{\partial x} - \frac{g_{12}}{h_2}\frac{\partial h_2}{\partial y}\right\}$$

$$C_{27} = -\frac{h_1 h_2^2}{g}$$

$$C_{28} = \frac{h_1 g_{12}}{g}$$

$$C_{34} = \frac{h_2}{g}\left\{\frac{\partial g_{12}}{\partial x} - h_1\frac{\partial h_1}{\partial y} - \frac{g_{12}}{h_1}\frac{\partial h_1}{\partial x}\right\}$$

$$C_{35} = \frac{1}{g}\left\{h_1 h_2\left[1+\left[\frac{g_{12}}{h_1 h_2}\right]^2\right]\frac{\partial h_2}{\partial x} - 2g_{12}\frac{\partial h_1}{\partial y}\right\}$$

$$C_{36} = \frac{g_{12}}{g}\left\{\frac{g_{12}}{h_2^2}\frac{\partial h_2}{\partial y} + \frac{\partial h_2}{\partial x} - \frac{1}{h_2}\frac{\partial g_{12}}{\partial y}\right\}$$

$$C_{37} = \frac{h_2 g_{12}}{g}$$

$$C_{38} = -\frac{h_1^2 h_2}{g}$$

Pressure Gradient Terms

Consistent with the boundary layer assumptions, the pressure remains constant in the boundary layer normal direction. Consequently, the pressure distribution calculated from the inviscid flow can be used to calculate the boundary layer edge velocities $u_e(x,y)$ and $v_e(x,y)$.

When the 3D BL equations are taken to the limit $\tilde{z} \to \infty$, $\mu \to 0$, the resulting equation set is called the surface Euler equations (see also Ref. 3). These equations characterize the edge streamlines resulting from a pressure distribution imposed on a body in the absence of viscous forces and normal pressure gradients. The energy equation reduces to $H_\infty = H_e = 1 + \frac{\gamma-1}{2}M_\infty^2$ and the momentum equations become:

$$\frac{u_e}{h_1}\frac{\partial u_e}{\partial x} + \frac{v_e}{h_2}\frac{\partial u_e}{\partial y} + C_{24}u_e^2 + C_{25}u_e v_e + C_{26}v_e^2 = \frac{1}{\rho_e}\left[C_{27}\frac{\partial p}{\partial x} + C_{28}\frac{\partial p}{\partial y}\right] \qquad (27)$$

$$\frac{u_e}{h_1}\frac{\partial v_e}{\partial x} + \frac{v_e}{h_2}\frac{\partial v_e}{\partial y} + C_{34}u_e^2 + C_{35}u_e v_e + C_{36}v_e^2 = \frac{1}{\rho_e}\left[C_{37}\frac{\partial p}{\partial x} + C_{38}\frac{\partial p}{\partial y}\right] \qquad (28)$$

Given $p(x,y)$ and initial values of u_e and v_e, the above equations can be solved for the boundary layer edge conditions of u_e and v_e at all downstream points.

Boundary Conditions

The 3D BL equations are parabolic in the normal direction $\tilde{z}$ and hyperbolic in the stream surface direction (x,y). This means that information travels instantaneously in the $\tilde{z}$ direction and propagates in the local streamline direction along planes parallel to the surface (see Wang[4]). A marching method in the x direction is the appropriate choice to solve this class of equations, with implicit solution of quantities in the normal direction. One can also obtain the solution in the crossflow direction y by marching. Special procedures must be incorporated if the crossflow direction changes sign. Consequently, an initial plane of data has to be specified as initial conditions to start the marching method. This can be obtained from a special subset of the general equations valid at the stagnation point or attachment line from where the boundary layer originates. Similarly, a plane of data corresponding to one side boundary has to be specified to start the solution march in the y direction. This plane of data is also obtained from a subset of the 3D

BL equations by making some simplifying assumptions.

The boundary conditions in the $\tilde{z}$ direction are as follows:

$\tilde{z} = 0$ (wall) $\quad u=0;\ v=0;\ \tilde{w} = \tilde{w}_w$ (suction or blowing velocity)

$$\frac{\partial H}{\partial \tilde{z}} = 0 \quad \text{for adiabatic wall or,}$$

$$H = H_w = \gamma M_\infty^2 T_w \quad \text{wall temperature specified or,}$$

$$\left.\frac{\partial T}{\partial \tilde{z}}\right|_w \quad \text{heat flux at wall specified.}$$

$\tilde{z} = \tilde{z}_e$ (BL edge) $\quad u = u_e,\ v = v_e$

$$H = H_e = 1 + \frac{\gamma - 1}{2} M_\infty^2$$

The total enthalpy remains a constant at the boundary layer edge. The edge velocities required as boundary condition are obtained by solving the surface Euler equations. Alternatively, they can also be obtained by interpolating the inviscid velocities to the boundary layer grid. In simpler cases, the inviscid velocities can be obtained by closed form solutions.

4. TRANSFORMATION OF THE 3D BL EQUATIONS

The boundary layer coordinate x is measured from the stagnation point or the attachment line. At x=0, the x momentum equation becomes singular for attachment line flow. If the body has a 3D stagnation point, the y momentum equation also becomes singular. Therefore, transformations are needed to remove these singularities, so that in transformed variables, the solution becomes finite and regular in all regions. In addition, in terms of physical variables, the boundary layer thickness grows as a function of the edge conditions. The transformation can be devised so that this growth is minimized, resulting in a nearly constant normal coordinate distribution. Obviously, this growth is a function of the pressure gradient and the type of flow (laminar or turbulent). We restrict ourselves to a transformation suitable for laminar flow.

Here we choose a Levy-Lees type of similarity transformation somewhat akin to the one employed by Vatsa and Davis[5]. Introduction of this transformation gives the similarity form of the equations at the stagnation point or attachment line.

The transformation is given by,

$$\xi = x \;\; ; \;\; \eta = y \tag{29}$$

$$\zeta = \psi \int_0^z \rho \, d\bar{z} = \frac{\phi}{\rho_e \mu_e s_1} \int_0^z \rho \, d\bar{z} \tag{30}$$

$$\phi = \sqrt{\rho_e \mu_e s_1 u_e} \quad ; \quad \psi = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}} = \frac{u_e}{\phi} \tag{31}$$

$$s_1 = \int_0^\xi h_1 \, d\xi \tag{32}$$

Note that although $\xi = x$, $\frac{\partial}{\partial x}$ is actually $\frac{\partial}{\partial x}\Big|_{y,z}$ (holding y and $\bar{z}$ constant) whereas, $\frac{\partial}{\partial \xi}$ is $\frac{\partial}{\partial \xi}\Big|_{\eta,\zeta}$ (holding η and ζ constant). Similarly, $\frac{\partial}{\partial y}\Big|_{x,z}$ and $\frac{\partial}{\partial \eta}\Big|_{\xi,\zeta}$ are different.

The subscript e refers to conditions at the BL edge. The quantity s_1 corresponds to the non-dimensional distance on the body in the x direction. It can be verified that, in the special case of zero pressure gradient flow, $u_e = \rho_e = 1$, ζ reduces to the similarity variable $\sqrt{\frac{Re_\infty}{s_1}} \int_0^z \rho \, dz$. The

transformation procedure involves the use of equations (29)-(32) in recasting the equations (8)-(11) in terms of ξ, η and ζ.

Transformation of the Continuity Equation

The continuity equation (8) is first integrated to give,

$$C_{13}\rho\bar{w} = -\frac{\partial}{\partial x}\int_0^z (C_{11}\rho u)d\bar{z} - \frac{\partial}{\partial y}\int_0^z (C_{12}\rho v)d\bar{z}$$

Transformation using $\frac{\partial}{\partial x} = \frac{\partial}{\partial \xi}+\frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta}$ and $\frac{\partial}{\partial y} = \frac{\partial}{\partial \eta}+\frac{\partial \zeta}{\partial y}\frac{\partial}{\partial \zeta}$ gives,

$$C_{13}\rho\bar{w} = -\frac{\partial}{\partial \xi}\int_0^z (C_{11}\rho u)d\bar{z} - \frac{\partial}{\partial \zeta}\int_0^z (C_{11}\rho u)d\bar{z}\,\frac{\partial \zeta}{\partial x}$$

$$-\frac{\partial}{\partial \eta}\int_0^z (C_{12}\rho v)d\bar{z} - \frac{\partial}{\partial \zeta}\int_0^z (C_{11}\rho v)d\bar{z}\,\frac{\partial \zeta}{\partial y}$$

Substituting $\rho d\bar{z} = \frac{\rho_e \mu_e s_1}{\phi}d\zeta$ and $\rho = \frac{\rho_e \mu_e s_1}{\phi}\frac{\partial \zeta}{\partial \bar{z}}$ gives,

$$C_{13}\bar{w}\,\frac{\rho_e\mu_e s_1}{\phi}\,\frac{\partial \zeta}{\partial \bar{z}} = -\frac{\partial}{\partial \xi}\int_0^\zeta (C_{11}u\frac{\rho_e\mu_e s_1}{\phi})\,d\zeta - C_{11}u\frac{\rho_e\mu_e s_1}{\phi}\,\frac{\partial \zeta}{\partial x}$$

$$-\frac{\partial}{\partial \eta}\int_0^\zeta (C_{12}v\frac{\rho_e\mu_e s_1}{\phi})\,d\zeta - C_{12}v\frac{\rho_e\mu_e s_1}{\phi}\,\frac{\partial \zeta}{\partial y}$$

Now let us define a new normal velocity w as given below:

$$w = \bar{w}\frac{s_1}{u_e}\frac{\partial \zeta}{\partial \bar{z}} + \frac{s_1}{h_1}F\frac{\partial \zeta}{\partial x} + \frac{s_1}{h_2}\frac{G\,v_r}{u_e}\frac{\partial \zeta}{\partial y} \qquad (33)$$

$$F = u/u_e \qquad (34)$$

$$G = v/v_r \qquad (35)$$

The velocity u is scaled by u_e and the velocity v by a reference velocity v_r. The reference velocity v_r can be u_e or unity. Choice of $v_r = v_e$ may result in large values of G near regions of cross-flow reversal and is to be avoided. With the definitions of w, F and G, and noting that $C_{11} = C_{13}/h_1$ and $C_{12} = C_{13}/h_2$, we get,

$$C_{13}\,\frac{\rho_e\mu_e u_e}{\phi}w = -\frac{\partial}{\partial \xi}\int_0^\zeta (C_{11}F\frac{\rho_e\mu_e s_1 u_e}{\phi})\,d\zeta - \frac{\partial}{\partial \eta}\int_0^\zeta (C_{12}Gv_r\frac{\rho_e\mu_e s_1}{\phi})\,d\zeta$$

Differentiating with respect to ζ and simplifying gives the transformed continuity equation,

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_1 \frac{\partial F}{\partial \xi} + \tilde{A}_2 F + \tilde{A}_3 \frac{\partial G}{\partial \eta} + \tilde{A}_4 G \qquad (36)$$

The expressions for $\tilde{A}_i$ are given in Table 2.

Transformation of the x Momentum Equation

The left-hand side of the x momentum equation (9) is transformed as follows:

$$\frac{u}{h_1}\frac{\partial u}{\partial x} + \frac{v}{h_2}\frac{\partial u}{\partial y} + \varpi\frac{\partial u}{\partial z} + C_{24}u^2 + C_{25}uv + C_{26}v^2 \rightarrow$$

$$\frac{u}{h_1}\left(\frac{\partial u}{\partial \xi}+\frac{\partial u}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial x}\right)+ \frac{v}{h_2}\left(\frac{\partial u}{\partial \eta}+\frac{\partial u}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial y}\right)+ \varpi\,\frac{\partial u}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial z} + C_{24}u^2 + C_{25}uv + C_{26}v^2$$

Multiplication by $\frac{s_1}{u_e^2}$ and replacing u, v by Fu_e and Gv_r respectively gives,

$$\frac{Fs_1}{h_1 u_e}\frac{\partial Fu_e}{\partial \xi} + \frac{Gs_1 v_r}{h_1 u_e^2}\frac{\partial Fu_e}{\partial \eta} + \frac{1}{u_e}\frac{\partial Fu_e}{\partial \zeta}\left[\frac{Fs_1}{h_1}\frac{\partial \zeta}{\partial x} + \frac{Gs_1 v_r}{h_2}\frac{\partial \zeta}{\partial y} + \varpi\frac{s_1}{u_e}\frac{\partial \zeta}{\partial z}\right]$$

$$+ C_{24}F^2 s_1 + C_{25}FGv_r\frac{s_1}{u_e} + C_{26}s_1 G^2\frac{v_r^2}{u_e^2} = \text{RHS} \cdot \left(\frac{s_1}{u_e^2}\right)$$

The term in the square brackets is equal to w from equation (33). In fact, the choice of w in the transformation of the continuity equation is based on the transformation of the momentum equations.

The pressure gradient terms on the right-hand side can be obtained from the surface Euler equations (27) and (28). Thus the term

$$\frac{1}{\rho}\left[C_{27}\frac{\partial p}{\partial x}+C_{28}\frac{\partial p}{\partial y}\right]$$

can be replaced by

$$\frac{\rho_e}{\rho}\left[\frac{u_e}{h_1}\frac{\partial u_e}{\partial x} + \frac{v_e}{h_2}\frac{\partial u_e}{\partial y} + C_{24}u_e^2 + C_{25}u_e v_e + C_{26}v_e^2\right]$$

When multiplied by $\frac{s_1}{u_e^2}$, the pressure gradient term becomes,

$$\theta\frac{s_1}{u_e^2}\left[\frac{u_e}{h_1}\frac{\partial u_e}{\partial x} + \frac{v_e}{h_2}\frac{\partial u_e}{\partial y} + C_{24}u_e^2 + C_{25}u_e v_e + C_{26}v_e^2\right]$$

Table 2. Summary of the Transformed 3D BL Equations

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_1 \frac{\partial F}{\partial \xi} + \tilde{A}_2 F + \tilde{A}_3 \frac{\partial G}{\partial \eta} + \tilde{A}_4 G$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_1 \frac{\partial F^2}{\partial \xi} + B_2 \frac{\partial FG}{\partial \eta} + B_3 F^2 + B_4 FG + B_5 G^2 + B_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_1 \frac{\partial FG}{\partial \xi} + C_2 \frac{\partial G^2}{\partial \eta} + C_3 F^2 + C_4 FG + C_5 G^2 + C_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_3 I - wH) = D_1 \frac{\partial FH}{\partial \xi} + D_2 \frac{\partial GH}{\partial \eta} + D_3 FH + D_4 GH + D_5$$

Transformation:

$$F = \frac{u}{u_e}$$

$$G = \frac{v}{v_r}$$

$$L = \frac{\partial F}{\partial \zeta} = F'$$

$$M = \frac{\partial G}{\partial \zeta} = G'$$

$$I = \frac{\partial H}{\partial \zeta} = H'$$

$$s_1 = \int_0^{\xi} h_1 d\xi$$

$$\phi = \sqrt{(\rho_e \mu_e s_1 u_e)} \quad ; \quad \psi = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}}$$

$$\xi = x$$

$$\eta = y$$

$$\zeta = \frac{\phi}{\rho_e \mu_e s_1} \int_0^z \rho \, dz = \psi(x,y) \int_0^z \rho \, dz$$

$$w = \bar{w} \frac{s_1}{u_e} \frac{\partial \zeta}{\partial z} + \frac{s_1}{h_1} F \frac{\partial \zeta}{\partial x} + \frac{s_1}{h_2} \frac{G v_r}{u_e} \frac{\partial \zeta}{\partial y}$$

Coefficients:

$$\tilde{A}_1 = \frac{-s_1}{h_1}$$

$$\tilde{A}_2 = \frac{-s_1}{C_{13}\phi}\frac{\partial}{\partial\xi}\left\{C_{13}\frac{\phi}{h_1}\right\}$$

$$\tilde{A}_3 = \frac{-s_1 v_r}{h_2 u_e}$$

$$\tilde{A}_4 = \frac{-s_1}{C_{13}\phi}\frac{\partial}{\partial\eta}\left\{C_{13}\phi\frac{v_r}{h_2 u_e}\right\}$$

$$B_1 = -\tilde{A}_1$$

$$B_2 = -\tilde{A}_3$$

$$B_3 = \frac{s_1}{h_1 u_e}\frac{\partial u_e}{\partial\xi} - \tilde{A}_2 + C_{24}s_1$$

$$B_4 = \frac{s_1 v_r}{h_2 u_e^2}\frac{\partial u_e}{\partial\eta} - \tilde{A}_4 + C_{25}\frac{s_1 v_r}{u_e}$$

$$B_5 = C_{26}\frac{s_1 v_r^2}{u_e^2}$$

$$\tilde{B}_3 = B_3 + \tilde{A}_2$$

$$\tilde{B}_4 = B_4 + \tilde{A}_4$$

$$B_6 = -\left\{\tilde{B}_3 + \tilde{B}_4 G_e + B_5 G_e^2 + (B_2 + \tilde{A}_3)\frac{\partial G_e}{\partial\eta}\right\}$$

$$C_1 = -\tilde{A}_1$$

$$C_2 = -\tilde{A}_3$$

$$C_3 = \frac{C_{34}s_1 u_e}{v_r}$$

$$C_4 = -\tilde{A}_2 + s_1 C_{35} + \frac{s_1}{h_1 v_r}\frac{\partial v_r}{\partial\xi}$$

$$C_5 = -\tilde{A}_4 + C_{36}\frac{s_1 v_r}{u_e} + \frac{s_1}{h_2 u_e}\frac{\partial v_r}{\partial\eta}$$

$$\tilde{C}_4 = C_4 + \tilde{A}_2$$

$$\tilde{C}_5 = C_5 + \tilde{A}_4$$

$$C_6 = -\left\{C_1\frac{\partial G_e}{\partial\xi} + (C_2 + \frac{\tilde{A}_3}{2})\frac{\partial G_e^2}{\partial\eta} + C_3 + \tilde{C}_4 G_e + \tilde{C}_5 G_e^2\right\}$$

$$D_1 = -\tilde{A}_1$$

$$D_2 = -\tilde{A}_3$$

$$D_3 = -\tilde{A}_2$$

$$D_4 = -\tilde{A}_4$$

$$D_5 = \frac{u_\infty^{*2}}{H_{ref}^*} \frac{1}{2} \frac{\partial}{\partial \zeta} \left\{ l \left(\frac{1-\sigma}{\sigma} \right) \frac{\partial}{\partial \zeta} q^2 \right\}$$

$$q^2 = F^2 u_e^2 + G^2 v_r^2 + 2FG \frac{v_r u_e g_{12}}{h_1 h_2}$$

$$\theta = \frac{\rho_e}{\rho} = \left\{ \frac{H}{(\gamma-1)M_\infty^2} - \frac{1}{2} q^2 \right\} \frac{\gamma-1}{\gamma T_e}$$

$$\theta' = \left\{ \frac{H'}{(\gamma-1)M_\infty^2} - q\, q' \right\} \frac{\gamma-1}{\gamma T_e}$$

$$l = \frac{\rho\mu}{\rho_e \mu_e} = \sqrt{\theta} \left(\frac{1+\tilde{S}}{\theta+\tilde{S}} \right)$$

$$\tilde{S} = \frac{S^*}{T_{ref}^* T_e}$$

$$l' = \frac{1+\tilde{S}}{2} \frac{\tilde{S}-\theta}{(\tilde{S}+\theta)^2} \frac{\theta'}{\sqrt{\theta}}$$

$$l_1 = l \left(1 + \frac{\varepsilon_1}{\mu} \Gamma \right)$$

$$l_2 = l \left(1 + \frac{\varepsilon_2}{\mu} \Gamma \right)$$

$$l_3 = \frac{l}{\sigma} \left(1 + \frac{\varepsilon_H}{\mu} \frac{\sigma}{\sigma_t} \Gamma \right)$$

$$\text{where, } \theta = \frac{\rho_e}{\rho} = \frac{T}{T_e} \tag{37}$$

The viscous term $\frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu(1+\frac{\varepsilon_1}{\mu}\Gamma)\frac{\partial u}{\partial \tilde{z}}\right\}$ multiplied by $\frac{s_1}{u_e^2}$ is transformed to,

$$\frac{s_1}{u_e^2}\frac{1}{\rho}\frac{\partial}{\partial \zeta}\left\{\mu(1+\frac{\varepsilon_1}{\mu}\Gamma)\frac{\partial F u_e}{\partial \zeta}\frac{\partial \zeta}{\partial \tilde{z}}\right\}\frac{\partial \zeta}{\partial \tilde{z}}$$

Substitution of $\frac{\partial \zeta}{\partial \tilde{z}} = \frac{\rho\phi}{\rho_e \mu_e s_1}$ reduces the viscous term to $\frac{\partial}{\partial \zeta}\left\{l_1\frac{\partial F}{\partial \zeta}\right\}$ where,

$$l_1 = \frac{\rho\mu}{\rho_e\mu_e}(1 + \frac{\varepsilon_1\Gamma}{\mu}) \tag{38}$$

Putting the different parts of the ξ momentum equation together and combining the $\frac{\partial}{\partial \zeta}$ terms into one and with

$$L = \frac{\partial F}{\partial \zeta} = F' \tag{39}$$

we get,

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = -Fw' + \frac{s_1F^2}{h_1u_e}\frac{\partial u_e}{\partial \xi} + \frac{s_1F}{h_1}\frac{\partial F}{\partial \xi} + \frac{s_1v_rFG}{h_2u_e^2}\frac{\partial u_e}{\partial \eta} +$$
$$\frac{s_1v_rG}{h_2u_e}\frac{\partial F}{\partial \eta} + C_{24}s_1F^2 + C_{25}s_1\frac{v_r}{u_e}FG + C_{26}\frac{s_1v_r^2}{u_e}G^2$$
$$- \theta\frac{s_1}{u_e}\left[\frac{1}{h_1}\frac{\partial u_e}{\partial x} + \frac{v_e}{h_2u_e}\frac{\partial u_e}{\partial y} + C_{24}u_e + C_{25}v_e + C_{26}\frac{v_e^2}{u_e}\right]$$

Substituting for $-Fw'$ by $F\,(\tilde{A}_1\frac{\partial F}{\partial \xi} + \tilde{A}_2F + \tilde{A}_3\frac{\partial G}{\partial \eta} + \tilde{A}_4G)$ from equation (36) and simplification gives :

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_1\frac{\partial F^2}{\partial \xi} + B_2\frac{\partial FG}{\partial \eta} + B_3F^2 + B_4FG + B_5G^2 + B_6\theta \tag{40}$$

The expressions for B_i are given in Table 2. Note that the coefficient B_6 in the equation corresponds to the pressure gradient term. It can be obtained from equation (39) by taking the limit $\zeta \to \infty$ and using the relation,

$$\left.\frac{\partial w}{\partial \zeta}\right|_\infty = \tilde{A}_2 + \tilde{A}_3\frac{\partial v_e}{\partial \eta} + \tilde{A}_4v_e \tag{41}$$

from the continuity equation (in the limit $\zeta \to \infty$).

Transformation of the y Momentum Equation

The y momentum equation is transformed in an identical fashion to the x momentum equation. In this case, we multiply both sides of the equation by $\frac{s_1}{u_e v_r}$. The $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, $\frac{\partial}{\partial z}$ derivatives are transformed to $\frac{\partial}{\partial \xi}$, $\frac{\partial}{\partial \eta}$, $\frac{\partial}{\partial \zeta}$ derivatives as explained in the case of the x momentum equation. Finally, we substitute for the new normal velocity w and define l_2 and M as

$$l_2 = \frac{\rho\mu}{\rho_e\mu_e}\left(1 + \frac{\varepsilon_2\Gamma}{\mu}\right) \tag{42}$$

$$M = \frac{\partial G}{\partial \zeta} = G' \tag{43}$$

Simplification and rearrangement of terms results in,

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_1\frac{\partial FG}{\partial \xi} + C_2\frac{\partial G^2}{\partial \eta} + C_3F^2 + C_4FG + C_5G^2 + C_6\theta \tag{44}$$

The expressions for C_i are given in Table 2.

Transformation of the Energy Equation

The left hand side of the energy equation is transformed to,

$$\frac{Fu_e}{h_1}\frac{\partial H}{\partial \xi} + \frac{G}{h_2}\frac{\partial H}{\partial \eta} + w\frac{\partial H}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial z} + \frac{\partial H}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial x}\frac{Fu_e}{h_1} + \frac{\partial H}{\partial \zeta}\cdot\frac{\partial \zeta}{\partial y}\frac{Gv_r}{h_2}$$

Multiplication by (s_1/u_e) and substitution for w from equation (33) gives

$$\frac{Fs_1}{h_1}\frac{\partial H}{\partial \xi} + \frac{s_1 G v_r}{h_2 u_e}\frac{\partial H}{\partial \eta} + w\frac{\partial H}{\partial \zeta}$$

The right-hand side multiplied by (s_1/u_e) simplifies to,

$$\frac{\partial}{\partial \zeta}\left\{ l_3\frac{\partial H}{\partial \zeta} - l\,\frac{1-\sigma}{\sigma}\,\frac{u_\infty^{*2}}{H_{ref}^*}\cdot \tfrac{1}{2}\cdot\frac{\partial}{\partial \zeta}q^2 \right\}$$

where,

$$l_3 = \frac{l}{\sigma}\left(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma\right) \tag{45}$$

$$l = \frac{\rho\mu}{\rho_e\mu_e} \tag{46}$$

Using the continuity equation (36) and some algebraic manipulation gives,

$$\frac{\partial}{\partial\zeta}(l_3\mathbf{I}-wH) = D_1\frac{\partial FH}{\partial\xi} + D_2\frac{\partial GH}{\partial\eta} + D_3FH + D_4GH + D_5 \qquad (47)$$

where **I** is the normal derivative of H

$$\mathbf{I} = \frac{\partial H}{\partial\zeta} = H' \qquad (48)$$

The coefficients D_1 to D_5 are given in Table 2. The term D_5 is unlike the other coefficients in that it is not just a function of the edge conditions. It depends on the variables F and G also. If the energy equation is solved decoupled from the momentum equations, F and G are known quantities and D_5 can be calculated explicitly.

The ratio θ can be obtained in terms of H and T_e from equations (13) and (16) as

$$\theta = \frac{\rho_e}{\rho} = \left[\frac{H}{(\gamma-1)M_\infty^2} - \frac{1}{2}q^2\right]\frac{\gamma-1}{\gamma T_e} \qquad (49)$$

The viscosity ratio I can be expressed as below:

$$I = \frac{\rho\mu}{\rho_e\mu_e} = \frac{1}{\theta}\left[\frac{\mu_s^* T^{*1.5}}{T^*+S^*}\right]\cdot\left[\frac{T_e^*+S^*}{\mu_s^* T_e^{*1.5}}\right]$$

Simplification gives,

$$I = \sqrt{\theta}\left[\frac{1+\tilde{S}}{\theta+\tilde{S}}\right] \; ; \; \tilde{S} = \frac{S^*}{T_{ref}^* T_e} = \frac{S}{T_e} \qquad (50)$$

The ratios I_1, I_2, I_3 are related to I as given by equations (38), (42) and (45). For laminar flow, $I=I_1=I_2=\sigma I_3$. The normal derivative of I (required subsequently) can be obtained by differentiation as,

$$I' = \frac{\partial I}{\partial\zeta} = \frac{1+\tilde{S}}{2}\frac{\tilde{S}-\theta}{(\tilde{S}+\theta)^2}\frac{\theta'}{\sqrt{\theta}} \qquad (51)$$

Similarly, the normal derivative of θ required at a later step is,

$$\theta' = \frac{\partial\theta}{\partial\zeta} = \left[\frac{H'}{(\gamma-1)M_\infty^2} - q\,q'\right]\frac{\gamma-1}{\gamma T_e} \qquad (52)$$

The transformed 3D BL equations with a complete definition of all the variables and coefficients are given in Table 2.

5. SPECIAL SUBSETS OF THE 3D BL EQUATIONS

The boundary planes required in solving the 3D BL equations are obtained as solutions of degenerate forms of the complete set at the boundaries. The general equations simplify to 2D-like forms at these boundaries, which can be solved independent of the 3D region. In some cases, the quasi-2D form at a boundary is the result of some simplifying assumptions. It is proposed here to present briefly the equations corresponding to these cases. The following types of boundary flows are encountered:

1. Stagnation point flow (STG BL)
2. Symmetry line flow (SYM BL)
3. Attachment line flow (ATT BL)
4. Infinite swept wing flow (INF BL)
5. Similarity-type flow (SIM BL)

A brief description of the corresponding equations and their transformed forms will now be presented. It is to be noted that after transformation, these equations can also be expressed in the general form given in Table 2, with some coefficients reducing to zero and some others taking on different values. Hence, these equations can be solved by the general 3D BL solver with appropriate edge conditions for each case.

Stagnation Point Flow (STG BL)

The basic assumption is that the location of the stagnation point is where the total inviscid velocity becomes zero on the body surface. The stagnation point may be of the 2D flow type (example: near the leading edge of an airfoil), or the 3D type (example: forebody of a fuselage at an angle of attack). The body in this vicinity is assumed to have some degree of bluntness. Starting of boundary calculations on a pointed body has to be obtained from similarity solutions. Since $u_e = 0$ and $v_e = 0$, the solution that is sought are the $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$ values within the boundary layer. Because of the similarity-type transformation used for the stagnation point flow and the general 3D flow, the solution profiles obtained from the stagnation point flow can be used directly as initial planes for the 3D solution.

Both the x and y momentum equations are singular at the stagnation point. Hence we differentiate the x momentum equation in the x direction and the y momentum equation in the y direction to obtain equations characterizing $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial y}$.

The continuity equation with u = v = 0 reduces to :

$$C_{11}\rho\frac{\partial u}{\partial x} + C_{12}\rho\frac{\partial v}{\partial y} + C_{13}\frac{\partial \rho\bar{w}}{\partial \bar{z}} = 0 \tag{52}$$

We define F and G for this case as:

$$F = \frac{f}{f_e} \; ; \; f_e = \frac{\partial u_e}{\partial x} \; ; \; f = \frac{\partial u}{\partial x} \tag{53}$$

$$G = \frac{g}{g_e} \; ; \; g_e = \frac{\partial v_e}{\partial y} \; ; \; g = \frac{\partial v}{\partial y} \tag{54}$$

Equation (52) then becomes

$$\frac{\rho f_e F}{h_1} + \frac{\rho g_e G}{h_2} + \frac{\partial \rho\bar{w}}{\partial \bar{z}} = 0 \tag{55}$$

Define a transformation,

$$\zeta = \psi \int_0^{\bar{z}} \rho \, d\bar{z} \tag{56}$$

$$\psi = \sqrt{\frac{f_e}{\rho_e \mu_e h_1}} \tag{57}$$

$$\text{Hence,} \quad \frac{\partial \zeta}{\partial \bar{z}} = \psi\rho \tag{58}$$

Equation (55) is then transformed to,

$$\rho F + \frac{\rho h_1 g_e}{h_2 f_e} + \psi \, \rho \, \frac{\partial \rho\bar{w}}{\partial \zeta} = 0 \tag{59}$$

$$\text{Define,} \quad w = \sqrt{\frac{h_1}{\rho_e \mu_e f_e}}\,\rho\bar{w} = \frac{h_1}{f_e}\,\psi\,\rho\bar{w} \tag{60}$$

$$\text{and} \quad C_{r,s} = \frac{h_1 g_e}{h_2 f_e} \tag{61}$$

Equation (59) simplifies to,

$$\frac{\partial w}{\partial \zeta} = -F - C_{r,s}G \tag{62}$$

Differentiating the x momentum equation (9) with respect to x and setting u = v = 0 gives,

$$\frac{1}{h_1}\left[\frac{\partial u}{\partial x}\right]^2 + \bar{w}\frac{\partial^2 u}{\partial \bar{z}\partial x} = \frac{\partial}{\partial x}\left\{\frac{1}{\rho}\left[C_{27}\frac{\partial p}{\partial x} + C_{28}\frac{\partial p}{\partial y}\right]\right\} + \frac{1}{\rho}\frac{\partial}{\partial \bar{z}}\left\{\mu\left(1+\frac{\varepsilon_1}{\mu}\Gamma\right)\frac{\partial^2 u}{\partial \bar{z}\partial x}\right\}$$

The pressure gradient term reduces to $(\frac{\rho_e}{\rho})\frac{1}{h_1}(\frac{\partial u_e}{\partial x})^2$ by taking the above equation to the limit $\bar{z} \to \infty$. Using the notations from equations (53) and (54) we get,

$$\frac{f_e^2}{h_1}F^2 + \bar{w}f_e\frac{\partial F}{\partial \bar{z}} = \frac{\theta}{h_1}f_e^2 + \frac{f_e}{\rho}\frac{\partial}{\partial \bar{z}}\left\{\mu(1+\frac{\varepsilon_1}{\mu}\Gamma)\frac{\partial F}{\partial \bar{z}}\right\}$$

Transforming $\frac{\partial F}{\partial \bar{z}}$ to $\frac{\partial F}{\partial \zeta} \cdot \psi\rho$ and using equation (6) for w, equation (38) for l_1 gives

$$F^2 + w\frac{\partial F}{\partial \zeta} = \theta + \frac{\partial}{\partial \zeta}(l_1\frac{\partial F}{\partial \zeta})$$

Using equation (62) for $\frac{\partial w}{\partial \zeta}$ results in

$$\frac{\partial}{\partial \zeta}(l_1\frac{\partial F}{\partial \zeta} - wF) = 2F^2 + C_{r,s}FG - \theta \qquad (63)$$

Transformation of the y derivative of the y momentum equation by a similar procedure gives

$$\frac{\partial}{\partial \zeta}(l_2\frac{\partial G}{\partial \zeta} - wG) = 2C_{r,s}G^2 + FG - C_{r,s}\theta \qquad (64)$$

The energy equation (11) with u=v=0 reduces to

$$\bar{w}\frac{\partial H}{\partial \bar{z}} = \frac{1}{\sigma}\frac{\partial}{\partial \bar{z}}\left\{\frac{\mu}{\sigma}(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma)\frac{\partial H}{\partial \bar{z}}\right\}$$

Transformation and simplification as before gives

$$w\frac{\partial H}{\partial \zeta} = \frac{\partial}{\partial \zeta}(l_3\frac{\partial H}{\partial \zeta})$$

Application of the transformed continuity equation gives

$$\frac{\partial}{\partial \zeta}(l_3\frac{\partial H}{\partial \zeta} - wH) = FH + C_{r,s}GH \qquad (65)$$

The complete transformed system for 3D stagnation point flow is given in Table 3. It may be noted that the equations are similar to the general form with some terms equal to zero and some coefficients defined differently. It can also be seen that the general transformation $\zeta = \sqrt{\frac{u_e}{\rho_e\mu_e s_1}}\int_0^{\bar{z}}\rho\, d\bar{z}$ tends to the stagnation point flow transformation $\zeta = \sqrt{\frac{f_e}{\rho_e\mu_e h_1}}\int_0^{\bar{z}}\rho\, d\bar{z}$ as $x \to 0$

Table 3. Summary of the Transformed STG BL Equations

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_2 F + \tilde{A}_4 G$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_3 F^2 + B_4 FG + B_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_4 FG + C_5 G^2 + C_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_3 I - wH) = D_3 FH + D_4 GH$$

Transformation:

$$F = \frac{f}{f_e} \quad ; \; f_e = \frac{\partial u_e}{\partial x} \quad ; \; f = \frac{\partial u}{\partial x}$$

$$G = \frac{g}{g_e} \quad ; \; g_e = \frac{\partial v_e}{\partial y} \quad ; \; g = \frac{\partial v}{\partial y}$$

$$L = \frac{\partial F}{\partial \zeta} = F'$$

$$M = \frac{\partial G}{\partial \zeta} = G'$$

$$I = \frac{\partial H}{\partial \zeta} = H'$$

$$\psi = \sqrt{\frac{f_e}{\rho_e \mu_e h_1}}$$

$$\xi = x$$

$$\eta = y$$

$$\zeta = \psi \int_0^z \rho \, d\tilde{z}$$

$$w = \sqrt{\frac{h_1}{\rho_e \mu_e f_e}} \, \rho \tilde{w}$$

Coefficients:

$$\tilde{A}_1 = \tilde{A}_3 = 0$$

$$\tilde{A}_2 = -1$$

$$\tilde{A}_4 = -C_{r,s}$$

$$B_1 = B_2 = B_5 = 0$$

$$B_3 = 2$$

$$B_4 = C_{r,s}$$

$$\tilde{B}_3 = B_3 + \tilde{A}_2$$

$$\tilde{B}_4 = B_4 + \tilde{A}_4$$

$$B_6 = -1$$

$$C_1 = C_2 = C_3 = 0$$

$$C_4 = 1$$

$$C_5 = 2C_{r,s}$$

$$\tilde{C}_4 = C_4 + \tilde{A}_2$$

$$\tilde{C}_5 = C_5 + \tilde{A}_4$$

$$C_6 = -C_{r,s}$$

$$D_1 = -\tilde{A}_1$$

$$D_2 = -\tilde{A}_3$$

$$D_3 = -\tilde{A}_2$$

$$D_4 = -\tilde{A}_4$$

$$D_5 = 0$$

$$\theta = \frac{\rho_e}{\rho} = \frac{H}{\gamma T_e M_\infty^2} \quad ; \quad \theta' = \frac{H'}{\gamma T_e M_\infty^2}$$

$$l = \frac{\rho\mu}{\rho_e \mu_e} = \sqrt{\theta}\left(\frac{1+\tilde{S}}{\theta+\tilde{S}}\right)$$

$$\tilde{S} = \frac{S^*}{T^*_{ref}\, T_e}$$

$$l' = \frac{1+\tilde{S}}{2}\,\frac{\tilde{S}-\theta}{(\tilde{S}+\theta)^2}\,\frac{\theta'}{\sqrt{\theta}}$$

$$l_1 = l_2 = l \quad ; \quad l_3 = \frac{l}{\sigma}$$

and $(u_e/s_1) \rightarrow (f_e/h_1)$. The general definition of $F = u/u_e$ tends to the stagnation point definition of $F = f/f_e$ as $x \rightarrow 0$. The solution G for stagnation point flow translates to G in general 3D flow by the relation

$$G \text{ (3D BL flow)} = G \text{ (STG BL flow)} \cdot (v_e / v_r) \qquad (66)$$

In the transformed system for STG BL flow, the only input from the external inviscid flow is the ratio $C_{r,s}$. The value of $C_{r,s} = 0$ corresponds to that of a 2D stagnation point flow (for example, flow past an airfoil at an angle of attack). The value of $C_{r,s} = 1$ corresponds to flow past an axisymmetric body at zero angle of attack (blunted cone, ellipsoid or any axisymmetric body). Values of $C_{r,s}$ in the range (0,1) correspond to flow past a general stagnation point. For fuselage-type bodies at moderate angles of attack ($\alpha \leq 6°$), $C_{r,s} = 1$ is a fairly good approximation. In this case, F and G solutions are identical. This avoids the difficulty of mapping the solution from the stagnation point flow coordinate system to the general 3D BL coordinate system defined on an infinitesimal circle around the stagnation point (as discussed in Ref. 2, page 32).

Symmetry Line Boundary Layer Flow (SYM BL)

In many boundary layer flows, notably in flow past bodies of revolution, lines of symmetry exist along windward and leeward planes. The flow along the root plane of a symmetric wing-alone configuration also belongs to this class. The solution along these planes can be obtained independent of the 3D region, and can be used as starting planes for y direction sweep in the 3D calculation. The crossflow velocity v_e is zero in the symmetry plane; however, the gradient of v_e in the orthogonal direction is non-zero. The quantities to be specified at the boundary layer edge are thus, the velocity u_e and the gradient $\partial v_e/\partial y$. The transformed equations for this type of flow are discussed below.

Along the symmetry plane, the following assumptions are used:

(1) The grid is orthogonal at the symmetry line; hence x and y correspond to orthogonal coordinates and $g_{12} = 0$. The derivative $\dfrac{\partial g_{12}}{\partial y}$ is not necessarily zero.

(2) The crossflow velocity v_e is equal to zero; $\partial v_e/\partial y = g_e$ is not necessarily equal to zero.

(3) The quantities u, ρ, p, h_1, h_2 are even functions on the symmetry line, i.e., their derivatives in the y direction are equal to zero.

The metric coefficients C_{24} and C_{34} reduce to zero with the above assumptions. The y momentum equation reduces to a singular form; hence its derivative in the y direction is taken. The derivative $\frac{\partial C_{34}}{\partial y}$ is non-zero and is retained in the equation. The resulting equations are given below:

$$\frac{\partial}{\partial x}(h_2\rho u) + h_1\rho g + h_1 h_2 \frac{\partial}{\partial \tilde{z}}(\rho \tilde{w}) = 0 \tag{67}$$

$$\frac{u}{h_1}\frac{\partial u}{\partial x} + \tilde{w}\frac{\partial u}{\partial \tilde{z}} = \theta\left(\frac{u_e}{h_1}\frac{\partial u_e}{\partial x}\right) + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_1}{\mu}\Gamma\right)\frac{\partial u}{\partial \tilde{z}}\right\} \tag{68}$$

$$\frac{u}{h_1}\frac{\partial g}{\partial x} + \frac{1}{h_2}g^2 + \tilde{w}\frac{\partial g}{\partial \tilde{z}} + \frac{\partial C_{34}}{\partial y}u^2 + C_{35}ug =$$

$$\theta\left\{\frac{u_e}{h_1}\frac{\partial g_e}{\partial x} + \frac{1}{h_2}g_e^2 + \frac{\partial C_{34}}{\partial y}u_e^2 + C_{35}u_e g_e\right\} + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_2}{\mu}\Gamma\right)\frac{\partial g}{\partial \tilde{z}}\right\} \tag{69}$$

$$\frac{u}{h_1}\frac{\partial H}{\partial x} + \tilde{w}\frac{\partial H}{\partial \tilde{z}} = \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\frac{\mu}{\sigma}\left(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma\right)\frac{\partial H}{\partial \tilde{z}} - \frac{u_\infty^{*2}}{H^*_{ref}}\mu\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial \tilde{z}}\left(\frac{u^2}{2}\right)\right\} \tag{70}$$

The transformation of the above equations to computational variables can be done in a manner identical to the 3D BL equations. In the present case, however, G and w are defined as,

$$G = \frac{g}{g_r} \tag{71}$$

$$w = \tilde{w}\frac{s_1}{u_e}\frac{\partial \zeta}{\partial \tilde{z}} + \frac{s_1}{h_1}F\frac{\partial \zeta}{\partial x} \tag{72}$$

The reference quantity g_r can be chosen as u_e or as unity. The transformed system and related coefficients are summarized in Table 4.

The solution F and G from STG BL equations can be used as the initial plane for the symmetry line flow. The solution G needs to be converted as given below, due to differring definitions.

$$G\ (\text{SYM BL}) = G\ (\text{STG BL}) \cdot \frac{g_e}{g_r} \tag{73}$$

Table 4. Summary of the Transformed SYM BL Equations

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_1 \frac{\partial F}{\partial \xi} + \tilde{A}_2 F + \tilde{A}_4 G$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_1 \frac{\partial F^2}{\partial \xi} + B_3 F^2 + B_4 FG + B_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_1 \frac{\partial FG}{\partial \xi} + C_3 F^2 + C_4 FG + C_5 G^2 + C_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_3 I - wH) = D_1 \frac{\partial FH}{\partial \xi} + D_3 FH + D_4 GH + D_5$$

Transformation:

$$F = \frac{u}{u_e}$$

$$G = \frac{g}{g_r} = \frac{\partial v}{\partial y}/g_r$$

$$L = \frac{\partial F}{\partial \zeta} = F'$$

$$M = \frac{\partial G}{\partial \zeta} = G'$$

$$I = \frac{\partial H}{\partial \zeta} = H'$$

$$s_1 = \int_0^{\xi} h_1 d\xi$$

$$\phi = \sqrt{(\rho_e \mu_e s_1 u_e)} \quad ; \quad \psi = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}}$$

$$\xi = x$$

$$\eta = y$$

$$\zeta = \frac{\phi}{\rho_e \mu_e s_1} \int_0^{z} \rho \, dz = \psi(x,y) \int_0^{z} \rho \, dz$$

$$w = \bar{w} \frac{s_1}{u_e} \frac{\partial \zeta}{\partial z} + \frac{s_1}{h_1} F \frac{\partial \zeta}{\partial x}$$

Coefficients:

$$\tilde{A}_1 = \frac{-s_1}{h_1}$$

$$\tilde{A}_2 = \frac{\tilde{A}_1}{h_2\phi}\frac{\partial(h_2\phi)}{\partial\xi}$$

$$\tilde{A}_3 = 0$$

$$\tilde{A}_4 = -\frac{g_r s_1}{h_2 u_e}$$

$$B_1 = -\tilde{A}_1$$

$$B_2 = B_5 = \tilde{B}_4 = 0$$

$$B_3 = -\tilde{A}_2 + \frac{s_1}{h_1 u_e}\frac{\partial u_e}{\partial\xi}$$

$$B_4 = -\tilde{A}_4$$

$$\tilde{B}_3 = B_3 + \tilde{A}_2$$

$$B_6 = -\tilde{B}_3$$

$$C_1 = -\tilde{A}_1$$

$$C_2 = 0$$

$$C_3 = \frac{\partial C_{34}}{\partial y}\frac{s_1 u_e}{g_r}$$

$$C_4 = \frac{s_1}{h_1 g_r}\frac{\partial g_r}{\partial\xi} - \tilde{A}_2 + C_{35}s_1$$

$$C_5 = -2\tilde{A}_4$$

$$\tilde{C}_4 = C_4 + \tilde{A}_2 \quad ; \quad \tilde{C}_5 = C_5 + \tilde{A}_4$$

$$C_6 = -\frac{s_1}{h_1 g_r}\frac{\partial g_e}{\partial\xi} - \frac{s_1 g_e^2}{h_2 u_e g_r} - C_3 - \frac{C_{35} g_e s_1}{g_r}$$

D_1, D_2, D_3, D_4 as in Table 2 (3D BL)

$$D_5 = \frac{u_\infty^{*2}}{H_{ref}^*}\, u_e^2\left(\frac{1-\sigma}{\sigma}\right)\left[I'FL + I(L^2+FL')\right]$$

$$L' = \frac{1}{I_1}\left\{\frac{1}{2}B_1\frac{\partial F^2}{\partial\xi} + \tilde{B}_3F^2 + B_6\theta - L\,I_1' + wL\right\}$$

θ, θ', I, I', I_1, I_1', I_2, I_2', I_3, I_3', as in Table 2 (3D BL) with $q^2 = F^2u_e^2$

Attachment Line Boundary Layer Flow (ATT BL)

In the calculation of flow past a wing geometry, an initial plane of data has to be calculated at the leading edge attachment line. This line is located from the inviscid flow as a curved line on the body at which the streamwise velocity is equal to zero. To a degree of approximation, this line is coincident with the line at which the pressure is a maximum in the vicinity of the leading edge of a wing.

The boundary layer equations for an attachment line are similar to the symmetry line equations, with symmetry along a y coordinate line. There is however, no symmetry in the metric coefficients and the grid need not be orthogonal. The following assumptions are valid:

(1) $u = 0$; $\frac{\partial u}{\partial x} = f$ is a finite value.

(2) v is finite but $\frac{\partial v}{\partial x} = 0$.

(3) The coordinate y is coincident with the attachment line.

(4) The metrics g_{12}, $\frac{\partial h_1}{\partial y}$, $\frac{\partial h_2}{\partial y}$ need not be equal to zero.

With these assumptions the continuity equation (8) reduces to,

$$C_{11}\rho f + \frac{\partial}{\partial y}(C_{12}\rho v) + C_{13}\frac{\partial}{\partial \check{z}}(\rho \check{w}) = 0 \qquad (74)$$

The x momentum equation reduces to a singular form on the attachment line (assuming that $C_{26} = 0$ on the attachment line; fairly good if the attachment line is not too curved). Differentiation of the x momentum equation in the x direction and simplification yields,

$$\frac{v}{h_2}\frac{\partial f}{\partial y} + \frac{1}{h_1}f^2 + \check{w}\frac{\partial f}{\partial \check{z}} + \frac{\partial C_{26}}{\partial x}v^2 + C_{25}fv =$$

$$\theta\left\{\frac{v_e}{h_2}\frac{\partial f_e}{\partial y} + \frac{1}{h_1}f_e^2 + \frac{\partial C_{26}}{\partial x}v_e^2 + C_{25}f_e v_e\right\} + \frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\mu\left(1+\frac{\varepsilon_1}{\mu}\Gamma\right)\frac{\partial f}{\partial \check{z}}\right\} \qquad (75)$$

The y momentum equation simplifies to,

$$\frac{v}{h_2}\frac{\partial v}{\partial y} + \check{w}\frac{\partial v}{\partial \check{z}} + C_{36}v^2 = \theta\left(\frac{v_e}{h_2}\frac{\partial v_e}{\partial y} + C_{36}v_e^2\right) + \frac{1}{\rho}\frac{\partial}{\partial \check{z}}\left\{\mu\left(1+\frac{\varepsilon_2}{\mu}\Gamma\right)\frac{\partial v}{\partial \check{z}}\right\} \qquad (76)$$

The energy equation becomes,

$$\frac{v}{h_2}\frac{\partial H}{\partial y} + w\frac{\partial H}{\partial z} = \frac{1}{\rho}\frac{\partial}{\partial z}\left\{\frac{\mu}{\sigma}\left(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma\right)\frac{\partial H}{\partial z} - \frac{u_\infty^{*2}}{H_{ref}^*}\mu\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial z}\left(\frac{v^2}{2}\right)\right\} \qquad (77)$$

The transformation used here is similar to the 3D BL transformation, already discussed. The following distinctions are to be noted:

$$\zeta = \psi\int_0^z \rho\, dz = \frac{\phi}{\rho_e\mu_e h_1}\int_0^z \rho\, dz \qquad (78)$$

$$\phi = \sqrt{\rho_e\mu_e h_1 f_e} \quad ; \quad \psi = \sqrt{\frac{f_e}{\rho_e\mu_e h_1}} = \frac{f_e}{\phi} \qquad (79)$$

The transformation defined here is exactly compatible with the the 3D BL transformation since $\frac{u_e}{s_1} \rightarrow \frac{f_e}{h_1}$ as $x \rightarrow 0$.

The normal velocity w has a new definition given as,

$$w = \bar{w}\frac{h_1}{f_e}\frac{\partial\zeta}{\partial z} + \frac{h_1 v_r}{h_2 f_e}G\frac{\partial\zeta}{\partial y} \qquad (80)$$

The complete transformed set of equations for attachment line flow is given in Table 5. The reference value for v can be chosen as the gradient f_e.

Infinite Swept Wing Flow (INF BL)

This type of flow is typical of the flow that occurs in the mid-section of a large aspect ratio wing at constant sweep angle. The flow at any chordwise section is assumed to be the same as the flow at any other chordwise station. If we define the boundary layer coordinate x along the wing surface in the streamwise direction and y along the wing surface parallel to the leading edge (see Fig. 2), then $\frac{\partial}{\partial y}$ quantities are assumed to be zero.

For a non-tapered wing, for example, if Λ is the sweep angle and α_c the surface tangent angle, the metric quantities simplify as given below:

$$\frac{\partial x'}{\partial x} = \cos(\alpha_c) \quad ; \quad \frac{\partial y'}{\partial x} = 0 \quad ; \quad \frac{\partial z'}{\partial x} = \sin(\alpha_c)$$

$$\frac{\partial x'}{\partial y} = \sin(\Lambda) \quad ; \quad \frac{\partial y'}{\partial y} = \cos(\Lambda) \quad ; \quad \frac{\partial z'}{\partial y} = 0$$

Table 5. Summary of the Transformed ATT BL Equations

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_2 F + \tilde{A}_3 \frac{\partial G}{\partial \eta} + \tilde{A}_4 G$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_2 \frac{\partial FG}{\partial \eta} + B_3 F^2 + B_4 FG + B_5 G^2 + B_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_2 \frac{\partial G^2}{\partial \eta} + C_4 FG + C_5 G^2 + C_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_3 I - wH) = D_2 \frac{\partial GH}{\partial \eta} + D_3 FH + D_4 GH + D_5$$

Transformation:

$$F = \frac{f}{f_e} = \frac{\partial u/\partial x}{\partial u_e/\partial x}$$

$$G = \frac{v}{v_r}$$

$$L = \frac{\partial F}{\partial \zeta} = F'$$

$$M = \frac{\partial G}{\partial \zeta} = G'$$

$$I = \frac{\partial H}{\partial \zeta} = H'$$

$$\xi = x$$

$$\eta = y$$

$$\zeta = \psi \int_0^z \rho \, d\check{z} \quad ; \quad \psi = \sqrt{\frac{f_e}{\rho_e \mu_e h_1}} \quad ; \quad \phi = (f_e \rho_e \mu_e h_1)^{1/2}$$

$$w = \check{w} \frac{h_1}{f_e} \frac{\partial \zeta}{\partial \check{z}} + \frac{h_1 v_r}{h_2 f_e} G \frac{\partial \zeta}{\partial y}$$

Coefficients:

$$\tilde{A}_1 = 0$$

$$\tilde{A}_2 = -1$$

$$\tilde{A}_3 = -\frac{h_1 v_r}{h_2 f_e}$$

$$\tilde{A}_4 = -\frac{h_1}{C_{13}\phi}\frac{\partial}{\partial\eta}\left\{\frac{C_{13}v_r\phi}{h_2 f_e}\right\}$$

$$B_1 = 0$$

$$B_2 = -\tilde{A}_3$$

$$B_3 = 2$$

$$B_4 = \frac{h_1 v_r}{h_2 f_e^2}\frac{\partial f_e}{\partial\eta} + C_{25}v_r\frac{h_1}{f_e} - \tilde{A}_4$$

$$B_5 = h_1\frac{v_r^2}{f_e^2}\frac{\partial C_{26}}{\partial\xi}$$

$$\tilde{B}_3 = B_3+\tilde{A}_2$$

$$\tilde{B}_4 = B_4$$

$$B_6 = -\left[\tilde{B}_3+\tilde{B}_4 G_e+B_5 G_e^2\right]$$

$$C_1 = 0$$

$$C_2 = -\tilde{A}_3$$

$$C_3 = 0$$

$$C_4 = -\tilde{A}_2$$

$$C_5 = \frac{h_1}{f_e h_2}\frac{\partial v_r}{\partial\eta} - \tilde{A}_4 + C_{36}\frac{h_1 v_r}{f_e}$$

$$\tilde{C}_4 = C_4+\tilde{A}_2$$

$$\tilde{C}_5 = C_5$$

$$C_6 = -\tilde{C}_5 G_e^2 - \frac{C_2}{2}\frac{\partial G_e^2}{\partial\eta}$$

$$D_1 = 0$$

$$D_2 = -\tilde{A}_3$$

$$D_3 = -\tilde{A}_2$$

$$D_4 = -\tilde{A}_4$$

$$D_5 = \frac{u_\infty^{*2}}{H_{ref}^*} v_r^2\left(\frac{1-\sigma}{\sigma}\right)\left[I'GM + I(M^2+GM')\right]$$

$$M' = \frac{1}{I_2}\left[\frac{1}{2}C_2\frac{\partial G^2}{\partial\eta}+\tilde{C}_5 G^2+C_6\theta+M(w-I_2')\right]$$

θ, θ', I, I', I_1, I_1', I_2, I_2', I_3, I_3', as in Table 2 (3D BL) with $q^2 = G^2 v_r^2$

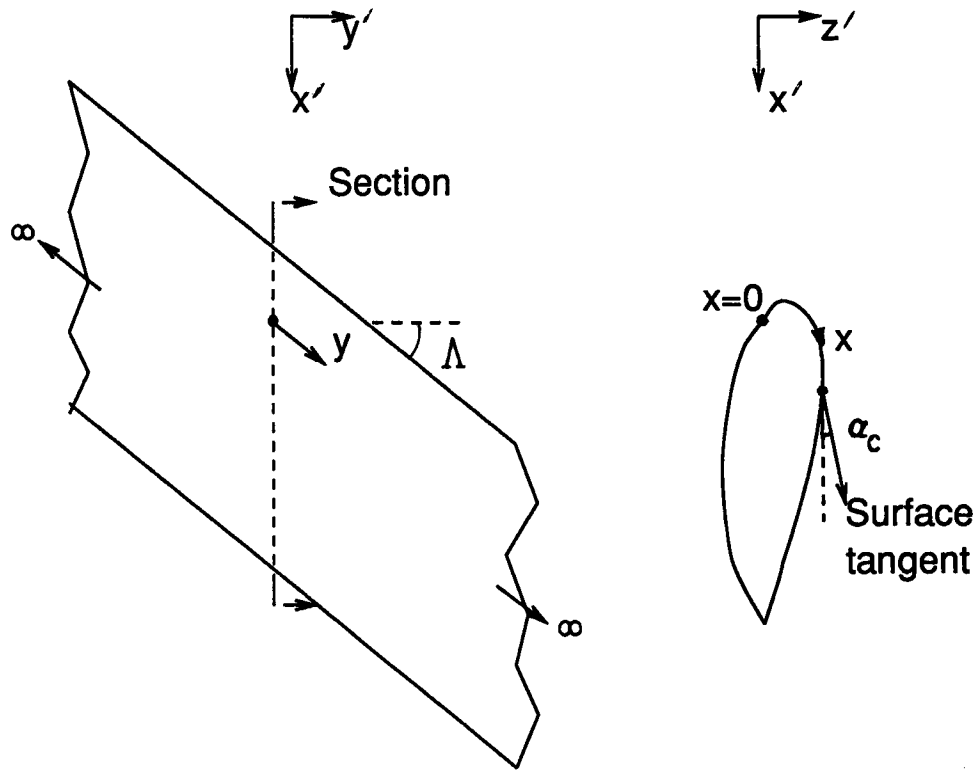


Fig. 2 Nomenclature Used for Infinite Swept Wing Flow

From equations 19 thru' 22, we get,

$$h_1 = 1 \quad ; \; h_2 = 1 \quad ; \; g_{12} = \sin(\Lambda)\cos(\alpha_c)$$

The metric coefficients listed in Table 1 simplify as follows:

$$C_{11} = C_{13} = \sqrt{\bar{g}} = \sqrt{1-g_{12}^2}$$

$$C_{25} = C_{26} = C_{35} = C_{36} = 0$$

$$C_{24} = -\frac{g_{12}}{\bar{g}}\frac{\partial g_{12}}{\partial x} \quad ; \; C_{34} = \frac{1}{\bar{g}}\frac{\partial g_{12}}{\partial x}$$

The 3D BL equations simplify to the following form for infinite swept wing flow:

$$\frac{\partial}{\partial x}(C_{11}\rho u) + C_{13}\frac{\partial}{\partial \tilde{z}}(\rho \tilde{w}) = 0 \tag{81}$$

$$\frac{u}{h_1}\frac{\partial u}{\partial x} + \tilde{w}\frac{\partial u}{\partial \tilde{z}} + C_{24}u^2 = \theta\left(\frac{u_e}{h_1}\frac{\partial u_e}{\partial x} + C_{24}u_e^2\right) + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_1}{\mu}\Gamma\right)\frac{\partial u}{\partial \tilde{z}}\right\} \tag{82}$$

$$\frac{u}{h_1}\frac{\partial v}{\partial x} + \tilde{w}\frac{\partial v}{\partial \tilde{z}} + C_{34}u^2 = \theta\left(\frac{u_e}{h_1}\frac{\partial v_e}{\partial x} + C_{34}u_e^2\right) + \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\mu\left(1+\frac{\varepsilon_2}{\mu}\Gamma\right)\frac{\partial v}{\partial \tilde{z}}\right\} \tag{83}$$

$$\frac{u}{h_1}\frac{\partial H}{\partial x} + \tilde{w}\frac{\partial H}{\partial \tilde{z}} = \frac{1}{\rho}\frac{\partial}{\partial \tilde{z}}\left\{\frac{\mu}{\sigma}\left(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma\right)\frac{\partial H}{\partial \tilde{z}} - \frac{u_\infty^{*2}}{H_{ref}^*}\mu\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial \tilde{z}}\left(\frac{q^2}{2}\right)\right\} \tag{84}$$

The edge velocities u_e and v_e may be obtained, for example, using the classical sweep theory. According to this theory[6], the inviscid velocity distribution past an infinite swept wing (ISW) is analogous to the flow past an airfoil section normal to the wing leading edge at a Mach number $M_\infty \cos(\Lambda)$, where Λ is the constant sweep angle. If we define $u_{e,2D}$ as the non-dimensional inviscid velocity distribution on the equivalent airfoil (referenced to the free-stream velocity, $M_\infty U_\infty^* \cos(\Lambda)$), the edge velocities on the infinite swept wing are given by the sweep theory to be,

$$u_e = \frac{u_{e,2D}\cos(\Lambda)}{[\cos^2(\alpha_c)\cos^2(\Lambda) + \sin^2(\alpha_c)]^{1/2}} \tag{85}$$

$$v_e = \sin(\Lambda)\,[1-\cos(\alpha_c)\,u_e] \tag{86}$$

For the calculation of the side boundary plane of solution on wings with taper, it is possible to relax some of the above assumptions. In this case, u_e and v_e are not related to a constant sweep angle Λ, but rather obtained from a 3D inviscid calculation. The variation of u_e and v_e in the y direction is however, neglected. Since the inviscid velocities are not related to a single sweep angle, this approach — called locally infinite swept wing assumption — comes closer to the real flow in the case of tapered wings. In addition, the metric h_1 is not assumed constant in the y direction and the metric h_2 is not assumed constant in the x direction. In this case, the coefficients C_{25}, C_{26}, C_{35} and C_{36} will be non-zero. Corresponding to this, the LHS of x momentum equation will have the additional terms $C_{25}uv + C_{26}v^2$ and the LHS of the y momentum equation will have the additional terms $C_{35}uv + C_{36}v^2$.

The transformation of the equations to computational coordinates can be done in the same manner as described for the 3D BL equations. The resulting equations and expressions are summarized in Table 6. The extra terms arising from the locally infinite swept wing (LINF) assumption are shown enclosed in boxes.

Infinite Swept Attachment Line

The initial conditions to solve the infinite swept wing (ISW or LINF) equations can be generated by solving the infinite swept attachment line equations. These equations are obtained by setting $\partial/\partial\eta$ terms to zero in the general attachment line (ATT BL) equations. The resulting equations for infinite (locally or otherwise) swept wing attachment line are as given below.

$$\frac{\partial w}{\partial \zeta} = -F + \boxed{\tilde{A}_4 G \;\; ; \;\; \tilde{A}_4 = -\frac{h_1 v_r}{C_{13} f_e}\partial/\partial\eta(C_{13}/h_2)} \tag{87}$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = 2F^2 + B_4 FG + B_5 G^2 + B_6\theta \tag{88}$$

$$\boxed{B_4 = -\tilde{A}_4 + C_{25} h_1 v_r/f_e \;\; ; \;\; B_5 = h_1 \frac{\partial C_{26}}{\partial x} v_r^2/f_e^2} \;\; ; \;\; B_6 = -1 - \boxed{h_1 C_{25} G_e - h_1 \frac{\partial C_{26}}{\partial x} G_e^2}$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = FG + C_5 G^2 + C_6\theta \tag{89}$$

$$\boxed{C_5 = -\tilde{A}_4 + \frac{h_1 v_r}{f_e} C_{36} \;\; ; \;\; C_6 = -h_1 C_{36} G_e^2}$$

Table 6. Summary of the Transformed INF BL Equations†

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_1 \frac{\partial F}{\partial \xi} + \tilde{A}_2 F + \boxed{\tilde{A}_4 G}$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_1 \frac{\partial F^2}{\partial \xi} + B_3 F^2 + \boxed{B_4 FG + B_5 G^2} + B_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_2 M - wG) = C_1 \frac{\partial FG}{\partial \xi} + C_3 F^2 + C_4 FG + C_5 G^2 + C_6 \theta$$

$$\frac{\partial}{\partial \zeta}(l_3 I - wH) = D_1 \frac{\partial FH}{\partial \xi} + D_3 FH + \boxed{D_4 GH} + D_5$$

Transformation:

$$F = \frac{u}{u_e}$$

$$G = \frac{v}{v_r}$$

$$L = \frac{\partial F}{\partial \zeta} = F'$$

$$M = \frac{\partial G}{\partial \zeta} = G'$$

$$I = \frac{\partial H}{\partial \zeta} = H'$$

$$s_1 = \int_0^{\xi} h_1 d\xi$$

$$\phi = \sqrt{(\rho_e \mu_e s_1 u_e)} \quad ; \quad \psi = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}}$$

$$\xi = x$$

$$\eta = y$$

$$\zeta = \frac{\phi}{\rho_e \mu_e s_1} \int_0^{z} \rho \, d\bar{z} = \psi(x,y) \int_0^{z} \rho \, d\bar{z}$$

$$w = \bar{w} \frac{s_1}{u_e} \frac{\partial \zeta}{\partial \bar{z}} + \frac{s_1}{h_1} F \frac{\partial \zeta}{\partial x}$$

---

† Boxed terms are non-zero only for locally infinite swept wing

Coefficients:

$$\tilde{A}_1 = \frac{-s_1}{h_1}$$

$$\tilde{A}_2 = \frac{-s_1}{C_{13}\phi}\frac{\partial}{\partial\xi}\left\{C_{13}\frac{\phi}{h_1}\right\}$$

$$\tilde{A}_4 = -\frac{s_1 v_r}{C_{13}u_e}\frac{\partial}{\partial\eta}\left\{\frac{C_{13}}{h_2}\right\}$$

$$B_1 = -\tilde{A}_1$$

$$B_3 = \frac{s_1}{h_1 u_e}\frac{\partial u_e}{\partial\xi} - \tilde{A}_2 + C_{24}s_1$$

$$B_4 = -\tilde{A}_4 + C_{25}\frac{s_1 v_r}{u_e}$$

$$B_5 = C_{26}\frac{s_1 v_r^2}{u_e^2}$$

$$\tilde{B}_3 = B_3 + \tilde{A}_2$$

$$\tilde{B}_4 = B_4 + \tilde{A}_4$$

$$B_6 = -\left\{\tilde{B}_3 + \tilde{B}_4 G_e + B_5 G_e^2\right\}$$

$$C_1 = -\tilde{A}_1$$

$$C_3 = \frac{C_{34}s_1 u_e}{v_r}$$

$$C_4 = -\tilde{A}_2 + s_1 C_{35} + \frac{s_1}{h_1 v_r}\frac{\partial v_r}{\partial\xi}$$

$$C_5 = -\tilde{A}_4 + C_{36}\frac{s_1 v_r}{u_e}$$

$$\tilde{C}_4 = C_4 + \tilde{A}_2$$

$$\tilde{C}_5 = C_5 + \tilde{A}_4$$

$$C_6 = -\left\{C_1\frac{\partial G_e}{\partial\xi} + C_3 + \tilde{C}_4 G_e + \tilde{C}_5 G_e^2\right\}$$

$$D_1 = -\tilde{A}_1$$

$$D_2 = -\tilde{A}_3$$

$$D_3 = -\tilde{A}_2$$

$$\boxed{D_4 = -\tilde{A}_4}$$

$$D_5 = \frac{u_\infty^{*2}}{H_{ref}^*} \frac{1}{2} \frac{\partial}{\partial \zeta} \left\{ l \left(\frac{1-\sigma}{\sigma} \right) \frac{\partial}{\partial \zeta} q^2 \right\}$$

D_5, θ, θ', l, l' etc. as given in Table 2 (3D BL)

$$\frac{\partial}{\partial\zeta}(l_3\mathbf{I}-wH) = FH - \boxed{\tilde{A}_4 GH} + D_5 \quad \text{with} \quad D_5 = \frac{u_\infty^{*2}}{H_{ref}^*}\left(\frac{1-\sigma}{\sigma}\right)\frac{\partial}{\partial\zeta}(IGG') \quad (90)$$

The terms in boxes are non-zero only if locally infinite assumption is used.

Similarity Boundary Layer Flow (SIM BL):

Similarity solutions are necessary to initiate the solution in cases where there is a flow singularity. Usually this involves a sharp-edged or sharp-tipped body such as flat plate, wedge or cone. Some aerodynamic bodies can be approximated by one of the above near the stagnation region.

Solution does not exist in physical variables at the singularity point; however, profiles can be generated in transformed variables. The similarity transformation is a sub-set of the present transformation and hence, for such type of flows, the solution is independent of the ξ direction. Flows with streamwise pressure gradients such as subsonic wedge and cone flows are also self-similar due to the transformation. The similarity BL flow equations are either 2-D or axisymmetric. Dropping the ξ and η direction gradients in the 3-D formulation results in the equation set and coefficients given in Table 7. The coefficients $\tilde{A}_2$, B_3 and B_6 are the edge parameters of importance. Depending on the type of flow, they are evaluated as given below.

Flat Plate Flow:

For the case of flat plate, $u_e=h_1=h_2=1$, $\rho_e\mu_e=1$ and $\frac{\partial\phi}{\partial\xi}=0$, resulting in,

$$\tilde{A}_2 = \frac{-x}{\sqrt{\rho_e\mu_e x}}\frac{\partial}{\partial\xi}[\sqrt{\rho_e\mu_e x}] = -\frac{1}{2} \;\; ; \;\; B_3 = D_3 = -\tilde{A}_2 \;\; ; \;\; B_6 = 0$$

This formulation is also valid for supersonic wedge flows if conditions downstream of the oblique shock are specified as free-stream.

Supersonic Flow Past a Cone of Half-angle α_h:

This axisymmetric flow additionally involves the metric h_2 which is a function of ξ. If the free-stream conditions correspond to conditions downstream of the shock, then,

$$u_e = \rho_e\mu_e = 1 \;\; ; \;\; h_1 = \frac{1}{\text{Cos}(\alpha_h)} \;\; ; \;\; h_2 = \pi r_i \text{ , where } r_i = \text{local radius} = \xi\,\tan(\alpha_h)$$

With ξ being measured along the cone axis, $s_1=h_1\,\xi$. Under these conditions,

Table 7. Summary of the Transformed SIM BL Equations

Equations:

$$\frac{\partial w}{\partial \zeta} = \tilde{A}_2 F$$

$$\frac{\partial}{\partial \zeta}(l_1 L - wF) = B_3 F^2 + B_6$$

$$\frac{\partial}{\partial \zeta}(l_3 \mathbf{I} - wH) = D_3 FH + D_5$$

Transformation:

$$F = \frac{u}{u_e} \; ; \; L = \frac{\partial F}{\partial \zeta} = F'$$

$$H = \frac{H^*}{H^*_{ref}} \; ; \; \mathbf{I} = \frac{\partial H}{\partial \zeta} = H'$$

$$\phi = \sqrt{(\rho_e \mu_e s_1 u_e)} \quad ; \quad \psi = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}}$$

$$s_1 = \int_0^{\xi} h_1 d\xi \; ; \; \xi = x \; ; \; \eta = y$$

$$\zeta = \frac{\phi}{\rho_e \mu_e s_1} \int_0^z \rho \, dz = \psi(x,y) \int_0^z \rho \, dz$$

$$w = \bar{w} \frac{s_1}{u_e} \frac{\partial \zeta}{\partial z} + \frac{s_1}{h_1} F \frac{\partial \zeta}{\partial x}$$

Coefficients:

$$\tilde{A}_2 = \frac{s_1}{h_1 h_2 \phi} \frac{\partial (h_2 \phi)}{\partial \xi} \; ; \; B_3 = D_3 = \frac{s_1}{u_e} \frac{\partial u_e}{\partial \xi} - \tilde{A}_2$$

$$\tilde{B}_3 = B_3 + \tilde{A}_2 \; ; \; B_6 = -\tilde{B}_3$$

$$D_5 = \frac{u^{*2}_{\infty}}{H^*_{ref}} u_e^2 \left(\frac{1-\sigma}{\sigma}\right) \left[l'FL + l(L^2 + FL') \right]$$

$$L' = \frac{1}{l_1} \{ - L l_1' + wL \}$$

θ, θ', l, l', l_1, l_1', l_3, l_3', as in Table 2 (3D BL) with $q^2 = F^2 u_e^2$

$$A_2 = - \frac{s_1 \mathrm{Cos}(\alpha_h)}{\pi r_i \sqrt{\rho_e \mu_e s_1}} \frac{\partial}{\partial \xi} [\pi r_i \sqrt{\rho_e \mu_e s_1}]$$

Simplification results in, $\tilde{A}_2 = - \frac{3}{2}$; $B_3 = D_3 = - \tilde{A}_2$; $B_6 = 0$

The factor of 3 for $\tilde{A}_2$ between flat plate flow and supersonic cone flow is related to the $\sqrt{3}$ factor employed in Mangler's transformation.

Subsonic Flow Past a Wedge with Included Angle β_w:

The inviscid flow near the stagnation point follows the rule, $u_e = u_1 x^m$, where u_1 is a constant and m depends on the angle of the wedge as given by, $\beta_w = \frac{2\pi m}{m+1}$. The expressions for ϕ in the ζ transformation takes the form,

$$\phi = \sqrt{\frac{m+1}{2}} \sqrt{\rho_e \mu_e u_1} \, s_1^{\frac{m+1}{2}} \qquad (91)$$

The coordinate ξ is measured along the surface; hence, $h_1=h_2=1$ and $s_1=\xi$. The coordinate ξ can also be measured in the flow direction; the metric h_1 will then be different, but the expressions for $\tilde{A}_2$ and B_3 remain the same.

$\tilde{A}_2$ is then obtained as, $\tilde{A}_2 = \frac{\xi}{\phi} \frac{\partial \phi}{\partial \xi} = - \frac{m+1}{2}$

$$B_3 = \frac{s_1}{u_e} \frac{\partial u_e}{\partial \xi} - \tilde{A}_2 = m - \tilde{A}_2 \; ; \; B_6 = - m$$

In the above, $\rho_e \mu_e$ is assumed to be constant with respect to ξ. Flat plate flow corresponds to m=0; $m=\frac{1}{3}$ corresponds to 3D stagnation point flow with $C_{r,s} = 1$ and m=1 corresponds to 2D stagnation point flow.

Subsonic Flow on Cone-tipped Bodies:

The conditions here are identical to subsonic wedge flow; additionally, the axisymmetric influence is brought in through h_2.

$$h_2 = \pi r_i = \pi \tan(\alpha_h) \, \xi \, \mathrm{Cos}(\alpha_h) = \pi \, \xi \, \mathrm{Sin}(\alpha_h)$$

Substitution in the general expression for $\tilde{A}_2$ yields $\tilde{A}_2 = - \frac{m+3}{2}$ and $B_3 = \frac{3(m+1)}{2}$. Three-dimensional stagnation point flow with $C_{r,s} = 1$ corresponds to m=1, in this case.

6. REVERSE TRANSFORMATION

Once the solution is obtained in computational coordinates (ξ, η, ζ) in the form of transformed variables F, G, F′, G′, H, H′, w and θ, there is need for a reverse transformation by which the solution and profiles can be expressed in physical quantities. It is also necessary to calculate quantities such as boundary layer and momentum thickness values, skin friction coefficients, streamline angles, velocity profiles, temperature profiles and heat transfer to the wall in physical units. For transition and turbulence calculations, derivatives of u^*, v^*, T^* in x^*, y^*, z^* coordinates are also of interest. The reverse transformation for the general 3D case is discussed in the following pages. The following quantities are assumed given:

- ζ, F, G, H, w, θ, I, I_1, I_2, I_3 and their normal derivatives are known as part of the solution.
- ξ, η, u_e, v_r, T_e, ρ_e, μ_e, h_1, h_2 and g_{12} are the known edge conditions and assumed constant for a given profile.
- u^*_∞, T^*_{ref}, H^*_{ref} and L^*, μ^*_{ref} and Re_{ref} are the known reference quantities.

In the general case of transformation from (x, y, z) to (ξ, η, ζ), the gradients of any function f in the transformed coordinates is given by,

$$\begin{bmatrix} f_\xi \\ f_\eta \\ f_\zeta \end{bmatrix} = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

The reverse transformation is given by,

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} \xi_x & \eta_x & \zeta_x \\ \xi_y & \eta_y & \zeta_y \\ \xi_z & \eta_z & \zeta_z \end{bmatrix} \begin{bmatrix} f_\xi \\ f_\eta \\ f_\zeta \end{bmatrix}$$

For the present 3D BL transformation,

$$\xi = x \;\; ; \;\; \eta = y$$

$$\zeta = \psi(x,y) \int_0^z \rho \, dz$$

$$\psi(x,y) = \frac{\phi}{\rho_e \mu_e s_1} = \sqrt{\frac{u_e}{\rho_e \mu_e s_1}}$$

The following conditions are valid:

$$x_\xi = \xi_x = y_\eta = \eta_y = 1$$

$$y_\xi = \xi_y = x_\eta = \eta_y = 0$$

Since the two square matrices are inverses of each other,

$$\begin{bmatrix} 1 & 0 & z_\xi \\ 0 & 1 & z_\eta \\ 0 & 0 & z_\zeta \end{bmatrix} \begin{bmatrix} 1 & 0 & \zeta_x \\ 0 & 1 & \zeta_y \\ 0 & 0 & \zeta_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From the above, we get the following relationships:

$$\frac{\partial \zeta}{\partial z} = \rho \psi = \frac{1}{(\frac{\partial z}{\partial \zeta})} \tag{92}$$

$$\frac{\partial \zeta}{\partial x} = -\frac{\partial z}{\partial \xi} \cdot \frac{\partial \zeta}{\partial z} \tag{93}$$

$$\frac{\partial \zeta}{\partial y} = -\frac{\partial z}{\partial \eta} \cdot \frac{\partial \zeta}{\partial z} \tag{94}$$

The quantities $\frac{\partial z}{\partial \xi}$ and $\frac{\partial z}{\partial \eta}$ correspond to the boundary layer growth rates in the physical plane in the two stream surface directions and can be obtained from the solution. Finally,

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -z_\xi \zeta_z \\ 0 & 1 & -z_\eta \zeta_z \\ 0 & 0 & \psi\rho \end{bmatrix} \begin{bmatrix} f_\xi \\ f_\eta \\ f_\zeta \end{bmatrix} \tag{95}$$

It may also be noted that $f_z = \sqrt{Re_{ref}}\, f_z$ from equation (7).

For example, $\frac{\partial u^*}{\partial x^*}$, the streamwise velocity gradient in physical units can be obtained as,

$$\frac{\partial u^*}{\partial x^*} = \frac{\partial u}{\partial x}\frac{u^*_\infty}{L^*} = \frac{u^*_\infty}{L^*}\left[\frac{\partial F u_e}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial \zeta}{\partial z}\frac{\partial u}{\partial \zeta}\right]$$

$$= \frac{u^*_\infty}{L^*}\left[\frac{\partial F u_e}{\partial \xi} - \sqrt{Re_{ref}}\, u_e \psi \rho \frac{\partial z}{\partial \xi} F'\right]$$

The righthand side terms are completely known from the solution and $\frac{\partial u^*}{\partial x^*}$ can be calculated.

The boundary layer thickness $t^*_{x,p}$ is defined as the normal distance z^* at which $(u/u_e) = p$. For example, $t^*_{x,0.99}$ is the boundary layer thickness corresponding to $u = 0.99\ u_e$. From the transformation we can write,

$$t^*_{x,p} = \frac{L^*}{\psi\sqrt{Re_{ref}}} \int_0^{\zeta_{x,p}} \frac{1}{\rho}\, d\zeta = L^* \sqrt{\frac{\mu_e s_1}{\rho_e u_e Re_{ref}}} \int_0^{\zeta_{x,p}} \theta\, d\zeta \qquad (96)$$

where $\zeta_{x,p}$ refers to the ζ value at which F is equal to p. Similarly,

$$t^*_{y,p} = \frac{L^*}{\psi\sqrt{Re_{ref}}} \int_0^{\zeta_{y,p}} \frac{1}{\rho}\, d\zeta = L^* \sqrt{\frac{\mu_e s_1}{\rho_e u_e Re_{ref}}} \int_0^{\zeta_{y,p}} \theta\, d\zeta \qquad (97)$$

where $\zeta_{y,p}$ refers to the ζ value at which G is equal to $(p\ v_e/v_r)$.

The displacement thickness in x direction is defined as,

$$\delta^*_x = \int_0^{\infty} \left(1 - \frac{\rho^* u^*}{\rho^*_e u^*_e} \right) dz^*$$

$$= \int_0^{\infty} \left(1 - \frac{F}{\theta} \right) \frac{\partial z^*}{\partial \zeta}\, d\zeta$$

$$= L^* \left[\frac{\mu_e s_1}{\rho_e u_e Re_{ref}} \right]^{1/2} \int_0^{\infty} (\theta - F)\, d\zeta \qquad (98)$$

The momentum thickness in the x direction is defined as,

$$\theta^*_x = \int_0^{\infty} \frac{\rho u}{\rho_e u_e} \left(1 - \frac{u}{u_e} \right) dz^*$$

$$= L^* \left[\frac{\mu_e s_1}{\rho_e u_e Re_{ref}} \right]^{1/2} \int_0^{\infty} F\,(1 - F)\, d\zeta \qquad (99)$$

Similar expressions can be written for δ^*_y and θ^*_y.

The following transformations can be obtained from the basic definitions.

$$u^* = F\, u_e\, u^*_\infty \qquad (100)$$

$$v^* = G\, v_r\, u_\infty^* \tag{101}$$

$$T^* = \theta\, T_e\, T_{ref}^* \tag{102}$$

$$q^* = u_\infty^* \left[F^2 u_e^2 + G^2 v_r^2 + 2 u_e v_r F G\, g_{12} / (h_1\, h_2) \right]^{1/2} \tag{103}$$

$$\rho^* = \theta\, \rho_e\, \rho_\infty^* \tag{104}$$

$$\mu^* = \theta\, l\, \mu_e\, \mu_{ref}^* \tag{105}$$

$$z^* = L^* \sqrt{\frac{\mu_e s_1}{\rho_e u_e Re_{ref}}} \int_0^\zeta \theta\, d\zeta \tag{106}$$

The skin friction coefficient $C_{f,x-\infty}$ is given by,

$$C_{f,x-\infty} = \frac{2\,\tau_w^*}{\rho_\infty^{*2} u_\infty^{*2}} = \frac{2}{\rho_\infty^* u_\infty^{*2}}\, \mu_w^* \frac{\partial u^*}{\partial z^*}\Big|_w$$

In terms of computed quantities, this can be written as,

$$\mu_w^* = l_{1,w}\, \theta_w\, \mu_e\, \mu_{ref}^*$$

$$\frac{\partial u^*}{\partial z^*}_w = \frac{u_\infty^* u_e \sqrt{Re_{ref}}}{L^*} \frac{\partial F}{\partial z}\Big|_w = \frac{u_\infty^*}{L^*}\, u_e \sqrt{Re_{ref}}\, \psi\, \rho_w\, F'_w$$

$$\text{and hence,} \quad C_{f,x-\infty} = \frac{2}{\sqrt{Re_{ref}}}\, \psi\, u_e\, \rho_e\, \mu_e\, l_{1,w}\, F'_w \tag{107}$$

The skin friction coefficient $C_{f,x-e}$ based on the edge conditions is given by,

$$C_{f,x-e} = \frac{2\,\tau_w^*}{\rho_e^{*2} u_e^{*2}} = \frac{C_{f,x-\infty}}{\rho_e u_e^2} \tag{108}$$

The skin friction coefficients in the y direction are given by,

$$C_{f,y-\infty} = \frac{2}{\sqrt{Re_{ref}}}\, \psi\, v_r\, \rho_e\, \mu_e\, l_{2,w}\, G'_w \tag{109}$$

$$\text{and,} \quad C_{f,y-e} = \frac{C_{f,y-\infty}}{\rho_e u_e^2} \tag{110}$$

The boundary layer normal velocity w^* is of interest in flows where mass injection or suction is employed at the wall. In such cases, w_w the value of normal velocity at the wall is to be specified as a boundary condition, which in turn has to be calculated from w_w^*, the physical suction or injection velocity. From equation (33) we get

$$\bar{w} = \left[w - \frac{s_1}{h_1} F \frac{\partial \zeta}{\partial x} - \frac{s_1}{h_2} G \frac{v_r}{u_e} \frac{\partial \zeta}{\partial y} \right] \frac{u_e}{s_1} \frac{\partial \bar{z}}{\partial \zeta}$$

Substituting for $\bar{w}$, $\frac{\partial \zeta}{\partial x}$ and $\frac{\partial \zeta}{\partial y}$, we get,

$$w^* = \frac{u_\infty^*}{\sqrt{Re_{ref}}} u_e \left[\frac{w}{s_1} \frac{\partial \bar{z}}{\partial \zeta} + \frac{F}{h_1} \frac{\partial \bar{z}}{\partial \xi} + G \frac{v_r}{h_2 u_e} \frac{\partial \bar{z}}{\partial \eta} \right] \qquad (111)$$

The normal velocity at the wall is given by,

$$w_w^* = \frac{u_\infty^* u_e}{\sqrt{Re_{ref}}} \frac{w_w}{s_1} \frac{1}{\psi \rho_w} = \frac{u_\infty^*}{\sqrt{Re_{ref}}} \sqrt{\frac{\rho_e \mu_e u_e}{s_1}} \left(\frac{w}{\rho} \right)_w \qquad (112)$$

The heat flux at the wall is given by

$$\dot{q}_w^* = - k_w^* \frac{\partial T^*}{\partial z^*} = - \frac{\mu_w^* c_p^*}{\sigma} T_{ref}^* \frac{\partial T}{\partial \zeta}\Big|_w \frac{\partial \zeta}{\partial z^*}\Big|_w$$

Substitution and simplification gives,

$$\dot{q}_w^* = - \frac{\mu_{ref}^* u_\infty^{*2}}{L^*} \frac{\gamma}{\gamma - 1} T_e I_{3_w} \theta'_w \rho_e \mu_e Re_{ref}^{1/2} \psi \qquad (113)$$

A positive heat flux indicates heat flow in the positive z direction. The quantities θ'_w and H'_w are related by the expression (see also eq. 13),

$$\theta'_w = \frac{H'_w}{T_e \gamma M_\infty^2}$$

7. DISCRETIZATION OF EQUATIONS

The two transformed 3D BL momentum equations (equations (40) and (44)) are:

$$\frac{\partial}{\partial\zeta}(l_1\frac{\partial F}{\partial\zeta}-wF) = B_1\frac{\partial F^2}{\partial\xi} + B_2\frac{\partial FG}{\partial\eta} + B_3F^2 + B_4FG + B_5G^2 + B_6\theta$$

$$\frac{\partial}{\partial\zeta}(l_2\frac{\partial G}{\partial\zeta}-wG) = C_1\frac{\partial FG}{\partial\xi} + C_2\frac{\partial G^2}{\partial\eta} + C_3F^2 + C_4FG + C_5G^2 + C_6\theta$$

To represent these equations in a vectorial form, define a vector $\vec{Q}$ as,

$$\vec{Q} \equiv \begin{bmatrix} l_1\frac{\partial F}{\partial\zeta}-wF \\ l_2\frac{\partial G}{\partial\zeta}-wG \\ F \\ G \end{bmatrix} = \begin{bmatrix} l_1L-wF \\ l_2M-wG \\ F \\ G \end{bmatrix} \tag{114}$$

The normal derivative of the vector can be obtained from the momentum equations. The first two elements of $\vec{Q}'$ thus correspond to the RHS of the momentum equations.

$$\vec{Q}' = \begin{bmatrix} B_1\frac{\partial F^2}{\partial\xi}+B_2\frac{\partial FG}{\partial\eta}+B_3F^2+B_4FG+B_5G^2+B_6\theta \\ C_1\frac{\partial FG}{\partial\xi}+C_2\frac{\partial G^2}{\partial\eta}+C_3F^2+C_4FG+C_5G^2+C_6\theta \\ L \\ M \end{bmatrix} \tag{115}$$

$\vec{Q}''$ can be obtained by differentiation as:

$$\vec{Q}'' = \begin{bmatrix} 2B_1\frac{\partial}{\partial\xi}FL+B_2\frac{\partial}{\partial\eta}(FM+GL)+2B_3FL+B_4(FM+GL)+2B_5GM+B_6\theta' \\ C_1\frac{\partial}{\partial\xi}(FM+GL)+2C_2\frac{\partial}{\partial\eta}GM+2C_3FL+C_4(FM+GL)+2C_5GM+C_6\theta' \\ \frac{1}{l_1}[\frac{1}{2}B_1\frac{\partial}{\partial\xi}F^2+B_2G\frac{\partial}{\partial\eta}F + \tilde{B}_3F^2 + \tilde{B}_4FG+B_5G^2+B_6\theta+L(w-l_1')] \\ \frac{1}{l_2}[C_1F\frac{\partial}{\partial\xi}G+\frac{1}{2}C_2\frac{\partial}{\partial\eta}G^2 + C_3F^2 + \tilde{C}_4FG\tilde{C}_5G^2+B_6\theta+M(w-l_2')] \end{bmatrix} \tag{116}$$

The first two elements of $\vec{Q}''$ are obtained by taking the ζ derivative of the first two elements of $\vec{Q}'$. The last two elements of $\vec{Q}''$ are obtained by expanding the LHS of the momentum equations and rearranging to get expressions for L′ and M′.

For example, from the x momentum equation,

$$I_1 L' + L I_1' - w L - F w' = RHS$$

$$L' = \frac{L}{I_1} (w - I_1') + \frac{1}{I_1} (F w' + RHS)$$

The term F w′ is then substituted by the continuity equation (36) and simplified.

Discretization in ζ Direction

The indices i, j and k are used to denote grid locations in the ξ, η and ζ solution space. The solution of the equations is obtained implicitly in the ζ direction and explicitly in the ξ and η directions. Hence, the discretization procedure for the ζ direction can be discussed independently of the other two directions. Assume for the present that for any function $\overline{f}$, the partial derivative $\frac{\partial \overline{f}}{\partial \xi}$ can be written as $(a_1 \overline{f}_{i,j} + \{a\overline{f}\})$, where $\{a\overline{f}\}$ is a symbolic notation for the weighted sum of the function values at other known locations. In a 3-point upwind differenced formulation, $\{a\overline{f}\} = a_2\overline{f}_{i-1,j} + a_3\overline{f}_{i-2,j}$. The coefficients a_1, a_2 and a_3 etc. depend on the grid distribution. Similarly, $\frac{\partial \overline{f}}{\partial \eta}$ for the present will be represented as $b_1 \overline{f}_{i,j} + \{b\overline{f}\}$. The details of ξ differencing and η differencing formulations will be dealt with at a later stage. For the present discussion on ζ differencing, we are only interested in the profile at a given (i,j) location.

A finite difference approximation accurate to fourth order in ζ, for uniform or non-uniform spacing is used here. The $O(\Delta\zeta^4)$ procedure allows the use of substantially fewer mesh points normal to the wall boundary than traditional $O(\Delta\zeta^2)$ procedures for a specified level of accuracy. A two-point compact scheme is used which is accurate to $O(\Delta\zeta^4)$ for uniform or non-uniform grids (see also Ref. 7). At each point midway between k and k-1, the following equation can be written ($\Delta\zeta = \zeta_k - \zeta_{k-1}$):

$$\vec{Q}_k - \vec{Q}_{k-1} - \frac{\Delta\zeta}{2} \left[\vec{Q}'_k + \vec{Q}'_{k-1} \right] + \frac{\Delta\zeta^2}{12} \left[\vec{Q}''_k - \vec{Q}''_{k-1} \right] + O(\Delta\zeta^5) = 0 \qquad (117)$$

The vector equation given above consists of 4 equations involving the unknowns (F, G, L, M) at locations k and k-1. If km is used to denote the total number of ζ direction grid points, then the above discretization can be applied to (km-1) pairs of points yielding 4x(km-1) equations in 4xkm unknowns. The additional 4 equations are simply the no-slip boundary conditions F = G = 0 at k=1 (wall) and F = 1 , G = v_e/v_r at k=km (BL edge). Thus, the solution of a profile at a given point (i,j) on the body reduces to the inversion of a system of size 4xkm. It will be shown below that the system is in a block triagonal form with km number of (4x4) block equations.

The equation (117) is first rewritten in the following form for convenience. To fourth order accuracy,

$$\vec{Q}_k - \vec{Q}_{k-1} - c_1\vec{Q}_k' - c_2\vec{Q}_{k-1}' - c_3\vec{Q}_k'' - c_4\vec{Q}_{k-1}'' = 0 \qquad (118)$$

where, $c_1 = c_2 = \Delta\zeta/2$; $c_3 = -\Delta\zeta^2/12$; $c_4 = \Delta\zeta^2/12$

As an example, let us consider the first element of equation (118). The complete expanded form is as below:

$$(l_1L - wF)_k - (l_1L - wF)_{k-1}$$

$$- c_1\left\{B_1\frac{\partial F^2}{\partial\xi} + B_2\frac{\partial FG}{\partial\eta} + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_k$$

$$- c_2\left\{B_1\frac{\partial F^2}{\partial\xi} + B_2\frac{\partial FG}{\partial\eta} + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_{k-1}$$

$$- c_3\left\{2B_1\frac{\partial FL}{\partial\xi} + B_2\frac{\partial}{\partial\eta}(FM+GL) + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_k$$

$$- c_4\left\{2B_1\frac{\partial FL}{\partial\xi} + B_2\frac{\partial}{\partial\eta}(FM+GL) + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_{k-1} = 0$$

Since we are interested only in ζ differencing at present, the $\frac{\partial}{\partial\xi}$ and $\frac{\partial}{\partial\eta}$ derivatives can be replaced by using the symbolic notation. We can also drop the i,j indices for convenience resulting in,

$$(I_1L - wF)_k - (I_1L - wF)_{k-1}$$

$$- c_1\Big\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\Big\}_k$$

$$- c_2\Big\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\Big\}_{k-1}$$

$$- c_3\Big\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\Big\}_k$$

$$- c_4\Big\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\Big\}_{k-1} = 0$$

The above equation in non-linear in the unknowns F, G, L and M. The equation is linearized by using Newton's linearization with successive iteration to a converged solution. If we denote by n, the current iteration number, the non-linear terms at iteration level n can be approximated in terms of quantities at level (n-1). For example, F_n^2 is linearized as,

$$F_n^2 \approx (F_{n-1} + \delta F)^2 \approx F_{n-1}^2 + 2F_{n-1}\delta F$$

$$\text{where, } \delta F = F_n - F_{n-1}$$

δF is the iterative update that is required at each iterative step. As another example, the linearization for FG takes the following form:

$$FG = (F_{n-1}+\delta F)\,(G_{n-1}+\delta G) \approx (FG)_{n-1} + F_{n-1}\delta G + + G_{n-1}\delta F$$

$$\text{where, } \delta G = G_n - G_{n-1}$$

The other non-linear terms in the above equation can be linearized in a similar manner. The complete linearized system (corresponding to the first element of the system of equations given by eq. (118)) is then as given below. We drop the iteration level index (n-1); all the known quantities in the following equation correspond to the (n-1)th iteration level.

$$
\begin{aligned}
&[\, I_1(L+\delta L) - w(F+\delta F)\,]_k - [\, I_1(L+\delta L) - w(F+\delta F)\,]_{k-1} \\
&- c_1 \Big\{ B_1[a_1F^2+2a_1F\delta F+\{aF^2\}] + B_2[b_1FG+b_1F\delta G+b_1G\delta F+\{bFG\}] \\
&\qquad + B_3[F^2+2F\delta F] + B_4[FG+F\delta G+G\delta F] + B_5[G^2+2G\delta G] + B_6\theta \Big\}_k \\
&- c_2 \Big\{ B_1[a_1F^2+2a_1F\delta F+\{aF^2\}] + B_2[b_1FG+b_1F\delta G+b_1G\delta F+\{bFG\}] \\
&\qquad + B_3[F^2+2F\delta F] + B_4[FG+F\delta G+G\delta F] + B_5[G^2+2G\delta G] + B_6\theta \Big\}_{k-1} \\
&- c_3 \Big\{ 2B_1[a_1FL+a_1F\delta L+a_1L\delta F+\{aFL\}] \\
&\quad + B_2[b_1(FM+GL)+b_1(F\delta M+M\delta F+G\delta L+L\delta G)+\{b(FM+GL)\}] + 2B_3FL \\
&\quad + B_4(FM+GL+F\delta M+M\delta F+G\delta L+L\delta G) + 2B_5(GM+G\delta M+M\delta G) + B_6\theta' \Big\}_k \\
&- c_4 \Big\{ 2B_1[a_1FL+a_1F\delta L+a_1L\delta F+\{aFL\}] \\
&\quad + B_2[b_1(FM+GL)+b_1(F\delta M+M\delta F+G\delta L+L\delta G)+\{b(FM+GL)\}] + 2B_3FL \\
&\quad + B_4(FM+GL+F\delta M+M\delta F+G\delta L+L\delta G) + 2B_5(GM+G\delta M+M\delta G) + B_6\theta' \Big\}_{k-1} = 0
\end{aligned}
$$

The unknown quantities in the above equation are,

$$
\vec{S}_{k-1} = \begin{bmatrix} \delta F_{k-1} \\ \delta G_{k-1} \\ \delta L_{k-1} \\ \delta M_{k-1} \end{bmatrix} \quad \text{and,} \quad \vec{S}_k = \begin{bmatrix} \delta F_k \\ \delta G_k \\ \delta L_k \\ \delta M_k \end{bmatrix} \tag{119}
$$

Let us denote the coefficient of δF_{k-1} by a_{11}^k. The first subscript 1 refers to the fact that this coefficient belongs to the first element of equation (118). The second subscript 1 refers to the fact that this coefficient corresponds to the unknown δF. The integers 2, 3, or 4 in the second subscript position refers to it being the coefficient of δG, δL or δM. The superscript k denotes that the coefficient belongs to the discretization at k and a point just below it. From the above equation, we can write down a_{11} as,

$$a^k_{11} = w_{k-1} - c_2[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_{k-1} - c_4[2L(a_1B_1+B_3)+M(b_1B_2+B_4)]_{k-1}$$

Let us denote by b^k_{11}, the coefficient of δF_k. This coefficient can be written as,

$$b^k_{11} = w_k - c_1[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_k - c_3[2L(a_1B_1+B_3)+M(b_1B_2+B_4)]_k$$

All the other coefficients can be similarly derived. Note that 'a' refers to coefficients of $\vec{S}_{k-1}$ and 'b' refers to coefficients of $\vec{S}_k$. The same procedure can be repeated for the other 3 equation elements. The discretization thus yields 16 coefficients of the type $a^k_{l,m}$ and 16 coefficients of the type $b^k_{l,m}$ corresponding to the ζ differencing between k and (k-1). All the terms which are not functions of $\vec{S}_{k-1}$ or $\vec{S}_k$ can be taken to right hand side. The RHS coefficient r^k_1 for the present case is then,

$$r^k_1 = -(i_1\dot{L} - w\dot{F})_k + (i_1\dot{L} - w\dot{F})_{k-1}$$

$$+ c_1\left\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_k$$

$$+ c_2\left\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_{k-1}$$

$$+ c_3\left\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_k$$

$$+ c_4\left\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_{k-1}$$

Similar expressions can be arrived at for r^k_2, r^k_3 and r^k_4. The linearized system thus reduces to the form,

$$\{a\}^k_{l,m}\vec{S}_{k-1} + \{b\}^k_{l,m}\vec{S}_k = [r]^k_l \qquad (120)$$

The 4x4 matrices of $\{a\}_{l,m}$, $\{b\}_{l,m}$ and the 4x1 vector $[r]_l$ are listed in Table 8.

Table 8. Coefficients of the Momentum Equations

Discretization at $(k+\frac{1}{2})$ with $\Delta\zeta = \zeta_k - \zeta_{k-1}$

$c_1 = \Delta\zeta/2$, $c_2 = \Delta\zeta/2$, $c_3 = -\Delta\zeta^2/12$, $c_4 = \Delta\zeta^2/12$

All quantities evaluated at iteration level (n-1)

$$a_{11} = w_{k-1} - c_2[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_{k-1} - c_4[2L(a_1B_1+B_3)+M(b_1B_2+B_4)]_{k-1}$$

$$a_{12} = -c_2[F(b_1B_2+B_4)+G(2B_5)]_{k-1} - c_4[L(b_1B_2+B_4)+M(2B_5)]_{k-1}$$

$$a_{13} = -(l_1)_{k-1} - c_4[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_{k-1}$$

$$a_{14} = -c_4[F(b_1B_2+B_4)+G(2B_5)]_{k-1}$$

$$a_{21} = -c_2[F(2C_3)+G(a_1C_1+C_4)]_{k-1} - c_4[L(2C_3)+M(a_1C_1+C_4)]_{k-1}$$

$$a_{22} = w_{k-1} - c_2[F(a_1C_1+C_4)+2G(b_1C_2+C_5)]_{k-1} - c_4[L(a_1C_1+C_4)+2M(b_1C_2+C_5)]_{k-1}$$

$$a_{23} = -c_4[2F(C_3)+G(a_1C_1+C_4)]_{k-1}$$

$$a_{24} = -(l_2)_{k-1} - c_4[F(a_1C_1+C_4)+2G(b_1C_2+C_5)]_{k-1}$$

$$a_{31} = -1 - c_4/(l_1)_{k-1}[F(a_1B_1+2\tilde{B}_3)+G(b_1B_2+\tilde{B}_4)]_{k-1}$$

$$a_{32} = -c_4/(l_1)_{k-1}[F(\tilde{B}_4)+G(2B_5)+B_2(b_1F+\{bF\})]_{k-1}$$

$$a_{33} = -c_2 - c_4/(l_1)_{k-1}[w-l'_1]_{k-1}$$

$$a_{34} = 0$$

$$a_{41} = -c_4/(l_2)_{k-1}[F(2C_3)+G(\tilde{C}_4)+C_1(a_1G+\{aG\})]_{k-1}$$

$$a_{42} = -1 - c_4/(l_2)_{k-1}[F(a_1C_1+\tilde{C}_4)+G(b_1C_2+2\tilde{C}_5)]_{k-1}$$

$$a_{43} = 0$$

$$a_{44} = -c_2 - c_4/(l_2)_{k-1}[w-l'_2]_{k-1}$$

$$b_{11} = w_k - c_1[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_k - c_3[2L(a_1B_1+B_3)+M(b_1B_2+B_4)]_k$$

$$b_{12} = -c_1[F(b_1B_2+B_4)+G(2B_5)]_k - c_3[L(b_1B_2+B_4)+M(2B_5)]_k$$

$$b_{13} = -(l_1)_k - c_3[2F(a_1B_1+B_3)+G(b_1B_2+B_4)]_k$$

$$b_{14} = -c_3[F(b_1B_2+B_4)+G(2B_5)]_k$$

$$b_{21} = -c_1[F(2C_3)+G(a_1C_1+C_4)]_k - c_3[L(2C_3)+M(a_1C_1+C_4)]_k$$

$$b_{22} = w_k - c_1[F(a_1C_1+C_4)+2G(b_1C_2+C_5)]_k - c_3[L(a_1C_1+C_4)+2M(b_1C_2+C_5)]_k$$

$$b_{23} = -c_3[2F(C_3)+G(a_1C_1+C_4)]_k$$

$$b_{24} = -(l_2)_k - c_3[F(a_1C_1+C_4)+2G(b_1C_2+C_5)]_k$$

$$b_{31} = -1 - c_3/(l_1)_k[F(a_1B_1+2\tilde{B}_3)+G(b_1B_2+\tilde{B}_4)]_k$$

$$b_{32} = -c_3/(l_1)_k[F(\tilde{B}_4)+G(2B_5)+B_2(b_1F+\{bF\})]_k$$

$$b_{33} = -c_1 - c_3/(l_1)_k[w-l'_1]_k$$

$$b_{34} = 0$$

$$b_{41} = -c_3/(l_2)_k[F(2C_3)+G(\tilde{C}_4)+C_1(a_1G+\{aG\})]_k$$

$$b_{42} = -1 - c_3/(l_2)_k[F(a_1C_1+\tilde{C}_4)+G(b_1C_2+2\tilde{C}_5)]_k$$

$$b_{43} = 0$$

$$b_{44} = -c_1 - c_3/(l_2)_k[w-l'_2]_k$$

$$r_1^k = (l_1L - wF)_k + (l_1L - wF)_{k-1}$$
$$+ c_1\left\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_k$$
$$+ c_2\left\{B_1[a_1F^2+\{aF^2\}] + B_2[b_1FG+\{bFG\}] + B_3F^2 + B_4FG + B_5G^2 + B_6\theta\right\}_{k-1}$$
$$+ c_3\left\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_k$$
$$+ c_4\left\{2B_1[a_1FL+\{aFL\}] + B_2[b_1(FM+GL)+\{b(FM+GL)\}] + 2B_3FL + B_4(FM+GL) + 2B_5GM + B_6\theta'\right\}_{k-1}$$

$$r_2^k = -(l_2M - wG)_k + (l_2M - wG)_{k-1}$$
$$+ c_1\left\{C_1[a_1FG+\{aFG\}] + C_2[b_1G^2+\{bG^2\}] + C_3F^2 + C_4FG + C_5G^2 + C_6\theta\right\}_k$$
$$+ c_2\left\{C_1[a_1FG+\{aFG\}] + C_2[b_1G^2+\{bG^2\}] + C_3F^2 + C_4FG + C_5G^2 + C_6\theta\right\}_{k-1}$$

$$+ c_3\Big\{C_1[a_1(FM+GL)+\{a(FM+GL)\}] + C_2[b_1G^2+\{bG^2\}] + C_3F^2 + C_4(FM+GL) + 2C_5G^2 + B_6\theta'\Big\}_k$$

$$+ c_4\Big\{C_1[a_1(FM+GL)+\{a(FM+GL)\}] + C_2[b_1G^2+\{bG^2\}] + C_3F^2 + C_4(FM+GL) + 2C_5G^2 + B_6\theta'\Big\}_{k-1}$$

$$r_3 = -F_k+F_{k-1}+c_1L_{k-1}+c_2L_k$$

$$+c_3/(l_1)_k\Big\{\tfrac{1}{2}B_1[a_1F^2+\{aF^2\}]+B_2G[b_1F+\{bF\}]+\tilde{B}_3F^2+\tilde{B}_4FG+B_5G^2+B_6\theta+L(w-l_1')\Big\}_k$$

$$+c_4/(l_1)_{k-1}\Big\{\tfrac{1}{2}B_1[a_1F^2+\{aF^2\}]+B_2G[b_1F+\{bF\}]+\tilde{B}_3F^2+\tilde{B}_4FG+B_5G^2+B_6\theta+L(w-l_1')\Big\}_{k-1}$$

$$r_4 = -G_k+G_{k-1}+c_1M_{k-1}+c_2M_k$$

$$+c_3/(l_2)_k\Big\{\tfrac{1}{2}C_2[a_1G^2+\{aG^2\}]+C_1F[b_1G+\{bG\}]+C_3F^2+\tilde{C}_4FG+\tilde{C}_5G^2+C_6\theta+M(w-l_2')\Big\}_k$$

$$+c_4/(l_2)_{k-1}\Big\{\tfrac{1}{2}C_2[a_1G^2+\{aG^2\}]+C_1F[b_1G+\{bG\}]+C_3F^2+\tilde{C}_4FG+\tilde{C}_5G^2+C_6\theta+M(w-l_2')\Big\}_{k-1}$$

Discretization of the Energy Equation

To simplify the solution algorithm, the energy equation is solved decoupled from the momentum equations. The discretization of the energy equation in the ζ direction can be done in a manner identical to the momentum equations. The transformed energy equation (equation (47)) is given below:

$$\frac{\partial}{\partial\zeta}(I_3\mathbf{I}-wH) = D_1\frac{\partial FH}{\partial\xi} + D_2\frac{\partial GH}{\partial\eta} + D_3FH + D_4GH + D_5 \qquad (121)$$

The $\vec{Q}$, $\vec{Q}'$ and $\vec{Q}''$ vectors in this case are as given below:

$$\vec{Q} = \begin{bmatrix} I_3\mathbf{I}-wH \\ H \end{bmatrix} \; ; \; \vec{Q}' = \begin{bmatrix} D_1\frac{\partial FH}{\partial\xi}+D_2\frac{\partial GH}{\partial\eta}+D_3FH+D_4GH+D_5 \\ \mathbf{I} \end{bmatrix}$$

$$\vec{Q}'' = \begin{bmatrix} D_1\frac{\partial}{\partial\xi}(F\mathbf{I}+HL)+D_2\frac{\partial}{\partial\eta}(G\mathbf{I}+HM)+D_3(F\mathbf{I}+HL)+D_4(G\mathbf{I}+HM)+D'_5 \\ \frac{1}{I_3}[D_1F\frac{\partial H}{\partial\xi}+D_2G\frac{\partial H}{\partial\eta}+D_5+\mathbf{I}(w-I'_3)] \end{bmatrix}$$

Application of the ζ differencing formula, equation (118) to the above system yields two equations in H and **I**. The coefficients of this system consists of F,G,L and M which are known at iteration level n, since they are solved decoupled. The equations are therefore linear in H and **I**. Iteration is however required since F,G,L and M are not the final converged values. As an example, the first equation of the discretized system is as follows:

$$(I_3\mathbf{I} - wH)_k - (I_3\mathbf{I} - wH)_{k-1} -$$

$$- c_1\Big\{D_1[a_1FH+\{aFH\}] + D_2[b_1GH+\{bGH\}] + D_3FH + D_4GH + D_5\Big\}_k$$

$$- c_2\Big\{D_1[a_1FH+\{aFH\}] + D_2[b_1GH+\{bGH\}] + D_3FH + D_4GH + D_5\Big\}_{k-1}$$

$$- c_3\Big\{D_1[a_1(F\mathbf{I}+HL)+\{aF(F\mathbf{I}+HL)\}]+D_2[b_1(G\mathbf{I}+HM)+\{b(G\mathbf{I}+HM)\}]$$

$$+ D_3(F\mathbf{I}+HL)+D_4(G\mathbf{I}+HM)+D_5'\Big\}_k$$

$$- c_4\Big\{D_1[a_1(F\mathbf{I}+HL)+\{aF(F\mathbf{I}+HL)\}]+D_2[b_1(G\mathbf{I}+HM)+\{b(G\mathbf{I}+HM)\}]$$

$$+ D_3(FI+HL)+D_4(GI+HM)+D_5' \Big\}_{k-1}$$

Here we solve for H and I at each point k, rather than the solution change at each iteration, δH and δI (note that the system for the momentum equations was set up to solve for the δ values). This is because there is no need for linearization in this case, and hence solving for H and I directly yields a simpler system. Defining the unknown vectors as,

$$\vec{S}_{k-1} = \begin{bmatrix} H_{k-1} \\ I_{k-1} \end{bmatrix} \quad \text{and,} \quad \vec{S}_k = \begin{bmatrix} H_k \\ I_k \end{bmatrix}$$

one can now write down the coefficients corresponding to the system,

$$\{a\}^k_{l,m}\vec{S}_{k-1} + \{b\}^k_{l,m}\vec{S}_k = [r]^k_l$$

For example, a^k_{12}, the coefficient of I_{k-1} from the first equation is given by,

$$a_{12} = -(l_3)_{k-1} - c_4[F(a_1D_1+D_3)+G(b_1D_2+D_4)]_{k-1}$$

The coefficients $\{a\}_{l,m}$, $\{b\}_{l,m}$ and $[r]^k_l$ are listed in Table 9.

Boundary Conditions for Momentum Equations

The ζ differencing presented above is a two-point fourth-order accurate scheme. However, the application of the ζ direction boundary conditions results in the discretized system being block tri-diagonal. The details of this feature are presented below.

The discretization at each mid-point of a solution profile at a fixed x and y location yields 4x(km-1) equations in the 4xkm unknowns, $\{\vec{S}_k\}_{k=1,km}$. The remaining 4 equations to close the system are obtained from the constraints at the wall and BL edge. These correspond to $\delta F_1 = 0$, $\delta G_1 = 0$ at the wall and $\delta F_{km} = 0$, $\delta G_{km} = 0$, assuming that the initial profile used is consistent with the no-slip conditions of $F_1 = 0$, $G_1 = 0$ and the edge conditions of $F_{km} = 1$ and $G_{km} = v_e/v_r$. An alternate boundary condition at BL edge is $\delta L_{km} = 0$, $\delta M_{km} = 0$ corresponding to the edge gradients being equal to zero. However, since the initial profile need not satisfy this condition exactly to fourth-order accuracy, this alternative is not preferred. The boundary conditions are thus easy to implement and do not compromise the fourth-order accuracy of the method.

Table 9. Coefficients of the Energy Equation

Discretization at $(k+\frac{1}{2})$ with $\Delta\zeta = \zeta_k - \zeta_{k-1}$

$c_1 = \Delta\zeta/2$, $c_2 = \Delta\zeta/2$, $c_3 = -\Delta\zeta^2/12$, $c_4 = \Delta\zeta^2/12$

F, G, L, M are from the solution of momentum equations at iteration level n

Coefficients are corresponding to the system,

$$\{a\}^k_{l,m}\vec{S}_{k-1} + \{b\}^k_{l,m}\vec{S}_k = [r]^k_l$$

where, $\vec{S}_{k-1} = \begin{bmatrix} H_{k-1} \\ I_{k-1} \end{bmatrix}$ and, $\vec{S}_k = \begin{bmatrix} H_k \\ I_k \end{bmatrix}$

$$a_{11} = w_{k-1} - c_2[F(a_1D_1+D_3)+G(b_1D_2+D_4)]_{k-1} - c_4[L(a_1D_1+D_3)+M(b_1D_2+D_4)]_{k-1}$$

$$a_{12} = -(I_3)_{k-1} - c_4[F(a_1D_1+D_3)+G(b_1D_2+D_4)]_{k-1}$$

$$a_{21} = -1 - c_4/(I_3)_{k-1}[Fa_1D_1+Gb_1D_2]_{k-1}$$

$$a_{22} = -c_2 - c_4/(I_3)_{k-1}[w-I'_3]_{k-1}$$

$$b_{11} = w_k - c_1[F(a_1D_1+D_3)+G(b_1D_2+D_4)]_k - c_3[L(a_1D_1+D_3)+M(b_1D_2+D_4)]_k$$

$$b_{12} = -(I_3)_k - c_3[F(a_1D_1+D_3)+G(b_1D_2+D_4)]_k$$

$$b_{21} = -1 - c_3/(I_3)_k[Fa_1D_1+Gb_1D_2]_k$$

$$b_{22} = -c_1 - c_3/(I_3)_k[w-I'_3]_k$$

$$r_1 = c_1[D_1\{aFH\}+D_2\{bGH\}+D_5]_k + c_2[D_1\{aFH\}+D_2\{bGH\}+D_5]_{k-1} +$$
$$c_3[D_1\{a(FI+GL)\}+D_2\{b(GI+HM)\}+D_5]_k +$$
$$c_4[D_1\{a(FI+GL)\}+D_2\{b(GI+HM)\}+D_5]_{k-1}$$

$$r_2 = c_3/(I_3)_k[D_1F\{aH\}+D_2G\{bH\}+D_5]_k + c_4/(I_3)_{k-1}[D_1F\{aH\}+D_2G\{bH\}+D_5]_{k-1}$$

At the wall, the boundary conditions of $\delta F_1 = 0$, $\delta G_1 = 0$ with δL_1, δM_1 as the unknowns shifts the discretized equations down by two rows. A block tri-diagonal system is produced as a result. Fig. 3 shows the structure of the block-triagonal system near the wall boundary. The sub-diagonal, diagonal and super-diagonal blocks α_k, β_k and γ_k are given by,

$$\alpha_k = \begin{bmatrix} a_{11}^k & a_{12}^k & a_{13}^k & a_{12}^k \\ a_{21}^k & a_{22}^k & a_{23}^k & a_{24}^k \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\beta_k = \begin{bmatrix} b_{11}^k & b_{12}^k & b_{13}^k & b_{14}^k \\ b_{21}^k & b_{22}^k & b_{23}^k & b_{24}^k \\ a_{31}^{k+1} & a_{32}^{k+1} & a_{33}^{k+1} & a_{34}^{k+1} \\ a_{41}^{k+1} & a_{42}^{k+1} & a_{43}^{k+1} & a_{44}^{k+1} \end{bmatrix}$$

$$\gamma_k = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_{31}^{k+1} & b_{32}^{k+1} & b_{33}^{k+1} & b_{34}^{k+1} \\ b_{41}^{k+1} & b_{42}^{k+1} & b_{43}^{k+1} & b_{44}^{k+1} \end{bmatrix}$$

The system can thus be represented as:

$$\alpha_k \vec{S}_{k-1} + \beta_k \vec{S}_k + \gamma_k \vec{S}_{k+1} = \delta_k$$

Note that β_k involves quantities at k, k-1 and k+1; α_k involves quantities at k-1 and k; γ_k involves quantities at k and k+1. The RHS vector of the block-triagonal system is of the form,

$$\delta_k = \left[r_1^k \, , \, r_2^k \, , \, r_3^{k+1} \, , \, r_4^{k+1} \right]^T$$

The boundary conditions at the wall and BL edge modify the diagonal block and the RHS vector resulting in,

$$\beta_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a_{31}^1 & a_{32}^1 & a_{33}^1 & a_{34}^1 \\ a_{41}^2 & a_{42}^2 & a_{43}^2 & a_{44}^2 \end{bmatrix}$$

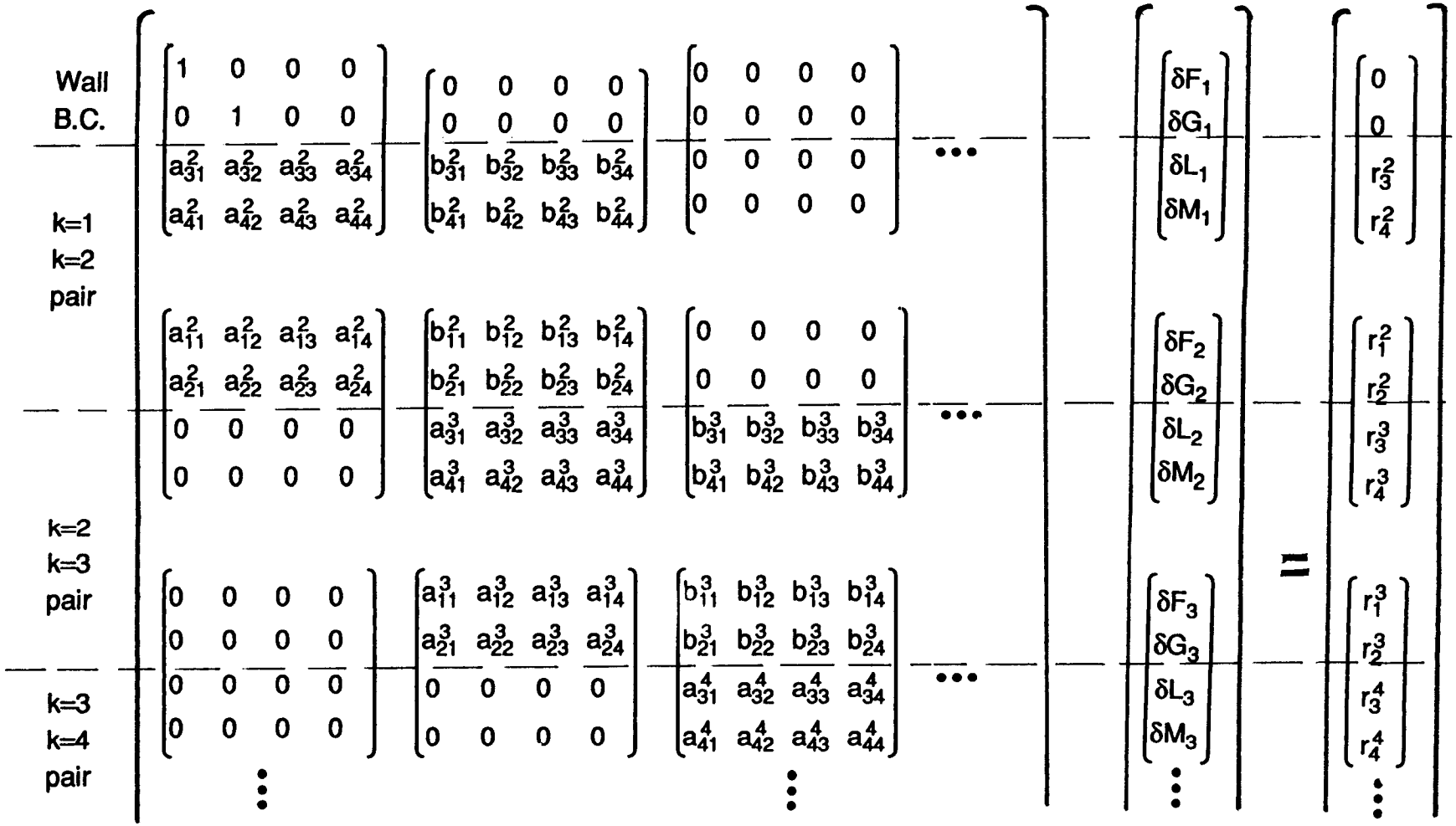


Fig.3 Block Tri-diagonal Structure of the Discretized System

$$\beta_{km} = \begin{bmatrix} b_{11}^{km} & b_{12}^{km} & b_{13}^{km} & b_{14}^{km} \\ b_{21}^{km} & b_{22}^{km} & b_{23}^{km} & b_{24}^{km} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\delta_1 = \begin{bmatrix} 0, 0, r_3^2, r_4^2 \end{bmatrix}^T \; ; \; \delta_{km} = \begin{bmatrix} r_1^{km}, r_2^{km}, 0, 0 \end{bmatrix}^T$$

Boundary Conditions for Energy Equation

Two types of boundary conditions at the wall can be specified. If the wall is insulated or adiabatic, zero heat flux condition at the wall is prescribed by setting $I_w = 0$. If the wall temperature or heat flux is specified, then the corresponding H_w or I_w is calculated from equation (13) or equation (113) and used in the RHS vector δ_1. If I_w is specified, the β_1 and δ_1 blocks at the wall become,

$$\beta_1 = \begin{bmatrix} 0 & 1 \\ a_{21}^2 & a_{22}^2 \end{bmatrix} \quad ; \quad \delta_1 = \begin{bmatrix} I_w \\ r^2{}_2 \end{bmatrix}$$

If total enthalpy or temperature distribution is specified, the corresponding β_1 and δ_1 would be,

$$\beta_1 = \begin{bmatrix} 1 & 0 \\ a_{21}^2 & a_{22}^2 \end{bmatrix} \quad ; \quad \delta_1 = \begin{bmatrix} H_w \\ r^2{}_2 \end{bmatrix}$$

The BL edge boundary condition of $H_{km} = H_e$ (see equation (15)) modifies β_{km} and δ_{km} blocks as below:

$$\beta_{km} = \begin{bmatrix} b_{11}^{km} & b_{12}^{km} \\ 1 & 0 \end{bmatrix} \quad ; \quad \delta_{km} = \begin{bmatrix} r^{km}{}_1 \\ H_e \end{bmatrix}$$

Discretization of the Continuity Equation

The continuity equation is solved explicitly assuming that F, G, L and M values are known. The equation can be integrated starting from the wall using the fourth order scheme. The transformed normal velocity w_w at the wall is required to start the integration. This can be calculated using equation (112) from a specified mass injection or suction rate at the wall, $(\rho^* w^*)_w$.

The integration is performed using the following formula:

$$w_k = w_{k-1} + \frac{\Delta\zeta}{2}(w'_k+w'_{k-1}) - \frac{\Delta\zeta^2}{12}(w''_k-w''_{k-1}) \quad , k=2,3,...km \qquad (122)$$

$$\text{where, } w' = \tilde{A}_1\frac{\partial F}{\partial \xi} + \tilde{A}_2 F + \tilde{A}_3\frac{\partial G}{\partial \eta} + \tilde{A}_4 G$$

$$\text{and } w'' = \tilde{A}_1\frac{\partial L}{\partial \xi} + \tilde{A}_2 L + \tilde{A}_3\frac{\partial M}{\partial \eta} + \tilde{A}_4 M$$

Normal Grid Exponential Stretching

The ζ transformation takes into account the laminar boundary layer growth. Usually, ζ_e=8 is sufficient. However, for turbulent flow, with the present transformation, the boundary layer edge may extend as far as ζ_e=100. A stretched grid may be of advantage in such a case.

For incompressible zero pressure gradient flow, a uniform distribution of points in ζ produces a uniform distribution in z^* with correct scaling (proportional to $\sqrt{x}$) for boundary layer growth. For compressible laminar flow with pressure gradient, the distribution obtained in the physical plane will be different.

In either of the above two situations, if a different normal grid distribution is desired, this can be provided by using an exponential stretching parameter, k_e. Since the normal differencing is basically a 2-point scheme, fourth-order accuracy is obtained for stretched grids as well. The stretching parameter is defined as,

$$\zeta = \zeta_e \cdot \left[\frac{k_e^{k-1}-1}{k_e^{km-1}-1}\right] \qquad (123)$$

As an example, the table given below gives the minimum and maximum grid spacings resulting from different values of the stretching parameter k_e. The values of ζ_e and km are assumed to be 8.0 and 41 respectively. The grid distances next to the wall and the boundary layer edge are tabulated.

k_e	$\Delta\zeta$ (k=2)	$\Delta\zeta$ (k=km)
1.0	0.2	0.2
1.005	0.1812	0.2201
1.01	0.1636	0.2412
1.02	0.1324	0.2867
1.05	0.0662	0.4440
1.10	0.0181	0.7437

Discretization in the ξ Direction

The BL solution is obtained by a marching scheme in the ξ direction. Hence all $\partial / \partial\xi$ quantities at a given location (i,j) can be written explicitly in terms of quantities at (i,j) and known values at (i-1), (i-2). For second order accuracy, a three-point upwind differencing formula is used. For example, the streamwise gradient of F is approximated by,

$$\left.\frac{\partial F}{\partial \xi}\right|_{i,j} = a_1 F_{i,j} + a_2 F_{i-1,j} + a_3 F_{i-2,j} \qquad (124)$$

The coefficients a_1, a_2 and a_3 depend on the grid distances,

$$\Delta\xi_i = \xi_{i,j} - \xi_{i-1,j}$$

$$\Delta\xi_{i-1} = \xi_{i-1,j} - \xi_{i-2,j}$$

They are given by,

$$\begin{aligned} a_1 &= (\Delta\xi_{i,j}^2 - \Delta\xi_{i-1,j}^2) / \Delta \\ a_2 &= -\Delta\xi_{i,j}^2 / \Delta \\ a_3 &= \Delta\xi_{i-1,j}^2 / \Delta \\ \Delta &= \Delta\xi_{i,j}\, \Delta\xi_{i-1,j}\, (\Delta\xi_{i,j} + \Delta\xi_{i-1,j}) \end{aligned} \qquad (125)$$

When i=2, the first order formula is used with,

$$a_1 = 1 / \Delta\xi_2 = 1 / (\xi_2 - \xi_1) \;;\; a_2 = -\Delta\xi_2 \;;\; a_3 = 0$$

A blending function ω is provided for transitioning from the first-order scheme to the second order scheme. For example, if the differencing is to become second order at $i \geq i_2$, then,

$$\omega = 0 \text{ for } i = 2$$

$$\omega = 1 \text{ for } i \geq i_2$$

$$\omega = (i-2)/(i_2-2) \text{ for } i_2 \geq i \geq 2$$

$$\left.\frac{\partial F}{\partial \xi}\right|_{i,j} = \omega \text{ [second order formula]} + (1-\omega) \text{ [first order formula]}$$

Referring to our earlier short notation,

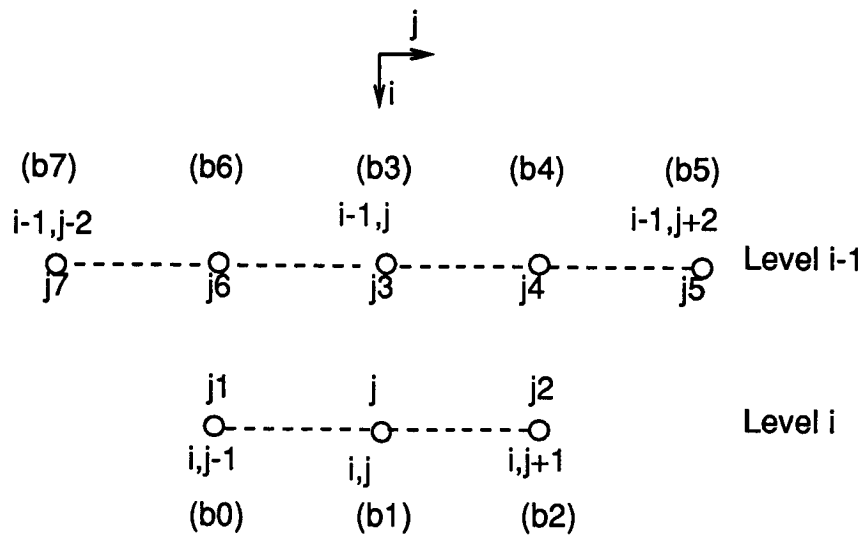
$$\{aF\} = a_2 F_{i-1,j} + a_3 F_{i-2,j}$$

Discretization in the η Direction

In order to provide flexibility in formulation, the differencing in η direction is designed to include a number of points surrounding a given location (i,j). The points included in the finite difference formula are,

At level i: j, j-1, j+1 (denoted as points j, j1 and j2)

At level i-1: j, j+1, j+2, j-1, j-2 (denoted as points j3, j4, j5, j6 and j7)



The finite difference formula is written as,

$$\left.\frac{\partial F}{\partial \eta}\right|_{i,j} = b_0 F_j + b_1 F_{j1} + b_2 F_{j2} + b_3 F_{j3} + b_4 F_{j4} + b_5 F_{j5} + b_6 F_{j6} + b_7 F_{j7} \quad (126)$$

Some of these coefficients $b_0, b_1, .., b_7$ may be set to zero to represent different schemes.

The present method is based on an explicit march in the η direction. Since the method is not implicit in η, b_2 will be zero for a positive η direction

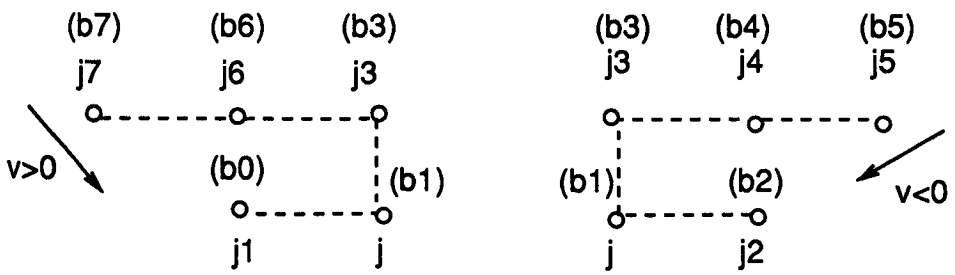
sweep and b_0 will be zero for a negative η direction sweep. For the case called Krause zig-zag differencing ('Z' scheme), in which a positive η direction sweep in used, the coefficients are:

$$b_1 = \frac{1}{2\Delta\eta_j} \;;\; b_0 = -\frac{1}{2\Delta\eta_j}$$
$$b_3 = -\frac{1}{2\Delta\eta_{j+1}} \;;\; b_4 = \frac{1}{2\Delta\eta_{j+1}} \qquad (127)$$
$$b_2 = b_5 = b_6 = b_7 = 0$$

The $\Delta\eta$-s are grid distances defined as below,

$$\Delta\eta_j = \eta_j - \eta_{j-1} \;;\; \Delta\eta_{j-1} = \eta_{j-1} - \eta_{j-2}$$
$$\Delta\eta_{j+1} = \eta_{j+1} - \eta_j \;;\; \Delta\eta_{j+2} = \eta_{j+2} - \eta_{j+1}$$

The five point differencing formula used by Tassa[8] et al. is an implicit cross-flow dependent differencing scheme. The finite differencing molecule is rotated depending on the local sign of v. This type of differencing can be used when points on the entire i line are solved simultaneously (i.e., implicit solution in η) and is used here in solving the surface Euler equations. The coefficients for this case are as given below:



$v > 0$

$$b_1 = 1/\Delta\eta_j$$
$$b_0 = -1/\Delta\eta_j$$
$$b_2 = b_4 = b_5 = 0$$
$$b_3 = 1/(\Delta\eta_j + \Delta\eta_{j-1})$$
$$b_6 = -1/\Delta\eta_{j-1}$$

$v < 0$

$$b_1 = -1/\Delta\eta_{j+1}$$
$$b_2 = 1/\Delta\eta_{j+1}$$
$$b_1 = b_6 = b_7 = 0$$
$$b_3 = 1/(\Delta\eta_{j+1} + \Delta\eta_{j+2})$$
$$b_4 = 1/\Delta\eta_{j+2}$$

(128)

$b_7 = \Delta\eta_j/\{\Delta\eta_{j-1}(\Delta\eta_j + \Delta\eta_{j-1})\}$ $\qquad$ $b_5 = -\Delta\eta_{j+1}/\{\Delta\eta_{j+2}(\Delta\eta_{j+1} + \Delta\eta_{j+2})\}$

The crossflow dependent finite difference can be represented by a single formula as given in Ref. 9, if we define an ε which takes the value of -1 for $v > 0$ and +1 for $v < 0$. At the boundary-adjacent points of j=2 and j=jm-1, the five-point scheme cannot be applied if the point $j+2\varepsilon$ point falls outside the computation region. In this case, a simple two-point scheme involving the boundary point and the boundary-adjacent point is used. This scheme (called the 'L' scheme here) has the following coefficients:

$$b_1 = 1 / \Delta\eta_j \;\; ; \;\; b_0 = -1 / \Delta\eta_j \qquad (129)$$

$$b_1 = b_3 = b_4 = b_5 = b_6 = b_7 = 0$$

8. 3D BL SOLUTION PROCEDURE

The solution procedure can be explained with the help of Fig. 4. The figure shows the 3D BL grid projected to the (ξ, η) plane. The grid lines in the ζ direction are thus normal to the plane on which the figure is drawn.

Solution at an Interior Point

For calculation of the 3D boundary layer at an interior (i,j) location, the initial boundary plane (i=1 plane) and the side boundary plane (j=1 plane) are assumed to be known. The first step involves the calculation of the i=2 plane which corresponds to a ξ direction sweep. The solution along the i=2 plane is obtained by a η direction march starting from the left boundary. The solution is thus obtained by sweeping in the j direction from j=2 to jm-1 for a fixed i.

The solution procedure at a particular (i,j) location can be explained with the help of Fig. 5. The solution is obtained by implicit solution in the k direction. The F, G, F′ and G′ profiles at each (i,j) location are obtained by inversion of a block-tridiagonal system as already explained. To start the iteration, the profiles are initialized by making them equal to the profiles at the corresponding (i-1,j) location. The edge coefficients $\tilde{A}_i$, B_i, C_i and D_i can be calculated from the known edge conditions and metrics. The coefficient D_5 is calculated based on the current solution F, G, F′ and G′. The linearized momentum equation is solved first by inversion of the discretized system. The continuity equation is then integrated from the given wall conditions to obtain new values of w. Subsequently, the energy equation is solved. The values of θ, θ', I, I′, I_i and I'_i are now updated based on the current solution (eq.s 49-52). The values of w_w and H'_w or H_w corresponding to the wall boundary conditions are updated based on the current solution and the input wall boundary values of w_w^* and T_w^* or $\dot{q}_w^*$ (equations 112 and 113). The calculations are repeated iteratively until convergence is obtained. Convergence is monitored by scanning values of $\delta F'$ and $\delta H'$. The criteria for convergence are:

$$\frac{|\delta F'_{max}|}{|F'_{max}|} < \varepsilon_F \quad ; \quad \frac{|\delta H'_{max}|}{|H'_{max}|} < \varepsilon_I$$

After convergence of solution at (i,j), calculation proceeds to the next j location as dictated by the η direction sweep. After the entire i line is solved,

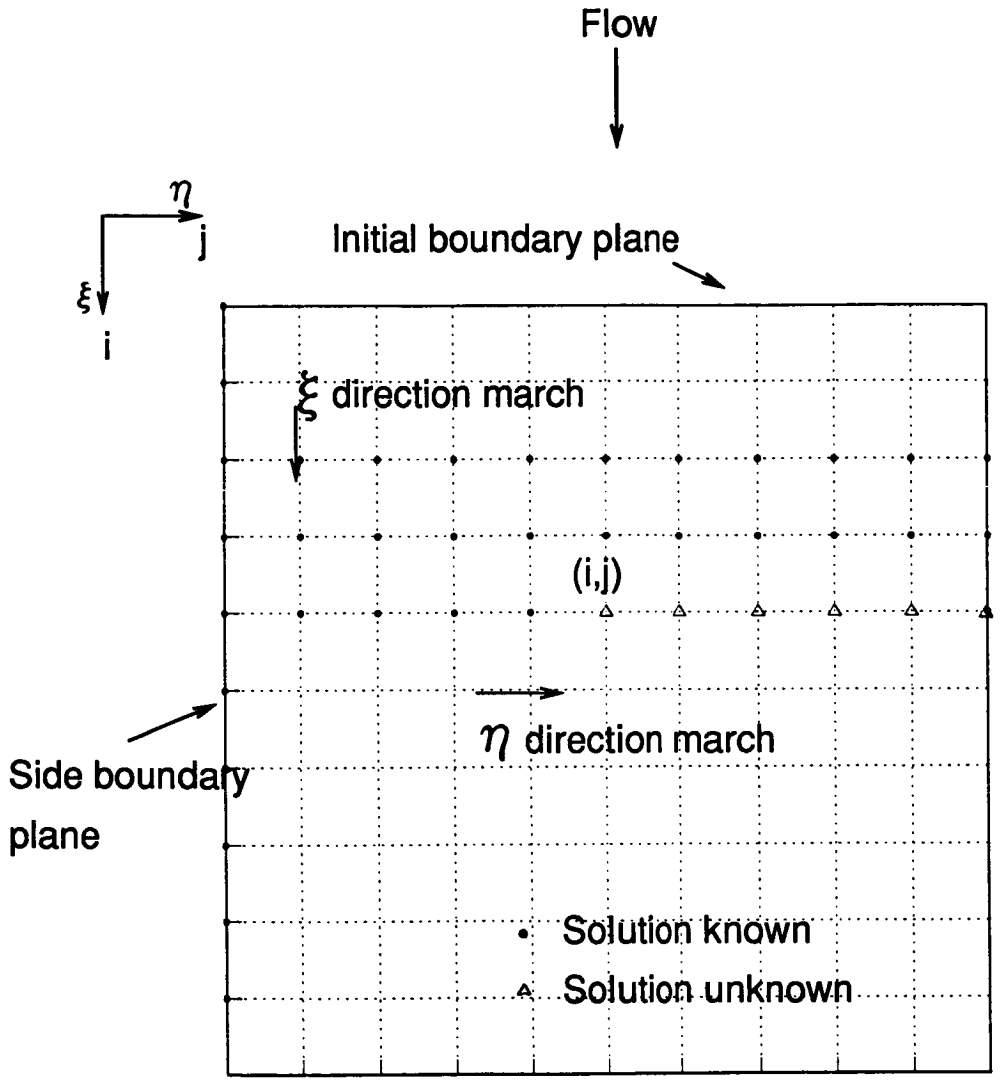


Fig 4. Solution Sweeps in the stream surface plane

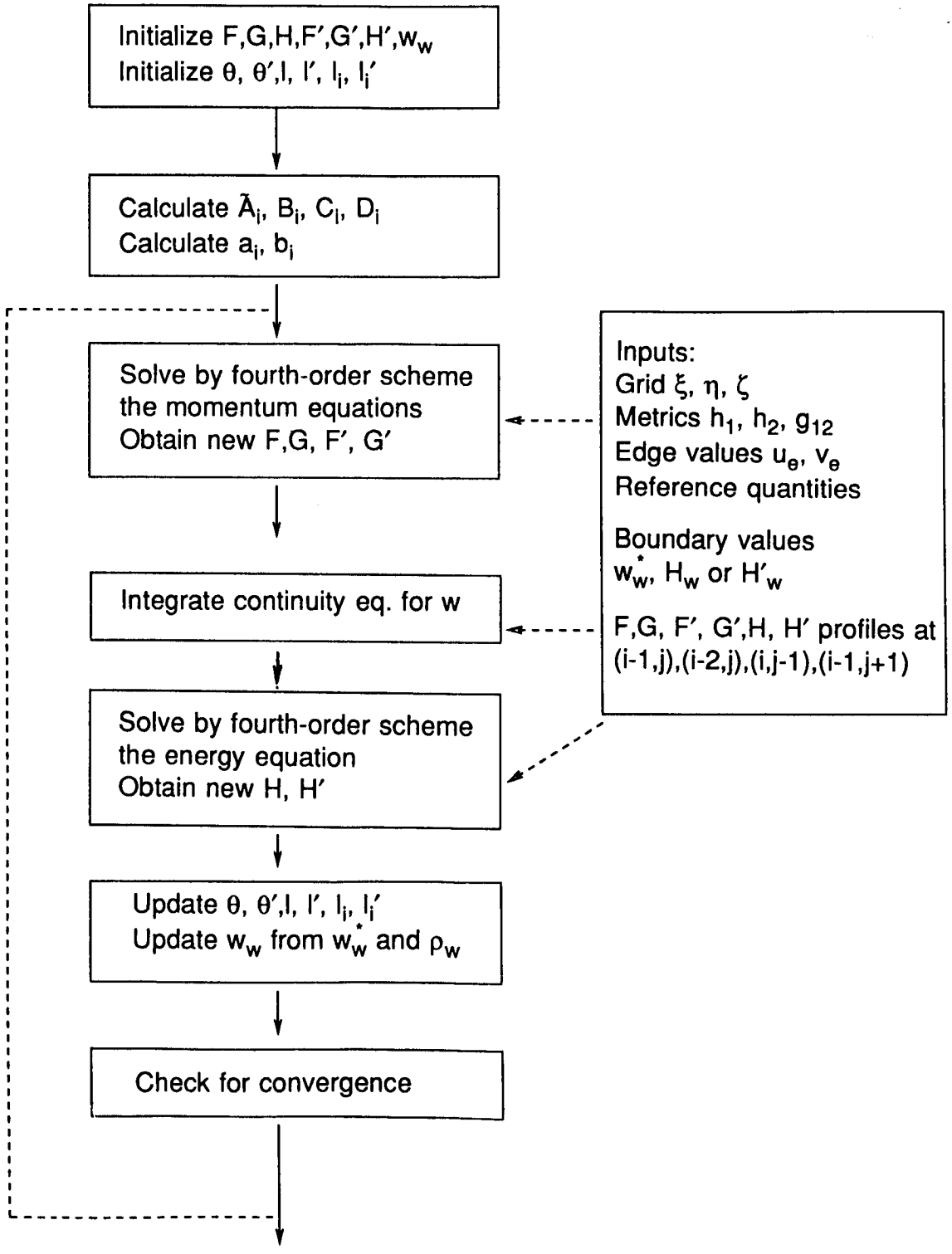


Fig.5 Solution Procedure for Profiles at (i,j)

the solution advances to (i+1) level. The solution at i and i-1 are saved for ξ differencing. Marching in the i direction is continued until proximity to streamwise separation is indicated by a large number of iterations needed for convergence.

Domain of Dependence

The differencing and solution march in the η direction has to honor the domain of dependence principle for an accurate and stable calculation. The nature of the 3D BL equations is such that it is parabolic in the ζ direction and hyperbolic in the stream surface direction. On a stream surface, the characteristics of the equations are the local streamlines. Since the equations are parabolic in the normal direction, information is transmitted instantaneously in that direction. Hence a wedge of region can be identified in the flow with its vertex on the wall normal line at (i,j) and with its sides coinciding with streamline at the BL edge and the limiting streamline at the wall. In a more general way, the wedge depends on the ratio $(\frac{v}{u})$ in the entire profile. The velocity vectors corresponding to minimum and maximum values of $(\frac{v}{u})$ determine the domain of dependence. According to the CFL stability condition, the η differencing stencil should be such that the numerical domain of dependence includes the physical domain of dependence.

The Krause zig-zag or the 'Z' scheme is designed such that the CFL condition is honored for moderate crossflow reversal situations. The maximum angle of crossflow in the negative η direction is $\frac{h_2\Delta\eta}{h_1\Delta\xi}$. For solution at a point (i,j), the CFL condition thus amounts to,

$$\frac{v}{u} < \frac{h_2\Delta\eta_{j+1}}{h_1\Delta\xi_i} \tag{130}$$

Alternatively, the stability test takes the form,

$$\frac{Gv_r h_1\Delta\xi_i}{Fu_e h_2\Delta\eta_{j+1}} > -1 \tag{131}$$

Improving the stability margin for a given flow condition involves reducing the streamwise step size, which effectively increases the domain of dependence wedge angle. Increasing the grid spacing in the η direction is not a viable alternative since flow features in that direction are liable to remain unresolved. If $h_1 = h_2 = 1$ and $\Delta\xi = \Delta\eta$, the 'Z' scheme can handle up to

135° flow direction angle. Frequently, the ξ direction spacing is much finer than the η direction spacing, which means that even larger crossflow angles can be handled. The computation at each (i,j) location is designed to check for CFL stability as given above.

Boundary Solutions

The solution at the initial and side boundaries can be obtained by solving one of the special subsets of the 3D BL equations given previously. These equations are in 2D form and can be solved independent of the 3D region.

For a fuselage-type geometry, the initial plane is generated using the stagnation point solution. For moderate angles of attack, the stagnation point on the body is quite close to the geometric extremity. In order to avoid the complexity of a special transformation and grid, we make the following assumptions.

(1) The stagnation point is at the geometric extremity

(2) The stagnation point solution is independent of the circumferential angle; this amounts to assuming $C_{r,s}=1$ in the stagnation point flow equations.

The initial plane solution thus consists of 3D stagnation point solution profiles at all j stations on the i=1 line.

The side boundary plane for a fuselage geometry is obtained by solving the symmetry line equations on the windward line of symmetry. For positive angle of attack, the windward line of symmetry is usually an outflow line. There are, however, exceptions to this rule depending on the fuselage shape. For negative angle of attack, the leeward symmetry line is the outflow line. In this case, the side boundary at j=jm is calculated and the 3D solution is calculated by a negative sweep in the η direction. The last point at the end of a η sweep is calculated by an 'L' scheme.

For a wing-type geometry, the initial plane of solution is provided by the attachment line equations. The side boundary presents somewhat of a problem. If we are looking at a wing-alone geometry, the crossflow velocity near the wing root may become negative (see Fig. 6). The presence of such a negative crossflow region depends on the angle of attack and the wing cross section. For real flow past wings attached to a fuselage, the displacement effect due to the fuselage usually prevents the occurrence of a negative crossflow region. The flow in the wing-fuselage junction region is not described by the boundary layer equations. Under such conditions, we

need a reasonable approximation to the flow near the wing root to start the 3D BL calculation.

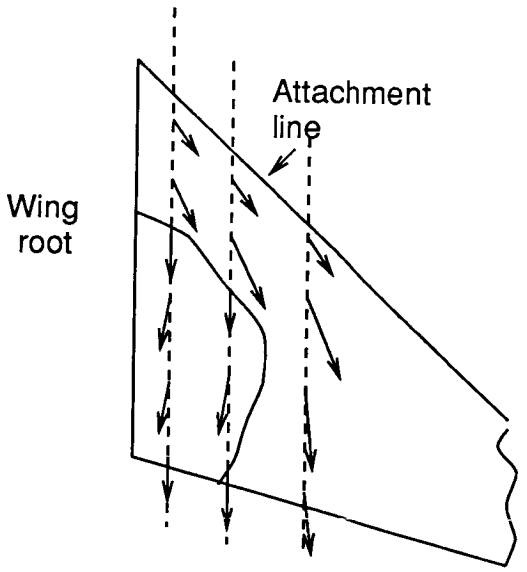


Fig.6 Region of Negative Crossflow on a Wing

The method adopted here is to assume locally infinite swept wing flow conditions at a chordwise plane close to the wing root. If this plane of solution contains regions of only moderate negative crossflow, a sweep in the η direction starting from this plane may provide stable solutions.

9. INTERFACE PROCEDURE

Our discussion so far assumed that the boundary layer edge conditions and the boundary layer grid are known to the required degree of accuracy, smoothness and resolution. Often the correct specification of the edge conditions constitute the most important part of a boundary layer calculation.

A separate computational procedure has been designed to address the need of correctly providing the interface between the inviscid results and the BL solver. In this interface program, attention is focussed on correctly interpolating the inviscid results from the inviscid grid to the BL grid. The interface procedure thus acts as a pre-processor for the BL solver, in which the inviscid results are suitably processed. Depending on the inviscid method used, the interface program can be modified. The BL solver, however, is essentially invariant and does not have to be modified depending on the flow case being solved. The interface can also have provisions to output the calculated BL edge conditions and graphically display them in order to check for smoothness and accuracy.

The Inviscid Method

We make the assumption that the inviscid results are available on an inviscid grid to the required degree of accuracy and resolution. Details of the inviscid procedure are not dealt with here. Usually, it involves solving the Euler equations, the full potential equations or a linearized version of the potential equations. For subsonic flow cases, a higher-order panel method is usually employed. For higher Mach numbers, solution of the Euler equations is necessary. To make the task of interpolation easier, certain restrictions have to be placed on the inviscid grid. They are given below:

(1) For a wing-type of geometry, the inviscid grid is assumed to be oriented in the constant span and constant chord directions. A grid oriented in any other way may produce large errors when interpolating to a body-oriented non-orthogonal BL grid. For a fuselage-type of body, the inviscid grid is assumed to be derived from sections normal to the body axis (defined in the positive x' direction).

(2) The inviscid grid distribution in the span-wise or circumferential direction is assumed to coincide with that of the BL grid. In this way, we avoid interpolation in the span-wise or circumferential (i.e. η) direction. Higher resolution required in the streamwise (ξ) direction can be provided in the BL grid.

The resolution in the η direction in the BL calculation is pre-determined by the inviscid grid.

The interface program must be provided with the following inputs from an inviscid calculation:

(1) The inviscid grid in Cartesian coordinates (x'_i, y'_i, z'_i)

(2) The Cartesian components of velocity on the inviscid grid (u'_i, v'_i, w'_i)

(3) The pressure or pressure coefficient values on the inviscid grid

It is not always necessary to specify the inviscid velocity components since the inviscid flow field is completely determined with the specification of the inviscid grid, the pressure, and the free stream conditions. However, the velocity components are required to correctly locate the inviscid attachment line on a wing.

The Interface Objectives

The interface procedure accomplishes the following[9]:

(1) Generate the boundary layer grid starting from the attachment line or stagnation point, with the required streamwise grid resolution

(2) Calculate the metric scale coefficients h_1, h_2, g_{12} for this grid

(3) Calculate the BL grid-oriented edge velocities u_e and v_e on the BL grid.

The details of how the above are accomplished depends to a large extent on the type of geometry being considered. Two types of geometries are defined, with each one having a separate interface program. The first one is a wing-type of geometry in which an inviscid attachment line can be located and the body can be mapped to a plane surface. The second one is a fuselage-type geometry which is assumed to have a stagnation point (or a point of singularity such as a sharp cone) at its geometric extremity. The geometry in this case can be mapped to a body of revolution.

The entire interface procedure can be skipped for bodies which can be analytically defined and whose inviscid flow conditions can be expressed in a closed form. The 2D flow test cases and some 3D flow test cases presented later do not require an interface procedure since the BL edge conditions can be exactly specified by analytical formula. The interface procedure is therefore required only for geometries and/or inviscid conditions defined numerically rather than by equations.

Interpolation Accuracy

The inviscid grid is typically a coarser grid than that is required for an accurate BL calculation, especially in the streamwise direction. Since we are assuming that the inviscid grid and BL grid are coincident in the η direction, interpolation is required only in the streamwise direction. Two options are provided in the interface procedure:

(1) Interpolate the Cartesian velocity components (u'_i, v'_i, w'_i) from the inviscid grid to the BL grid and then compute the edge velocities

(2) Interpolate the pressure or pressure coefficient from the inviscid grid to the BL grid and calculate u_e and v_e by solving the surface Euler equations.

In either case, the edge conditions going into the 3D BL solver must be carefully monitored for smoothness and accuracy, since an accurate BL calculation depends to a large extent on the quality of edge conditions prescribed. The smoothness of edge condition assumes great importance especially when the boundary layer calculation results are used for a stability analysis. Non-physical oscillations entering into the BL results via edge conditions can severely degrade the conclusions arrived at in a stability analysis. Graphic visualization of the edge conditions generated by the interface and checking for smoothness in ξ and η direction is recommended.

BL Grid and Metrics

For a wing-type geometry, the BL grid is generated starting from the inviscid attachment line. The attachment line is in general a curved line depending on the angle of attack, taper and twist of the wing. The inviscid attachment line is first located approximately corresponding to the maximum pressure line in the neighborhood of the wing leading edge. The Cartesian velocity components are then interpolated to this line. The BL edge velocities u_e in the streamwise direction and v_e along the attachment line at this location are then computed. The velocity conversion from Cartesian to BL grid coordinates is done using the the following inversion:

$$\begin{bmatrix} x'_x & x'_y & \frac{\Delta_1}{\Delta} \\ y'_x & y'_y & \frac{\Delta_2}{\Delta} \\ z'_x & z'_y & \frac{\Delta_3}{\Delta} \end{bmatrix} \begin{bmatrix} \frac{u_e}{h_1} \\ \frac{v_e}{h_2} \\ \frac{w_e}{h_3} \end{bmatrix} = \begin{bmatrix} u'_e \\ v'_e \\ w'_e \end{bmatrix} \qquad (132)$$

where,

$$\Delta_1 = y'_x \cdot z'_y - z'_x \cdot y'_y$$

$$\Delta_2 = x'_y \cdot z'_x - z'_y \cdot x'_x$$

$$\Delta_3 = x'_x \cdot y'_y - x'_y \cdot y'_x$$

$$\Delta = (\Delta_1^2 + \Delta_2^2 + \Delta_3^2)^{1/2}$$

The u_e values obtained on the attachment line should be zero by definition. A relocation of this line can be done based on the non-zero value of u_e so as to make it exactly equal to zero. Usually this involves only very small relocations of the attachment line since $\partial u_e/\partial s_1$ values are large near the attachment line.

Once the attachment line is exactly determined, the BL grid can be developed off it over the lower or upper surface. The grid distribution in the streamwise direction is made such that the arc lengths Δs_1 have a step distribution that can be selected as required. Once the BL grid is generated, the metric coefficients h_1, h_2 and g_{12} can be calculated based on eq.s 19-21. The measurement of the BL coordinate x and y is arbitrary. They can be measured along x′ and y′ or along x and y. The values of h_1, h_2 and g_{12} will accordingly be different.

For a fuselage type of body, the BL grid is generated starting from the geometric extremity since we assume the stagnation point to be at this point. However, one has to be careful about the placement of the i=2 point of the BL grid. This has to be located downstream of the actual inviscid stagnation point to avoid negative streamwise velocities in the numerical computation. For moderate angles of attack < 6 ° the i=2 point can be placed reasonably close to the geometric extremity without producing any uncharacteristic fluctuations near the stagnation point.

Spline Interpolation

Quantities on the inviscid grid can now be interpolated to the BL grid. Interpolation is required only in the streamwise direction. The independent variable for streamwise interpolation is the surface arc length s_1, which can be calculated for both grids starting from a convenient reference point such as the leading edge or the stagnation point. Interpolation is done using smoothed tension splines. The degree of smoothing and the amount of tension to use is dependent on the original input data and cannot be fixed a

priori. This has to be done on an interactive basis with graphic display of the results; no automation of this selection is possible.

In the general case, this is a surface interpolation problem. If interpolation is done by smoothing in the ξ direction, followed by smoothing in the η direction, this can introduce non-smooth variation in the ξ direction back again. We have conveniently avoided this type of error-coupling by having both the grids identically distributed in the η direction.

This interpolation procedure is done for pressure. Optionally, the Cartesian velocity components can also be interpolated to the BL grid and edge velocities can then be calculated using equation (130). Another option is to calculate (u_e, v_e) from the interpolated pressure field by solving the surface Euler equations; this method is more accurate in most cases.

Once (u_e, v_e) are obtained at all BL grid points, the edge quantities of ρ_e and T_e required in the 3D BL solver can be calculated. Calculation of T_e is based on the assumption that total enthalpy at BL edge is equal to the free stream total enthalpy value (equation (16)). Calculation of ρ_e is based on the free stream conditions and the assumption of an isentropic process from free stream to BL edge. If there is a shock present, ρ_e can be calculated knowing the conditions downstream of the shock.

Solution of the Surface Euler (S.E.) Equations

The surface Euler equations from equations (27) and (28) are:

$$\frac{u_e}{h_1}\frac{\partial u_e}{\partial x} + \boxed{\frac{v_e}{h_2}\frac{\partial u_e}{\partial y}} + C_{24}u_e^2 + C_{25}u_e v_e + C_{26}v_e^2 = r^*_{e,1} \qquad (133)$$

$$\frac{u_e}{h_1}\frac{\partial v_e}{\partial x} + \boxed{\frac{v_e}{h_2}\frac{\partial v_e}{\partial y}} + C_{34}u_e^2 + C_{35}u_e v_e + C_{36}v_e^2 = r^*_{e,2} \qquad (134)$$

where, $r^*_{e,1}$ and $r^*_{e,2}$ are the known pressure gradient terms on the RHS of equations (27) and (28). The terms outlined in solid lines are zero for a locally infinite swept wing (LINF) and the terms outlined in dashed lines are zero for symmetry line flow (SYM BL). In the latter case, the entire y momentum equation vanishes and the x momentum reduces to a Bernoulli-type equation. Note that (x,y) and (ξ,η) are interchangeable in the above set since derivatives in the ζ direction are zero.

The above equations can be discretized using 3-point backward differencing for $\partial/\partial x$ quantities and the 5-point scheme for $\partial/\partial y$ quantities. Each i line is solved implicitly for (u_e, v_e) from a block tridiagonal system. The discretization proceeds as given below:

$$\frac{(u_e+\delta u_e)_j}{h_1}\left[a_1(u_e+\delta u_e)_j+\{au_e\}\right]+\frac{(v_e+\delta v_e)_j}{h_2}\left[b_1(u_e+\delta u_e)_j+b_0(u_e+\delta u_e)_{j-1}+b_2(u_e+\delta u_e)_{j+1}\right.$$
$$\left.+\{bu_e\}\right]+C_{24}(u_e+\delta u_e)_j^2+C_{25}(u_e+\delta u_e)_j(v_e+\delta v_e)_j+C_{26}(v_e+\delta v_e)_j^2 = r^*_{e,1}$$

$$\frac{(u_e+\delta u_e)_j}{h_1}\left[a_1(v_e+\delta v_e)_j+\{av_e\}\right]+\frac{(v_e+\delta v_e)_j}{h_2}\left[b_1(v_e+\delta v_e)_j+b_0(v_e+\delta v_e)_{j-1}+b_2(v_e+\delta v_e)_{j+1}\right.$$
$$\left.+\{bv_e\}\right]+C_{34}(u_e+\delta u_e)_j^2+C_{35}(u_e+\delta u_e)_j(v_e+\delta v_e)_j+C_{36}(v_e+\delta v_e)_j^2 = r^*_{e,2}$$

The quantities $\{au_e\}$, $\{bu_e\}$, $\{av_e\}$, $\{bv_e\}$ in the above refer to the explicit part of the finite difference formula. For example, $\{bu_e\}$ corresponds to $b_3u_{i-1,j}+ b_4u_{i-1,j+1}+b_5u_{i-1,j+2}$ for $v<0$ and $b_3u_{i-1,j}+ b_6u_{i-1,j-1}+b_7u_{i-1,j-2}$ for $v>0$. Collecting coefficients corresponding to a block tridiagonal system as given below, the expressions for the matrix elements can be derived. These are given in Table 10.

$$\begin{bmatrix} \ddots & & & \\ & \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} & \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \begin{bmatrix} \delta u_e \\ \delta v_e \end{bmatrix}_{j-1} \\ \begin{bmatrix} \delta u_e \\ \delta v_e \end{bmatrix}_{j} \\ \begin{bmatrix} \delta u_e \\ \delta v_e \end{bmatrix}_{j+1} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \begin{bmatrix} r_{e,1} \\ r_{e,2} \end{bmatrix}_j \\ \vdots \end{bmatrix}$$

Boundary-adjacent points which require a $j+2\varepsilon$ point outside the computation region are finite-differenced using a 2-point 'L' scheme.

Table 10. Coefficients from the Surface Euler Equations

$$a_{11} = \frac{b_0}{h_2} v_{e,j}$$

$$a_{12} = 0$$

$$b_{11} = \frac{1}{h_1}\left[2a_1 u_{e,j} + \{au_e\}\right] + \frac{b_1}{h_2} v_{e,j} + 2C_{24}u_{e,j} + C_{25}v_{e,j}$$

$$b_{12} = \frac{1}{h_2}\left[b_1 u_{e,j} + b_0 u_{e,j-1} + b_2 u_{e,j+1} + \{bu_e\}\right] + C_{25}u_{e,j} + 2C_{26}v_{e,j}$$

$$c_{11} = \frac{b_2}{h_2} v_{e,j}$$

$$c_{12} = 0$$

$$r_{e,1} = r^*_{e,1} - \frac{u_{e,j}}{h_1}\left[a_1 u_{e,j} + \{au_e\}\right] - \frac{v_{e,j}}{h_2}\left[b_1 u_{e,j} + b_0 u_{e,j-1} + b_2 u_{e,j+1} + \{bu_e\}\right]$$

$$- C_{24}u^2_{e,j} - C_{25}u_{e,j}v_{e,j} - C_{26}v^2_{e,j}$$

$$a_{21} = 0$$

$$a_{22} = \frac{b_0}{h_2} v_{e,j}$$

$$b_{21} = 2C_{34}u_{e,j} + C_{35}v_{e,j} + \frac{1}{h_1}\left[a_1 v_{e,j} + \{av_e\}\right]$$

$$b_{22} = \frac{a_1}{h_1} u_{e,j} + \frac{1}{h_2}\left[2b_1 v_{e,j} + b_0 v_{e,j-1} + b_2 v_{e,j+1} + \{bv_e\}\right] + C_{35}u_{e,j} + 2C_{36}v_{e,j}$$

$$c_{21} = 0$$

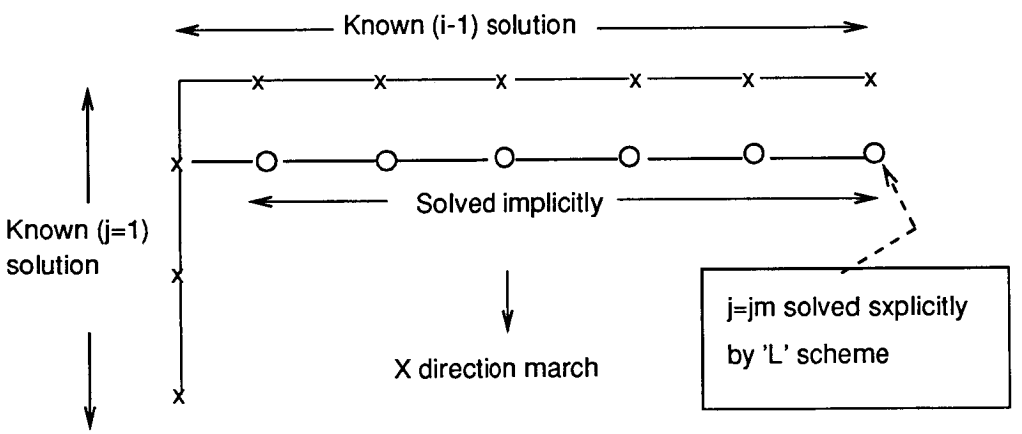
$$c_{22} = \frac{b_2}{h_2} v_{e,j}$$

$$r_{e,2} = r^*_{e,2} - \frac{u_{e,j}}{h_1}\left[a_1 v_{e,j} + \{av_e\}\right] - \frac{v_{e,j}}{h_2}\left[b_1 v_{e,j} + b_0 v_{e,j-1} + b_2 v_{e,j+1} + \{bv_e\}\right]$$

$$- C_{34}u^2_{e,j} - C_{35}u_{e,j}v_{e,j} - C_{36}v^2_{e,j}$$

The solution procedure for the surface Euler equations is similar to the general 3D BL solution procedure. The initial solution is $u_e = v_e = 0$ at the stagnation point for a fuselage-type body and $u_e = 0$ for a wing-type body. The attachment line velocity v_e is obtained from equation (130) in the attachment line relocation procedure. For an infinite swept wing, this value is a constant equal to the sine of the sweep angle Λ.

The left boundary is calculated by solving the symmetry line equation or the LINF equation set. The solution in the interior is obtained implicitly by block tridiagonal inversion as shown in the sketch below. Iteration is required for the nonlinear terms.



The j=jm point can be included in the implicit procedure by finite differencing using an 'L' scheme. Alternatively, for a fuselage type body, it can be obtained explicitly by solving the leeward symmetry line. The implicit 5 point scheme is dropped to an 'L' scheme at the boundary adjacent points if the $j+2\varepsilon$ point is outside the computation region.

10. RESULTS

A brief outline of the program structure will first be presented before going into a discussion of the test cases and results. Further details are given in the Appendix, which is intended as a program manual.

A general 3D BL solver cannot be designed to be used as a canned program with no user interaction. Depending on the flow geometry, flow conditions, inviscid results and outputs desired, some amount of user interaction is necessary. For a given problem, some routines may have to be modified or separate routines added. The user is thus assumed to possess a good amount of familiarity with the program structure, algorithms and data flow. The material covered in the present report and the Appendix should be adequate for this purpose.

Program structure

The program ensemble is arranged in a UNIX file tree structure. The various program divisions are tabulated below.

Program Group	Sub-directory Name
3D stagnation point solution	stg_bl
Symmetry line BL solution	sym_bl
Wing attachment line BL solution	att_bl
Infinite swept wing BL solution	inf_bl
Similarity-type BL flow solution	sim_bl
3D BL solution	3d_bl
Interface program for wing	int_wng
Interface program for fuselage	int_fus
Momentum and energy equation solver routines, matrix inversion, spline and library routines, 'include' blocks	lib

A number of test and validation cases have been run. The following is a list of these cases.

Case 1. 3D stagnation point boundary layer solution

Case 2. Flow past a flat plate (wall temperature or heat flux specified)

Case 3. Flow past a cone at α=0°, M_∞=6.8 with mass injection

Case 4. Symmetry line flow on an ellipsoid at α=6°

Case 5. Flow past a NACA 0012 airfoil at α=0°, M_∞=0.5

Case 6. Flow past a swept elliptical cylinder, α=5°, Λ=40°

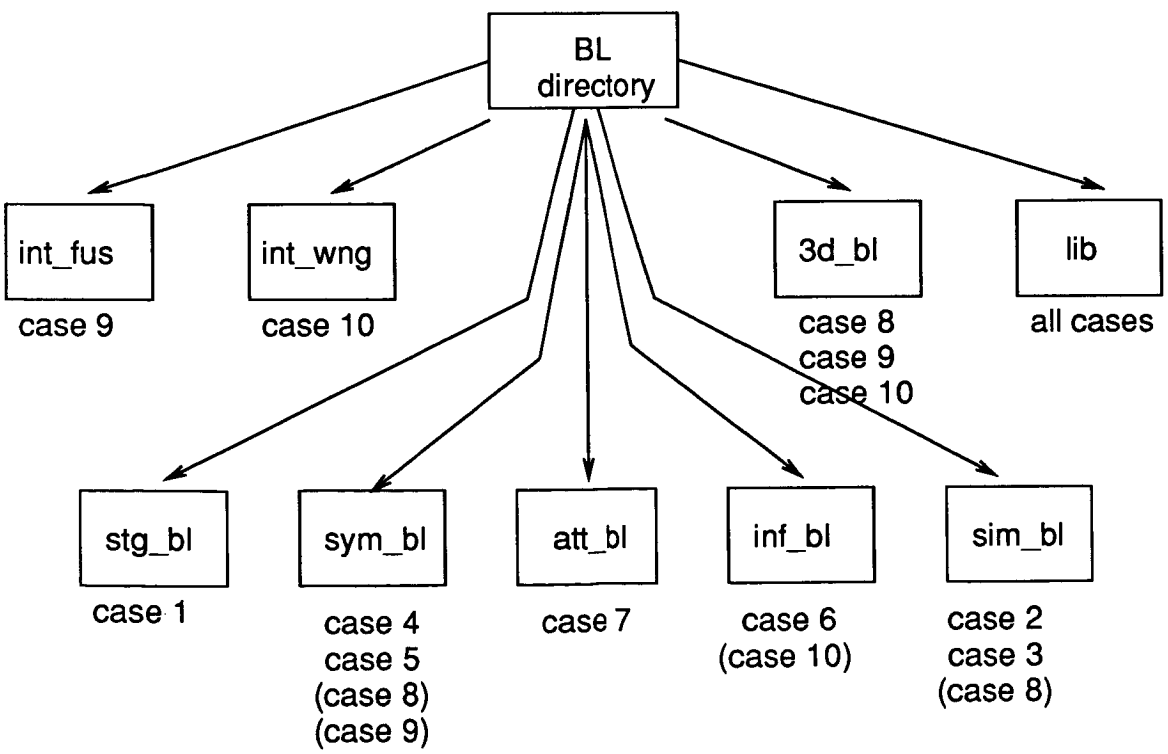
Case 7. Flow along attachment line on a swept cylinder

Case 8. Flow past a cylinder normal to a flat plate

Case 9. BL solution on a ellipsoid at angle of attack

Case 10. Flow past a constant sweep and zero taper wing

The sketch below indicates the location of each of the corresponding programs in the program package.



The sub-directory called 'lib' contains the routines to solve the momentum, energy and continuity equations and some other routines called by these routines. All the test cases listed above use identical routines in 'lib' with changes only in array dimensions. As seen in previous sections, the equations corresponding to all the above cases can be expressed in the same general form. Only the coefficients A_i, B_i, C_i and D_i are different which are defined externally in the calling program. Two-dimensional and axisymmetric cases can also be run with jm set to 1 or 3.

The interface program is required only in cases 9 and 10, where the BL grid and the inviscid results are not analytically defined. All the other cases have an analytic treatment of the interface and is included as a routine in the BL solution program itself. Graphic programs are essential to monitor results at intermediate and final stages. These consist of line plots to look at ξ and η direction variations of various parameters and contour plots. Since these programs are machine-dependent, they are not included in the program package.

The results corresponding to the ten test cases will now be presented. Detailed input information for each case is presented in the Appendix. The Appendix also lists the various subroutines and their functions. The present discussion is limited to the computational and physical aspects of the results. The numerical and programming details are given in the Appendix.

Case 1: 3D Stagnation Point Flow

This test case consists of flow at a general 3D stagnation point. The solution is required on the wall normal at the inviscid stagnation point of the body in 3D flow. Since there is no flow at the stagnation point, the solution profiles consist of $\partial u/\partial x$ and $\partial v/\partial y$ values referenced to the corresponding edge values. The solution is used as starting profiles for flow past 2D or 3D blunt bodies.

The inputs for this case consist of:

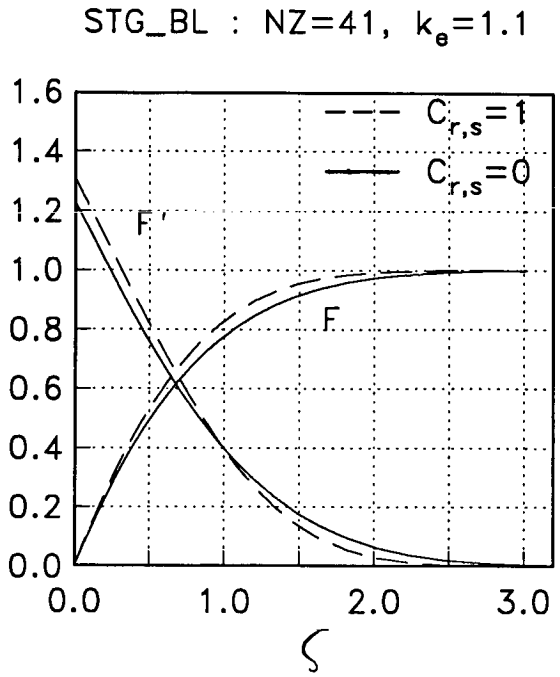
(1) The quantity $C_{r,s} = \dfrac{h_1 g_e}{h_2 f_e}$ representing the ratio of velocity gradients in two orthogonal directions. A value of $C_{r,s} = 0$ corresponds to 2D stagnation point flow. For example, this would correspond to the starting profile required for calculating the flow past an airfoil. A value of $C_{r,s} = 1$ corresponds to flow past an axisymmetric body at zero angle of attack. The velocity gradient is independent of the direction in this case. Flow past an ellipsoid of revolution at zero angle of attack is an example. Values of $C_{r,s}$ in the range 0 to 1 represent 3D stagnation point flow on a general body where $\partial v_e/\partial s_2 = g_e/h_2$ is smaller than $\partial u_e/\partial s_1 = f_e/h_1$. For flow past a fuselage body at moderate angle of attack, $C_{r,s}$ is very close to unity.

(2) The free-stream conditions of M_∞, p^*_∞ and T^*_∞. If the flow is supersonic, conditions downstream of the shock are to be input.

(3) To obtain physical quantities such as heat transfer rate, boundary layer thickness, it is necessary to specify also the inviscid velocity gradient, $\partial u_e/\partial s_1 = f_e/h_1$. However, for obtaining the solution in transformed variables, this quantity is not required.

(4) If the wall boundary condition is non-adiabatic, the wall temperature or wall heat flux can be specified. Transpiration at the wall w^*_w can also be specified. In both instances, f_e/h_1 needs to be specified to transform the physical heat flux or mass flux at the wall to the corresponding computational quantity.

The example presented here correspond to adiabatic wall conditions only. Fig. 7 shows the solution profiles F and F′ for 2D as well as 3D stagnation point flow. The first plot shows the profiles with 41 points in the boundary layer ($\zeta_e = 6$) and 9 points. Exponential stretching with k_e=1.1 was used for the ζ distribution. It can be noted that with just 9 points in the boundary layer, a reasonably good solution has been obtained. The solution compares exactly with other published results (Ref. 10, for example).



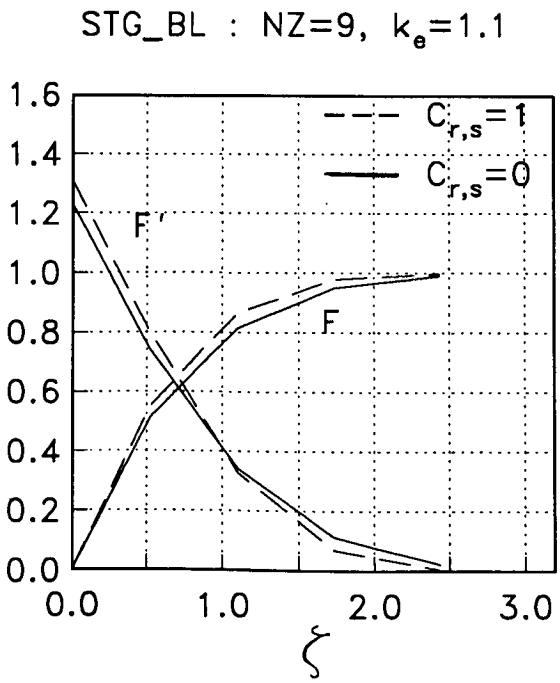


Fig.7 3D Stganation Point Profiles

An order of accuracy estimation was done for this simple case, by successively halving $\Delta\zeta$. A number of runs were made with km=9,17,33 and 65 and with ζ_e=6, k_e=1.0 (no stretching). The absolute error relative to the solution with km=129 on the profiles of F and F′ at different ζ locations was computed. The corresponding variation with step size is plotted on a log-log scale in Fig. 8. The slopes of error decrease on F as well as F′ is approximately unity. Since the x scale is magnified four times compared to the y scale, this indicates that the profiles are accurate to the fourth order. It can also be seen that 5 digit accuracy is obtained for km in the range 30 to 40. High accuracy with relatively few number of points is a feature of the present fourth-order accurate method.

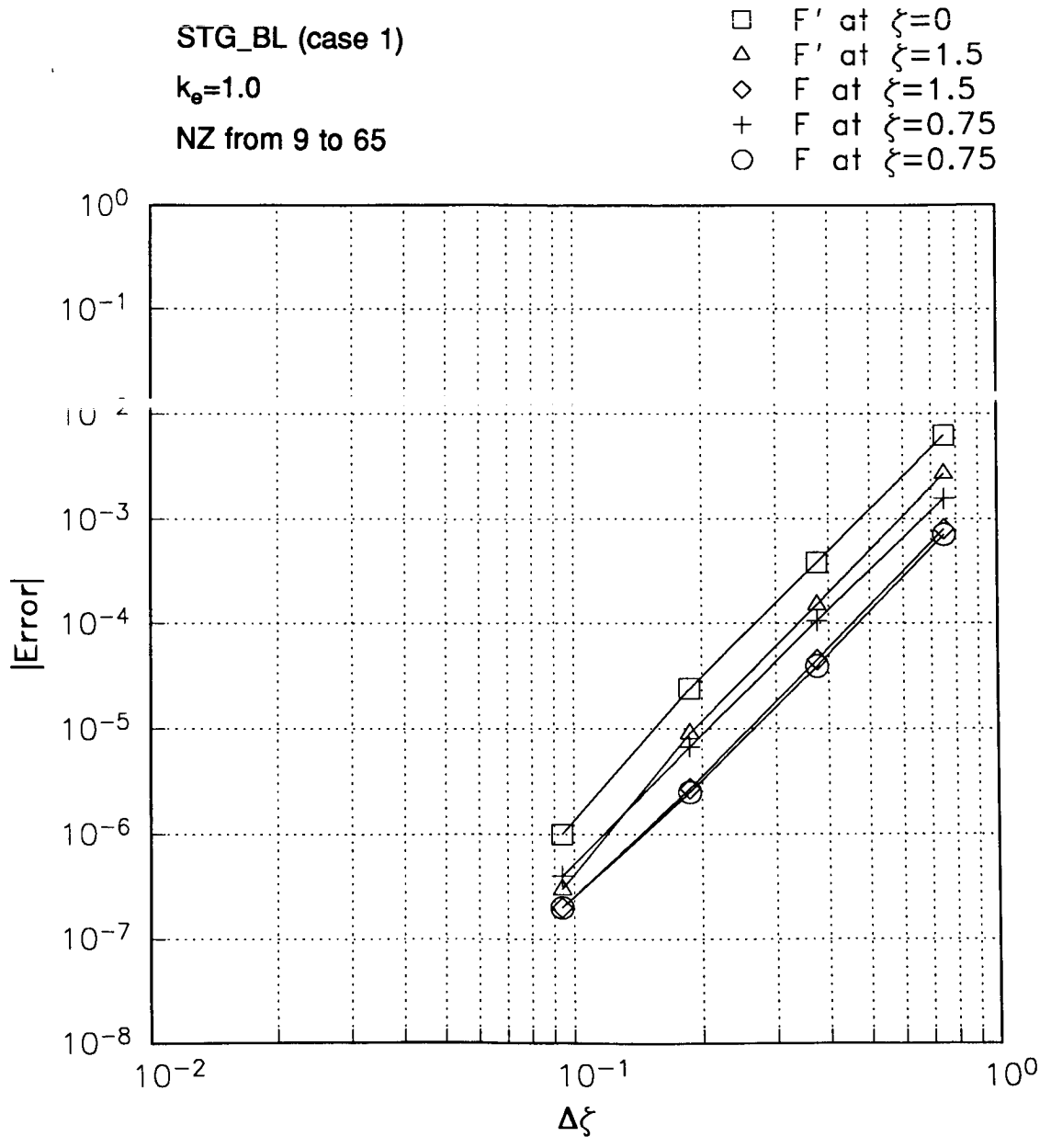


Fig.8 3D Stagnation Point Flow, Variation of Error with Δζ

Case 2: Flow Past Flat Plate

This is a 2D test case designed to validate the generation of similarity profiles at a sharp leading edge and then calculate the 2D flow downstream for a specified wall temperature or heat flux. The similarity profiles are generated by solving the equations set given in Table 7. The 2D flow downstream is calculated based on the symmetry line equations presented in Table 4 with the crossflow velocity gradients set to zero.

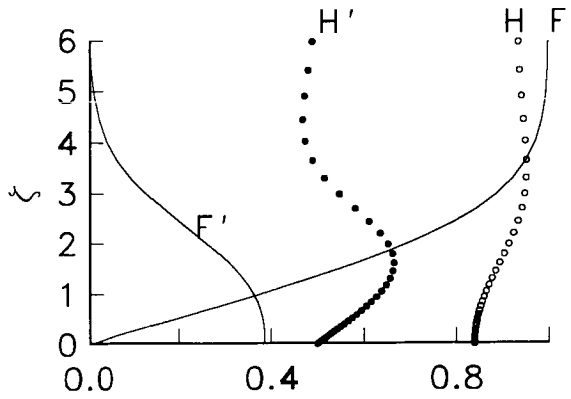
The free-stream conditions are specified as M_∞=3.0, p^*_∞ =400 lb/ft^2 and T^*_∞=400°R. The reference length (corresponding to ξ=1.0) is input as 5 ft. Calculation was done at 21 streamwise points with $\Delta\xi$=0.005. Initial runs at lower Mach numbers representing nearly incompressible conditions gave a $C_{f,x-\infty}\sqrt{Re_x}$ value of 0.664 corresponding to the Blasius solution.

The program was then run with adiabatic wall conditions at M_∞=3. The adiabatic wall temperature was obtained as 1007.3°R which compares well with the approximate relation $T^*_{ad} = T^*_\infty [1+\frac{\gamma-1}{2}M^2_\infty\sqrt{\sigma}]$ =1011°R. The program was then run with two types of wall conditions. The first run consisted of an initial adiabatic wall condition for the first 11 points (for $\xi \leq 0.05$) followed by a linear decrease in the wall temperature from 1107.3°R to 820°R for the next 10 points. This run thus corresponds to wall condition type 1 (iwall=1) with a specified temperature distribution from ξ=0.05 to ξ=0.10. The heat flux for this case is calculated. The second run consisted of specifying the heat flux from ξ=0.05 to ξ=0.1 as equal to the heat flux values obtained from the previous run. The wall temperature values calculated by the program were output and verified to be identical to the distribution used in the previous run. This validates the 3 types of wall boundary conditions, viz., adiabatic wall (iwall=0), wall temperature specified (iwall=1) and wall heat flux specified (iwall=2). The boundary layer results for this case are given in the Appendix.

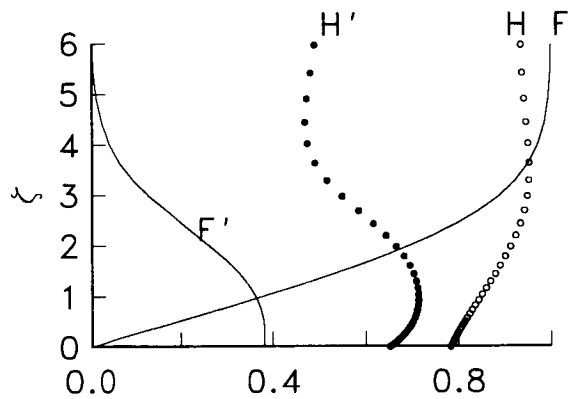
Fig. 9 shows the solution profiles at 3 stations corresponding to ξ=0.05, ξ=0.075 and ξ=0.10. The station ξ=0.05 is the last point with adiabatic boundary condition (i=11). The stations ξ=0.075 and ξ=0.10 have wall temperature specified as boundary condition. Profiles of F, F′ , H and H′ are shown. H′ at the wall is equal to zero at ξ=0.05 which is the adiabatic condition. The ξ=0.075 and ξ=0.10 profiles have finite values of H′ at the wall.

Fig. 10 shows profiles of T^*, u^* and the normal velocity w^* in physical dimension z^*. The normal coordinate transformation used and the exponential stretching is responsible for clustering more points near the wall. The BL

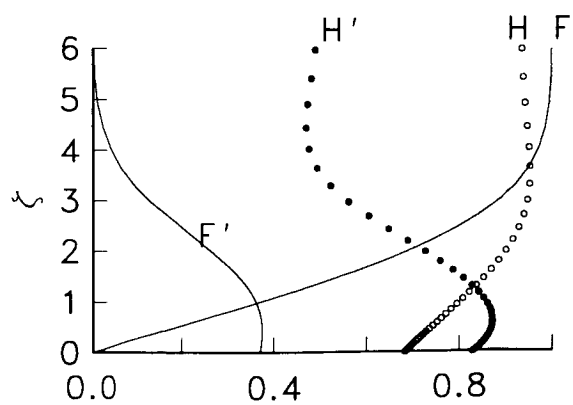
FLOW PAST FLAT PLATE (Wall heat flux specified)
SIM BL : M_∞=3.0 : NZ=41 : k_e=1.10 : L^*=5 ft.



i=11 ξ=0.05
X scale for H from 0.0 to 3.0
X scale for H' from −0.5 to 0.5



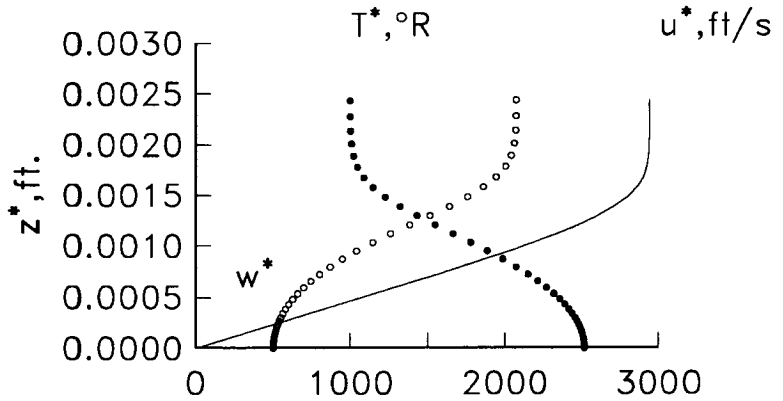
i=16 ξ=0.075
X scale for H from 0.0 to 3.0
X scale for H' from −0.5 to 0.5



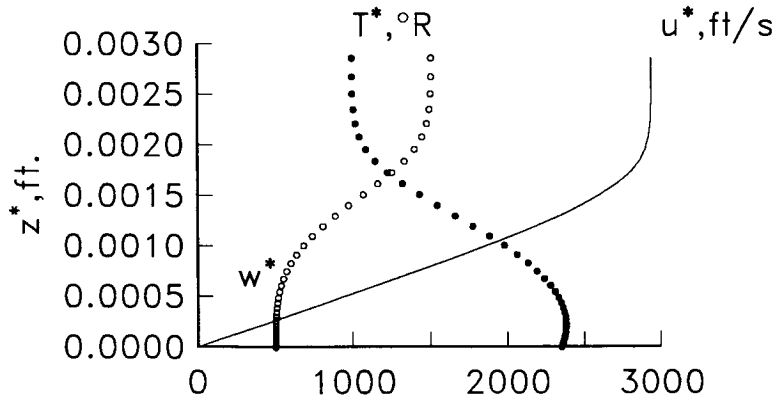
i=21 ξ=0.10
X scale for H from 0.0 to 3.0
X scale for H' from −0.5 to 0.5

Fig. 9 Solution Profiles for Flow Past a Flat Plate

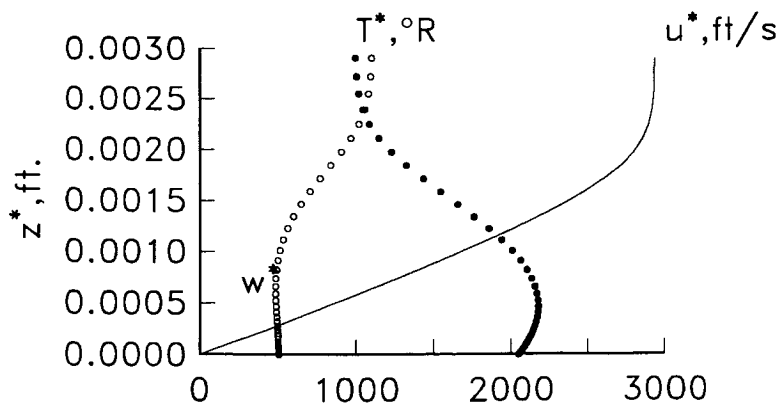
FLOW PAST FLAT PLATE (Wall heat flux specified)
SIM BL : M_∞=3.0 : NZ=41 : k_e=1.02 : L^*=5 ft.



i=11 x^*=0.25 ft.
X scale for T* from 0 to 1200
X scale for w* from −2 to 10



i=16 x^*=0.375 ft.
X scale for T* from 0 to 1200
X scale for w* from −2 to 10



i=21 x^*=0.5 ft.
X scale for T* from 0 to 1200
X scale for w* from −2 to 10

Fig. 10 Solution Profiles in Physical Plane for Flat Plate Flow

thickness increases in the physical plane as can be seen by comparing the 3 profiles. At ξ=0.05, the value of $\partial T^*/\partial z^*$ is zero at the wall since the wall is specified as adiabatic at that point. At the other stations, $\partial T^*/\partial z^*_w$ is positive as a result of the cold wall boundary condition $T^*_w < T^*_{ad}$.

The boundary layer normal velocities are calculated from Eq.(111). For the present case, Eq.(111) simplifies to:

$$w^* = \frac{u^*_\infty u_e \theta w}{\rho_e s_1 \sqrt{Re_{ref}}\psi} + \frac{u^*_\infty u_e F}{h_1 L^*}\cdot\frac{\partial z^*}{\partial \xi}$$

The boundary layer growth rate $\frac{\partial z^*}{\partial \xi}$ is calculated from the solution profiles. The normal velocity profile compares well with the normal velocity given by the Blasius solution (Ref. 10) when run under low Mach number conditions. The non-dimensional velocity corresponding to Ref. 10 is obtained as $\frac{v^*}{u^*_\infty}\sqrt{Re_{ref}\frac{\rho s}{l\mu_e\theta^2}}$.

The skin friction corresponding to compressible laminar flow on a flat plate can be approximately calculated as given in Ref. 10. In the present case, the computed value of $C_{f,x-\infty}\sqrt{Re_x}$ is 0.6088 for the adiabatic case, which compares well with the $C_{f,x-\infty}\sqrt{Re_x}$ value given graphically in Ref. 10, page 237.

The variation of the boundary layer thickness, wall heat flux and skin friction coefficient for this case are plotted with distance ξ in Fig. 11. The linear decrease in T^*_w is the input producing a corresponding change in the wall heat flux. The first 11 points have zero heat flux condition. Note that different scales are used for the different quantities.

FLOW PAST FLAT PLATE (Wall heat flux specified)

SIM BL : M_∞=3.0 : NZ=41 : k_e=1.10 : L^*=5 ft.

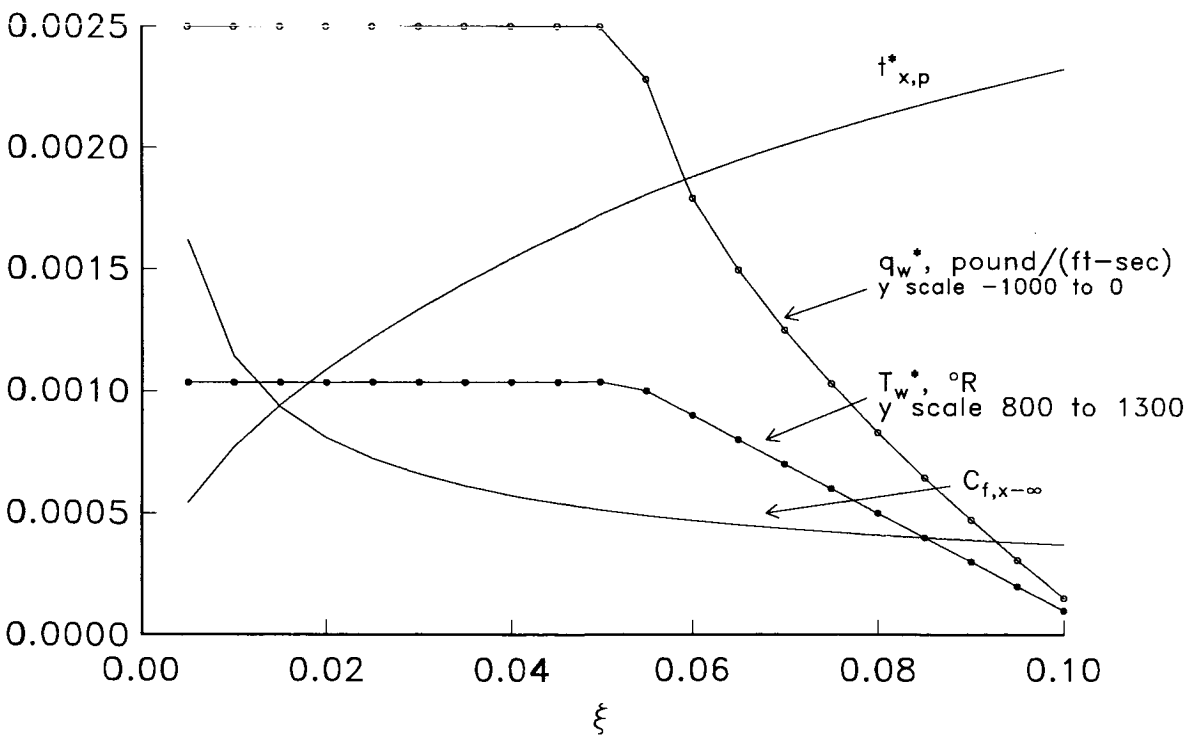


Fig. 11 Axial Variation of Results for Flat Plate Flow

Case 3: Supersonic Flow Past a Cone

This test case involves flow past a cone at a free stream Mach number of 7.4 at 0° angle of attack and with mass injection at the wall. The same conditions have been used as a test case by Harris [11].

The cone is assumed to be pointed with a semi vertex angle of 5°. The total pressure and temperature values ahead of the body are $p_0^* = 4.14 \times 10^6$ Pa. and $T_0 = 833°K$. The inputs to the program are the conditions downstream of the oblique shock. From standard supersonic cone tables, the conditions downstream of the shock at a free stream Mach number of 7.4 are obtained as M_∞=6.833, p_∞^*=1260.12 Pa. and T_∞^*=82.53 °K. These are used as reference conditions.

The cone has zero mass injection from x^*=0.0 to x^*=0.096 m. Downstream of this point, three types of wall conditions are input:

(1) $\rho_w w_w = 0$, zero mass injection

(2) $\rho_w^* w_w^* = -0.090117$ Pa.sec/m, wall suction

(3) $\rho_w^* w_w^* = 0.090117$ Pa.sec/m, wall blowing

The wall temperature is specified as T_w^*=316.65°K

The boundary layer on this body is calculated by first obtaining the initial solution profile at the pointed tip by solving the similarity equations presented in Table 7. The solution downstream is obtained by marching in the x direction and solving the symmetry boundary layer equations (Table 4) with the crossflow velocity gradient set to zero. In the present case, x is measured along the axis of the cone. Hence the metric h_1 will be equal to secant of the half angle of the cone (from eq. 24). The metric h_2 for this axisymmetric case is the tangent of the half angle multiplied by (πx) (eq. 25) and g_{12} is equal to zero. The inviscid velocity is equal to unity in accordance with conical flow.

The boundary layer grid has a step size of $\Delta\xi$=0.005 from ξ=0.0 to ξ=0.09. From ξ=0.09 to ξ=0.13, $\Delta\xi$ value is decreased to 0.001 where mass injection introduces larger streamwise changes. For $\xi > 0.13$, $\Delta\xi$ reverts back to 0.005. The grid in the wall normal direction has 41 points with ζ_e=8.0 and stretching corresponding to k_e=1.02.

Fig. 12 presents the results obtained for the three cases of wall condition and plotted similar to the results given in Ref. 11 (the x dimension in Ref. 11 is multiplied by a factor of 3.281). Fig. 12a shows the variation of the skin friction along a ray of the cone as a function of the axial distance.

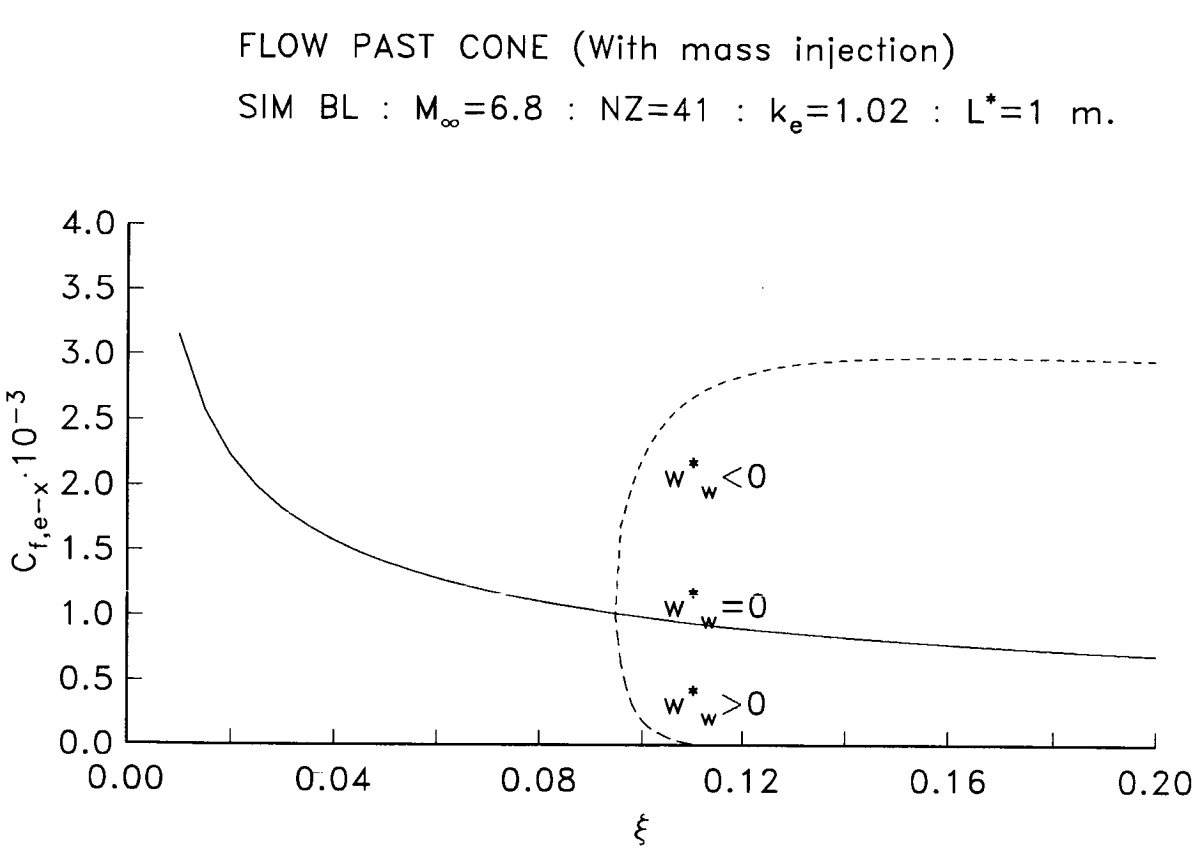
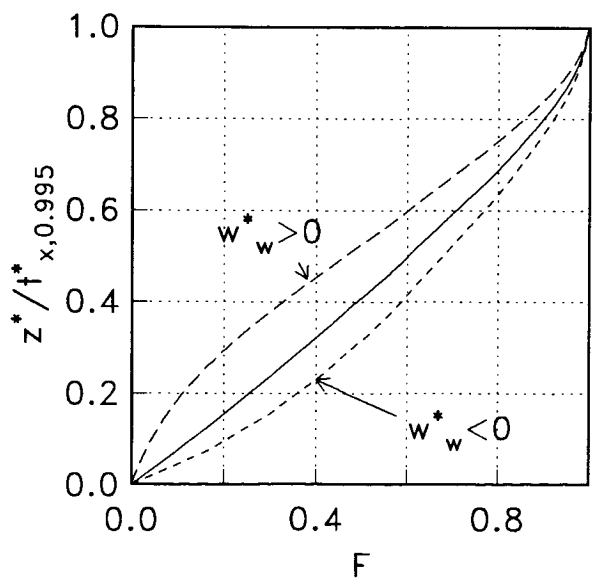


Fig. 12a Variation of Skin Friction for Supersonic Cone Flow



Profile at ξ=0.1

Fig. 12b Effect of Mass Injection on Velocity Profiles

The mass injection/suction at ξ=0.096 produces the expected decrease/increase in $C_{f,x-\infty}$. The values and trend of the curve agree well with the tabulated results in Ref. 11. For the wall blowing case, the boundary layer separates at about ξ=0.11 as determined by the large increase in iterations needed for convergence.

Fig. 12b shows the velocity profiles at the station ξ=0.1 of the cone symmetry line with or without blowing/suction. The profile for the case $w_w^*<0$ is fuller than the profile for the case $w_w^*>0$ as expected. The profiles compare very well with results from Ref. 11.

The solution of supersonic flow past a cone at 0° angle of attack is basically similar to flow past a flat plate. The laminar supersonic cone rule is applicable to this case (see Ref. 12). The results from this case can be verified to follow the rule $C_{f,cone} = \sqrt{3}\, C_{f,flat\ plate}$ for the same local Reynolds number.

Case 4: Symmetry Line Flow on an Ellipsoid at α=6°

This test case was selected to validate the equations and solution scheme for symmetry line flow. The cross flow velocity is zero on the symmetry line; however, the cross-flow velocity gradient is non-zero and has a strong effect on the boundary layer solution. The cross flow gradient influence is brought into the system of equations by solving the equation obtained by taking the η derivative of the η momentum equation along with the ξ momentum equation. The complete equations for symmetry line flow are given in Table 4.

The symmetry line flow past an ellipsoid at α along the windward and leeward lines is a good test case because the inviscid conditions and the metrics can be specified in closed form, thereby eliminating extraneous errors in the solution arising from geometry and the inviscid flow. The grid is orthogonal everywhere on the ellipsoid (g_{12}=0). The metrics h_1 and h_2 are functions only of ξ.

An ellipsoid of revolution with a fineness ratio e_e (ratio of semi-minor to semi-major axes) of 0.25 is considered here. The body is assumed to be 2 feet long. The flow is assumed at a Mach number of 0.05 at an angle of attack of α=6°. The inflow conditions correspond to p^*_∞= 2116 lb/ft^2, T^*_∞=520°R.

The streamwise coordinate ξ is measured along the axis of the body, with ξ=0 at the leading extremity and ξ=1 at the aft end. The cross-flow coordinate η is measured along the surface in a direction perpendicular to the axis. At the windward line of symmetry, η=0 and at the leeward line of symmetry, η=1. Corresponding to the above definition of coordinates, the metrics can be obtained analytically as,

$$h_2 = \pi r_e$$

where r_e is the non-dimensional radius at any location given as,

$$r_e = e_e \sqrt{\xi(1-\xi)}$$

$$h_1 = \sqrt{1+\left(\frac{\partial r}{\partial \xi}\right)^2}$$

$$\text{where, } \frac{\partial r}{\partial \xi} = -\frac{1}{2} r_e (2\xi-1)$$

$$g_{12} = 0$$

The metrics h_1 and h_2 are thus functions only of ξ. It can be seen that $h_1 \to \infty$ as $\xi \to 0$ or $\xi \to 1$. The extremities of the body are thus geometric singularities in the present coordinate system.

The inviscid velocities at any location (ξ,η) can be obtained as follows:

$$u_e = \frac{1}{h_1}\left\{q_1 - q_2\frac{\partial r}{\partial \xi}\,\mathrm{Cos}(\pi\eta)\right\}$$

$$v_e = q_2\mathrm{Sin}(\pi\eta)$$

$$q_1 = \frac{2\mathrm{Cos}\alpha}{2-q_3} \;;\; q_2 = \frac{2\mathrm{Sin}\alpha}{2-q_4}$$

$$q_3 = \frac{2e_e^2}{e_s^2}(e_t - e_s) \;;\; q_4 = \frac{e_e^2}{e_s^2}\left(\frac{e_s}{e_e^2} - e_t\right)$$

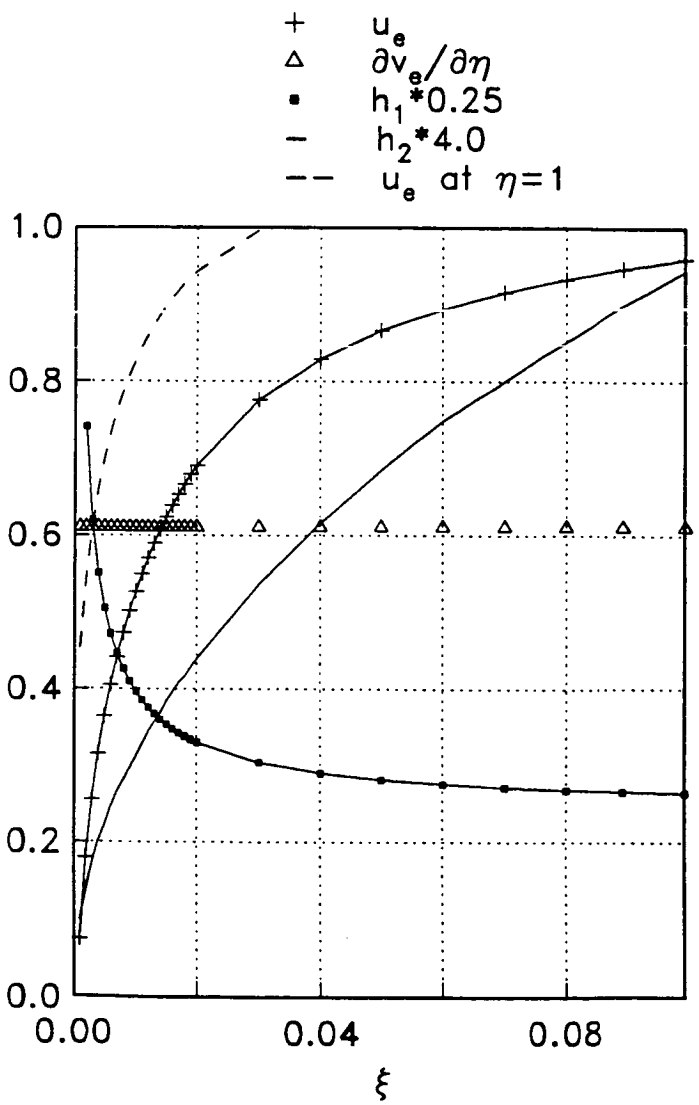
$$e_s = \sqrt{1-e_e^2} \;;\; e_t = \tfrac{1}{2}\ln\left\{\frac{1+e_s}{1-e_s}\right\}$$

Fig. 13 shows the axial variation of u_e, $\frac{\partial v_e}{\partial \eta}$, h_1 and h_2 along the axis on the windward line for the first 10% length of the body (till ξ=0.1). The stagnation point is located on the η=0 line at ξ=0.0005 approximately. Hence we place the i=2 grid point downstream of this point at ξ=0.001. The cross flow velocity gradient is a constant for all ξ.

The initial profiles are generated by solving the 3D stagnation point equations with $C_{r,s}$=1. The symmetry lines along the windward and leeward rays are solved by a ξ direction march. The edge coefficient $\tilde{A}_2$ is an important quantity since this determines the smoothness of the solution near the starting plane. The value of $\tilde{A}_2$ is equal to -1 in the STG BL equations. In the SYM BL equations the value of $\tilde{A}_2$ at i=2 should be close to -1 for a smooth continuation of the solution starting from the initial stagnation point profile. Numerical evaluation of $\tilde{A}_2$ from the equation,

$$\tilde{A}_2 = -\frac{s_1}{h_1 h_2\sqrt{\rho_e\mu_e s_1 u_e}}\frac{\partial}{\partial \xi}\sqrt{\rho_e\mu_e s_1 u_e}$$

by first order backward differencing at i=2 gives a value of $\tilde{A}_2$ close to -1.5, which will give a non-smooth solution in the ξ direction. This is because $\tilde{A}_2$ is in $\frac{0}{0}$ form as written above as $\xi \to 0$ and first order differencing does not produce the correct value. However, if we rewrite the above equation as,



ξ	u_e	$\partial v_e/\partial \eta$	h_1	h_2	$u_{e,\eta=1}$
0.001	0.076	0.611	4.072	0.025	0.453
0.002	0.180	0.611	2.961	0.035	0.546
0.003	0.255	0.611	2.482	0.043	0.611
0.004	0.315	0.611	2.204	0.050	0.661
0.005	0.364	0.611	2.019	0.055	0.702
0.006	0.405	0.611	1.886	0.061	0.735
0.007	0.442	0.611	1.785	0.066	0.764
0.008	0.473	0.611	1.705	0.070	0.788
0.009	0.502	0.611	1.640	0.074	0.810
0.010	0.527	0.611	1.586	0.078	0.829
0.011	0.550	0.611	1.541	0.082	0.846
0.012	0.571	0.611	1.502	0.086	0.861
0.013	0.590	0.611	1.468	0.089	0.875
0.014	0.608	0.611	1.439	0.092	0.887
0.015	0.624	0.611	1.412	0.096	0.899
0.016	0.639	0.611	1.389	0.099	0.909
0.017	0.653	0.611	1.368	0.102	0.919
0.018	0.666	0.611	1.350	0.104	0.928
0.019	0.679	0.611	1.333	0.107	0.936
0.020	0.690	0.611	1.317	0.110	0.943
0.030	0.776	0.611	1.214	0.134	0.996
0.040	0.829	0.611	1.159	0.154	1.026
0.050	0.867	0.611	1.125	0.171	1.045
0.060	0.894	0.611	1.102	0.187	1.058
0.070	0.916	0.611	1.085	0.200	1.067
0.080	0.933	0.611	1.072	0.213	1.073
0.090	0.947	0.611	1.062	0.225	1.078
0.100	0.959	0.611	1.054	0.236	1.082

Fig.13 Variation of Metrics and Inviscid Conditions on Symmetry Line
Ellipsoid at 6° angle of attack

$$\tilde{A}_2 = -\frac{1}{2} + \frac{\tilde{A}_1}{\chi}\frac{\partial \chi}{\partial \xi} \quad \text{where,} \quad \chi = h_2\sqrt{\rho_e \mu_e u_e}$$

then the numerical evaluation gives the correct behavior of $\tilde{A}_2$ as $\xi \rightarrow 0$.

Fig. 14 shows the skin friction and boundary layer thickness (99% value) variation along the windward and leeward lines until separation near the rear stagnation point. These values agree well with results given by Wang[13] (note that for comparisons with the present results, Wang's results must be multiplied by $2\sqrt{2}$ due to different definitions of $C_{f_{-\infty}}$ and ξ).

Fig. 15 shows the streamwise velocity profiles and the crossflow velocity gradient profiles at different axial stations on the windward and leeward symmetry lines. Note that the crossflow velocity gradient on the leeward line has a cross-over profile at ξ=0.6 and ξ=0.81.

Fig. 16 shows the variation of t^* and C_{f-x} with ξ for the first 1% of the body on the windward symmetry line. The ξ distribution used for this run is: ξ_1=0.0, ξ_2=0.001, ξ_3=0.0011, ξ_4=0.0012 ... ξ_{22}=0.003. It can be seen from Fig. 16(a) that the variation of t^* is not smooth near the stagnation point. This is due to the fact that we have assumed the stagnation point to be at ξ=0, which is not physically correct and feeds in wrong edge values of s_1, $\frac{\partial u_e}{\partial \xi}$ and $\frac{\partial \chi}{\partial \xi}$. The stagnation point is approximately at ξ=0.0005 and correspondingly, we can have a grid originating from this point with ξ_1=0.0005, ξ_2=0.001, ξ_3=0.0011, ξ_4=0.0012 etc. Fig. 16b shows the boundary layer thickness and skin friction variation for this new grid. It can be seen that the variation is smooth in the nieghborhood of the stagnation point.

Fig. 17 shows the corresponding improvement in the solution near the nose region on the leeward line. In this case, ξ_1 is taken to be equal to -0.0005 since the stagnation point is on the windward line. The edge values of $\frac{\partial h_1}{\partial x}$, $\frac{\partial h_2}{\partial x}$, $\frac{\partial u_e}{\partial x}$, and $\frac{\partial \chi}{\partial x}$ at i=2 are evaluated using forward differences in this case to avoid differencing across the metric singularity at ξ=0.

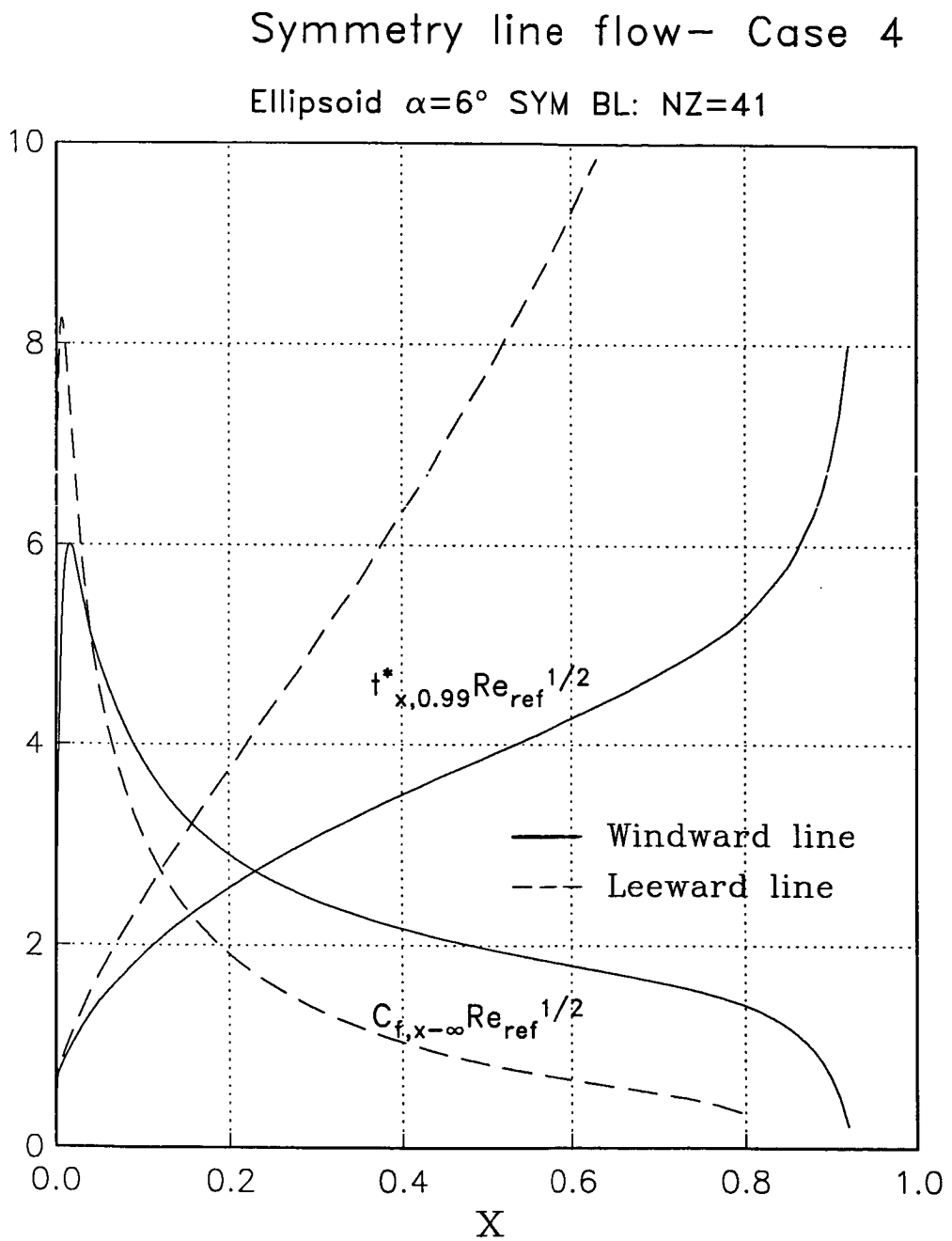


Fig.14 Boundary Layer Results on Symmetry Lines
Ellipsoid at 6° angle of attack

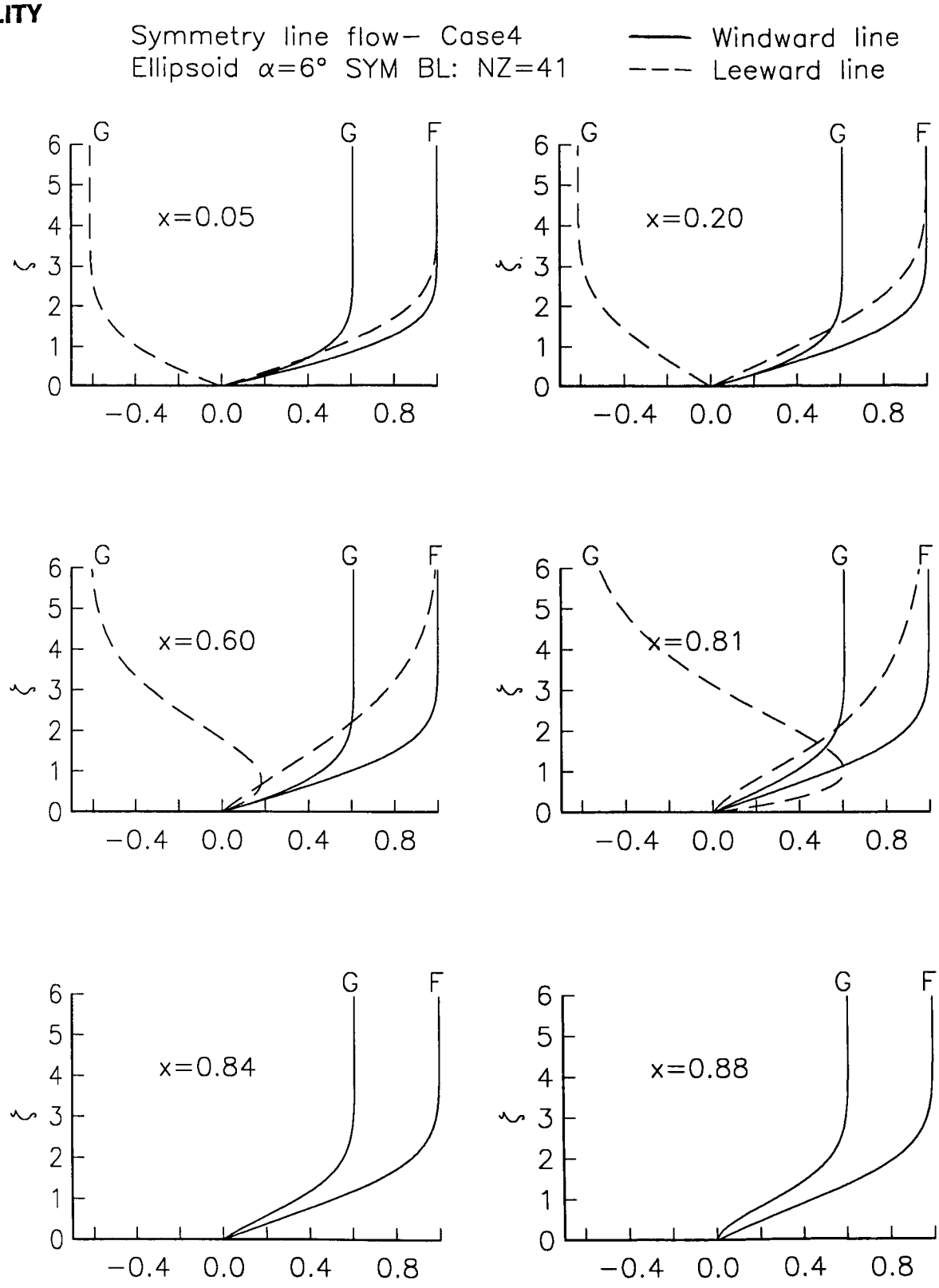


Fig.15 Boundary Layer Profiles on Symmetry Lines
Ellipsoid at 6° angle of attack

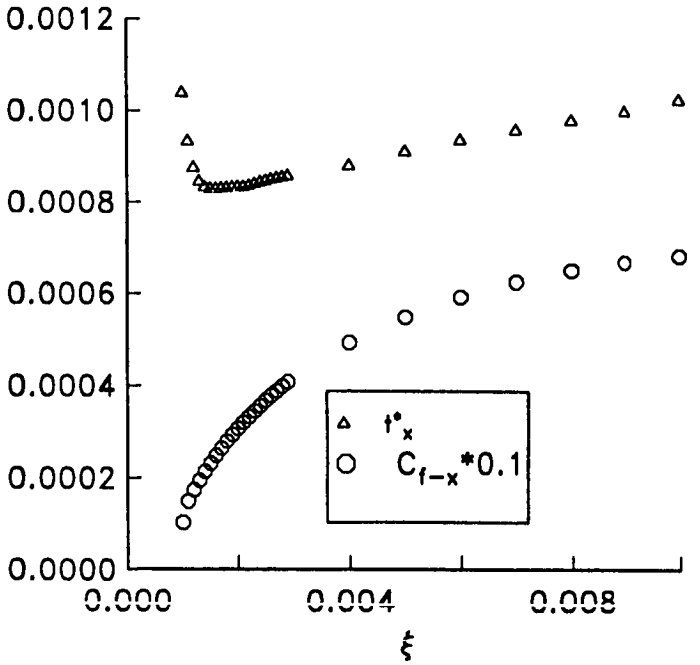
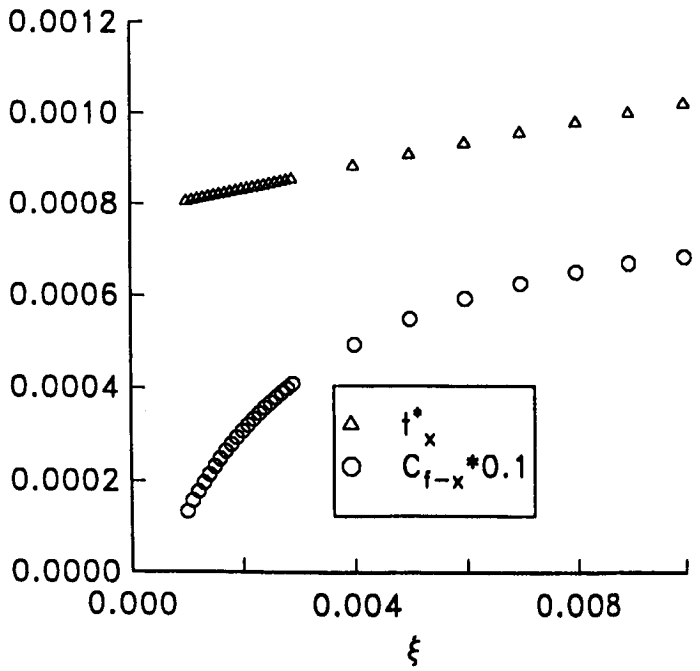


Fig. 16a

New ξ distribution stag. pt. location assumed to be at ξ=0.0



New ξ distribution with edge coefficients adjusted for stag. pt. located at ξ=0.0005

Fig. 16a BL Results (windward line) with Grid Originating from ξ=0.0

Fig. 16b BL Results with Grid Originating from ξ=0.0005

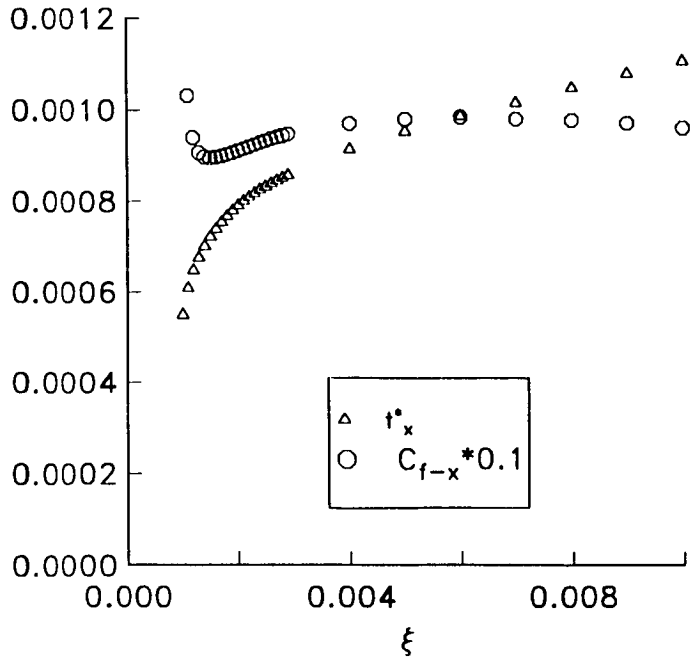
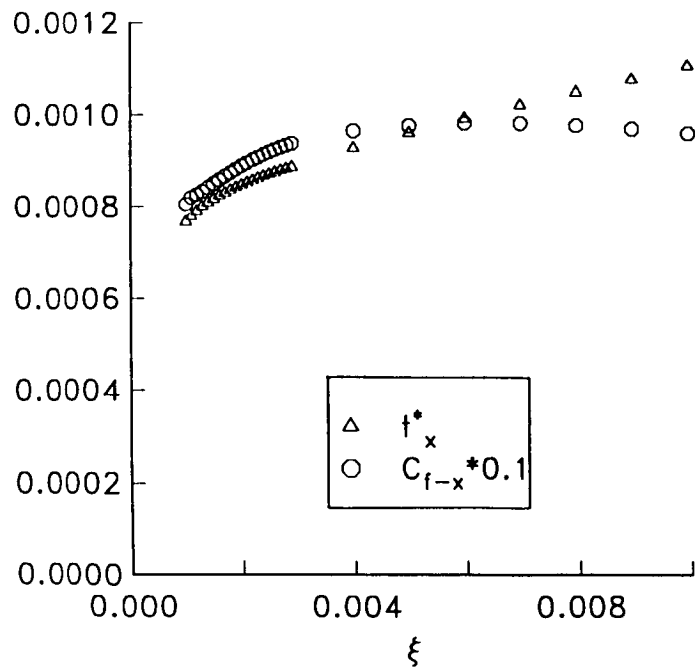


Fig. 17a

New ξ distribution stag. pt. location assumed to be at ξ=0.0



New ξ distribution with edge coefficients adjusted for stag. pt. located at ξ=0.0005

Fig. 17a BL Results (leeward line) with Grid Originating from ξ=0.0

Fig. 17b BL Results with Grid Originating from ξ=-0.0005

Case 5: 2D Subsonic Flow Past an Airfoil

This test case is that of two-dimensional flow past an airfoil at zero angle of attack and comparison with 2D Navier-Stokes results at two chord Reynolds numbers. The airfoil has NACA 0012 cross section and the free stream conditions correspond to M_∞ =0.5, p^*_∞=2116 lb/ft^2 and T^*_∞=520°R. The free stream Reynolds numbers based on chord length are Re_∞=5,000 and Re_∞=1 million. The reference lengths correesponding to these two Reynolds numbers are L^*=1.4133x10^{-3} and L^*=0.28266 ft. respectively.

The inviscid results in the form of pressure coefficient values on the airfoil surface are obtained from a 2D navier-Stokes calculation as presented in Ref. 14. The N-S calculation is obtained on a 161x65 grid. The edge velocity u_e is calculated from the pressure distribution assuming isentropic flow.

The boundary layer calculation is done by first obtaining the 2D stagnation point solution at i=1 on the leading edge. The BL grid distribution over the airfoil surface is the same as the grid used for the Navier-Stokes run; consequently, no interpolation is involved in this case. The streamwise coordinate ξ is defined and measured along the airfoil surface, so that h_1 is equal to unity. The boundary layer downstream of the stagnation point is calculated by solving the symmetry line equations with the cross flow gradient terms suppressed.

Fig. 18 shows the variation of the skin friction coefficient and the boundary layer thickness on the airfoil with surface arc length for Reynolds number of 5,000. Shown also are the skin friction values computed from the Navier Stokes run, which compare well with the present results.

Fig. 19 shows the results for Reynolds number of 1 million. The Navier-Stokes result shows an increase in skin friction near ξ=0.45, where the turbulence model was turned on. The boundary layer calculation does not include turbulence modelling and predicts laminar separation at this point.

Further details about this test case can be found in the Appendix.

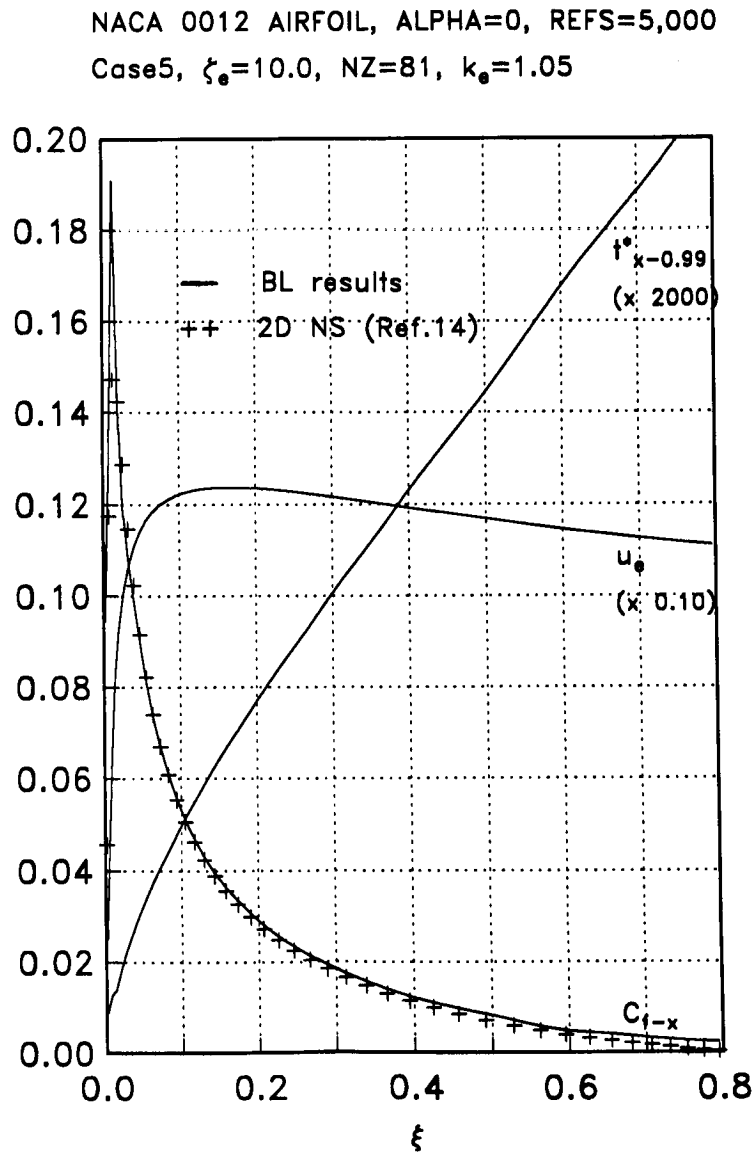


Fig.18 BL Results for Case5, Chord Reynolds number=5,000

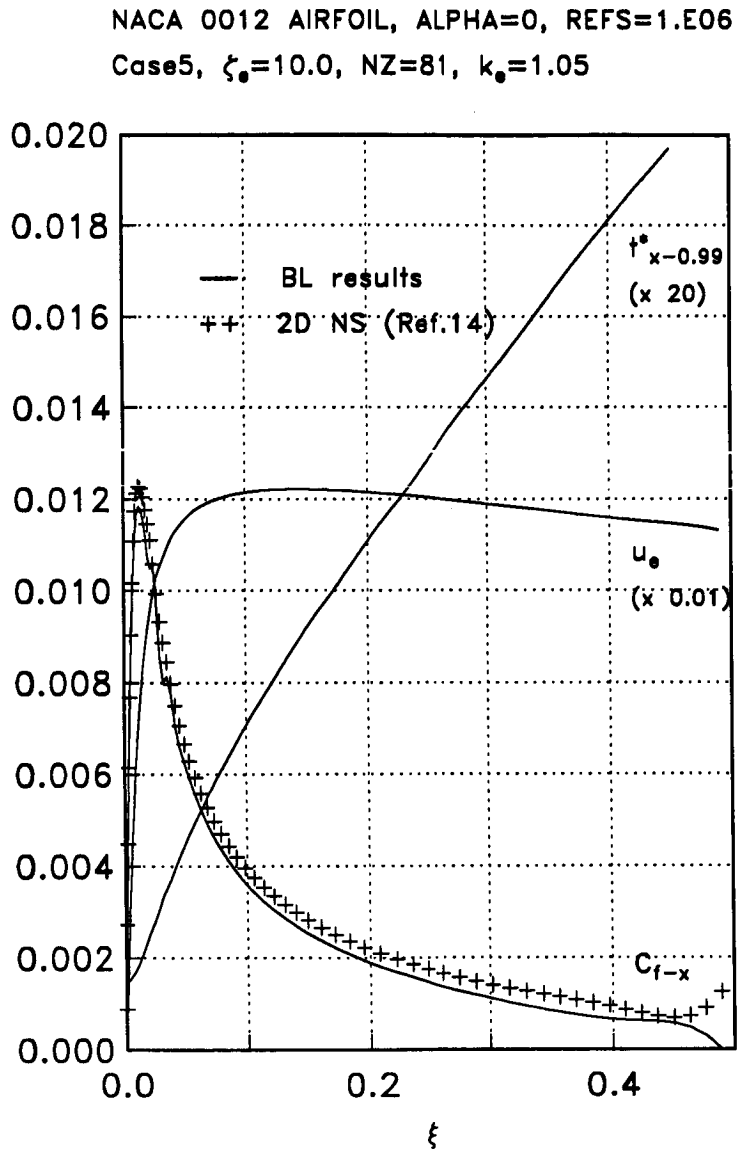


Fig.19 BL Results for Case5, Chord Reynolds number=1 Million

Case 6: Flow Past an Infinite Swept Elliptical Wing

This test case has been selected to validate the solution of the infinite swept wing equations presented in Table 6. The elliptical wing considered here has a sweep Λ of 40° and a fineness ratio of 0.125 for the elliptical cross section normal to the leading edge. Flow is assumed to be at an angle of attack α of 5° and low Mach number (M_∞=0.1, p^*_∞=2116 lb/ft^2 and T^*_∞=520°R). The reference length is the chord length normal to the leading edge (L^*=1 ft). The geometry corresponds to that used in a study by Monnoyer[15].

The boundary layer coordinate used for this study is an orthogonal one. A non-orthogonal grid similar to that for a wing can also be used; however, in the present case, the orthogonal grid yields simpler expressions for the inviscid velocities. The ξ direction is assumed to be in a direction normal to the wing leading edge over the wing surface and the η direction parallel to the leading edge. Both ξ and η are measured in their respective directions, so that $h_1 = h_2 = 1$ and g_{12}=0 due to orthogonality. Note that ξ is zero at the stagnation point, not the leading edge.

The inviscid conditions can be calculated by potential theory. Ref. 16, for example, gives a closed form expression for the complex potential for this flow, from which the complex velocity can be obtained by differentiation. The absolute of the complex velocity $u_{e,2D}$ past the normal cross section of the wing can then be calculated.

By classical sweep theory, the incompressible flow past the swept wing in the orthogonal coordinate system is given by, $u_e = u_{e,2D}\,\mathrm{Cos}(\Lambda)$ and $v_e = \mathrm{Sin}(\Lambda)$. Fig. 20 shows the inviscid edge velocity $u_{e,2D}$ for an ellipse of fineness ratio 0.125 at α=5°. The plots show the cross section of the ellipse and the $u_{e,2D}$ distribution near the leading edge, at different magnifications. Also shown is the variation of $u_{e,2D}$ with ξ.

The pressure coefficient C_p on the body for incompressible flow is given by $1-(u_{e,2D}\,\mathrm{Cos}\Lambda)^2 - \mathrm{Sin}^2\Lambda$. The C_p distribution calculated for this body is significantly different from Ref. 15 (see Fig. 21). Since the input conditions do not agree, the results from this case and Ref. 15 cannot be compared.

The boundary layer solution for this type of flow starts with the solution at the attachment line which is also assumed to be infinitely swept. The Eq.s 87-90 are applicable in this case. From this starting solution, the flow over the elliptical wing is calculated by solving the infinite swept wing equations of Table 6. Fig. 22 shows the skin friction and displacement thickness

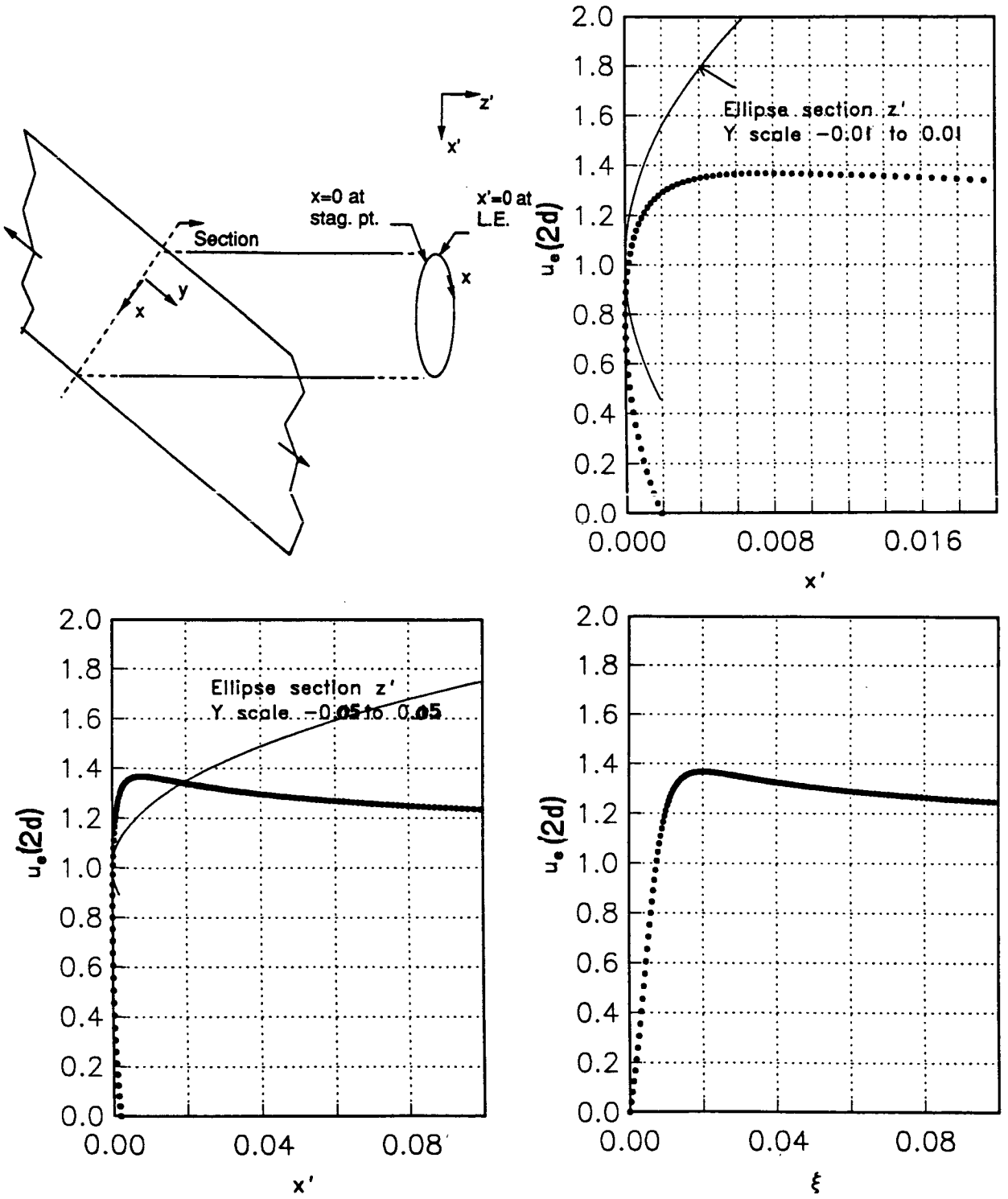


Fig.20 Inviscid Conditions for Elliptical Wing

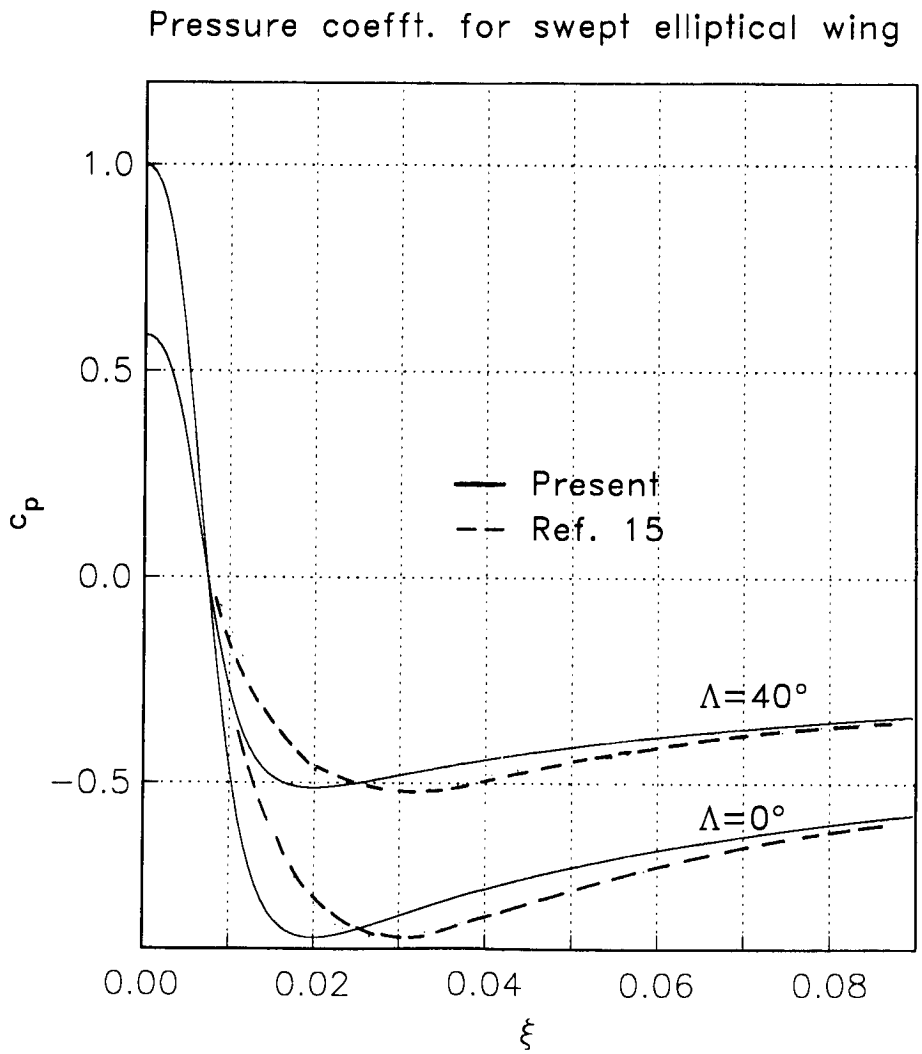
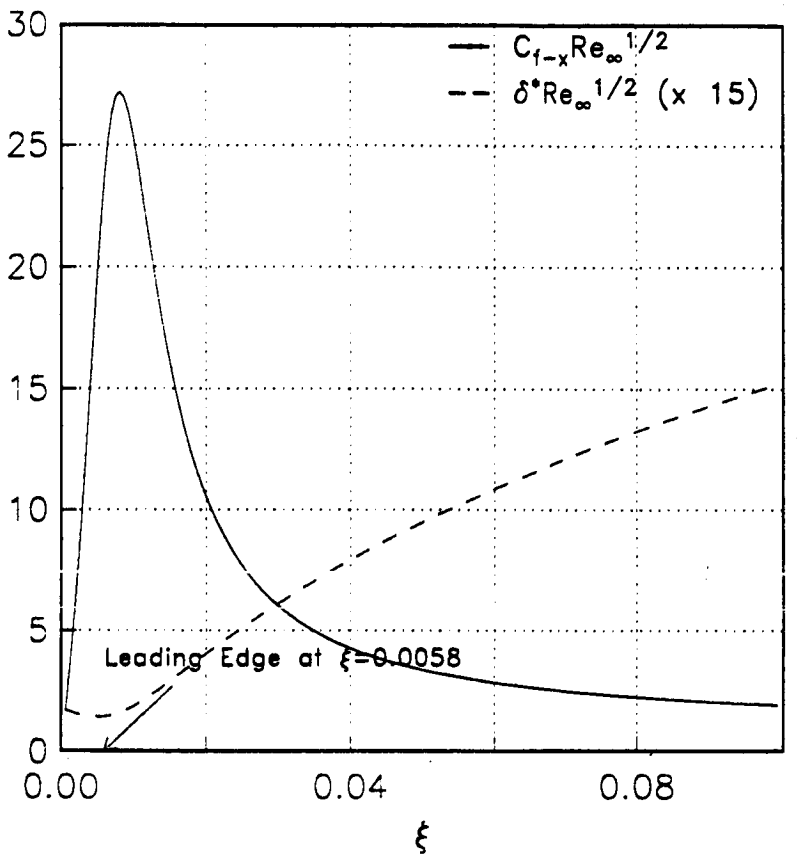
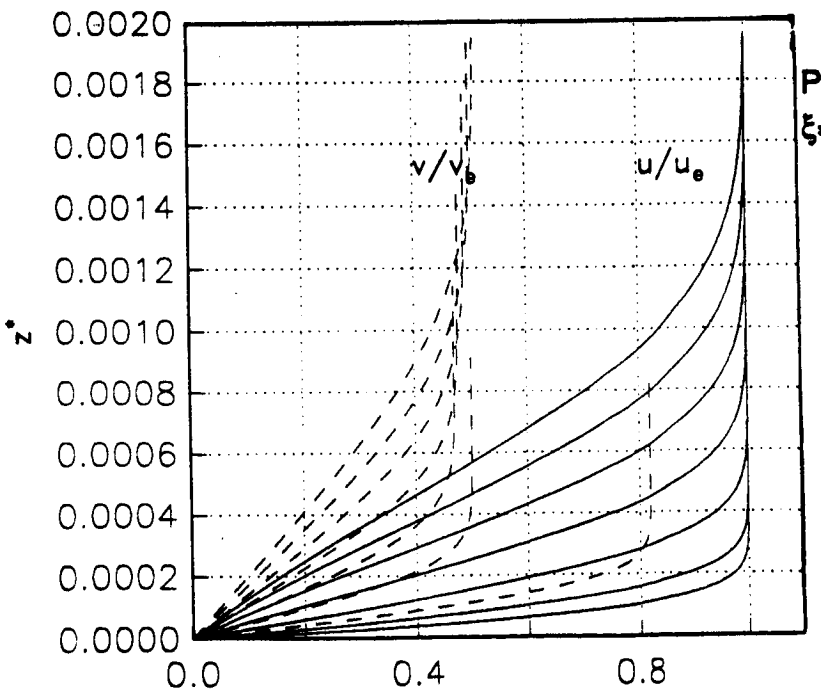


Fig. 21 Variation of Pressure Coefficient for Elliptical Wing

Swept Elliptical Wing, $\Lambda=40°, \alpha=5°$





Profiles at
ξ=0.006,0.012,0.020,
0.03,0.045,0.062,0.082

Fig.22 Boundary Layer Results for Elliptical Wing

values (multiplied by the square root of the free stream Reynolds number) on the body for the first 10% part of the wing. The figure also shows the streamwise and crossflow velocity profiles at several stations.

Case 7: Supersonic Attachment Line Flow on Swept Cylinder

This configuration corresponds to a cylinder of 1 inch radius and 60 inches span with a sweep of 60°. The flow is supersonic at M_∞=3.5, p^*_∞=188.78 psf and T^*_∞=172.75 °R and is at zero angle of attack. Fig. 23 is a sketch of the geometry with definition of the coordinate system employed. The y=0 plane is assumed to correspond to a symmetry plane. Boundary layer solution of the attachment line at the leading edge is the objective.

The inviscid flow past this geometry has a small subsonic region around the symmetry plane leading edge. This region extends between one and two radius lengths from the symmetry plane. Outside this region, the flow is fully supersonic, with an oblique shock in front of the leading edge. The present calculation is done only in this supersonic region. The subsonic region has sharp gradients and the inviscid grid used is not able to resolve fully the flow in this region.

The inviscid flow past this body has been calculated by a finite volume Runge-Kutta integration solver for the Euler equations (Ref. 17). Fig. 24a and Fig. 24b show the contours of the pressure coefficient and the absolute velocity plotted in (y,θ) plane for one half of the elliptical cross section from θ=0° to θ=90°. The large gradients occurring near the symmetry plane can be seen. The inviscid results required for the attachment line flow are the velocity values along the attachment line and the streamwise gradient of u_e. The variation of these quantities with span distance is shown in Fig. 25a. It can be seen that the quantities tend to constant values at about 20 radius lengths away from the symmetry plane.

The coordinate system used for this case is a non-orthogonal one. The y coordinate is measured in a direction perpendicular to the chord. The x coordinate is measured along the surface of the elliptical cross section in the stream direction. The reference length L^* for non-dimensionalization is the radius of the cylinder. The value of x is zero at the attachment line and equal to π/2 at θ=90°, where θ is the angle from the leading edge on the projected circular cross section, as shown in Fig. 23. The metric coefficients can be obtained from the coordinate system definition as,

$$h_1 = \frac{\pi}{2}\sqrt{\frac{\sin^2\theta}{\cos\Lambda} + \cos^2\theta}$$

$$h_2 = \frac{1}{\cos\Lambda}$$

$$g_{12} = \frac{\pi}{2}\sin\theta\sin\Lambda$$

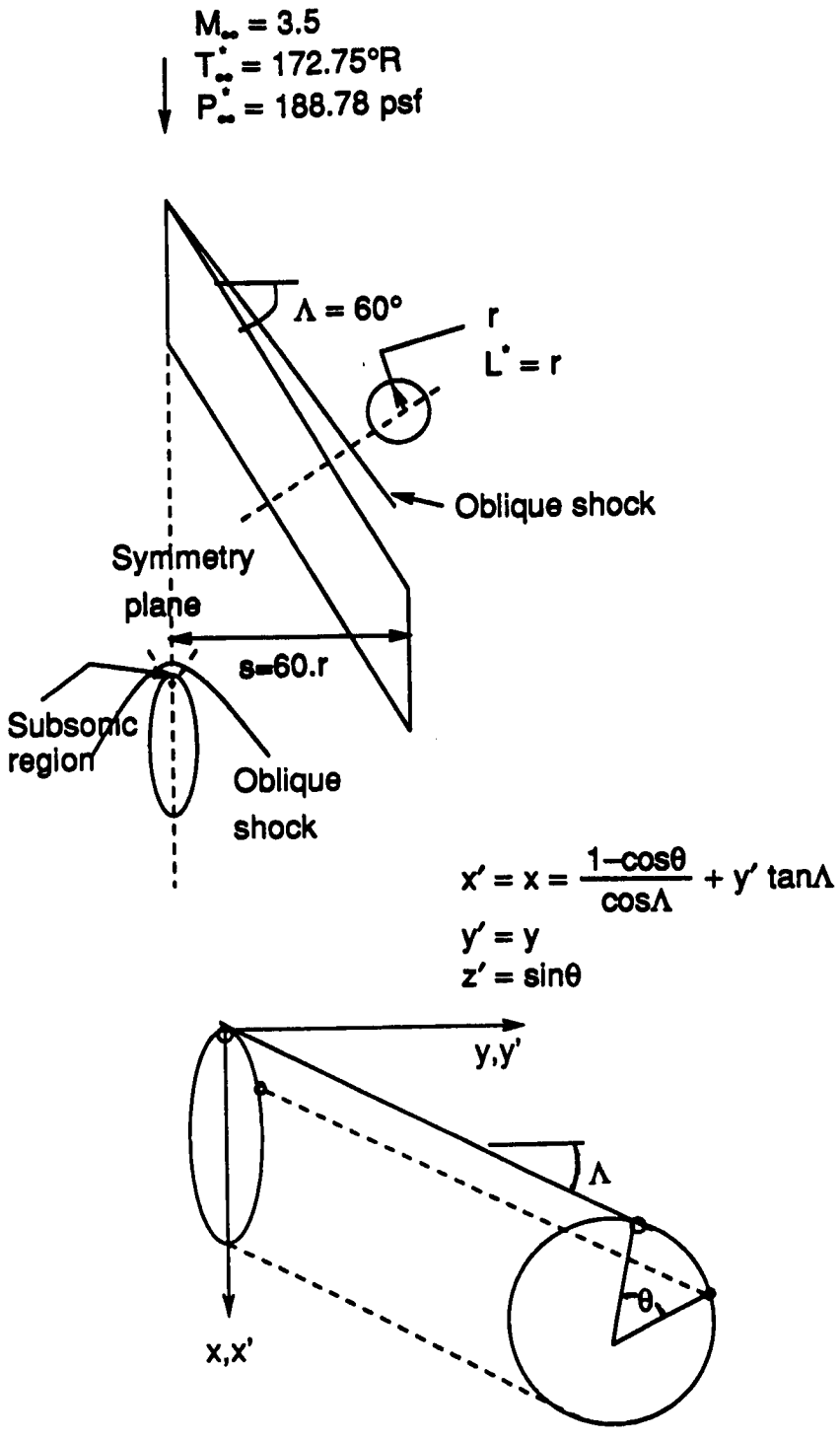


Fig.23 Coordinate System for Swept Cylinder

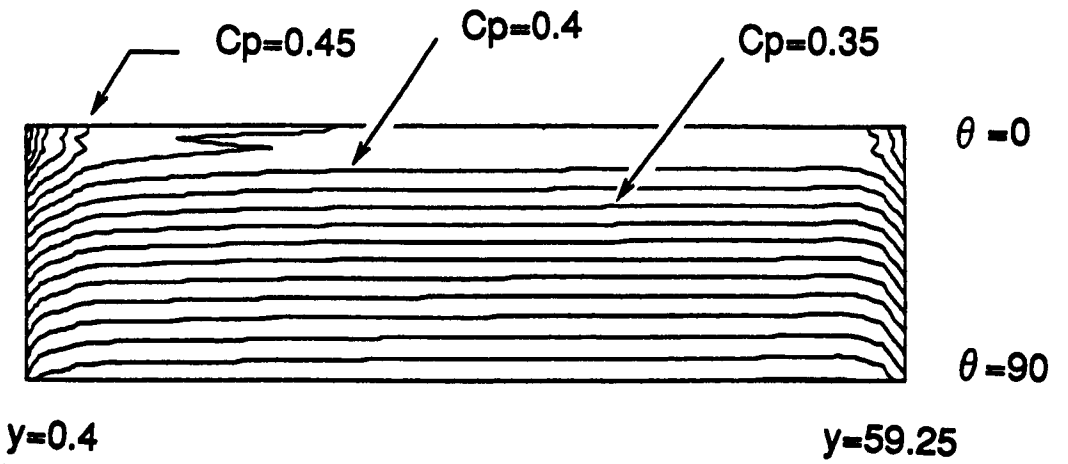


Fig.24a. Contour Plot of Pressure Coefficient for Swept Cylinder

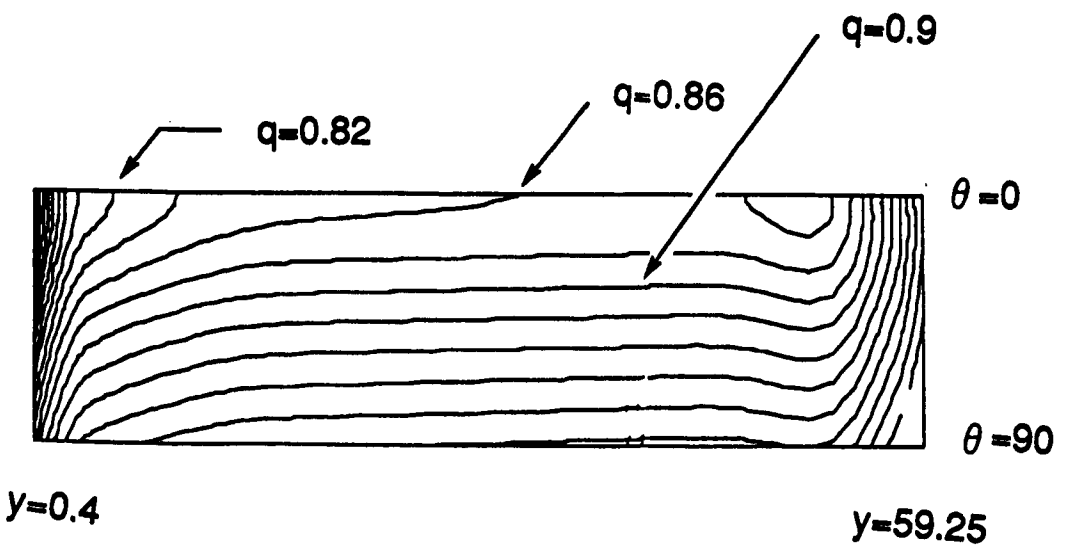


Fig.24b. Contour Plot of Absolute Velocity for Swept Cylinder

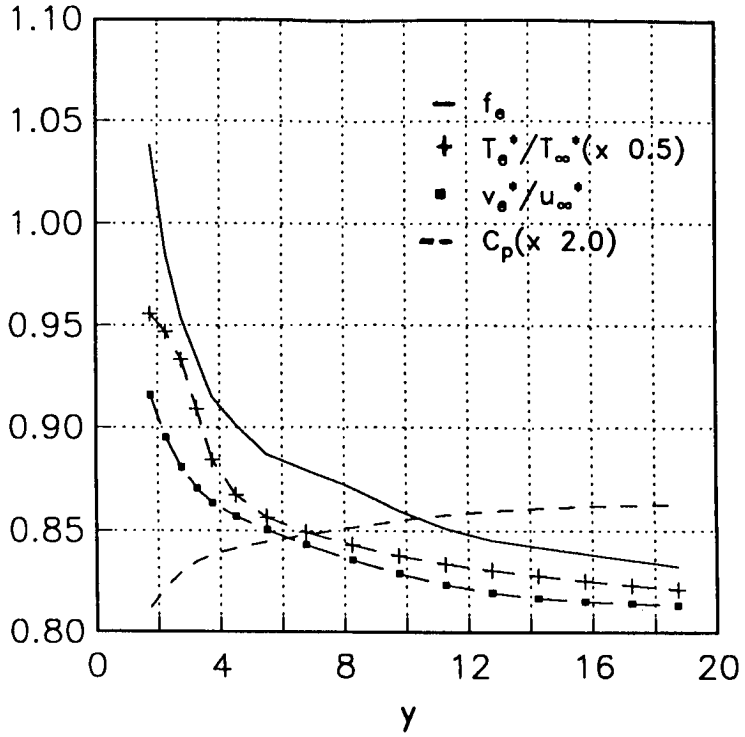


Fig.25a. Inviscid Results Along Attachment Line for Swept Cylinder

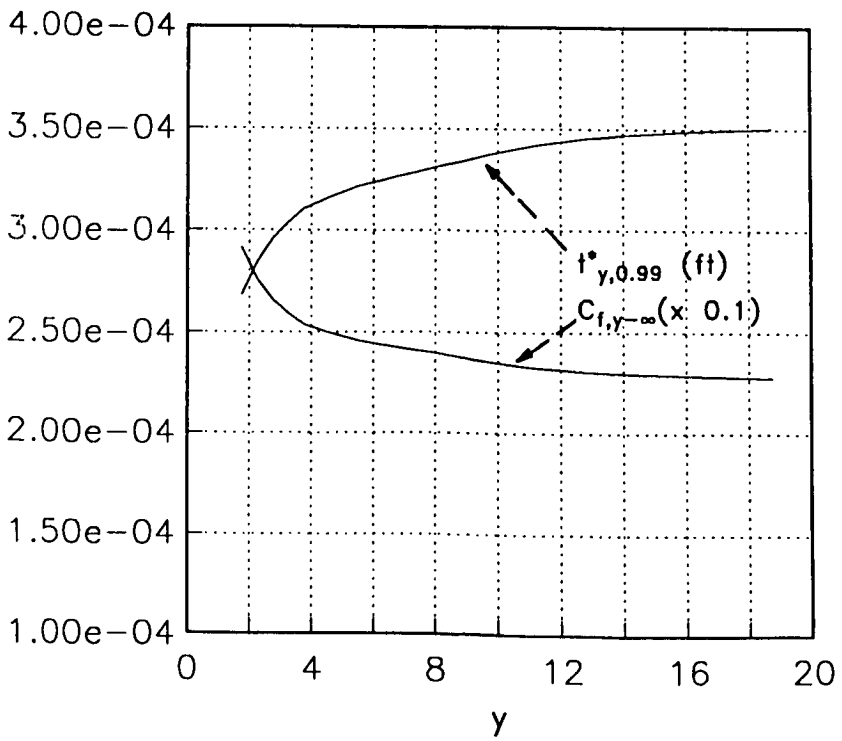


Fig.25b. BL Results Along Attachment Line for Swept Cylinder

At θ=0 on the attachment line, the grid becomes orthogonal with $h_1=\pi/2$, $h_2=\frac{1}{\cos\Lambda}$, $g_{12}=0$.

The BL calculation is started from y=1.8 approximately and marched till about y=20. At the first station j=1, the infinite swept attachment line equations are solved to initialize the calculation. Stations downstream in the y direction are solved from the general attachment line equations. The η derivatives are calculated using a 2 point backward scheme. Fig. 25b shows the variation of boundary layer thickness and skin friction along the attachment line. For y>20, the results tend to the infinite swept case. The starting profile assumption of infinite swept attachment line, though a severe one, does not seem have affected the results considerably. The same procedure is also applicable for calculating attachment line flow on blunt leading edge wings.

Case 8: Flow Past Cylindrical Post on Flat Plate

This is the first case involving solution of the full 3D BL equations. The flow geometry consists of a cylindrical obstruction mounted normal to a flat plate in subsonic flow and is a standard test case for 3D boundary layer code validation. Calculations made by Fillo and Burbank[18] and Cebeci[19] are referred to here for comparison of results.

The radius of the cylinder (0.061 m) is used as the reference length. The flat plate extends to a length of 7.49 radii in front of the cylinder center. The free-stream conditions are specified as $p_\infty^*=1.0133\text{x}10^5$ Pa, $T_\infty^*=288.15°K$ and $M_\infty=0.08963$, which corresponds to a free stream velocity of 30.5 m/s. Fig. 26 shows a sketch of the configuration.

The coordinate system for the boundary layer calculation is an orthogonal one aligned in the streamwise and crossflow directions. The metric coefficients h_1 and h_2 are equal to unity and g_{12} is equal to zero. The $\eta=0$ line corresponds to a line of symmetry. The computation region is restricted to ξ [0,2.8] and η [0,1] since only laminar attached flow is of interest here. A step size of $\Delta\xi=\Delta\eta=0.1$ is used giving a 29x11 boundary layer grid.

The inviscid flow is given by analytical formula as,

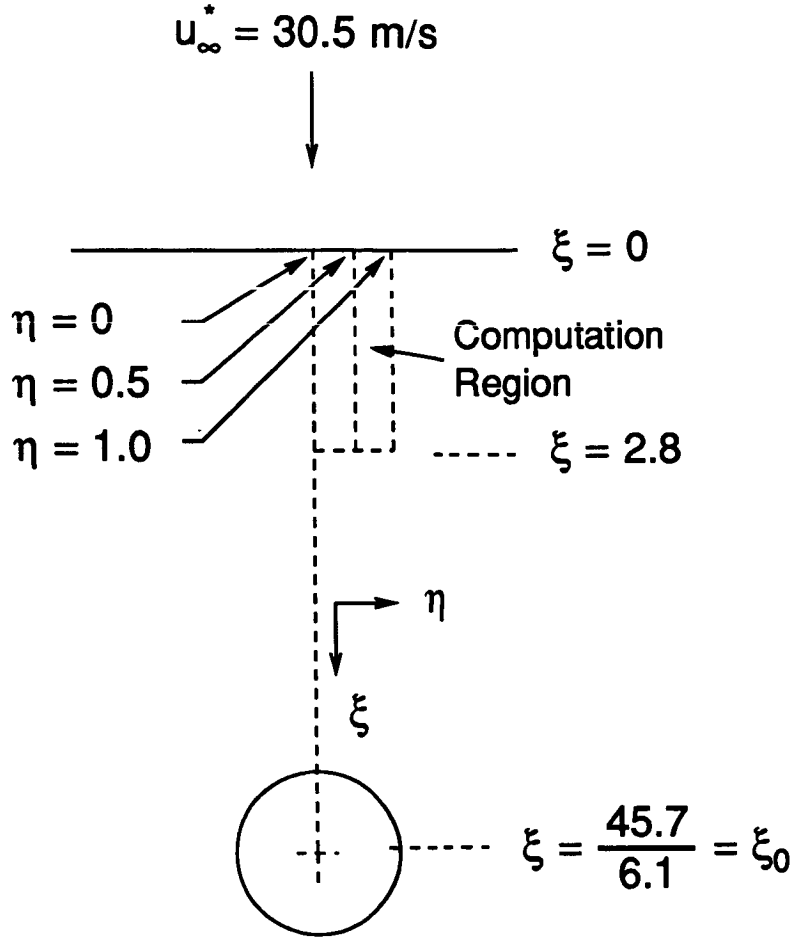
$$u_e = 1 + \frac{\eta^2-(\xi-\xi_0)^2}{[(\xi-\xi_0)^2+\eta^2]^2} \quad ; \quad \xi_0 = 7.254$$

$$v_e = \frac{-2\eta(\xi-\xi_0)}{[(\xi-\xi_0)^2+\eta^2]^2}$$

Note that v_e is zero at $\eta=0$ and is positive in the computation region.

To start the 3D BL computation, an initial plane of data is to be provided at $\xi=0$, the leading edge of the flat plate, which is a flow singularity. Similarity solution is therefore required at this location. This is obtained as explained in Case 2 by solving the similarity equations. For the η direction sweep, it is also required to specify the solution on the side boundary at $\eta=0$. The flow here is of the symmetry line type. The solution is obtained by a procedure as explained in Case 4. The definition of G is different depending on the equations solved. It is equal to zero for the initial profile, since it is assumed that the upstream influence of the cylinder at that location is negligible. For the symmetry line solution G is defined as $\partial v/\partial y$ and in the 3D region G is defined as equal to v_e. The different definitions are to be properly accounted for when calculating ξ and η derivatives of G near the boundary.

$$u_e = 1 + \frac{\eta^2-(\xi-\xi_0)^2}{[(\xi-\xi_0)^2+\eta^2]^2} \quad ; \quad v_e = \frac{-2\eta(\xi-\xi_0)}{[(\xi-\xi_0)^2+\eta^2]^2}$$



L^* = Cylinder radius = 0.061 m

Fig.26 Flow Configuration for Cylinder on Flat Plate

The 3D solution is obtained by marching in ξ direction and for each ξ location, marching in the positive η direction. Two η differencing schemes are used here; the first one is the 'Z' scheme as given by Eq. 127; the second scheme is the 'L' scheme as given by Eq. 129. The edge coefficients are evaluated as given in Table 2. Certain simplifications are possible since metric quantities are constant. Since the crossflow velocity is always positive in the computation region, the calculations are stable for any step size.

Fig. 27 shows a comparison of the F' values at the wall at three span locations. Data from Ref. 18 as well as the 'Z' scheme and the 'L' scheme results are shown. In all the three cases, good agreement can be seen.

Fig. 28 shows contour plots of F'_w and G'_w in the computation region. The variation is smooth at both initial and side boundaries. Table 11 is a table of the values of F'_w and G'_w at η=0.0., η=0.5 and at η=1.0, for several ξ stations and can be directly compared with similar tables in Ref. 18 and Ref. 19. Agreement is obtained to four digit accuracy in all cases.

Fig. 29 contains plots of the boundary layer parameters $t^*_{x-0.99}$, θ^*_x, $C_{f-x,\infty}$ and $C_{f-y,\infty}$ plotted in the η direction for four ξ stations. The variation near the boundaries is smooth and continuous.

Details of the program and routines used are presented in the Appendix.

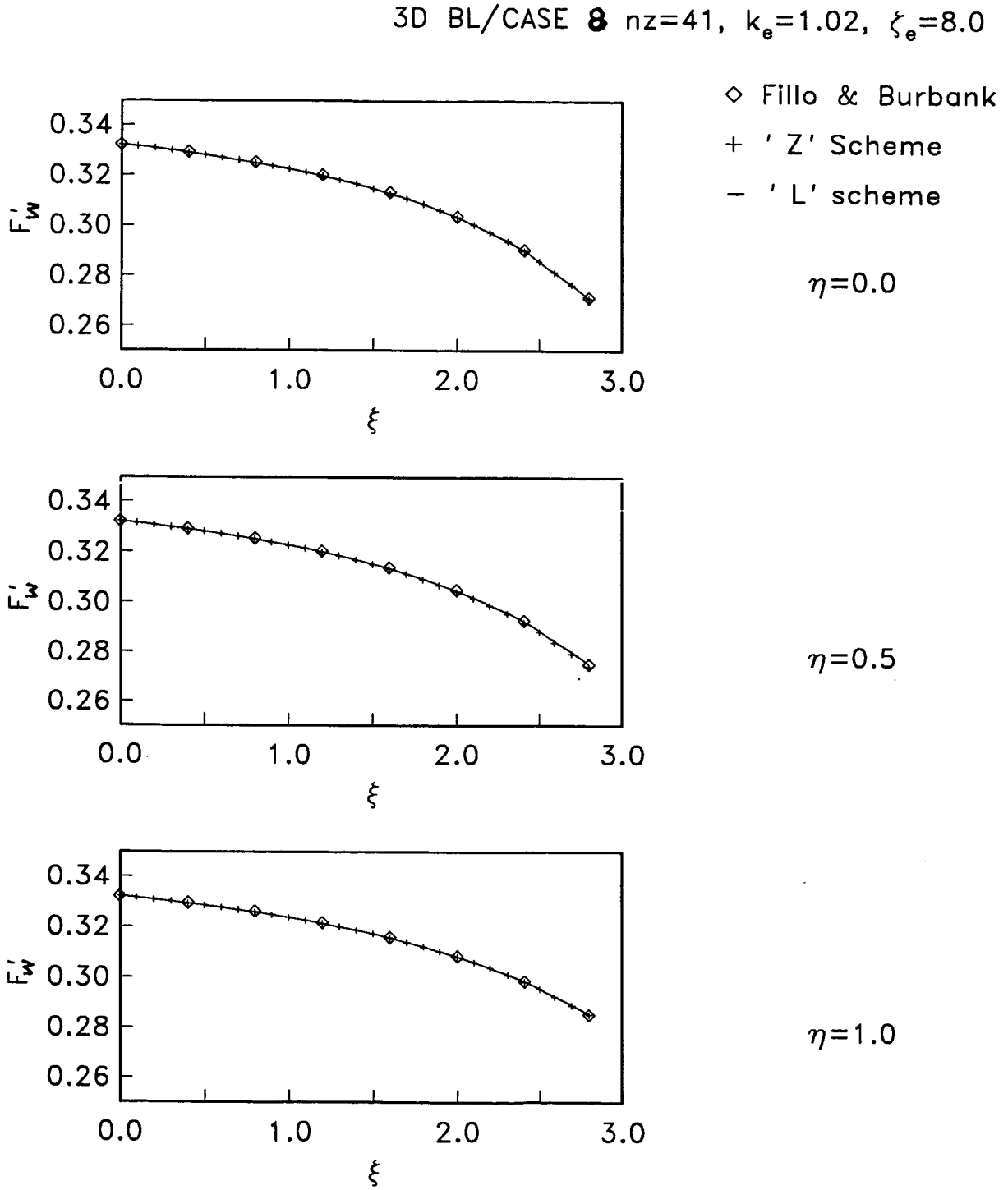
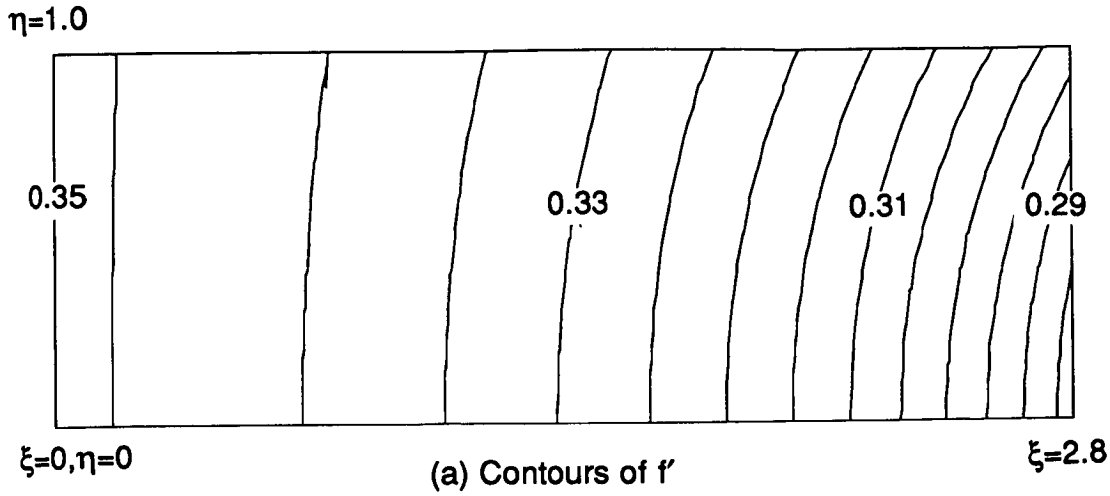
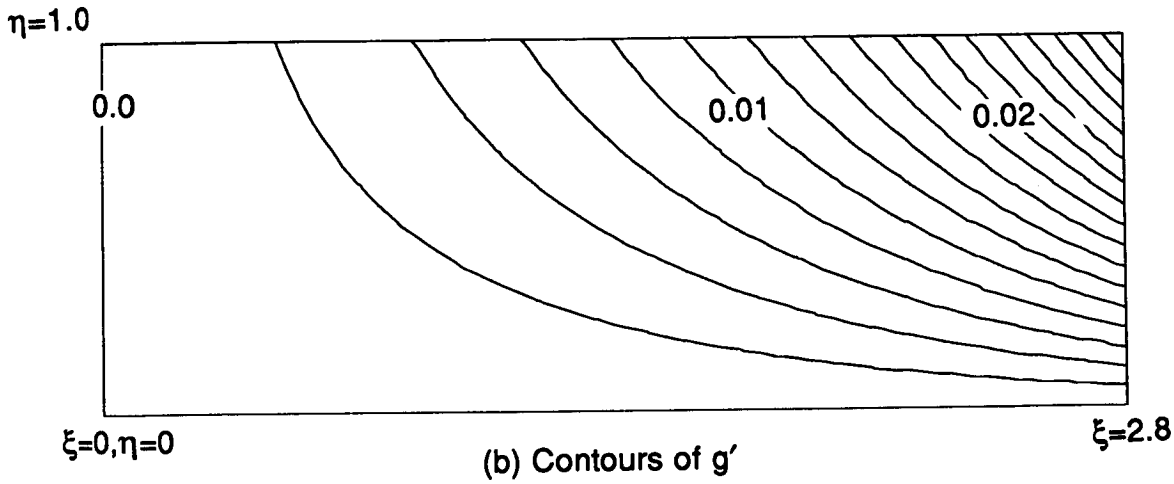


Fig.27 F'_w Variation for Cylinder on Flat Plate Flow



(a) Contours of f′



(b) Contours of g′

Fig.28. Contour Plots of F'_w and G'_w

'Z' scheme	y=0.0		y=0.50		y=1.0	
x(i)	f'(j=1)	g'(j=1)	f'(j=6)	g'(j=6)	f'(j=11)	g'(j=11)
0.0	0.33207	0.00000	0.33207	0.00000	0.33207	0.00000
0.4	0.32889	0.00000	0.32896	0.00081	0.32926	0.00156
0.8	0.32481	0.00000	0.32498	0.00184	0.32573	0.00350
1.2	0.31953	0.00000	0.31988	0.00320	0.32129	0.00605
1.6	0.31257	0.00000	0.31319	0.00508	0.31560	0.00953
2.0	0.30313	0.00000	0.30417	0.00777	0.30820	0.01445
2.4	0.29000	0.00000	0.29177	0.01174	0.29834	0.02157
2.8	0.27130	0.00000	0.27430	0.01776	0.28511	0.03214

'L' scheme	y=0.0		y=0.50		y=1.0	
x(i)	f'(j=1)	g'(j=1)	f'(j=6)	g'(j=6)	f'(j=11)	g'(j=11)
0.0	0.33207	0.00000	0.33207	0.00000	0.33207	0.00000
0.4	0.32889	0.00000	0.32898	0.00082	0.32926	0.00156
0.8	0.32481	0.00000	0.32505	0.00184	0.32574	0.00350
1.2	0.31953	0.00000	0.32000	0.00320	0.32130	0.00605
1.6	0.31257	0.00000	0.31339	0.00508	0.31562	0.00953
2.0	0.30313	0.00000	0.30449	0.00778	0.30823	0.01445
2.4	0.29000	0.00000	0.29231	0.01175	0.29844	0.02157
2.8	0.27130	0.00000	0.27523	0.01778	0.28536	0.03214

Table 11. Numerical Values of F'_w and G'_w for Cylinder on Flat Plate Flow

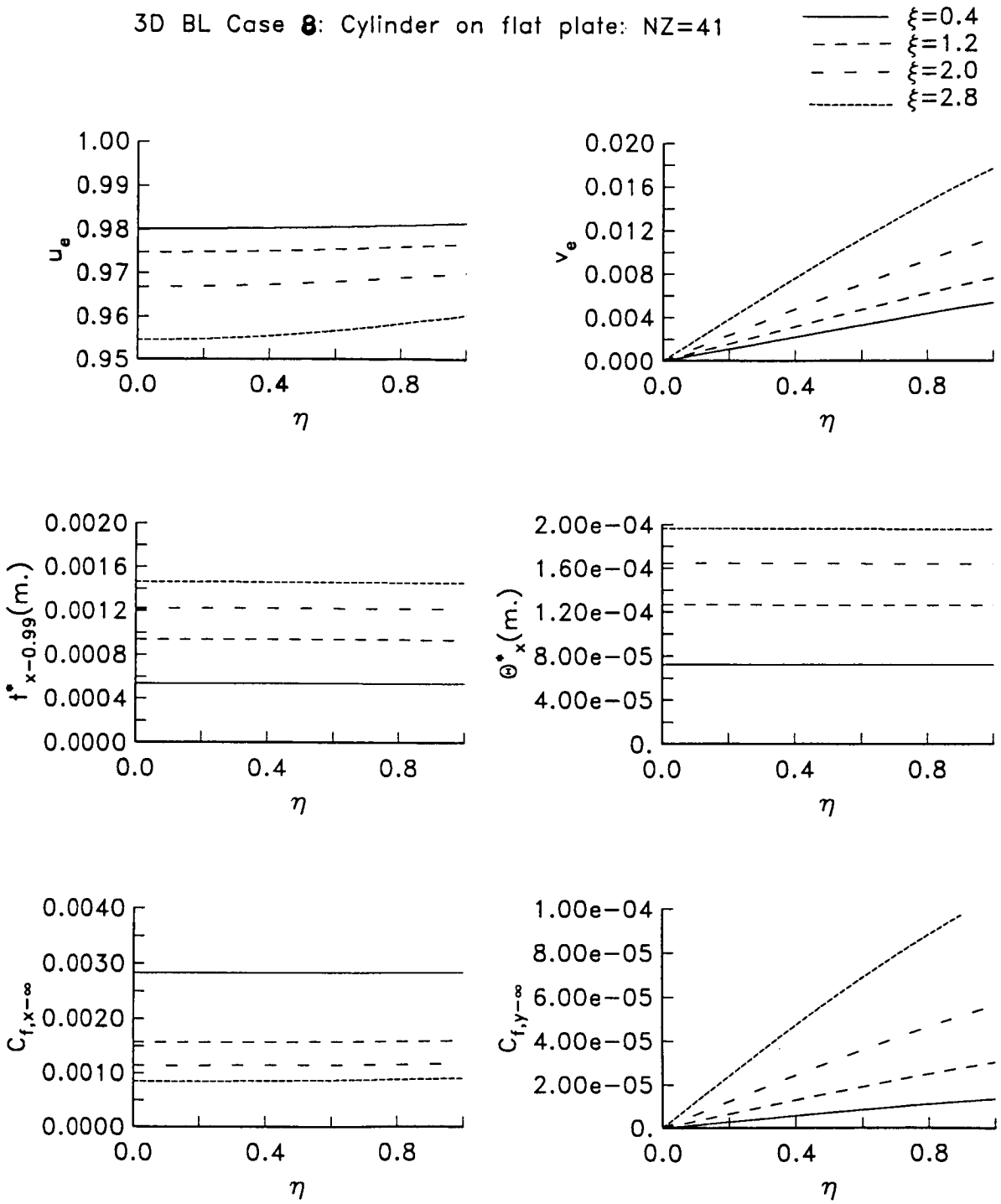


Fig.29. Boundary Layer Results for Cylinder on Flat Plate Flow

Case 9: 3D Flow Past Ellipsoid at Angle of Attack

Overview

This test case involves a fuselage-type body in subsonic flow at an angle of attack. The incompressible flow can be expressed in an analytic form. The boundary layer edge conditions can thus be obtained exactly.

In order to simulate the conditions of flow past a fuselage, the BL solution was also obtained by defining the body in discrete coordinates. The body surface is made up of discrete panels and the inviscid flow is calculated by the Hess potential panel code[20]. The pressure coefficient values obtained from this code are at the center of the panels. A more densely packed BL surface grid is then generated. The pressure field is interpolated to this grid. The edge velocities corresponding to the BL grid are then calculated by an interface program in which the surface Euler equations are solved. The results at this stage can be compared with the exact solution.

The boundary layer solution for the first one quarter of the body ($\xi \leq 0.25$) is then obtained from the exact edge conditions as well as the numerically obtained edge conditions.

Flow Conditions

An ellipsoid of revolution with a fineness ratio e_e (ratio of semi-minor to semi-major axes) of 0.25 is considered here. The body is assumed to be 2 feet long. The flow is assumed at a Mach number of 0.05 at an angle of attack of α=3°. The inflow conditions correspond to p^*_∞= 2116 lb/ft^2, T^*_∞=520°R. This corresponds to a free-stream Reynolds number of 3.54x10^5 per foot or a reference Reynolds number of 4.765x10^8 per foot.

Exact Forms for Metrics and Inviscid Flow

The exact definition of the metrics, and the inviscid flow based on potential theory have been discussed under Case 4.

Numerical Evaluation of Metrics and Inviscid Flow

For the numerical calculation of the inviscid flow, the body is represented by means of 1998 panels (54 streamwise and 37 circumferential). The panels are made smaller near the leading region to obtain adequate resolution in the streamwise direction. The panel size in the circumferential direction is kept constant (corresponding to a 5° angular distribution). The Hess code results are obtained at the center of these panels on a

53x36 grid. The inviscid results at each panel centroid consist of the Cartesian coordinates, the three Cartesian velocity components and the pressure coefficient.

The interface program generates the BL edge conditions from the inviscid results. The streamwise resolution provided by the inviscid code is inadequate for BL solution. A BL grid is generated with 77 points in the interval ξ [0.0,0.25] and 38 points in the circumferential direction. The circumferential distribution is the same as the inviscid grid with two points added corresponding to the windward and leeward lines. The metrics h_1, h_2 and g_{12} are then calculated numerically as given by eqs. 19-20.

The pressure coefficient is interpolated from the inviscid grid to the BL grid using tension spline fit. Since circumferential locations coincide interpolation needs to be done only in the ξ direction. Smoothing can be introduced at this stage, if necessary. However, the extent of smoothing to be used depends on the quality of the inviscid results. In the present case, smoothing was not required since the inviscid results have adequate smoothness.

The surface Euler equations are then solved as described previously. A three-point backward formula is used for the streamwise gradients and the five-point implicit formula of Ref. 8 for the crossflow gradients. The resulting velocities u_e and v_e are compared with the exact values to verify accuracy of the numerical results.

Fig. 30 shows streamwise plots of the edge velocity u_e and the inviscid pressure distribution used for this calculation. The plots are selectively shown at six locations of η=0.014, 0.208, 0.402, 0.596, 0.790, 0.984 corresponding to j values of 2, 9, 16, 23, 30, 37. The results are shown in the range ξ [0.0,0.05] where the variation is more pronounced. Shown in these plots are the C_p values from the inviscid code and the interpolated pressure distribution. There are 13 inviscid points in this range, whereas the BL grid has 38 points. The plots show that interpolation of pressure is accurate. The plots also show a comparison of the calculated u_e values with the exact u_e values. The surface Euler solution is initialized at i=2 (ξ=0.000127) with the exact values of u_e and v_e. The windward and leeward are first solved from the x momentum equation of the surface Euler set with v_e set to zero. The interior region is then solved using the five-point scheme. Subsequent runs using the 'L' scheme and the 'Z' scheme also gave comparable results, although the crossflow velocities from the 'Z' scheme showed a slight departure from the exact results at larger values of ξ.

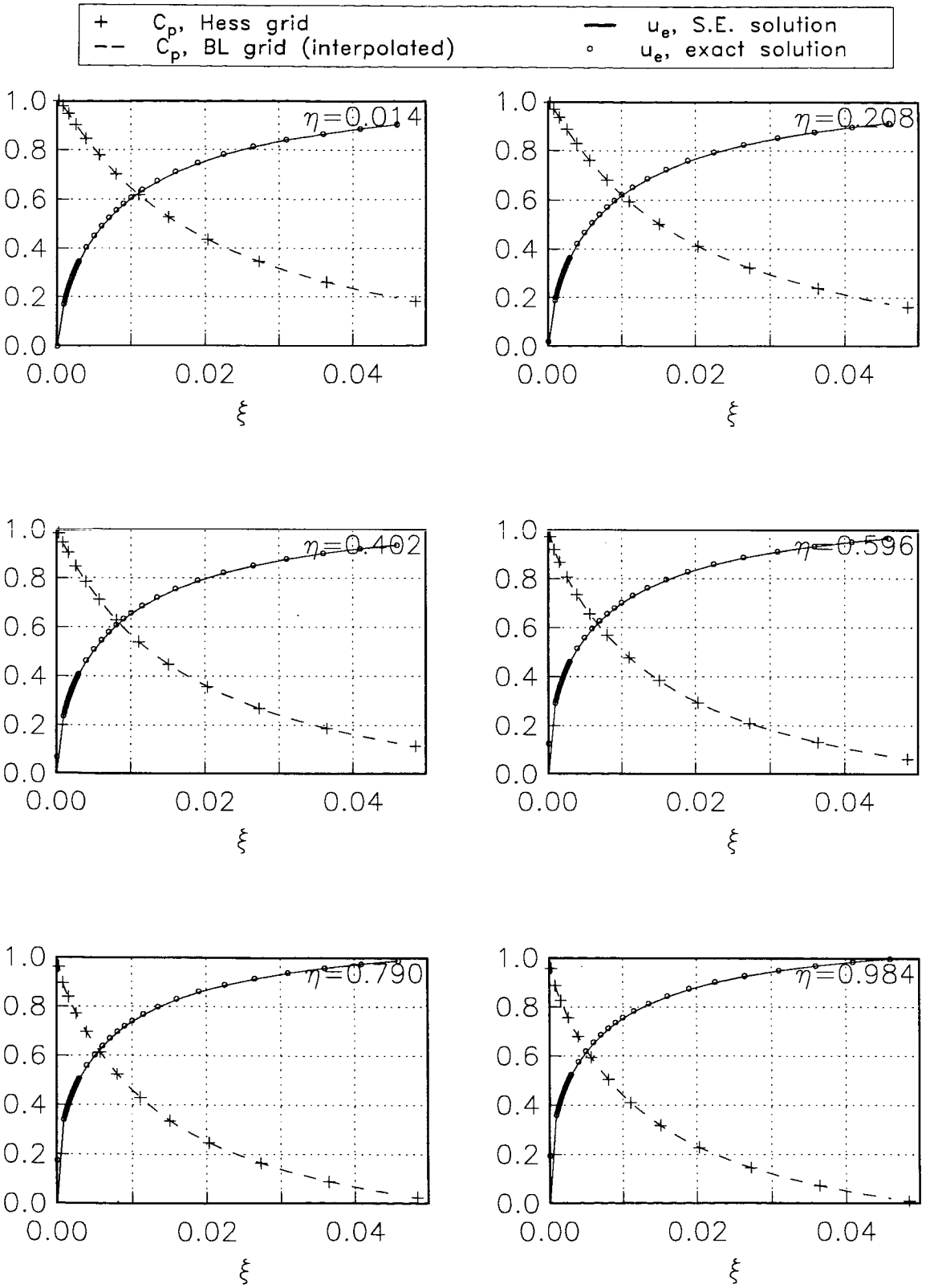


Fig. 30 Comparison of Surface Euler Solution and Exact Solution Streamwise Plots: Ellipsoid at α=3°

Fig. 31 shows a comparison of the exact values of u_e and v_e with the surface Euler solution in the η direction. The plots are selectively shown at ξ=0.002, 0.005, 0.037, 0.091, 0.151, 0.211 corresponding to i locations of 12, 24, 36, 48, 60, 72. The plots of v_e are scaled up by a factor of 4. The agreement between the exact values and the numerical solution is thus good in both directions.

Boundary Layer Results from Exact Inviscid Input

As discussed under Case 4, the solution is first obtained on the windward symmetry line. At α=3°, the stagnation point is located at x=0.000125. The i=2 point is hence located just downstream of this point. The windward line symmetry solution at i=2 is obtained by using forward differences for $\frac{\partial \chi}{\partial \xi}$ and $\frac{\partial u_e}{\partial \xi}$. Similarly the streamwise gradients of the metrics at this point are calculated using forward differences, so as reduce the error in the starting region due to the imprecise location of the stagnation point (as pointed out in Case 4).

The solution in the 3D flow region is obtained by a sweep in the η direction using the 'Z' scheme. Since the crossflow is positive everywhere, an 'L' scheme can also be used. Fig.s 32 and 33 are plots of the variation of $C_f \sqrt{Re_\infty}$ and $t_x^* \sqrt{Re_\infty}$ in the ξ and η directions. The variation in both directions is found to be smooth and continuous and agrees well with results given by Wang [13].

Fig. 34 shows plots of the velocity profiles and their first and second derivative in the normal direction at a particular ξ and η location. It can be seen that the solution is smooth to the second derivative level, even though there are only 15 points within the boundary layer. Smoothness of the higher derivatives of the solution is a requirement for stability calculations and this can be efficiently achieved by the present method.

Boundary Layer Results from Numeric Inviscid Input

The boundary layer calculation here is identical to the exact calculation except that the input metrics and inviscid edge data are numerically obtained. The numerical errors resulting from the geometry and inviscid data can thus be estimated.

Fig. 35 shows a comparison of the skin friction coefficient and boundary layer thickness resulting from exact as well as numeric treatment of edge

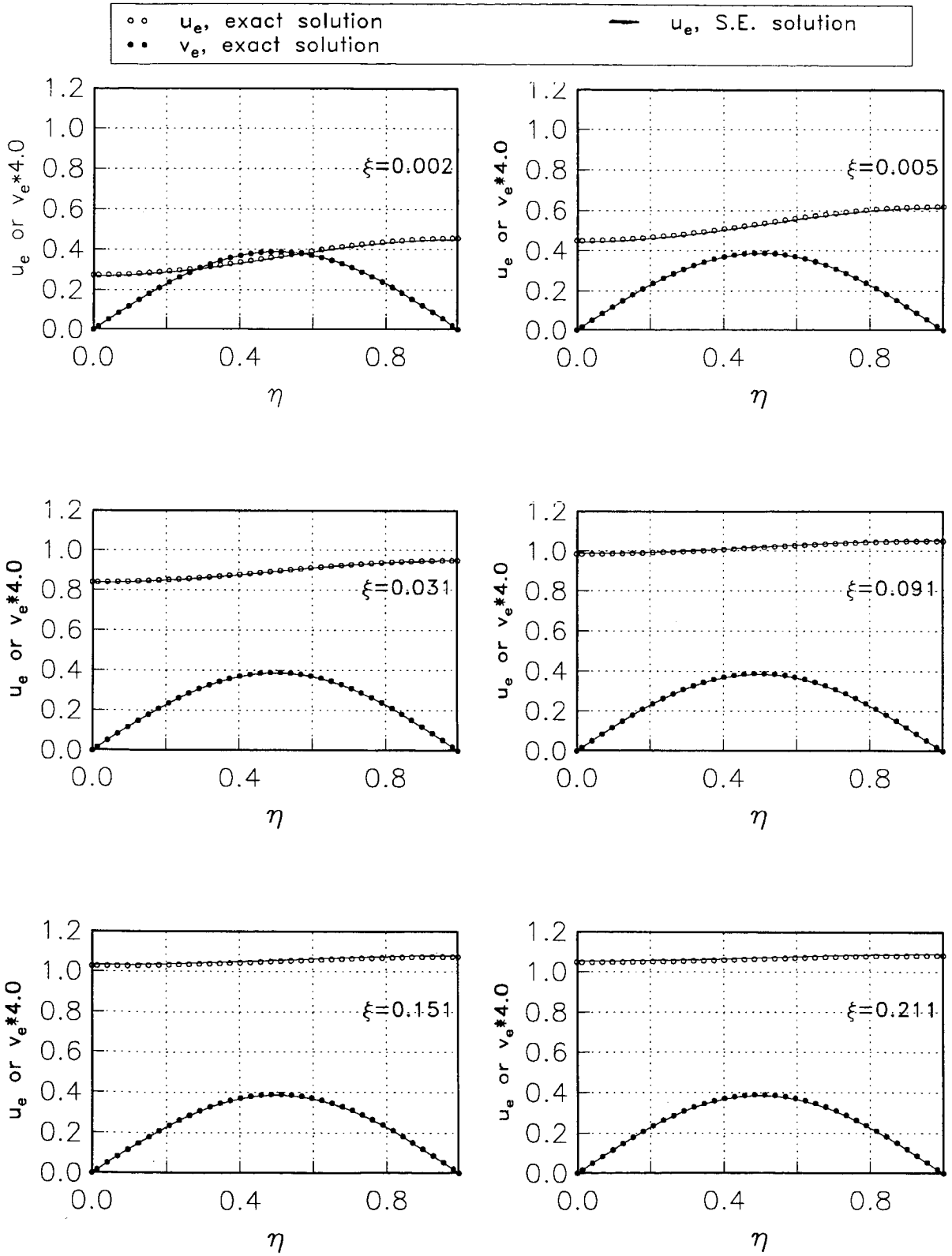


Fig. 31 Comparison of Surface Euler Solution and Exact Solution Circumferential Plots: Ellipsoid at α=3°

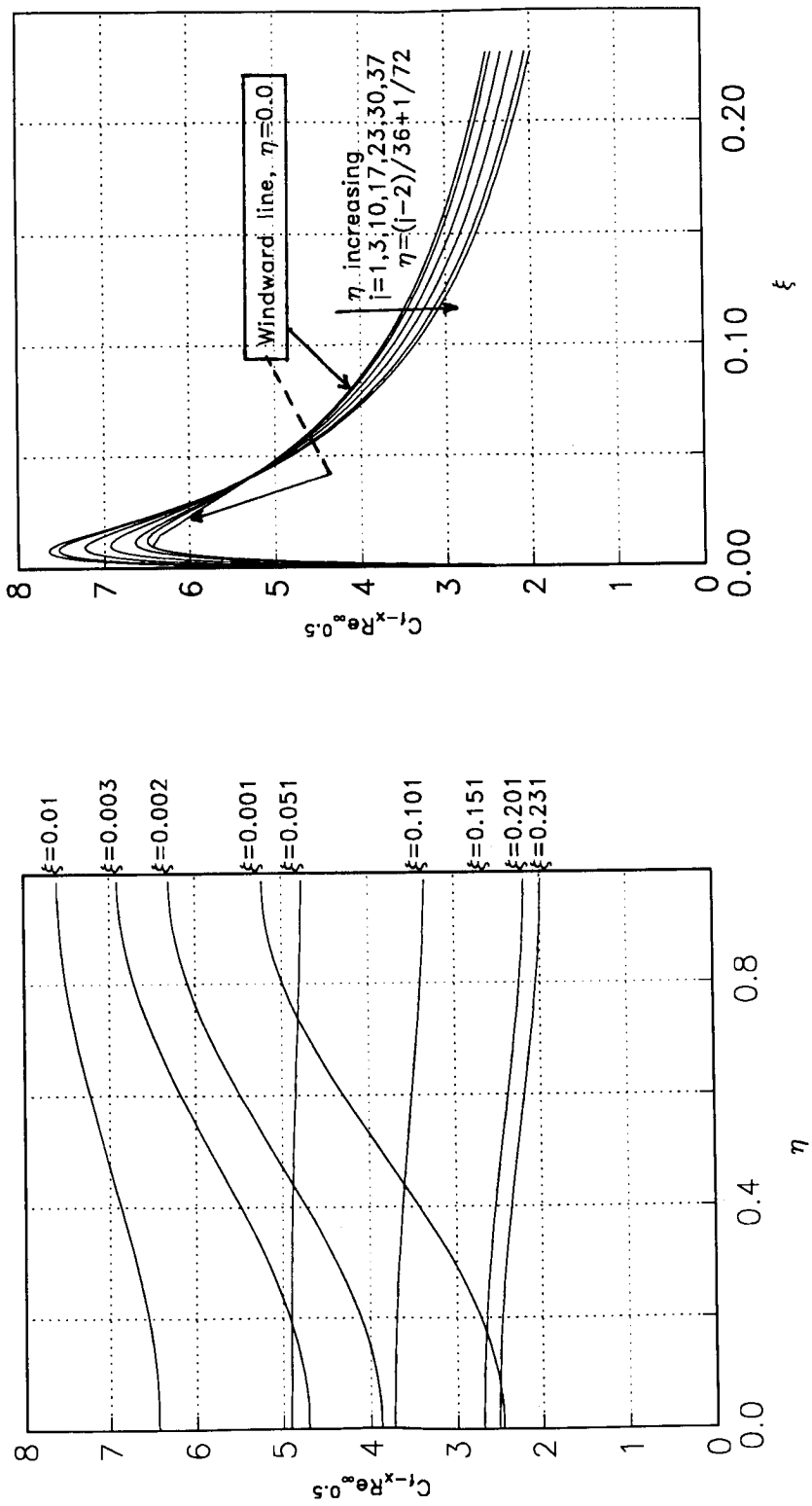


Fig. 32 Ellipsoid Flow Boundary Layer solution from Exact Edge Conditions
Variation of Skin Friction Coefficient in ξ and η

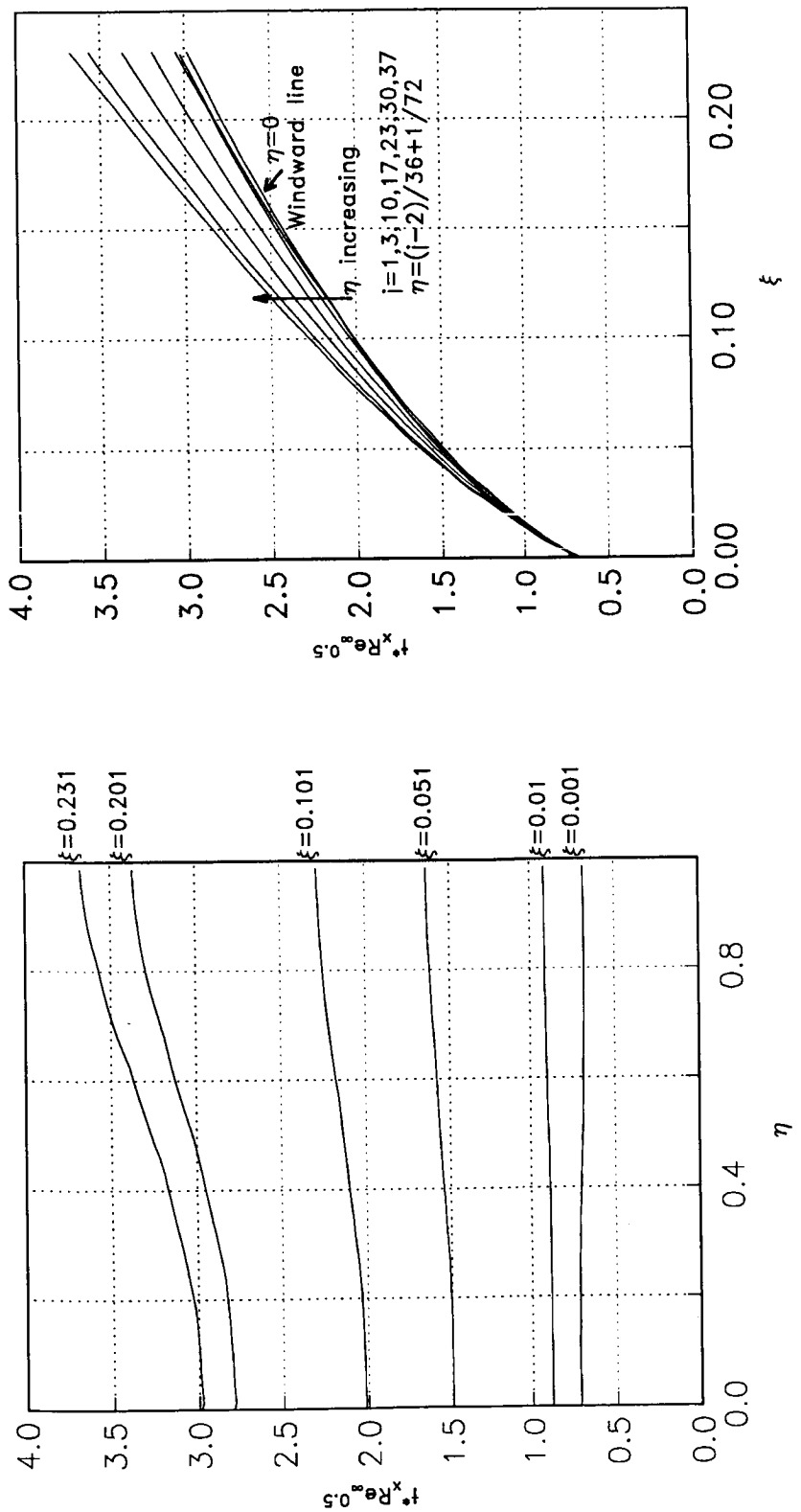


Fig. 33 Ellipsoid Flow Boundary Layer solution from Exact Edge Conditions
Variation of Boundary Layer Thickness in ξ and η

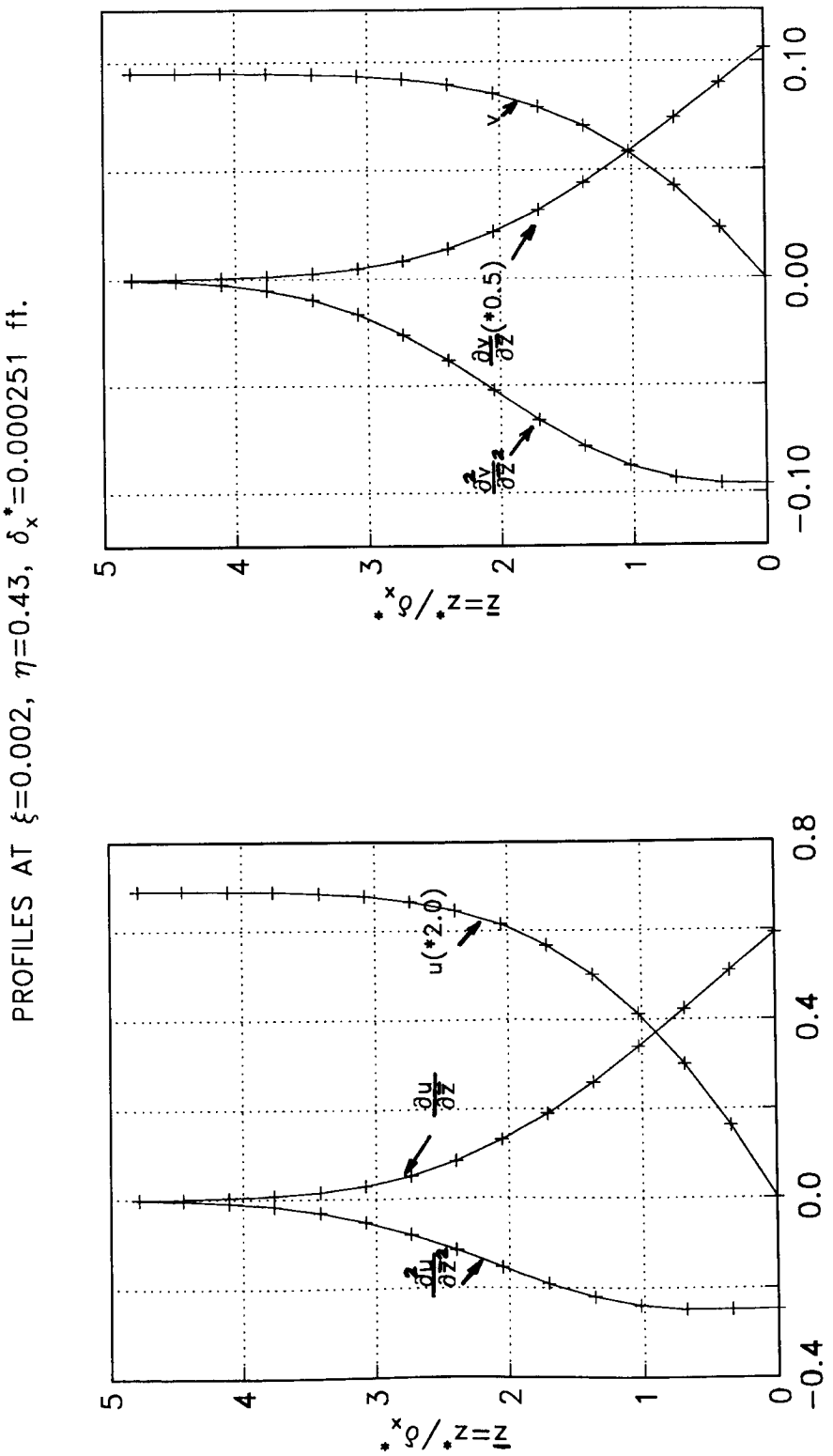


Fig. 34 Ellipsoid Flow at α=3°; Profiles of (u, u′, u″) and (v, v′, v″)

data and metrics. There is a good agreement between the two sets of results in both ξ and η directions. This study shows that the present numerical interface procedure is accurate and similar results can be expected when obtaining BL solutions on fuselage bodies defined in discrete coordinates.

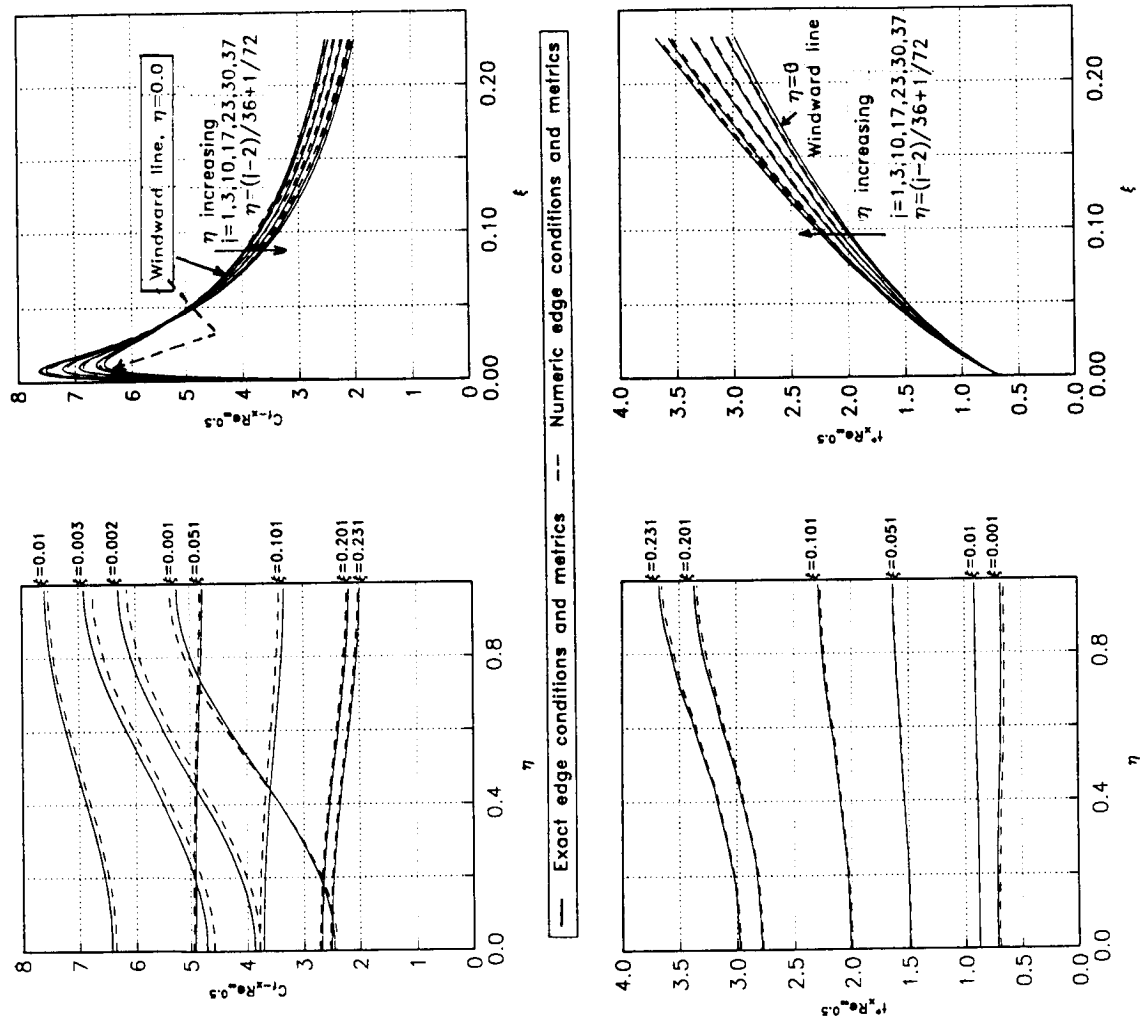


Fig. 35 Comparison of BL Results Obtained from Exact and Numeric Edge Conditions

Case 10: Flow Past Swept Wing at Angle of Attack

Geometry and Flow Conditions

A swept wing with zero taper and a straight leading edge is considered here. The wing has NACA 0012 cross section in the streamwise direction and is swept back by 20°. The chord length is taken to be equal to 1 ft. and the span distance (normal to the chord line) is 1.5 ft. Flow is at an angle of attack of 2° and a Mach number of 0.5. The free-stream conditions of p_∞^* and T_∞^* are same as given for Case 9.

The present calculation is done for the upper surface only and assumes laminar flow until separation. However, the calculation does not take any advantage of the analytical nature of the wing section or planform and can be applied to solve the flow over the upper or lower surface of a general wing with taper, sweep or cranked leading edge. The method used is general and can be used for transonic or supersonic flow. The only change required would be in the calculation of the edge density values which are calculated here assuming isentropic flow.

Inviscid Flow

The inviscid flow past the wing is calculated using the Hess panel code[20]. These results were also checked against Euler and Navier-Stokes calculations and found to be accurate for the present inflow conditions.

The Hess panel code version used here had a limitation of a maximum of 2,000 panels. Assuming 21 panels distributed in the spanwise direction, the streamwise wrap-around direction is limited to just 95 points. The panels are distributed in the wrap-around direction such that more points are clustered on the upper surface especially in the attachment line region. The panel size near the stagnation region is about 0.1% of the chord length. Elsewhere, the panel size is 2% of the chord. Fig. 36(a) shows the distribution of Δs, the panel surface arc length as a function of s, the arc length on the entire wing. A step-wise distribution has been used.

The inviscid results are obtained at the panel centroids. The corresponding grid has 89 points in the wrap-around direction and 20 points in the span direction. The results from the Hess code are obtained in the form of Cartesian velocity components and the pressure coefficient at each inviscid grid point.

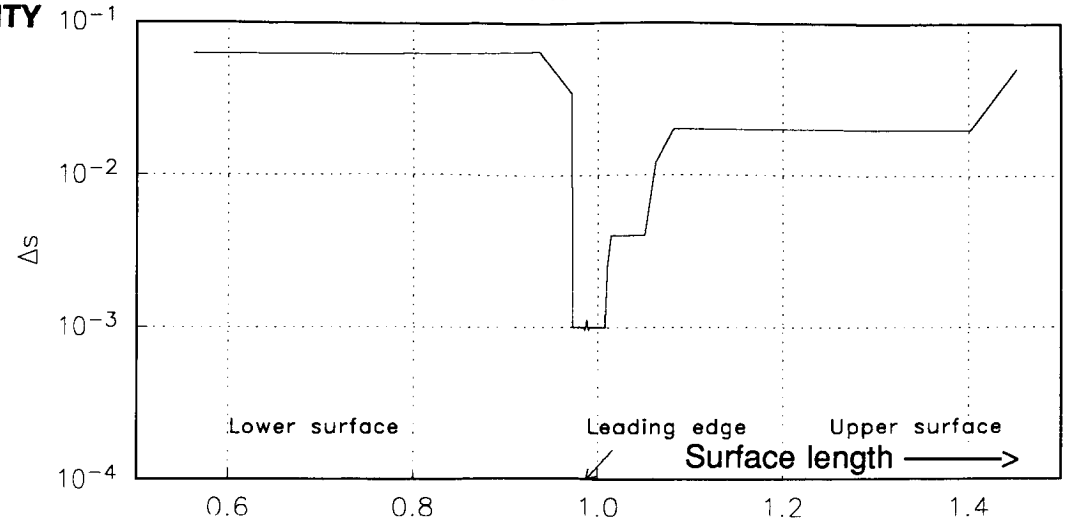


Fig. 36(a) Inviscid Grid Panel Length Distribution (at mid-span)

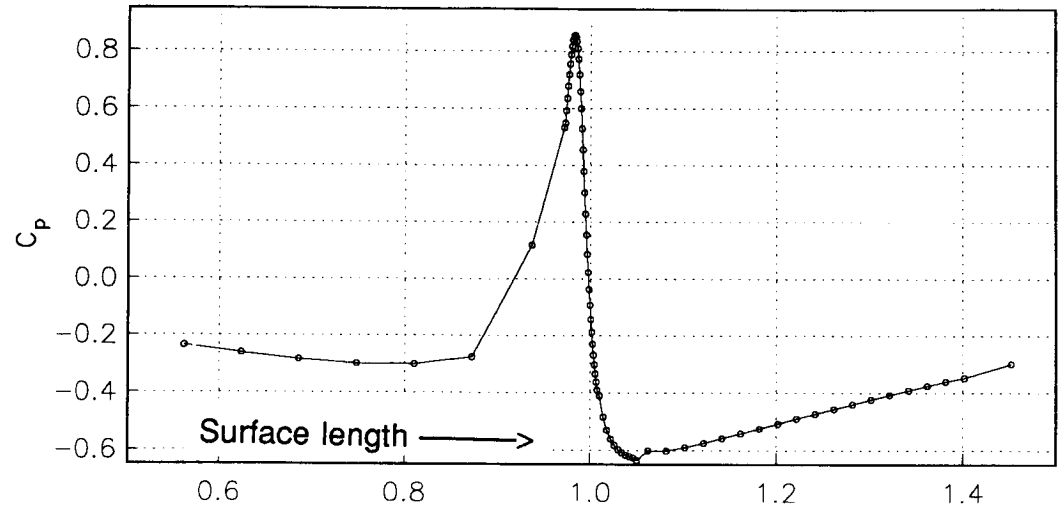


Fig. 36(b) Inviscid C_p Distribution

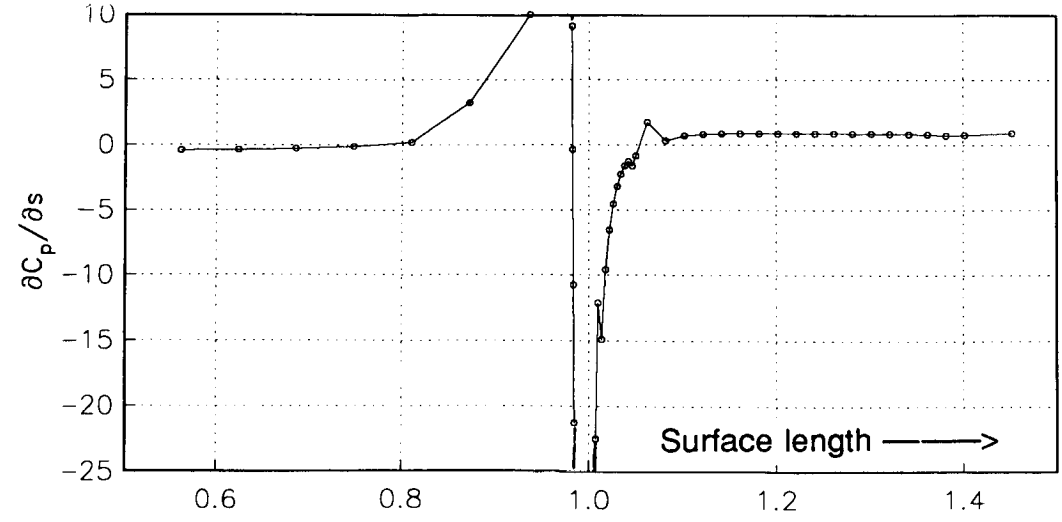


Fig. 36(c) Inviscid Pressure Gradient in x Direction (at mid-span)

Coordinates and Metrics

The boundary layer coordinate ξ is defined to be zero at the attachment line of the wing. The ξ coordinate line runs in the chord direction on the wing surface and is measured along the surface of the wing. The value of ξ is defined as the surface arc length starting from the attachment line divided by a normalizing chordwise length such as the the chord length at a particular span location, or the total arc length from the attachment line to the wing trailing edge. This normalization is done so that the coordinate lines in the η direction will be approximately along constant chord ratio points.

The metric h_1 is thus a constant at a given span section and is equal to the normalizing length. In the present case, the local chord length is used as the normalizing length, so that h_1 is equal to unity everywhere. For a tapered wing h_1 will vary in the span direction as a function of the planform shape.

The coordinate η is measured along the Cartesian coordinate y'. The coordinate h_2 will then be equal to inverse of the cosine of the sweep angle, if the attachment line is straight. Otherwise, h_2 will depend in the wing curvature and taper in the span direction.

Interpolation of Pressure

Fig. 36 (b) shows the variation of C_p obtained from the inviscid code as a function of the surface length s at a span station near the mid-section of the wing. It can be observed that more number of points are clustered on the upper surface. The attachment line region is where C_p has a peak and enough number of points are clustered in this region. The attachment line and the maximum C_p line need not be coincident (see Ref. 2), but they are very close to each other. Fig. 36(b) shows that the C_p distribution is not entirely smooth. Fluctuations can be seen in the regions where the panel size is changed. The fluctuations are especially remarkable when we look at the pressure gradients as in Fig. 36(c). Therefore, when the pressure is interpolated to a BL grid, some kind of smoothing has to be employed to prevent non-physical fluctuations entering into the BL calculations.

The wing interface program is designed to process the inviscid results and interpolate them smoothly to a BL grid. Fig. 37(a) shows the surface length distribution corresponding to the BL grid on the upper surface of the wing. Fig. 37(b) shows the smooth interpolated pressure at a mid-span location. Tension spline interpolation with smoothing has been used. The smoothing parameter is set such that the standard deviation on the error in

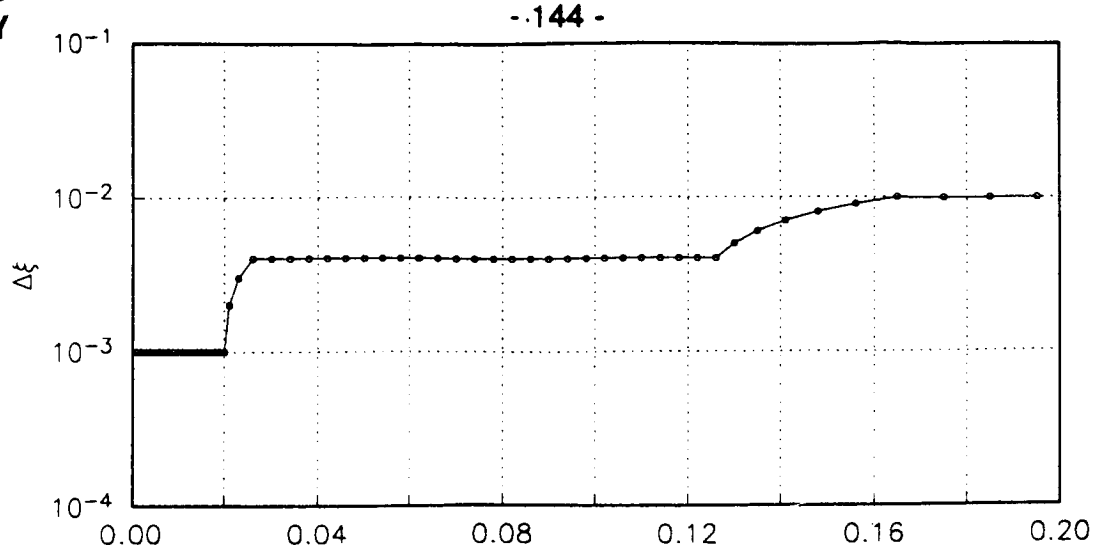


Fig. 37(a) BL Grid Distribution Near Attachment Line

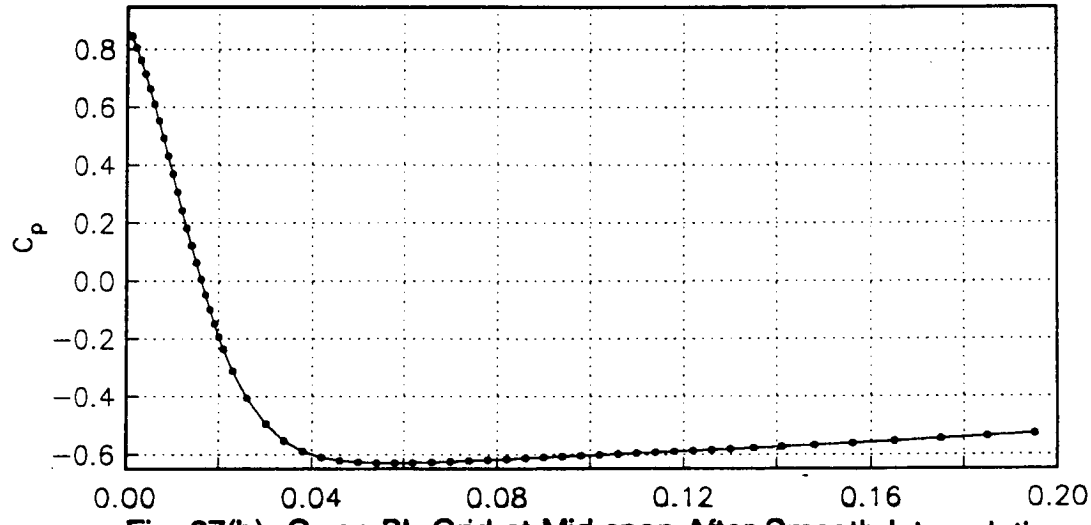


Fig. 37(b) C_p on BL Grid at Mid-span After Smooth Interpolation

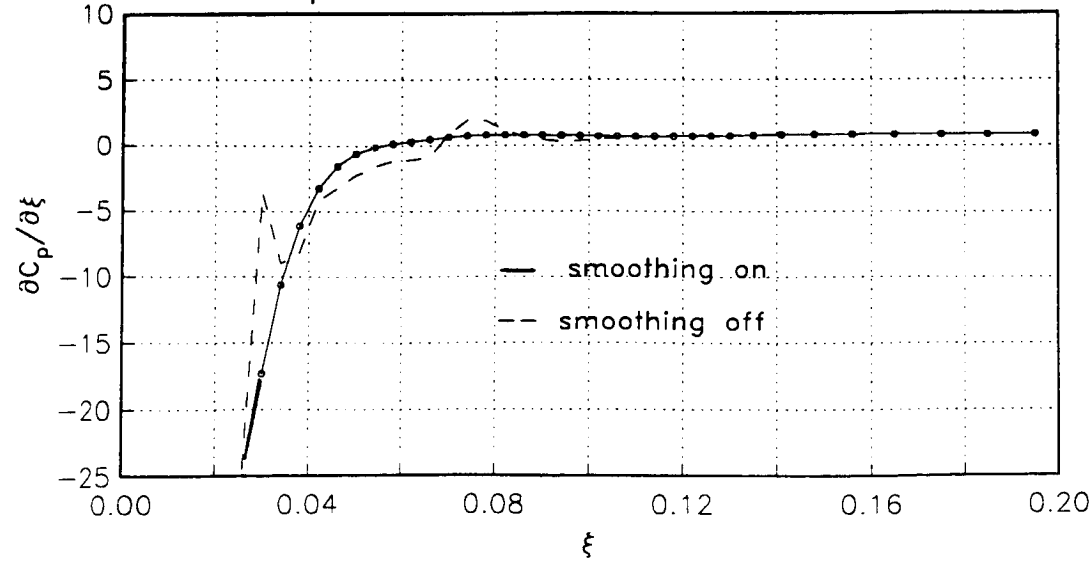


Fig. 37(c) Pressure Gradient in ξ on BL Grid at Mid-span

the input C_p distribution is assumed to be 0.01 near the attachment line ($\xi \leq 0.01$) and equal to 0.05 elsewhere. These smoothing parameter values are dependent on the quality of the inviscid results and will vary depending on the case. The resulting pressure gradients are smoothly varying as can be seen from Fig. 37(c). In comparison, the pressure gradients resulting from an interpolation without smoothing (shown as a dashed line) has unacceptable levels of fluctuation.

Attachment Line Relocation

Since the inviscid grid may in general be coarse and not aligned with the wing attachment line, the interface procedure includes a routine to accurately locate the attachment line. The criterion for relocating the line is that the streamwise velocity at any point on the attachment line should be equal to zero. In fact, the velocity vector on the attachment line should be exactly tangential to it.

To ensure this condition, an initial attachment line is first located on the wing surface as that corresponding to the maximum C_p line. However, since a general attachment line is curved, the velocity on this line may not be tangential to the grid line. The attachment line is then relocated as a function of the velocity error. The relocation calculation is repeated in an iterative fashion until convergence is obtained.

The relocation procedure involves the following steps:

- Assume initially that the attachment line is located at the line of maximum pressure
- Generate a temporary BL grid off this line and interpolate the velocity components to the BL grid points
- Compute the streamwise surface velocity u_e by transformation
- Move the attachment line as a function of the streamwise velocity; the relocation extent δs is $= u_e/k_u$, where k_u is an estimated value of $(\partial u_e/\partial s)$ at the attachment line.
- Repeat the steps above with the new attachment line and iterate until $u_e < \varepsilon_u$ on the attachment line.

The interpolation involved in the above steps is based on the surface length as the independent variable. The wing cross section coordinates required for calculation of the metrics and velocity transformation are also obtained by spline interpolation. A reference section geometry at each span location is used in the interpolation procedure.

Fig. 38 shows a magnified picture of the shifting of the attachment line as a result of the relocation. The initial attachment line is located along inviscid grid points and shows up as a horizontal line. The u_e on this line is non-zero. Several iterations later, the attachment line is relocated as a curved line as u_e on this line converges to zero within a specified tolerance level.

Surface Euler Solution

A BL grid with the required clustering of points on the wing surface is now generated originating from the true attachment line. The metrics are then calculated and the pressure is accurately and smoothly interpolated as explained previously. The surface Euler solution is then obtained using the 5-point implicit scheme. The solution procedure is similar to the one discussed under case 9. The only difference is that the left boundary is an infinite swept wing type instead of plane of symmetry. Fig. 39(a) shows the variation of the obtained solution u_e and v_e at two span locations near the root and tip of the wing. In order to demonstrate the error that can occur by interpolation of velocities, Fig. 39(b) shows a magnified plot of the interpolated velocities near the attachment line region. The variation is obviously not smooth and a BL calculation cannot be performed with this type of edge conditions. The surface Euler solution, on the contrary, is smooth and accurate. The output from the interface procedure consists of the velocities u_e and v_e, the metrics h_1, h_2 and g_{12} and the grid distribution x and y.

Boundary Layer Solution

The boundary solution starts with the solution of the general attachment line equations at ξ=0. The solution is analogous to case #7 except that in this case, the flow is subsonic and none of the metric terms or their gradients are assumed to be zero. The solution is restricted to the region 0.1875<y<1.2375. The flow outside this region close to the wing root and wing tip are excluded from the calculation. The attachment line solution itself starts with the solution at y=0.1875, which is assumed to be locally infinite swept. The inputs to the attachment line solution consist of the metrics, the velocity v_e and the stream-wise velocity gradient f_e. The required results to start the 3D BL computation are the profiles of v/v_e and f/f_e. The span-wise variation of the input quantities and the the results in the form of $\delta^*_{y-0.99}$ and $C_{f-y,\infty}$ are plotted in Fig. 40. The profiles at different locations on the attachment line are given in Fig. 41. Fig. 40 and Fig. 41 show that the attachment line boundary layer corresponds to that of an

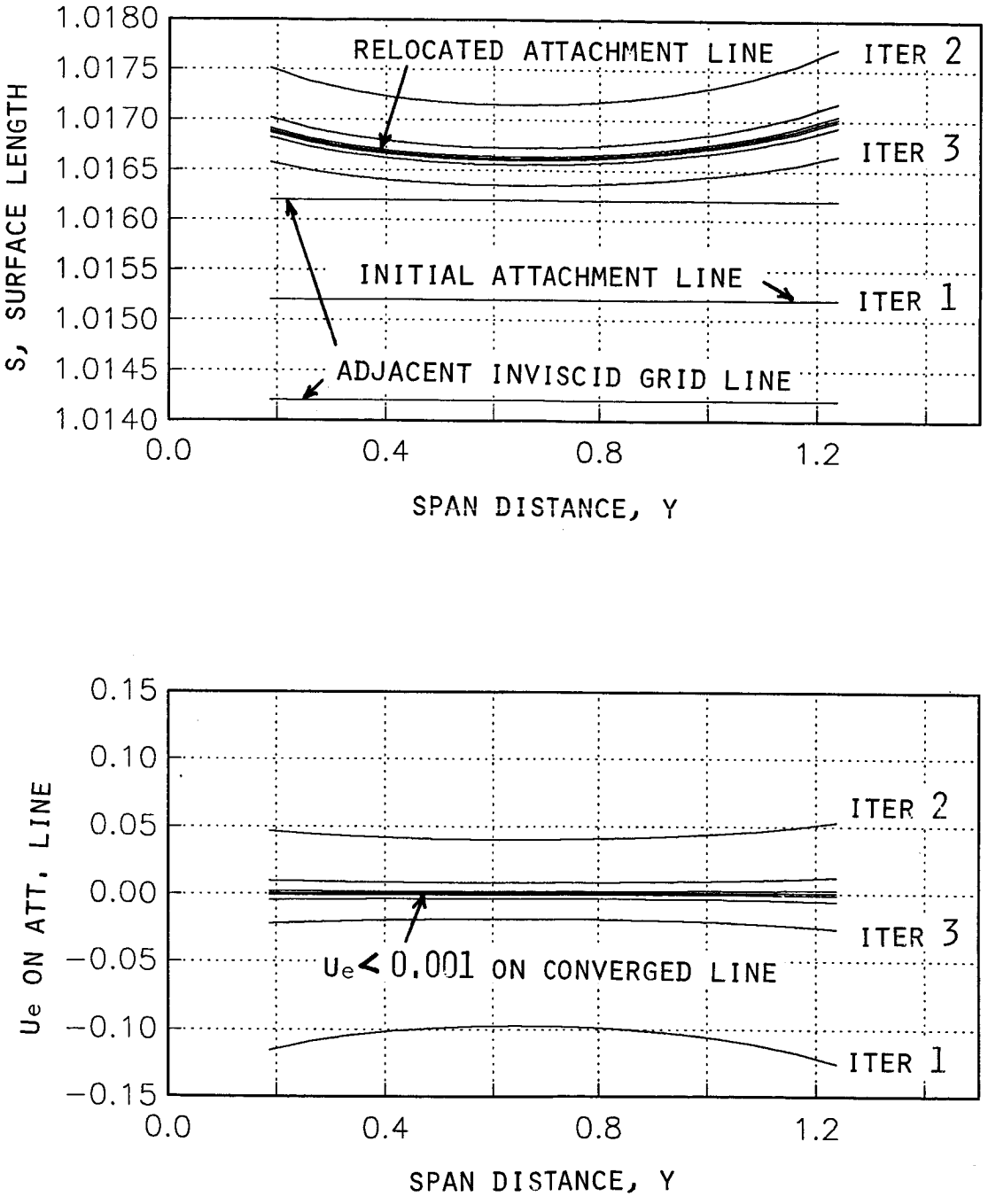


Fig. 38 Attachment Line Relocation Convergence

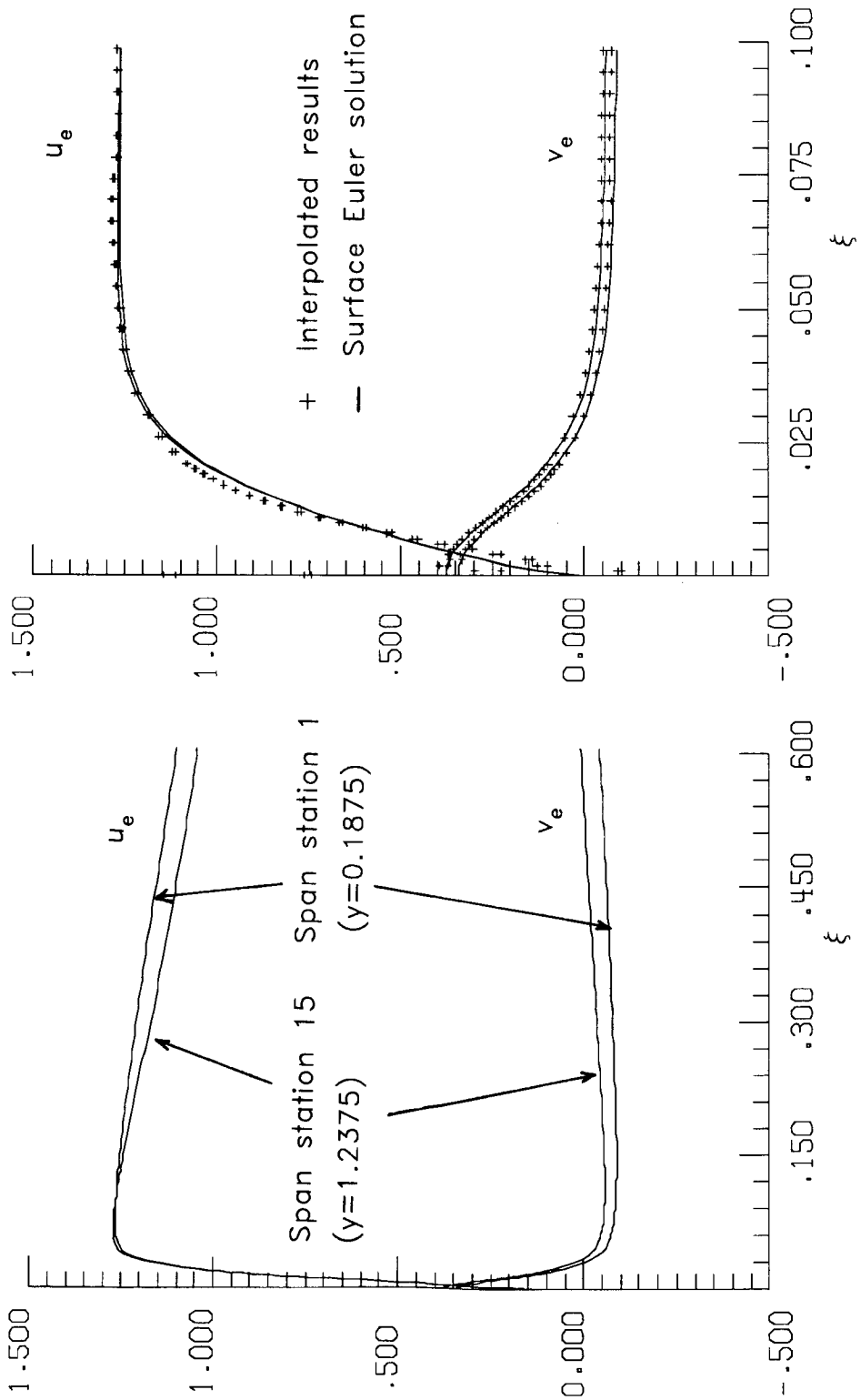


Fig. 39(a) Surface Euler Solution at Two Span Stations Fig. 39(b) Enlarged Plot Near Attachment Line With Comparison With Interpolated Velocities

Symbol	Variable	Y-axis scale
∘ ∘	v_e	0.30 – 0.40
• •	$\delta_{y-0.99}$	0.00013 – 0.00015 (ft.)
▪ ▪	$C_{f,y-\infty}$	0.0024 – 0.0028
△ △	f_e	120 – 140

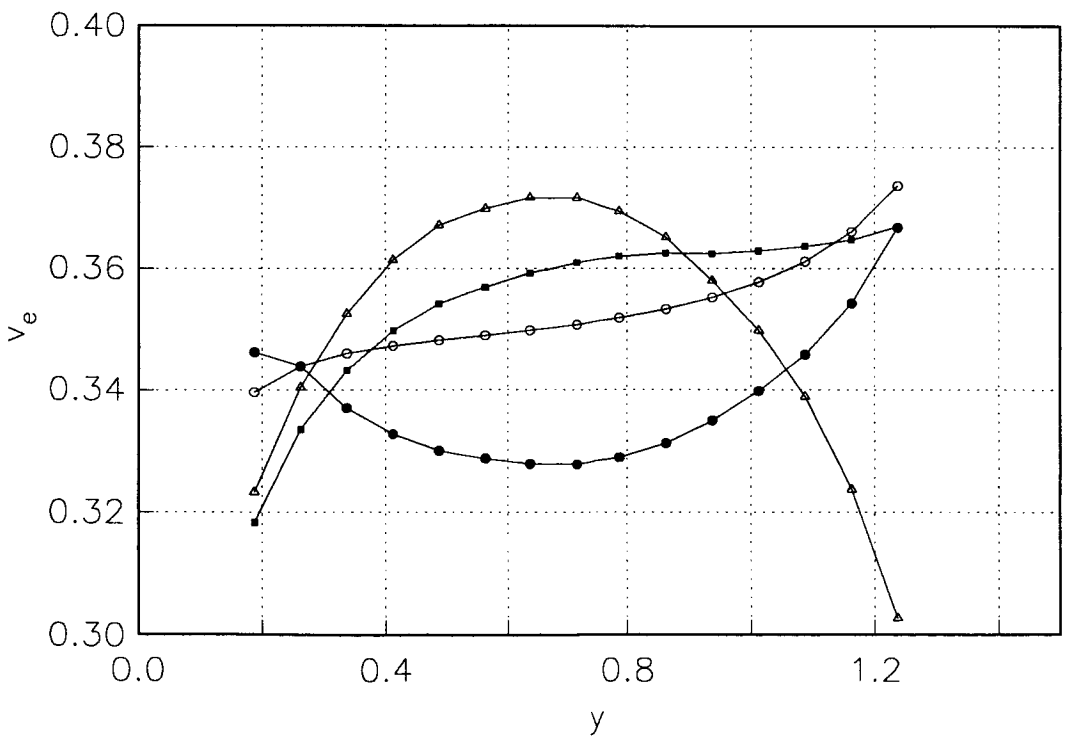


Fig. 40 BL Results on the Attachment Line

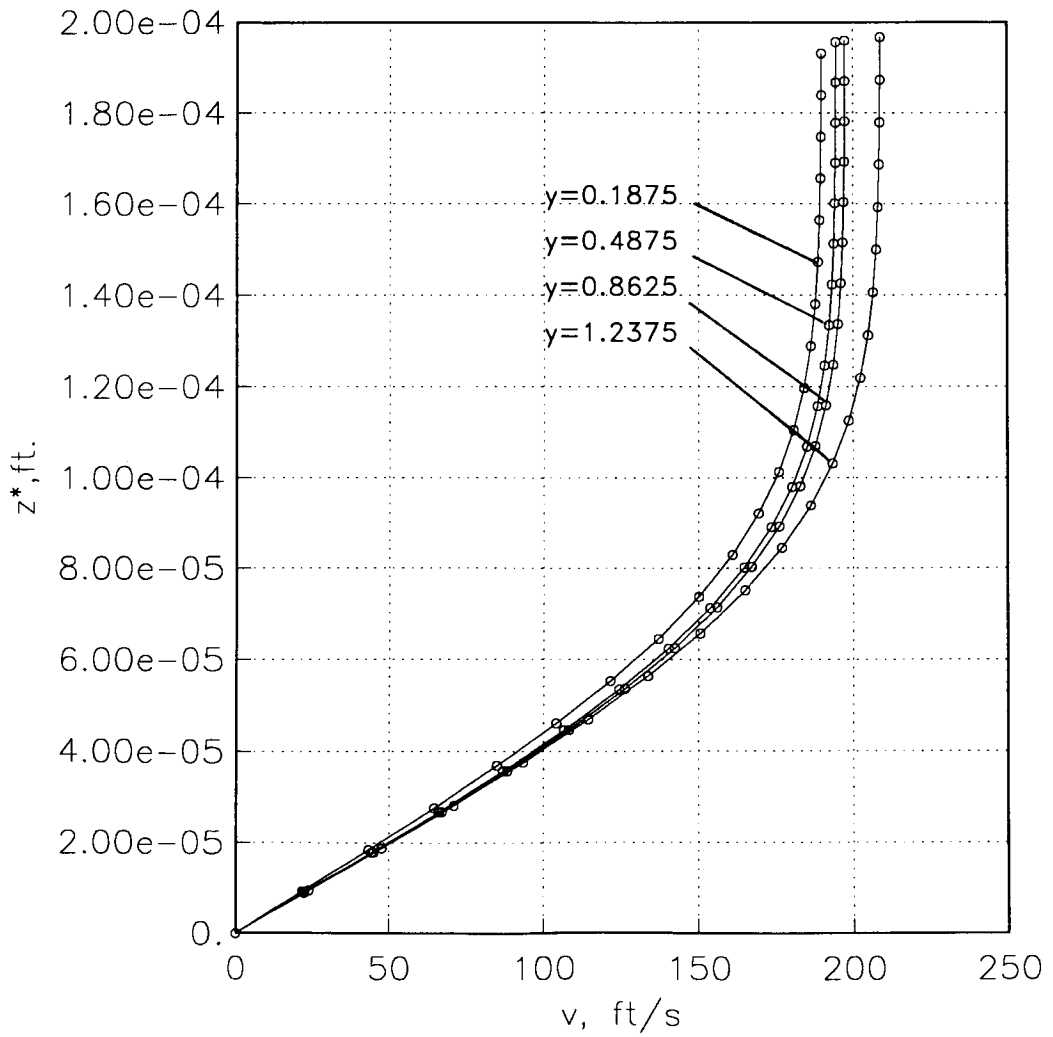


Fig. 41 Velocity Profiles on the Attachment Line

accelerating flow.

The side boundary solution is obtained by solving the LISW equations at the j=1 boundary corresponding to y=0.1875. Fig. 42 shows the results of $C_{f-x,\infty}$, $C_{f-y,\infty}$ and $\delta^*_{x-0.99}$ along the side boundary. The calculation is terminated at ξ=0.47 approximately, where the boundary layer separates. $C_{f-x,\infty}$ reaches a peak value of 0.007 at about ξ=0.017 and $C_{f-y,\infty}$ has a negative peak at ξ=0.026. The boundary layer thickness increases steadily except near the separation region where the growth is exponential. Fig. 43 shows the velocity profiles in the ξ and η direction at 9 stations along the left boundary. The profiles are plotted in actual physical units in order to show the growth of the boundary layer. It can be seen that the crossflow velocity profiles show an inflexion as the edge velocity vector turns inward earlier than the velocity vector near the wall. This inflexion in the profile is seen to persist till about ξ=0.3 and is of interest in the study of crossflow separation.

With the initial and side boundary planes of solution available, the 3D equations are now solved. Crossflow gradients are treated using either an 'L' scheme or a 'Z' scheme. The results in either case are almost identical in the present case. Fig. 44 shows contour plots of the BL results on the wing surface (approximated by the ξ, y′ plane). It can be seen that the contours are predominantly parallel and in the span direction, showing that there is very little departure from the infinite swept wing flow. As separation is approached one can notice an increase in the three-dimensionality of the flow.

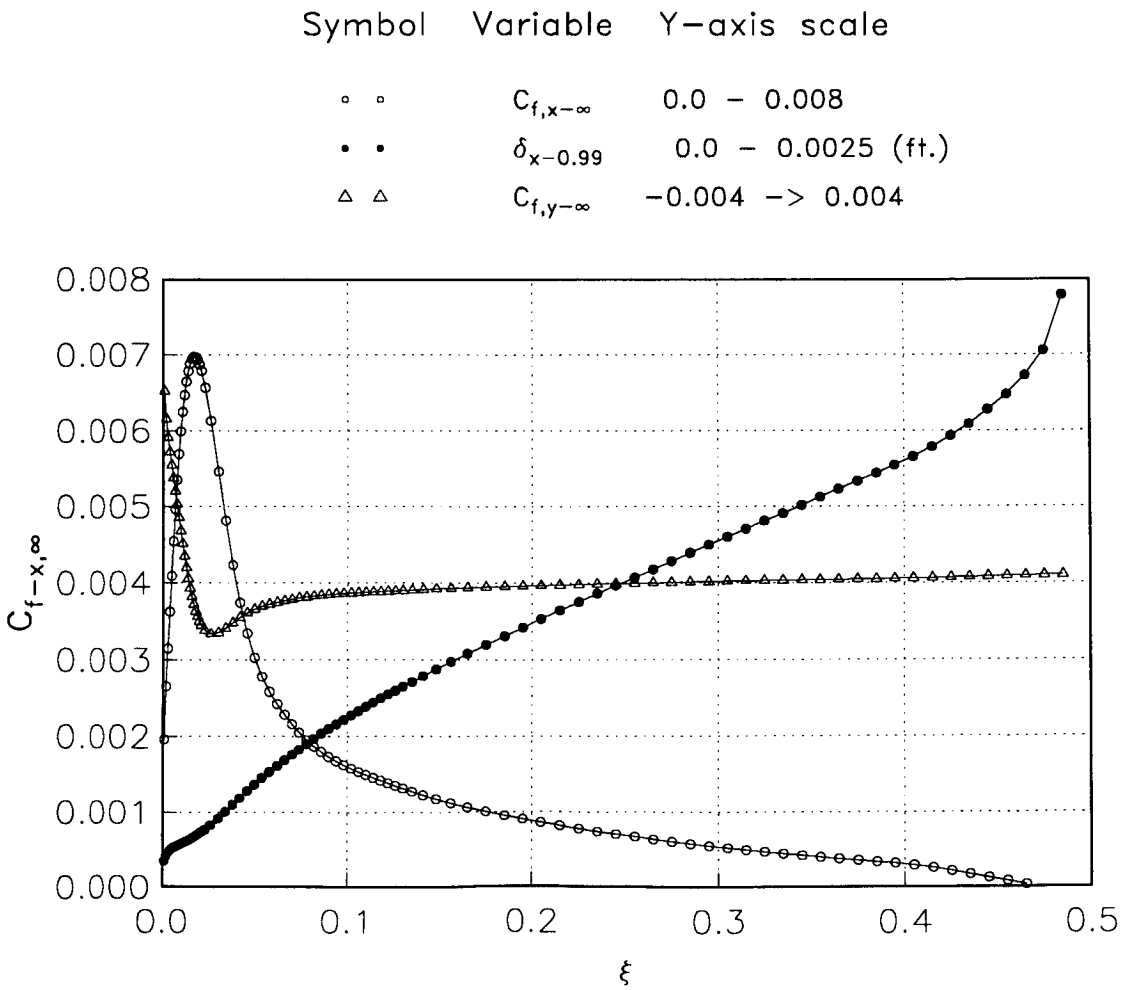


Fig. 42 BL Results on the Left (Locally Infinite Swept) Boundary

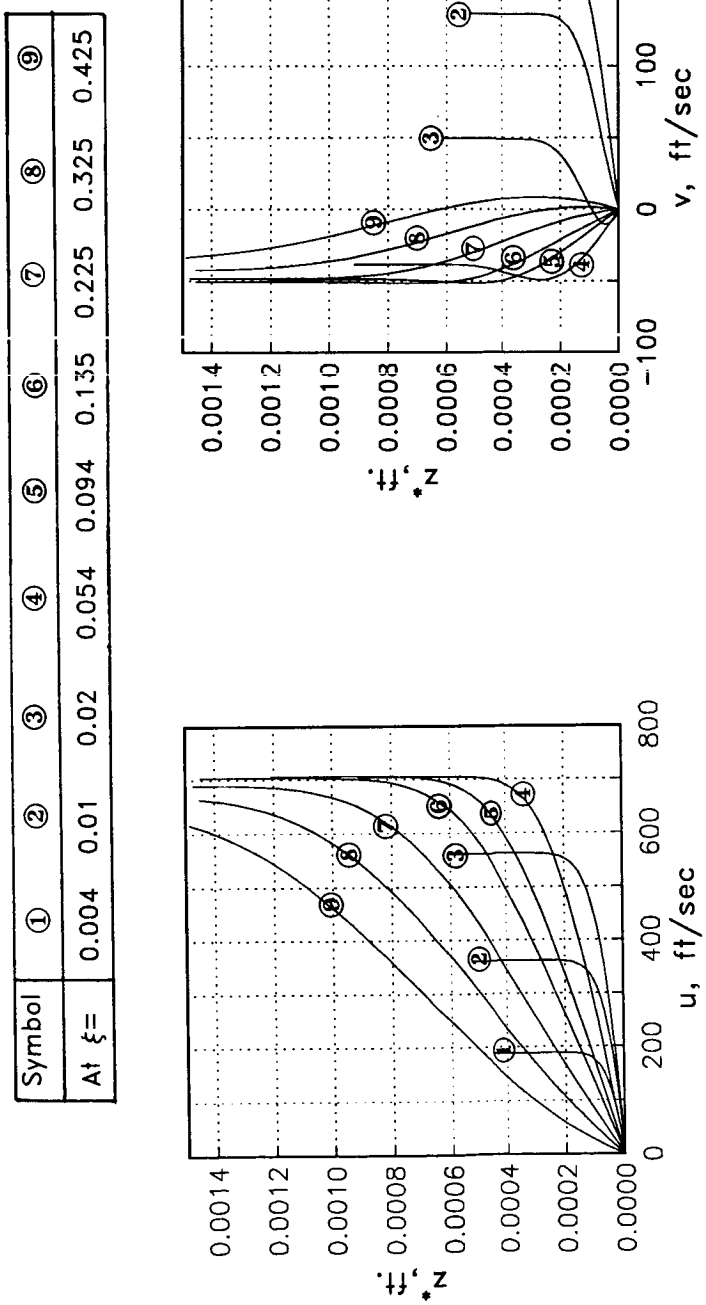
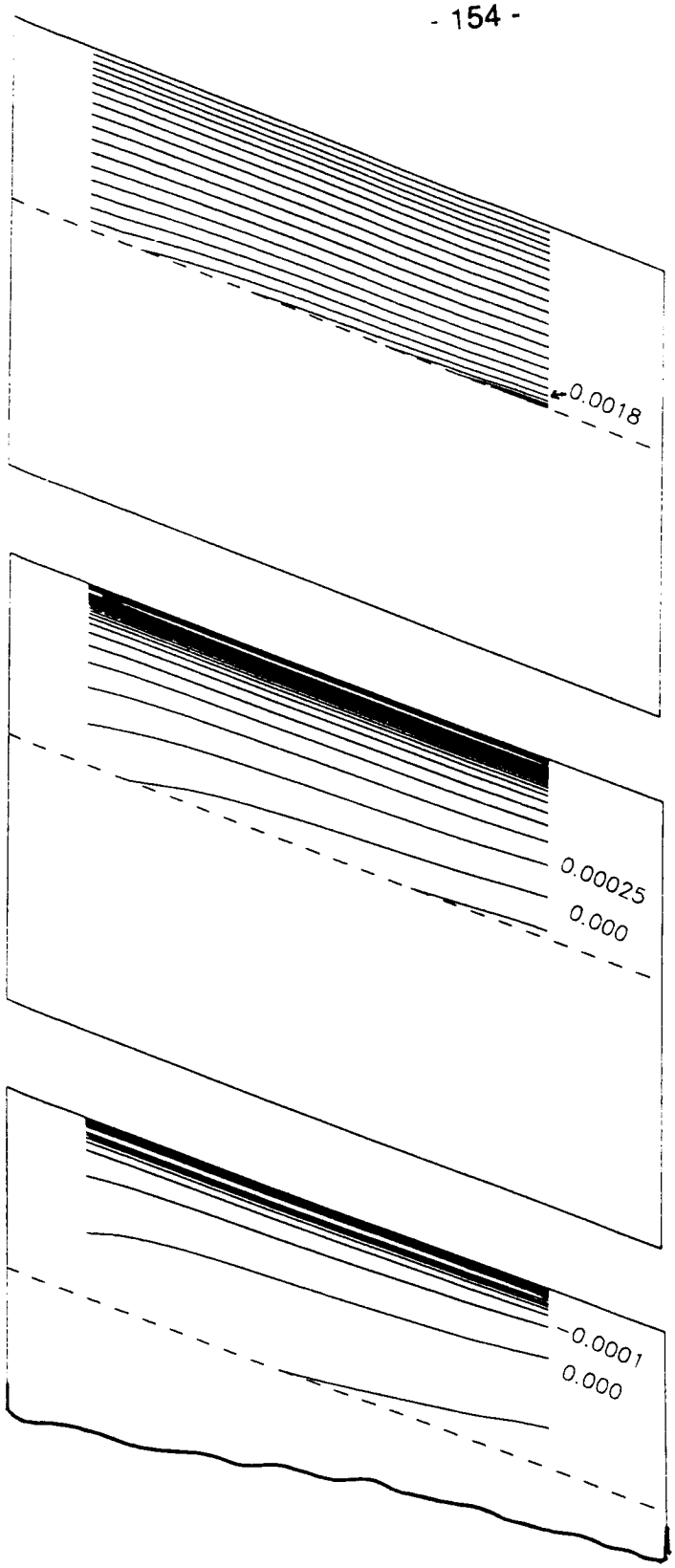


Fig. 43 Velocity Profiles on the Left (Locally Infinite Swept) Boundary



Contours of $t^{*}_{x,0.99-\infty}$
from 0.0000 to 0.0025
contour interval 0.0001 ft.

Contours of $C_{f,x-\infty}$
from 0.000 to 0.008
contour interval 0.00025

Contours of $C_{f,y-\infty}$
from −0.001 to 0.003
contour interval 0.0001

Fig. 44 BL Results on the Wing Upper Surface

11. CONCLUDING REMARKS

A fourth-order accurate method to solve the 3D boundary layer equations has been presented. The method has been applied to a number of 2D and 3D cases over a range of speeds and validated against standard known results.

The 3D boundary layer equations in curvilinear coordinates are first non-dimensionalized with reference quantities. As a result, the body dimensions are absorbed into the reference Reynolds number. Hence for laminar flow, the dimensions of the body enter only as a scale factor. The solution in transformed quantities is in fact, independent of the body length. In real flows however, the body dimension feeds into the viscous solution through viscous-inviscid interaction, turbulence and surface roughness.

Transformations are subsequently defined to absorb the boundary layer growth and to avoid problems at the singularity point. The transformation has been found to work well for laminar flow cases. The edge of the boundary layer is usually within ζ=8. For zero pressure gradient flows, the growth of ζ in the ξ-η plane is zero, whereas in other cases, the growth is a function of the pressure gradient. For turbulent flows, however, the edge of the boundary layer corresponds to large values of ζ. In this case, the value of ζ_e and the number of points km have to be carefully chosen. The distribution of the ζ grid can be uniform or stretched.

A streamwise marching method to solve the transformed equations has been presented. The solution in the surface normal direction is obtained to fourth-order accuracy by a two-point compact differencing method. The differencing used in the stream surface plane is of second-order. However, accuracy in the ξ direction can be improved by using a dense mesh in that direction. The grid distribution in the cross-flow direction is presently set up to be the same as that of the inviscid grid in order to avoid interpolation errors.

An interface procedure has been presented to accurately obtain the inviscid edge conditions. The surface Euler equations are solved here on the BL grid with the interpolated inviscid pressure distribution to obtain smooth and accurate edge velocity values.

The metrics of transformation of the equations from the Cartesian system to the curvilinear system are dependent on the way the curvilinear surface grid is defined and measured. It can be orthogonal or non-orthogonal

and the curvilinear coordinates need not necessarily be measured along the same direction as they are defined. The six surface gradients defined at each grid point uniquely determine the transformation. In the validation cases presented in the report, the curvilinear coordinates have been defined in several different ways to demonstrate the advantage of using the curvilinear coordinate system.

The program software implementing the solution procedure has been presented in detail. The programs are organized so that the 3D boundary layer solution or solution of any of its subsets can be accomplished by calling identical subroutines, but with different edge coefficients.

Finally, practical application of the software to solving boundary layer flow past real wings and fuselages have been discussed. The examples presented deal with a simple fuselage and wing geometry. However, the programs are written in a general way so that it should be possible to apply these programs to any fuselage or wing shape where the boundary layer flow assumptions hold good. Only minor modifications may be necessary depending on the case. Adequate details of the program structure have been presented in this report to enable a user to make the necessary modifications.

REFERENCES

1. Robert, K., "Higher-order Boundary Layer Equations for Three-dimensional, Compressible Flow," DLR-FB 76-36, 1976.
2. Hirchel, E.H. and Kordulla, W., *Shear Flow in Surface Oriented Coordinates,* Notes on Numerical Fluid Mechanics, Volume 4, Vieweg, 1981.
3. Anderson, O.L., "Calculation of Three-Dimensional Boundary Layers On Rotating Turbine Blades,", *Three-Dimensional Flow Phenomena in Fluid Machinery,* Proceedings of the Winter Annual Meeting, November 1985, ASME, 1985, pp. 121-132.
4. Wang, K.C., "On the Determination of the Zones of Influence and Dependence for Three-Dimensional Boundary Layer Equations," *Fluid Mech.* (1971), Vol.48, part 2, pp. 397-404.
5. Vatsa, V.N., and Davis, R.T., "The Use of Levy-Lees Variables in Three-dimensional Boundary-layer Flows," NASA CR-112315, 1973.
6. Shapiro, A.H., *The Dynamics and Thermodynamics of Compressible Fluid Flow* , Vol. 1, The Ronald Press Company, New York, 1953.
7. Iyer, V. and Harris, J.E., "Three-Dimensional Compressible Boundary Layer Calculations to Fourth Order Accuracy on Wings and Fuselages," AIAA 89-0130, January 1989.
8. Tassa, A., Atta, E.H. and Lemmerman, L.A., "A New Three-Dimensional Boundary Layer Calculation Method," AIAA-82-0224, January 1982.
9. Iyer, V. and Harris, J.E., "Solution of the Surface Euler Equations for Accurate Three-Dimensional Boundary Layer Analysis of Aerodynamic Configurations," AIAA Paper 87-1154-CP, January 1987.
10. Schlichting, H., *Boundary Layer Theory* , McGraw-Hill, Seventh Edition, 1979.
11. Harris, J.E. and Blanchard, D.K., "Computer Program for Solving Laminar, Transitional, or Turbulent Compressible Boundary Layer Equations for Two-Dimensional and Axisymmetric Flow," NASA TM-83207, February 1982.
12. White, F.M., *Viscous Fluid Flow* , McGraw-Hill, 1974.
13. Wang, K.C., "Three-dimensional Boundary Layer Near the Plane of Symmetry of a Spheroid at Incidence," *Journal of Fluid Mechanics,* (1970), Vol.43, Part 1, pp. 187-209.

14. Swanson, R.C. and Turkel, E., "Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations," AIAA Paper 87-1107-CP, June 1987.

15. Monnoyer, F., "Three Dimensional Higher Order Boundary Layers,", Von Karman Institute for Fluid Dynamics Technical Note 155, April 1985.

16. Currie, I.G., *Fundamental Mechanics of Fluids* , McGraw-Hill Book Company, 1974.

17. Moitra, A., "Euler Solutions for High-Speed Flow About Complex Three-Dimensional Configurations,", AIAA Paper 86-0246, January 1986.

18. Fillo, J.A. and Burbank, R., "Calculation of Three-Dimensional Laminar Boundary Layer Flows," *AIAA Journal,* V.10, No.3(1972), pp.353-355.

19. Cebeci, T., "Calculation of Three-Dimensional Boundary Layers, II. Three-Dimensional Flows in Cartesian Coordinates," *AIAA Journal* , Vol.13, No.8, August 1975, pp.1056-1064.

20. Hess, J.L., "Calculation of Potential Flow About Arbitrary Three-Dimensional Lifting Bodies," Report MDC J5679/01 (Contract N00019-C-71-0524), Douglas Aircraft Co., October 1972. (available from DTIC as AD 755 480.)

APPENDIX: PROGRAM DETAILS

The boundary layer solution program is designed to provide for user interaction and flexibility in application. Depending on the particular problem being analyzed, program modifications will be necessary. Additional inputs or outputs may be required. This implies that the user has to be familiar with the structure of the program, variables used and the program logic. The theoretical base has already been presented. In this Appendix, additional information will be provided to enable the user to modify the programs as required. A formal input/output description is therefore not given. Instead, it is proposed to present a description of the program structure, the various subroutines and their function in the overall program. A list of computer variables and its corresponding mathematical symbols are given. Information is also given about the library subroutines and on running the program. A UNIX environment is assumed, though the programs can easily be adapted to and can be run on non-UNIX machines as well.

A summary of the procedure and details of inputs required to run each of the test cases is included. Also given are descriptions of the subroutines and a list of the program variables. Finally, a complete listing of the programs required to run Case 10 of flow past a swept wing is given.

Program Size

The BL program size is mainly dependent on the BL grid dimensions, im [nx] †, jm [ny], and km [nz]. These are set using PARAMETER statements in the 'include' COMMON blocks file called 'lib/com'. The program memory requirement calculated from the variable dimension statements alone is approximately, 15(im.jm)+20(jm.km)+70(km). Added to this will be the memory to store the program itself (which is much less than array storage requirements). As an example, assuming single precision variables and im=100, jm=30, km=50, the program size will be in the nieghbourhood of 300 Kilobytes. The advantage of this small size is two-fold: desktops or workstations are adequate, and faster execution (interactive processing is possible). All the example cases in this report were run on a SUN 3/150 workstation with 2 Megabytes of memory.

---

† Computer variable names are enclosed in square brackets

Units

Since the governing equations are non-dimensionalized by reference quantities, units are important only when specifying inputs such as free-stream pressure p_∞^* [pfs], free-stream temperature T_∞^* [tfs], wall temperature T_w^* [tws(i,j)], wall normal velocity w_w^* [wws(i,j)] and reference length L^* [reflen]. In addition, units are involved in the calculation of output quantities such as boundary layer thickness $t_{x,p}^*$ [deltax], displacement thickness δ_x^* [dispx] momentum thickness θ_x^* [thetax] and wall heat transfer rate $\dot{q}_w$ [qdot]. A flag variable [iunit] is provided in the programs to operate in either US units [iunit=0] or SI units [iunit=1].

[iunit]=0 => U.S. Customary units (US)

p_∞^* [pfs] in lb/ft^2

T_∞^* [tfs] and T_w^* [tws(i,j)] in ° R

L^* [reflen] in ft.

w_w^* [wws(i,j)] in ft/sec

R^* [rstar] = 1716.6 $ft^2/(\ sec^2 °R\ )$

S^* [sstar] = 198.6 °R

μ_s^* [amustd] = 2.27e–08 $lb\ sec/(\ ft^2\ °R^{1/2})$

ρ_w^* in $lb\ sec^2/ft^4$

$\dot{q}_w^*$ in lb/(ft-sec)

[iunit]=1 => International system of units (SI)

p_∞^* [pfs] in Pascal (i.e. $Newton/m^2$)

T_∞^* [tfs] and T_w^* [tws(i,j)] in ° K

L^* [reflen] in m.

w_w^* [wws(i,j)] in m/sec

R^* [rstar] = 287.059 $m^2/(\ sec^2 °K\ )$

S^* [sstar] = 110.333 °K

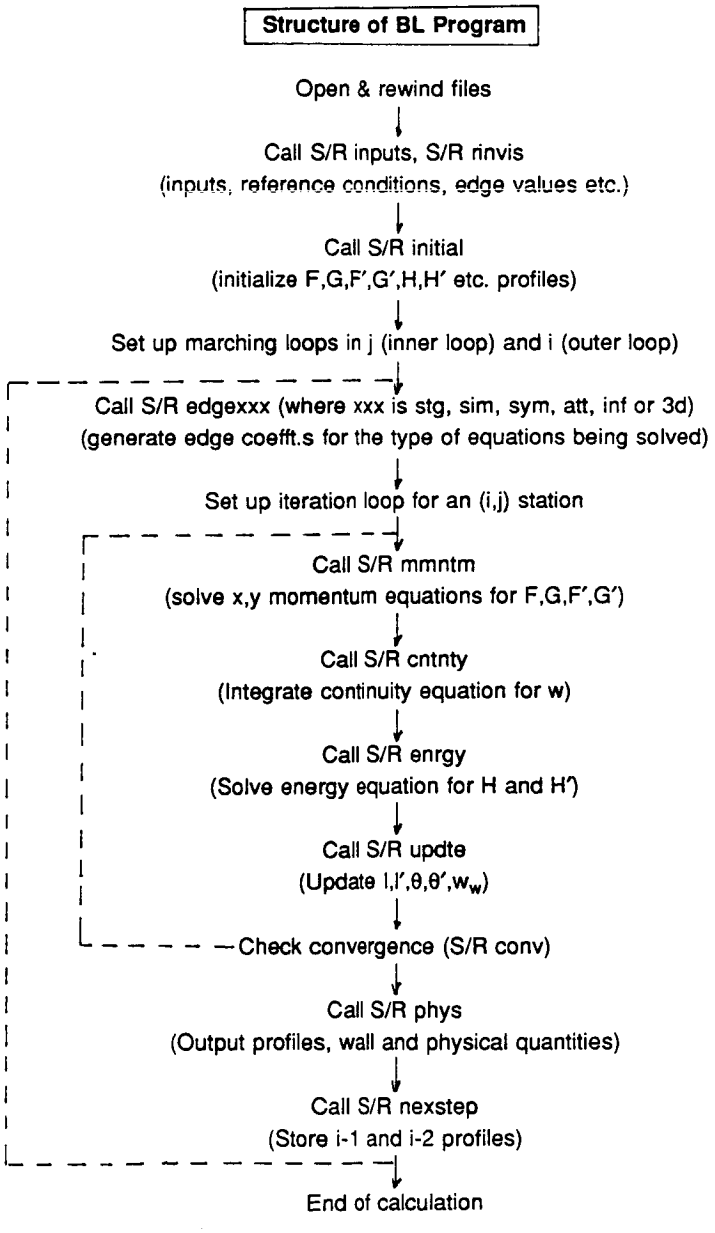
μ_s^* [amustd] = 2.27e–08 $lb\ sec/(\ ft^2\ °R^{1/2})$

ρ_w^* in $Newton\ sec^2/m^4$

$\dot{q}_w^*$ in Newton/(m-sec)

Program Structure of the BL Solver

The program is designed so that it can be run for a 2D case, axisymmetric case or a 3D case with appropriate modifications. Running a 2D case involves setting the dimension jm to 1 and suppressing the cross-flow gradients. The axisymmetric case can be treated as a symmetry line flow with metric h_2 defined by the local radius of the body at each location. In all cases, the flow equations are solved by the identical subroutines. The program structure is outlined in the following sketch:



The program first reads the inputs (S/R inputs), followed by reading/calculation of the BL grid, metrics and edge conditions (S/R rinvis). The input quantities in S/R inputs are directly included in the subroutine itself rather than reading from a file. This is done to keep the program flexible. However, it also means that the program will have to be recompiled each time any of the inputs are changed. Most of the inputs and calculated reference conditions (computed in S/R ref of file lib/blk.f) are output to file fort.2.

The inputs in S/R rinvis corresponding to the edge conditions and metrics [x, y, ue, ve, h1, h2, g12] are, in some cases, read from unit 20. In other cases, these quantities are analytically generated by FORTRAN statements in a part of this subroutine. This subroutine S/R rinvis also includes calculation of ρ_e, T_e, μ_e [roe, te, amue] from the above quantities. The calculation of ρ_e presently assumes that there are no shocks present in the flow between free-stream and the BL edge. If the body is sharp-tipped and in supersonic flow, the conditions downstream of the oblique shock can be specified as the free-stream condition with no other change. Otherwise, the isentropic relation to calculate ρ_e will have to be replaced by an expression for ρ_e from p, u_e and v_e.

The profiles at i=1 are initialized in S/R initial. Usually an initial Falkner-Skan or Karman-Polhausen profile is used to start the iterative process. The viscosity ratios l, l_1, l_2 and l_3 are initialized to unity and their normal gradients to zero.

At each point (i,j), depending on the type of flow, the edge coefficients are then generated. The expressions for these coefficients are listed in Tables 2-7. These coefficients are functions of the gradients of the edge conditions and metrics in ξ and η directions. These coefficients are calculated in S/R edgexxx, where 'xxx' stands for one of the following:

- stg for stagnation point flow
- sim for similarity profiles
- sym for symmetry line flow
- inf for infinite swept wing (locally or otherwise) flow
- att for attachment line flow
- 3d for general 3-D flow

These subroutines also contain statements to calculate a_i [a1,a2,a3] and b_i [b0,b1,b2,...b7], the ξ and η direction differencing coefficients. The flags [ix] and [iy] are also specified in these subroutines, which can

selectively suppress the gradients in the ξ or η directions to simulate 2D flow, similarity flow etc.. Most of these quantities are in the common block called 'edge'. This common block is a part of the 'include' file lib/com. The common blocks are included in the program by the use of the 'include' statement. For compilers which do not support the 'include' feature, the entire common block file can be copied to the program in place of the 'include' statement.

The solution of the x and y momentum equations at location (i,j) is done in S/R mmntm (located in file lib/blk.f). This routine calls another routine called S/R getfg (also located in lib/blk.f). S/R getfg obtains the values of F, G, F′ and G′ [f,g,fp,gp] at (i,j,k) and 26 other nieghboring points corresponding to different combinations of {i= i,i1,i2}, {j= j,j1,j2,j3,j4,j5,j6,j7} and {k= k,k1,k2}. These 108 values (evaluated once per iteration at a given (i,j)) are contained in common blocks in the 'include' file lib/comloc. The block-tridiagonal system elements of [a], [b], [c] and [r] are then computed as given in Table 8. The solution of the system is efficiently achieved through S/R blklu (LU decomposition with full pivoting), S/R blksolv (inversion of LU decomposed system) and a number of other other supporting routines, all contained in the file lib/lib.f. The F, G, F′ and G′ values are then updated.

The continuity equation is then solved in S/R cntnty (located in file lib/blk.f). This involves fourth-order accurate integration of the continuity equation with the current (n-th) iteration values of F, G, F′ and G′ (obtained by another call to S/R getfg). The transformed normal velocity at the wall, w_w [ww] is used here as the initial value for the integration. The value of w_w is updated at the end of each iteration in S/R updte from the physical normal velocity w_w^* [wws] and the current solution.

The solution of the energy equation is achieved in S/R enrgy and follows closely the procedure for solving the momentum equations. However, in the present case, h and h′ [h,hp] values at (i,j) and surrounding points are evaluated by a call to S/R gethi in addition to a call to S/R getfg. Furthermore, the calculation of D_5 [dd5] and D'_5 [dd5p] is also accomplished in the subroutine S/R gethi, based on the current solution profile. The block-tridiagonal system corresponding to Table 9 is then formed. The wall condition type [iwall=0,1 or 2] determines the elements [b] and [r] at k=1. If iwall=1 (temperature specified), the wall total enthalpy [hw] is specified; otherwise [iwall=0 or 2], the normal gradient of the total enthalpy [hpw] is the specified quantity. The system is inverted using the same subroutines as in S/R mmntm.

S/R updte is the final step in the iterative loop, wherein the viscosity and the density ratios and the wall normal velocity are updated based on the current solution.

Convergence is tested on the ratios $\frac{|\delta F'_{max}|}{|F'_{max}|}$ and $\frac{|\delta H'_{max}|}{|H'_{max}|}$. Upon convergence, the physical quantities and profiles are calculated in S/R phys based on expressions given in Chap. 6.

The η and ξ direction sweeps are continued till [ny]=[nylim] and [nx]=[nxlim]. The storing of the profiles corresonding to the i-1 and i-2 levels at the end of a η sweep is accomplished in S/R nextep. Differring definitions of F and G at the boundary compared to the interior are also suitably handled in this subroutine.

The dimensions of various arrays [nx], [ny], [nz] are set depending on the case, in the 'include' block lib/com. When the array are changed, the subroutines in the library files lib/lib.f and lib/blk.f are recompiled and loaded along with the main program. The main program is case-dependent, whereas, the library files are essentially the same except for changes in array dimensions.

Given below are decriptions of all of the subroutines used in the BL calculation. This is followed by a list of all the computer variables used in the program and corresponding symbols. Subsequent to this, brief descriptions of each of the test cases and the inputs are given.

A complete listing of the programs required to run Case10 (flow past a swept wing) is also given subsequently. The complete set of programs for all the test cases can also be made available upon request.

Subroutine inputs †

- Read in inputs to the BL program
- Called from main program
- Makes calls to s/r rinvis and s/r ref
- Main inputs are:

  p^*_∞ [pfs]

  T^*_∞ [tfs]

  M_∞ [amach]

  L^* [reflen]

  Wall condition type:

  [iwall]=0 --> adiabatic wall

  [iwall]=1 --> wall temperature specified

  [iwall]=2 --> wall heat flux specified

  Units used:

  [iunit]=0 --> US units

  [iunit]=1 --> SI units

  Wall heat flux [qd] (if [iwall]=2), dimension ([nx],[ny])

  Wall mass injection rate, [wws], dimension ([nx],[ny])

  Number of stations in i and j direction, [nppi] and [nppj] for profile output

  Profile output stations, [ip] and [jp], dimension maximum of 100

  Normal grid parameters, ζ_e [zmax], k_e [ak]

  limits for i and j sweeps [nxlim] and [nylim]

  Convergence limits [epsf] and [epsh]

  BL edge locating constants [fedge] and [gedge]

  Other inputs as given in S/R rinvis and S/R ref

† Program variable names are enclosed in square brackets

Subroutine ref

- Located in library program file lib/blk.f
- Calculates reference quantities
- Called by S/R inputs
- Some of the reference quantities calculated are:

Reference temperature T^*_{ref} [trs]

Free-stream sonic velocity c^*_∞ [velson]

Free-stream velocity u^*_∞ [ufs]

Reference viscosity μ^*_{ref} [amuref]

Free-stream viscosity μ^*_∞ [amuinf]

Reference Reynolds number per unit length [rereff]

Free stream Reynolds number per unit length [refsf]

Reference total enthalpy H^*_{ref} [hrefs]

Edge enthalpy, H_e [htot]

Subroutine rinvis

- Called from S/R inputs
- Read/specify the following:

  BL grid [x] (dimension [nx]) and [y] (dimension [ny])

  Edge velocities [ue] and [ve] (both dimensioned to ([nx],[ny]))

  Velocity gradients [fe] (dimension [ny]) or [vey] (dimension [nx]), if required

  Metrics h_1 [h1], h_2 [h2] and g_{12} [g12], all dimensioned to ([nx],[ny])

- Calculate T_e [te] and ρ_e [roe], both dimensioned to ([nx],[ny])

  Calculation of [te] assumes constant total enthalpy at BL edge

  Calculation of [roe] based on isentropic process from free-stream

Subroutine initial

- Called once from the main program

- Initializes profiles to start the iterative calculation

Initialize F [f] and G [g]; fourth-order plynomial used

f(j,nz) is generally set to 1; g(j,nz) set to $\frac{v_e}{v_r}$

Initialize F′ [fp] and G′ [gp] by differentiating F and G

Initialize H and H′; H set to H_e; H′ set to zero

Initialize w,I_1,I_2,I_3,I'_1,I'_2,I'_3,I,I′,θ and θ′

Subroutine updte

- Called from main program after each iteration

- Updates quantities I_1,I_2,I_3,I'_1,I'_2,I'_3,I,I′,θ and θ′

- If mass injection present, update w_w

  w_w is a function of current solution and the specified w_w^*

- Update I_w if q_w^* is specified (iwall=2).

- Simplified turbulence and transition models can be included

  ε_1, ε_2, ε_H and σ can be used in eq.s (38), (42), (45) and (46) to calculate I_1, I_2 and I_3

  $\varepsilon_1 = \varepsilon_2 = \varepsilon_H = \sigma = 0$ assumed

Subroutine edgexxx

- xxx stands for one of the following:

  stg 3D stagnation point flow

  sim Similarity flow

  sym Symmetry line flow

  3d 3D BL flow

  att Attachment line flow

  inf Infinite swept wing flow

- Called from main program once for each (i,j) location

- Evaluates the following edge coefficients:

  $\tilde{A}_1,\tilde{A}_2,\tilde{A}_3,\tilde{A}_4$ or [aa1s],[aa2s],[aa3s],[aa4s]
  $B_1,B_2,...,B_6,\tilde{B}_3,\tilde{B}_4$ or [bb1],[bb2],...,[bb6],[b3s],[b4s]
  $C_1,C_2,...,C_6,\tilde{C}_4,\tilde{C}_5$ or [cc1],[cc2],...,[cc6],[cc4s],[cc5s]
  D_1,D_2,D_3,D_4 or [dd1],[dd2],[dd3],[dd4]
  ξ differencing coefficients, a_1,a_2,a_3 or [a1],[a2],[a3]
  η differencing coefficients, $b_0,b_1,b_2,...,b_7$ or [b0],[b1],[b2],...,[b7]
  Corresponding j indices, [j1],[j2],[j3],[j4],[j5],[j6],[j7]

- specifies edge value g(j,nz)

- specifies ξ and η differencing flags, [ix] and [iy]
  [ix]=0 indicates no x differencing, [iy]=0 indicates no y differencing

- evaluates metric factors $C_{24},C_{25},C_{26},C_{34},C_{35},C_{36}$
  The quantities below may be evaluated depending on the case:
  $\partial h_1/\partial x,\partial h_1/\partial y$, $\partial h_2/\partial x,\partial h_2/\partial y$, $\partial g_{12}/\partial x,\partial g_{12}/\partial y$, $\partial u_e/\partial x,\partial u_e/\partial y$, $\partial v_e/\partial x,\partial v_e/\partial y$

Subroutine nexstep

- Called at the end of each η sweep

- F,F′,G,G′,H,H′ values at i and (i-1) levels are stored in arrays before advancing to the next i level

  [f1],[fp1],[g1],[gp1],[hh1],[hp1] arrays used to store (i-1) level profiles

  [f2],[fp2],[g2],[gp2],[hh2],[hp2] arrays used to store (i-2) level profiles

Subroutine phys

- Transformation to physical quantities after solution at (i,j) is obtained

- Calculates the following quantities:

  $t^*_{x,p}$, $t^*_{y,p}$, $\theta^*_{x,p}$, $\theta^*_{y,p}$, $C_{f,x-\infty}$, $C_{f,y-\infty}$, $\dot{q}^*_w$ and other quantities as required

- If i=ip(...) or j=jp(...), outputs profiles of z^*,u^*,v^*,T^*,ρ^* etc..

Subroutine mmntm

- Located in lib/blk.f

- Called once per iteration for each (i,j)

- Forms the block-tridiagonal matrix corresponding to the discretized momentum equations

- Inverts the system and updates F,F′,G,G′

- Calls S/R getfg to evaluate F,F′,G,G′ values at point (i,j,k) and 26 points surrounding it, required in consructing the matrix; S/R getfg called km times from S/R mmntm

- S/R blklu and S/R blksolv in lib/lib.f called to invert the system

Subroutine cntnty

- Located in lib/blk.f; called once per iteration for each (i,j)

- Integrates the continuity equation to fourth-order accuracy with a given initial value [ww] of wall normal velocity

- Calls S/R getfg (km-1) times to evaluate F,F′,G,G′ values at point (i,j,k) and 26 points surrounding it, required in the integration

Subroutine enrgy

- Located in lib/blk.f; called once per iteration for each (i,j)

- Evaluates the block-tridiagonal matrix corresponding to the discretized energy equation

- Inverts the system and updates H, H′

- Calls S/R gethi and S/R getfg to evaluate H,H′,F,F′,G,G′ values at (i,j,k) and 26 surrounding points ; each subroutine called km times.

- S/R blklu and S/R blksolv in lib/lib.f called to invert the system

Subroutine getfg

- Evaluates F,F′,G,G′ values at (i,j,k) and 26 surrounding points

- The points are (i,j), (i,j-1), (i,j+1), (i-1,j), (i-1,j+1), (i-1,j+2), (i-1,j-1), (i-1,j-2), (i-2,j) at levels k, (k-1) and (k+1)

- Boundary adjacent points handled separately

Subroutine gethi

- Evaluates H,H′ values at (i,j,k) and 26 surrounding points
- Evaluates D_5 and D'_5 once per iteration

Library subroutines

- Located in lib/lib.f
- Given below is a list of library subroutines/ functions:

  Function deri22: calculates derivative of yp (2D array) with respect to xp (2D array) at point (i,j) in the direction i

  Function derj22: calculates derivative of yp (2D array) with respect to xp (2D array) at point (i,j) in the direction j

  Function derj21: calculates derivative of yp (2D array) with respect to xp (1D array) at point (i,j) in the direction j

  Function deri21: calculates derivative of yp (2D array) with respect to xp (1D array) at point (i,j) in the direction i

  Function derk: calculates derivative of yp (1D array) with respect to xp (1D array) at point k

  Function sumint: Integrates function yp (1D array) with respect to xp (1D array) from 1 to i by trapezoidal formula

  Function sumx21: Integrates function yp (2D array) with respect to xp (1D array) from 1 to i by trapezoidal formula

  Function sumx22: Integrates function yp (2D array) with respect to xp (2D array) from 1 to i by trapezoidal formula

  S/R blklu: LU decomposition of a block-tridiagonal matrix (full pivoting)

  S/R blksolv: Inversion of a LU decomposed system; supporting routines are; S/R cmult, S/R lu, S/R solve, S/R ax

  Function quad: quadratic interpolation using (x1,y1), (x2,y2), (x3,y3)

  Function stlin: Linear interpolation at x given (x1,y1), (x2,y2)

  S/R solv3: Solves a 3x3 system [a].[x]=[b]

  Functions sind, cosd, tand: sin, cos and tan of angles expressed in degrees.

Important Variables Used in the Program

Program Variable	Symbol	Description
a	-	lower diagonal block of block-tridiagonal sytem Maximum dimension a(4,4,nz) in s/r mmntm and a(2,2,nz) in s/r enrgy
aa1s	$\bar{A}_1$	Coefficient of transformed continuity equation, evaluated in s/r edgexxx (where xxx stands for stg, sim, sym, 3d, inf or att)
aa2s	$\bar{A}_2$	Coefficient of transformed continuity equation (s/r edgexxx)
aa3s	$\bar{A}_3$	Coefficient of transformed continuity equation (s/r edgexxx)
aa4s	$\bar{A}_4$	Coefficient of transformed continuity equation (s/r edgexxx)
ak	k_e	Exponential stretching factor for ζ (s/r inputs)
akl	-	$h_2\sqrt{\rho_e\mu_e u_e}$ at i (s/r edgexxx)
akl1	-	$h_2\sqrt{\rho_e\mu_e u_e}$ at i-1
akl2	-	$h_2\sqrt{\rho_e\mu_e u_e}$ at i-2
akx	-	ξ derivative of $h_2\sqrt{\rho_e\mu_e u_e}$
al	l	$\frac{\rho\mu}{\rho_e\mu_e}$ (s/r updte), dimensioned to al(nz)
all	l_3	$\frac{l}{\sigma}(1+\frac{\varepsilon_H}{\mu}\frac{\sigma}{\sigma_t}\Gamma)$ (s/r updte), dimensioned to all(nz)
allp	l'_3	Normal derivative of l_3 (s/r updte), array allp(nz)
alp	l'	Normal derivative of l (s/r updte), array alp(nz)
alpha	α	Angle of attack in degrees
al1	l_1	$l\,(1+\frac{\varepsilon_1}{\mu}\Gamma)$ (s/r updte), dimensioned to al1(nz)
al1p	l'_1	Normal derivative of l_1 (s/r updte), dimensioned to al1p(nz)
al2	l_2	$l\,(1+\frac{\varepsilon_2}{\mu}\Gamma)$ (s/r updte), dimensioned to al2(nz)
al2p	l'_2	Normal derivative of l_2 (s/r updte), dimensioned to al2p(nz)
amach	M_∞	Mach number, input in s/r inputs
amue	μ_e	non-dimensional edge viscosity dimensioned to amue(nx,ny) calculated in s/r rinvis
amui	μ_∞	non-dimensional free-stream viscosity (s/r ref)
amuinf	μ^*_∞	Dimensional free-stream viscosity at T^*_∞; units, Pa.sec (SI) or lb.sec/(ft**2) (US)
amuref	μ^*_{ref}	Dimensional reference viscosity at T^*_{ref}; units, Pa.sec (SI) or lb.sec/(ft**2) (US)
amustd	μ^*_s	Sutherland viscosity coefficient Pa.sec/(°K)**0.5 (SI) or lb.sec/(ft**2)/(°R)**0.5 (US)

a1,a2,a3	a_1,a_2,a_3	coefficients of ξ derivative, Eq. 124
b	-	diagonal block of block-tridiagonal system maximum dimension b(4,4,nz) in s/r mmntm and b(2,2,nz) in s/r enrgy
bd	b_1	η differencing coefficient, eq 126
bd1	b_0	η differencing coefficient, eq 126
bd2	b_2	η differencing coefficient, eq 126
bd3 to bd7	b_3 to b_7	η differencing coefficients, eq 126
bet	β	$\gamma/(\gamma-1)$
b1	B_1	coefficient, x momentum (s/r edgexxx) where xxx stands for stg,sim,sym,3d,inf ot att
b2	B_2	coefficient, x momentum (s/r edgexxx)
b3	B_3	coefficient, x momentum (s/r edgexxx)
b3s	$\tilde{B}_3$	coefficient, x momentum (s/r edgexxx)
b4	B_4	coefficient, x momentum (s/r edgexxx)
b4s	$\tilde{B}_4$	coefficient, x momentum (s/r edgexxx)
b5	B_5	coefficient, x momentum (s/r edgexxx)
b6	B_6	coefficient, x momentum (s/r edgexxx)
c	-	super-diagonal block of block-tridiagonal sytem Maximum dimension c(4,4,nz) in s/r mmntm and c(2,2,nz) in s/r enrgy
cc1	C_1	coefficient, y momentum (s/r edgexxx) where xxx stands for stg,sim,sym,3d,inf ot att
cc2	C_2	coefficient, y momentum (s/r edgexxx)
cc3	C_3	coefficient, y momentum (s/r edgexxx)
cc4	C_4	coefficient, y momentum (s/r edgexxx)
cc4s	$\tilde{C}_4$	coefficient, y momentum (s/r edgexxx)
cc5	C_5	coefficient, y momentum (s/r edgexxx)
cc5s	$\tilde{C}_5$	coefficient, y momentum (s/r edgexxx)
cc6	C_6	coefficient, y momentum (s/r edgexxx)
cfxi	$C_{f,x-\infty}$	Skin friction coefficient in x direction
cfyi	$C_{f,y-\infty}$	Skin friction coefficient in y direction
crs	$C_{r,s}$	Stagnation point parameter (eq. 61)
c24l	C_{24}	Metric coefficient
c25l	C_{25}	Metric coefficient
c26l	C_{26}	Metric coefficient

c34l	C_{34}	Metric coefficient
c35l	C_{35}	Metric coefficient
c36l	C_{36}	Metric coefficient
dd1	D_1	coefficient, energy equation (s/r edgexxx) where xxx stands for stg,sim,sym,3d,inf ot att
dd2	D_2	coefficient, energy equation (s/r edgexxx)
dd3	D_3	coefficient, energy equation (s/r edgexxx)
dd4	D_4	coefficient, energy equation (s/r edgexxx)
dd5k	D_5	coefficient D_5 at point k
dd5k1	-	coefficient D_5 at point k-1
dd5k2	-	coefficient D_5 at point k+1
dd5pk	D'_5	D_5' at point k
dd5pk1	-	D_5' at point k-1
dd5pk2	-	D_5' at point k+1
dd6	-	$(\gamma-1)M_\infty^2\frac{1-\sigma}{\sigma}$
dd7	Cos β	$\frac{g_{12}}{h_1h_2}$
delfp	$\delta F'$	Change in fp of profile in any iteration
delhp	$\delta H'$	Change in hp of profile in any iteration
delta	t^*	boundary layer thickness, ft (US) or m (SI)
dfpmax	$\delta F'_{max}$	Maximum in profile of delfp
dhpmax	$\delta H'_{max}$	Maximum in profile of delhp
dispx	δ_x^*	Displacement thickness in x direction, in ft (US) or m (SI)
dpx	-	pressure gradient in x direction, local value
dpy	-	pressure gradient in y direction, local value
dxi	Δx_i	Step size in ξ at i = x_i-x_{i-1}
dxi1	Δx_{i-1}	Step size in ξ at i-1 = x_{i-1}-x_{i-2}
epsf	ε_F	Convergence limit for momentum equations
epsh	ε_I	Convergence limit for energy equation
erru	-	Error tolerance on u_e for S.E. solution
errv	-	Error tolerance on v_e for S.E. solution
f	F	u/u_e, dimensioned to f(ny,nz)
fe	f_e	$\partial u_e/\partial\xi$, dimensioned to fe(ny)

fedge	-	value of F to find edge of boundary layer, =0.99, for example
fp	F'	dimensioned to fp(ny,nz)
fp1	$F'_{i-1,j}$	dimensioned to fp1(ny,nz)
fp2	$F'_{i-2,j}$	dimensioned to fp2(ny,nz)
f1	$F_{i-1,j}$	dimensioned to f1(ny,nz)
f2	$F_{i-2,j}$	dimensioned to f2(ny,nz)
g	G	v/v_r dimensioned to G(ny,nz)
gam	γ	ratio of specific heats
gp	G'	dimensioned to gp(ny,nz)
gp1	$G'_{i-1,j}$	dimensioned to gp1(ny,nz)
gp2	$G'_{i-2,j}$	dimensioned to gp2(ny,nz)
gref	v_r	Reference quantity for v
grefx	$\partial v_r/\partial x$	
grefy	$\partial v_r/\partial y$	
g1	$G_{i-1,j}$	dimensioned to g1(ny,nz)
g12	g_{12}	metric coefficient, dimension g12(nx,ny)
g12l	g_{12}	local value of g_{12} at i,j
g12xl	$\partial g_{12}/\partial x$	local value of x gradient of g_{12}
g12yl	$\partial g_{12}/\partial y$	local value of y gradient of g_{12}
g2	$G_{i-2,j}$	dimensioned to g2(ny,nz)
h	H	non-dimensional total enthalpy, dimensioned to h(ny,nz)
hcon	-	$\frac{1}{M_\infty^2(\gamma-1)}$
hh1	H_{i-1}	H at i-1 level, dimensioned to hh1(ny,nz)
hh2	H_{i-2}	H at i-2 level, dimensioned to hh2(ny,nz)
hp	H'	normal gradient of H, dimensioned to hp(ny,nz)
hpw	H'_w	local value of H' at the wall, i.e. at (i,j,1)
hp1	H'_{i-1}	H'at i-1 level, dimensioned to hp1(ny,nz)
hp2	H'_{i-2}	H'at i-2 level, dimensioned to hp2(ny,nz)
hrefs	H^*_{ref}	Reference total enthalpy value, eq.(14), $\frac{ft^2}{sec^2}$ (US) or $\frac{m^2}{sec^2}$ (SI)
htot	H_e	BL edge total enthalpy, eq.(15)
hw	H_w	local value of H at the wall, i.e. at (i,j,1)

h1	h_1	metric coefficient, array h1(nx,ny)
h2	h_2	metric coefficient, array h2(nx,ny)
i	i	index in the x direction
ii	-	maximum dimension of inviscid grid in x direction
iord	-	i value at which x differencing becomes 2nd order
ip	-	i values for which profiles are to be output, array ip(nppi)
it	-	iteration number
itmax	-	maximum number of iterations
iunit	-	unit flag, =0 for US units, =1 for SI units
iwall	-	wall condition flag, =0 for adiabatic wall =1, temperature specified, =2 heat flux specified
ix	-	flag, =0 indicates no discretization in the x direction
iy	-	flag, =0 indicates no discretization in the y direction
j	j	index in the crossflow or y direction
jbl	-	flag for boundary-adjacent point, =0 for interior point =1 if j=2, =2 of j=nylim-1, =3 if j=ny
ji	-	maximum dimension of inviscid grid in y direction
j1	j_1	j-1
j2	j_2	j-2
j3	j	j
j4	j_4	j+1 at level i-1
j5	j_5	j+2 at level i-1
j6	j_6	j-1 at level i-1
j7	j_7	j-2 at level i-1
k	k	index in the z or ζ direction
kmaxe	-	k value corresponding to largest $\delta H'$ in an iteration
kmaxm	-	k value corresponding to largest $\delta F'$ in an iteration
k1	k_1	k-1
k2	k_2	k+1
nppi	-	number of i stations at which profile output is required
nppj	-	number of j stations at which profile output is required
nx	im	maximum dimension of BL grid in x direction
nxlim	-	value of i at which streamwise march is to be stopped
ny	jm	maximum dimension of BL grid in y direction

nylim	-	value of j at which y march is to be stopped
nz	km	maximum dimension of grid in the normal (ζ) direction
nz1	-	km-1
om	ω	factor controlling x differencing order
pb	C_p	pressure coefficient on the BL grid array dimensioned to pb(nx,ny)
ph	-	pressure coefficient on the inviscid grid array dimensioned to ph(ii,ji)
pfs	p_∞^*	free stream static pressure, lb/ft^2 or Pa.
phil	ϕ	value $\sqrt{\rho_e \mu_e u_e s_1}$ at i,j
pi	π	
prl	σ	Prandtl number (laminar value)
psi	ψ	$\sqrt{u_e/(\rho_e \mu_e s_1)}$
qd	-	dimensional heat flux, lb/(ft.sec), input s/r inputs
qdot	-	dimensional heat flux, calculated s/r phys
reflen	L^*	reference length, ft or m.
refs	Re_∞	free-stream Reynolds number based on x^*
refsf	-	free-stream Reynolds number per unit length
reinff	-	Free stream Reynolds number per unit length
rereff	-	Reference Reynolds number per unit length
reref	Re_{ref}	Reference Reynolds number based on L^*
roe	ρ_e	non-dimensional edge density, array roe(nx,ny)
rofs	ρ_∞^*	free-stream density, lb sec^2/ft^4 (US) or Pa sec^2/m^2 (SI)
rstar	R^*	Gas constant, sec^2°R (US) or sec^2°K (SI)
s	-	partial derivatives of (x,y) w.r.t. (xb,yb,zb); array s(6,nx,ny)
sstar	S^*	Reference temperature for Sutherland formula, °R (US) or (SI)
stild	$\tilde{S}$	$\frac{S^*}{T_{ref}^* T_e}$
stilde	S	$\frac{S^*}{T_{ref}^*}$
s1	s_1	distance in x direction over surface
te	T_e	edge temperature, array te(nx,ny)
tf	-	free-stream temperature, non-dimensional
tfs	T_∞^*	free-stream temperature, °R (US) or °K (SI)

thetax	θ_x^*	momentum thickness in x direction, ft. or m.
the1	θ	ratio, $\rho_e/\rho = T/T_e$
the2	θ'	normal gradient of θ
tphys	T^*	dimensional value of temperature, °R or °K
trs	T^*_{ref}	reference temperature, u_∞^2/R^*
tws	T^*_w	dimensional value of wall temperature
u	u	non-dimensional vcelocity in x direction
uden	-	magnitude of reference velocity for F
ue	u_e	edge velocity, array (nx,ny)
ufs	u^*_∞	free stream velocity, ft/s or m/s
uh	-	Inviscid velocity component (Cartesian) in x' direction
us	-	Inviscid surface velocity, interpolated
uphys	u^*	dimensional streamwise velocity, ft/s or m/s
v	v	non-dimensional velocity in y direction
vden	-	magnitude of reference velocity for G
ve	v_e	edge velocity, array (nx,ny)
vh	-	Inviscid velocity component (Cartesian) in y' direction
us	-	Inviscid surface velocity, interpolated
velson	-	speed of sound, ft/s or m/sec
vey	g_e	y gradient of v_e
w	w	transformed normal velocity
wh	-	Inviscid velocity component (Cartesian) in z' direction
ws	-	Inviscid surface velocity, interpolated
ww	w_w	wall value of w
wws	w^*_w	wall value of w^*
x	ξ	streamwise coordinate, array x(nx)
xb	x'	Cartesian coordinate, array xb(nx,ny)
xh	-	Cartesian coordinate of inviscid grid, array xh(ii,ji)
xs	-	surface coordinate for inviscid grid, array xs(ii,ji)
y	η	crossflow coordinate, array y(ny)
yb	y'	cartesian coordinate, array yb(nx,ny)
yh	-	Cartesian coordinate of inviscid grid, array yh(ii,ji)
ys	-	surface coordinate for inviscid grid, array ys(ii,ji)

z	ζ	normal coordinate, array z(nz)
zh	-	Cartesian coordinate of inviscid grid, array zh(ii,ji)
zb	z'	cartesian coordinate
zcon	-	integrated normal distance scale for normal coordinate
zmax	ζ_e	edge value of ζ
zphys	z^*	dimensional value of normal coordinate, ft. or m., array zphys(nz)

Case 1: Program Notes

Program

File name is stg_bl/case1.f

Library routines are in lib/blk.f and lib/lib.f

To Run Program

Edit lib/com and set [nx]=1, [ny]=1, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile stg_bl/case1.f with f77 -o case1 case1.f ../lib/blk.o ../lib/lib.o

Run with : case1

Inputs

[crs], [pfs], [tfs], [amach] in S/R inputs

[fe(1)], [h1(1,1)] if physical quantities are required in S/R rinvis

Variations

Different [crs] values in S/R inputs (0 to 1); different values of [ak]

[iwall]=1 in S/R inputs, wall temperature [tws(1,1)] to be specified

[iwall]=2 in S/R inputs, wall heat flux [qd(1,1)] to be specified

[wws(1,1)] transpiration at the wall in units of velocity can be specified

Different [nz] values by editing lib/com and recompiling lib/blk.f

Program Description

This program corresponds to obtaining solution profiles at a single station i=1 and j=1. The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis. Calculation of reference quantities is done in S/R ref in lib/blk.f. The initial profiles are specified in S/R initial.

The edge coefficients as given in Table 1 are evaluated in S/R edgestg. The momentum equations are solved in S/R mmntm, the continuity equation in S/R cntnty and the energy equation in S/R enrgy. The subroutines getfg and gethi obtain the F, G, F′, G′, H, H′ values at points (i,j) and surrounding points depending on the x and y differencing flags [ix] and [iy]. In the present case, since [ix] and [iy] are set to zero in S/R edgestg, there is no x or y differencing. The block-tridiagonal system is solved by library routines contained in lib/lib.f. Update of viscosity and density ratios during the iteration is done in S/R updte. After convergence of the iteration, the physical quantities and profiles are output in S/R phys.

Results

fort.2	has output of input data and reference conditions
fort.3	profile output from S/R phys
fort.4	has output of physical quantities.
fort.10	has profile output; see S/R phys for the corresponding write statements; these write statements to be modified as required.

Caution

(1) To simulate a very cold wall case, [tws(1,1)] cannot be set to zero since formulation involves density ratio [the1] and quantities [al], [alp] etc. which are functions of [the1]. Setting [tws(1,1)] to a small value can be done for this case.

(2) [roe(i,j)] is calculated in S/R rinvis assuming isentropic process between free-stream and BL edge. If a detached shock is present in front of the blunt body, [roe(i,j)] is to be calculated based on normal shock relations. Alternatively, conditions downstream of the shock can be specified.

```
Output file stg_bl/fort.2_case1
-------------------------------

Stagnation point flow Case 1

Physical quantities are in U.S. units

Wall condition : Adiabatic wall

Inputs

nx, ny, nz =   1,   1,  41
zmax =    6.00
reflen =   1.000000
pfs =   2116.000
tfs=     520.00
mach no =      0.500
trs=     182.00
ufs =     558.85
ref Re no per unit length =  0.9049514e+07
fs  Re no per unit length =  0.3537902e+07
fs sonic vel =    1117.70
amuinf =  0.3745792e-06
amuref =  0.1464415e-06
rofs =  0.2371347e-02
hrefs =  0.3123120e+07
f-edge =   0.990000


nxlim=    1
iwall= 0
iunit= 0
ak=   1.100000
epsf=  0.1000000e-04
epsh=  0.1000000e-04
crs=         0.

Wall conditions (tws,qd used only if iwall=1 or 2)

 i          wws             tws             qd

  1         0.        400.0000       -600.0000


zeta stretching factor ak=    1.10000
zeta distribution

        0.       0.1355647e-01  0.2846859e-01  0.4487193e-01
 0.6291560e-01   0.8276363e-01  0.1045965      0.1286126
 0.1550304       0.1840899      0.2160553      0.2512174
 0.2898956       0.3324416      0.3792422      0.4307230
 0.4873517       0.5496434      0.6181642      0.6935372
 0.7764474       0.8676486      0.9679699       1.078323
  1.199712        1.333240       1.480121        1.641689
  1.819415        2.014912       2.229960        2.466513
```

```
  2.726721        3.012949        3.327801        3.674138
  4.055108        4.474175        4.935149        5.442221
  6.000000

fe=  25.00000

iterating at i=  1 j=  1

iter=  1 dfp max= 0.3409062e+00 dhp max= 0.1957642e-06 at ( 1,39)
iter=  2 dfp max= 0.2060392e-01 dhp max= 0.1957642e-06 at ( 1,39)
iter=  3 dfp max= 0.3460014e-02 dhp max= 0.1953470e-06 at ( 1,39)
iter=  4 dfp max= 0.7061199e-03 dhp max= 0.1439067e-06 at ( 2,34)
iter=  5 dfp max= 0.1581609e-03 dhp max= 0.1503779e-06 at ( 2,33)
iter=  6 dfp max= 0.3731292e-04 dhp max= 0.3007552e-06 at ( 3,33)
iter=  7 dfp max= 0.8927242e-05 dhp max= 0.2255658e-06 at ( 9,33)

end of calculation


Output file stg_bl/fort.4_case1
-------------------------------

dispx= 0.6617032e-04 deltax= 0.2441325e-03
thetax= 0.2984530e-04
qdot=         0.
```

Case 2: Program Notes

Program

File name is sim_bl/case2.f

Library routines are in lib/blk.f and lib/lib.f

To Run Program

Edit lib/com and set [nx]=21, [ny]=3, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile sim_bl/case2.f with f77 -o case2 case2.f ../lib/blk.o ../lib/lib.o

Run with : case2

Inputs

[pfs], [tfs], [amach], [reflen] in S/R inputs; [iwall]=0

[nppi] and [ip] refer to i locations where profile output is required

[x(i),i=1,nx] distribution in S/R rinvis; [h1], [h2], [ue] are equal to 1

US units are used, [iunit]=0

Variations

[iwall]=1 in S/R inputs, wall temperature [tws(i,1),i=1,nx] to be specified

[iwall]=2 in S/R inputs, wall heat flux [qd(1,1),i=1,nx] to be specified

[amach]=0.05 to approximate incompressible flow

Program Description

This program corresponds to obtaining solution profiles at [nx] stations (at j=1) of a flat plate. However, [ny] is set to 3 since y gradient of the metrics are calculated in S/R edgesym (equal to zero in the present case). The y direction equation and gradients are suppressed. The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis. The non-dimensional free-stream velocity [ue(i,1),i=1,nx] is equal to unity as also are the metrics [h1(i,1)] and [h2(i,1)]. Calculation of reference quantities are done in S/R ref in lib/blk.f. The initial profiles are specified in S/R initial.

The edge coefficients corresponding to the similarity profile at leading edge i=1 are evaluated in S/R edgesmc (see Table 7). The x and y differencing flags [ix] and [iy] are set to zero. The edge coefficients corresponding the 2D flow downstream of the leading edge are evaluated in S/R edgesym, which are from Table 4 with the edge cross flow velocity gradient [dvy(i)] equated to zero. The x differencing flag [ix] is now set to 1 here which means that streamwise gradients will be calculated. The F, G, F′, G′, H, H′ profiles at i-1 and i-2 locations are stored at the end of iteration convergence in S/R nexstep. The momentum equations are solved in S/R mmntm, the continuity equation in S/R cntnty and the energy equation in S/R enrgy. The subroutines getfg and gethi obtain the

F, G, F′, G′, H, H′ values at points (i,j) and surrounding points depending on the x and y differencing flags [ix] and [iy]. When wall heat flux is specified, the corresponding H′ at the wall is calculated from equation (113). Since this equation involves quantities based on the current solution, H′ at the wall will be recalculated in S/R updte based on the current solution. The physical quantities of BL thickness [deltax], momentum thickness [thetax], displacement thickness [dispx], heat flux [qdot], skin friction coefficient [cfxi] as well as profiles at selected stations are output in S/R phys.

Results

fort.2	has output of input data and reference conditions
fort.3	has profile output; see S/R phys for the corresponding write statements; these write statements to be modified as required.
fort.4	wall and integrated quantities (S/R phys)
fort.10	profile output

file sim_bl/fort.2_case2, partial listing


```
 Flow past flat plate : Test case # 2

 Physical quantities are in U.S. units

 Wall condition : Wall heat flux specified

 Inputs

 nx, ny, nz =  21,   3,  41
 zmax =    8.00
 reflen =   5.000000
 pfs =   400.0000
 tfs=     400.00
 mach no =      3.000
 trs=    5040.00
 ufs =    2940.86
 ref Re no per unit length =  0.1105351e+07
 fs  Re no per unit length =  0.5649077e+07
 fs sonic vel =     980.29
 amuinf =  0.3033745e-06
 amuref =  0.1550445e-05
 rofs =  0.5827506e-03
 hrefs =  0.2402400e+07
 f-edge =   0.990000


 nxlim=   21
 iwall= 2
 iunit= 0
 ak=   1.100000
 epsf=  0.1000000e-04
 epsh=  0.5000000e-04
 iord=  5

 Wall conditions

  i          wws            tws            qd

  1          0.       1007.270             0.
  2          0.       1007.270             0.
  3          0.       1007.270             0.
  4          0.       1007.270             0.
  5          0.       1007.270             0.
  6          0.       1007.270             0.
  7          0.       1007.270             0.
  8          0.       1007.270             0.
  9          0.       1007.270             0.
 10          0.       1007.270             0.
 11          0.       1007.270             0.
 12          0.       1000.000      -86.67000
```

13	0.	980.0000	-283.2600
14	0.	960.0000	-401.2100
15	0.	940.0000	-500.1100
16	0.	920.0000	-588.1200
17	0.	900.0000	-668.2100
18	0.	880.0000	-742.4200
19	0.	860.0000	-811.5800
20	0.	840.0000	-877.1000
21	0.	820.0000	-939.2600

stretching factor ak= 1.10000
zeta distribution

0.	0.1807530e-01	0.3795813e-01	0.5982924e-01
0.8388747e-01	0.1103515	0.1394620	0.1714835
0.2067071	0.2454532	0.2880738	0.3349565
0.3865274	0.4432555	0.5056563	0.5742972
0.6498023	0.7328579	0.8242190	0.9247162
1.035263	1.156865	1.290627	1.437765
1.599617	1.777654	1.973494	2.188919
2.425886	2.686550	2.973280	3.288684
3.635628	4.017266	4.437068	4.898850
5.406810	5.965567	6.580199	7.256294
8.000000			

Profiles will be output at 3 stations
i= 11 i= 15 i= 21 i=
iflt= 1

i	x	ue
1	0.	1.000000
2	0.5000000e-02	1.000000
3	0.1000000e-01	1.000000
4	0.1500000e-01	1.000000
5	0.2000000e-01	1.000000
6	0.2500000e-01	1.000000
7	0.3000000e-01	1.000000
8	0.3500000e-01	1.000000
9	0.4000000e-01	1.000000
10	0.4500000e-01	1.000000
11	0.5000000e-01	1.000000
12	0.5500000e-01	1.000000
13	0.6000000e-01	1.000000
14	0.6500000e-01	1.000000
15	0.7000000e-01	1.000000
16	0.7500000e-01	1.000000
17	0.8000000e-01	1.000000
18	0.8500000e-01	1.000000
19	0.9000000e-01	1.000000
20	0.9500000e-01	1.000000
21	0.9999999e-01	1.000000

file sim_bl/fort.4_case2

Results when run under adiabatic conditions

i	x	del99	dispx	cfre	cfxi	t-wall	qdot
2	0.5000000e-02	0.5451318e-03	0.3411990e-03	0.6088464	0.1620127e-02	1007.275	0.
3	0.1000000e-01	0.7709304e-03	0.4825265e-03	0.6088482	0.1145606e-02	1007.275	0.
4	0.1500000e-01	0.9441908e-03	0.5909703e-03	0.6088498	0.9353858e-03	1007.274	0.
5	0.2000000e-01	0.1090255e-02	0.6823926e-03	0.6088508	0.8100692e-03	1007.274	0.
6	0.2500000e-01	0.1218941e-02	0.7629370e-03	0.6088514	0.7245486e-03	1007.274	0.
7	0.3000000e-01	0.1335280e-02	0.8357536e-03	0.6088524	0.6614205e-03	1007.273	0.
8	0.3500000e-01	0.1442266e-02	0.9027158e-03	0.6088529	0.6123568e-03	1007.273	0.
9	0.4000000e-01	0.1541847e-02	0.9650433e-03	0.6088533	0.5728079e-03	1007.273	0.
10	0.4500000e-01	0.1635374e-02	0.1023582e-02	0.6088533	0.5400484e-03	1007.274	0.
11	0.5000000e-01	0.1723834e-02	0.1078950e-02	0.6088539	0.5123353e-03	1007.273	0.
12	0.5500000e-01	0.1807972e-02	0.1131611e-02	0.6088540	0.4884927e-03	1007.273	0.
13	0.6000000e-01	0.1888364e-02	0.1181929e-02	0.6088543	0.4676964e-03	1007.272	0.
14	0.6500000e-01	0.1965472e-02	0.1230190e-02	0.6088545	0.4493483e-03	1007.273	0.
15	0.7000000e-01	0.2039667e-02	0.1276628e-02	0.6088545	0.4330028e-03	1007.273	0.
16	0.7500000e-01	0.2111255e-02	0.1321435e-02	0.6088548	0.4183207e-03	1007.273	0.
17	0.8000000e-01	0.2180496e-02	0.1364772e-02	0.6088550	0.4050374e-03	1007.272	0.
18	0.8500000e-01	0.2247604e-02	0.1406775e-02	0.6088551	0.3929441e-03	1007.272	0.
19	0.9000000e-01	0.2312764e-02	0.1447558e-02	0.6088552	0.3818730e-03	1007.272	0.
20	0.9500000e-01	0.2376140e-02	0.1487224e-02	0.6088553	0.3716880e-03	1007.272	0.
21	0.9999999e-01	0.2437866e-02	0.1525859e-02	0.6088553	0.3622767e-03	1007.272	0.

file sim_bl/fort.4_case2

Results for iwall=1, wall temperature specified

i	x	del99	dispx	cfre	cfxi	t-wall	qdot
2	0.5000000e-02	0.5451317e-03	0.3411985e-03	0.6088454	0.1620124e-02	1007.270	-0.7315939
3	0.1000000e-01	0.7709293e-03	0.4825249e-03	0.6088482	0.1145606e-02	1007.270	-0.2586575
4	0.1500000e-01	0.9441890e-03	0.5909681e-03	0.6088501	0.9353863e-03	1007.270	-0.2111930
5	0.2000000e-01	0.1090254e-02	0.6823899e-03	0.6088513	0.8100700e-03	1007.270	0.
6	0.2500000e-01	0.1218939e-02	0.7629337e-03	0.6088521	0.7245495e-03	1007.270	0.
7	0.3000000e-01	0.1335277e-02	0.8357506e-03	0.6088529	0.6614210e-03	1007.270	0.
8	0.3500000e-01	0.1442263e-02	0.9027128e-03	0.6088533	0.6123573e-03	1007.270	0.
9	0.4000000e-01	0.1541844e-02	0.9650405e-03	0.6088536	0.5728081e-03	1007.270	-0.1293287
10	0.4500000e-01	0.1635371e-02	0.1023580e-02	0.6088539	0.5400489e-03	1007.270	0.
11	0.5000000e-01	0.1723832e-02	0.1078947e-02	0.6088542	0.5123356e-03	1007.270	0.1156751
12	0.5500000e-01	0.1806032e-02	0.1129680e-02	0.6094378	0.4889611e-03	1000.000	-86.67883
13	0.6000000e-01	0.1880208e-02	0.1173786e-02	0.6109868	0.4693345e-03	980.0000	-283.2556
14	0.6500000e-01	0.1948849e-02	0.1213569e-02	0.6125151	0.4520499e-03	959.9999	-401.2092
15	0.7000000e-01	0.2012855e-02	0.1249809e-02	0.6140729	0.4367141e-03	940.0000	-500.1109
16	0.7500000e-01	0.2072852e-02	0.1283016e-02	0.6156431	0.4229847e-03	920.0000	-588.1194
17	0.8000000e-01	0.2129280e-02	0.1313531e-02	0.6172200	0.4106022e-03	900.0000	-668.2148
18	0.8500000e-01	0.2182472e-02	0.1341604e-02	0.6188032	0.3993644e-03	880.0000	-742.4174
19	0.9000000e-01	0.2232689e-02	0.1367430e-02	0.6203939	0.3891102e-03	859.9999	-811.5812
20	0.9500000e-01	0.2280151e-02	0.1391165e-02	0.6219944	0.3797090e-03	839.9999	-877.0996
21	0.9999999e-01	0.2325044e-02	0.1412946e-02	0.6236057	0.3710533e-03	820.0000	-939.2571

file sim_bl/fort.4_case2

Results for iwall=2, wall heat flux specified

i	x	del99	dispx	cfre	cfxi	t-wall	qdot
2	0.5000000e-02	0.5451318e-03	0.3411990e-03	0.6088464	0.1620127e-02	1007.275	0.
3	0.1000000e-01	0.7709304e-03	0.4825265e-03	0.6088482	0.1145606e-02	1007.275	0.
4	0.1500000e-01	0.9441908e-03	0.5909703e-03	0.6088498	0.9353858e-03	1007.274	0.
5	0.2000000e-01	0.1090255e-02	0.6823926e-03	0.6088508	0.8100692e-03	1007.274	0.
6	0.2500000e-01	0.1218941e-02	0.7629370e-03	0.6088514	0.7245486e-03	1007.274	0.
7	0.3000000e-01	0.1335280e-02	0.8357536e-03	0.6088524	0.6614205e-03	1007.273	0.
8	0.3500000e-01	0.1442266e-02	0.9027158e-03	0.6088529	0.6123568e-03	1007.273	0.
9	0.4000000e-01	0.1541847e-02	0.9650433e-03	0.6088533	0.5728079e-03	1007.273	0.
10	0.4500000e-01	0.1635374e-02	0.1023582e-02	0.6088533	0.5400484e-03	1007.274	0.
11	0.5000000e-01	0.1723834e-02	0.1078950e-02	0.6088539	0.5123353e-03	1007.273	0.
12	0.5500000e-01	0.1806037e-02	0.1129686e-02	0.6094359	0.4889596e-03	1000.017	-86.66998
13	0.6000000e-01	0.1880213e-02	0.1173789e-02	0.6109861	0.4693340e-03	980.0002	-283.2599
14	0.6500000e-01	0.1948786e-02	0.1213517e-02	0.6125490	0.4520749e-03	959.9877	-401.2098
15	0.7000000e-01	0.2012768e-02	0.1249739e-02	0.6141149	0.4367439e-03	939.9987	-500.1099
16	0.7500000e-01	0.2072760e-02	0.1282943e-02	0.6156841	0.4230128e-03	919.9982	-588.1200
17	0.8000000e-01	0.2129190e-02	0.1313461e-02	0.6172570	0.4106268e-03	899.9979	-668.2100
18	0.8500000e-01	0.2182382e-02	0.1341535e-02	0.6188365	0.3993859e-03	879.9872	-742.4200
19	0.9000000e-01	0.2232605e-02	0.1367367e-02	0.6204227	0.3891281e-03	859.9975	-811.5801
20	0.9500000e-01	0.2280073e-02	0.1391107e-02	0.6220199	0.3797246e-03	839.9941	-877.1000
21	0.9999999e-01	0.2324970e-02	0.1412892e-02	0.6236280	0.3710666e-03	819.9958	-939.2601

Case 3: Program Notes

Program

File name is sim_bl/case3.f

Library routines are in lib/blk.f and lib/lib.f

To Run Program

Edit lib/com and set [nx]=101, [ny]=3, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile sim_bl/case3.f with f77 -o case3 case3.f ../lib/blk.o ../lib/lib.o

Run with : case3

Inputs

[pfs], [tfs], [amach], [reflen] in S/R inputs; [iunit]=1 (SI units)

These correspond to conditions downstream of the oblique shock

[iwall]=1; Wall temperature [tws(i,1),i=1,nxlim] specified

[nppi] and [ip] refer to i locations where profile output is required

[x(i),i=1,nx] distribution in S/R rinvis

[h1], [h2] are defined in S/R rinvis for a cone; [ue] is equal to 1

Mass injection [wws(i,1)] at wall specified in S/R inputs

Note that [wws] is input as density multiplied by normal velocity

Variations

[wws(i,1)] set to different values (suction, blowing or no injection)

Program Description

This program corresponds to obtaining solution profiles at [nx] stations (at j=1) of a cone in supersonic flow. However, [ny] is set to 3 since y gradient of the metrics are calculated in S/R edgesym (equal to zero in the present case). The y direction equation and gradients are suppressed. The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis. The non-dimensional free-stream velocity [ue(i,1),i=1,nx] is equal to unity The metrics [h1(i,1)] and [h2(i,1)] are functions of x and the cone half-angle. Calculation of reference quantities are done in S/R ref in lib/blk.f. The initial profiles are specified in S/R initial.

The edge coefficients corresponding to the similarity profile at leading edge i=1 are evaluated in S/R edgesmc (see Table 7). The x and y differencing flags [ix] and [iy] are set to zero. The edge coefficients corresponding the 2D flow downstream of the leading edge are evaluated in S/R edgesym, which are from Table 4 with the edge cross flow velocity gradient [dvy(i)] equated to zero. The x differencing flag [ix] is now set to 1 here which means that streamwise gradients will be calculated. The F, G, F′, G′, H, H′ profiles at i-1 and i-2 locations

are stored at the end of iteration convergence in S/R nexstep. The momentum equations are solved in S/R mmntm, the continuity equation in S/R cntnty and the energy equation in S/R enrgy. The subroutines getfg and gethi obtain the F, G, F′, G′, H, H′ values at points (i,j) and surrounding points depending on the x and y differencing flags [ix] and [iy]. The transformed velocity at the wall [ww] is calculated from equation (112). Since this equation involves quantities based on the current solution, [ww] is recalculated in S/R updte based on the current solution. The physical quantities of BL thickness [deltax], momentum thickness [thetax], displacement thickness [dispx], heat flux [qdot], skin friction coefficient [cfxi] as well as profiles at selected stations are output in S/R phys.

Results

fort.2	has output of input data and reference conditions
fort.3	has profile output; see S/R phys for the corresponding write statements; these write statements to be modified as required.
fort.4	wall and integrated quantities (S/R phys)
fort.1	profile output

file sim_bl/fort.2_case3: partial listing

Flow past a cone with mass injection: Case 3

Physical quantities are in S.I. units

Wall condition : Temperature specified

Inputs

```
nx, ny, nz = 101,   3,  41
zmax =    8.00
reflen =   1.000000
pfs =   1260.120
tfs=      82.53
mach no =      6.833
trs=    5394.64
ufs =    1244.42
ref Re no per unit length =  0.6306532E+06
fs  Re no per unit length =  0.1167644E+08
fs sonic vel =     182.12
amuinf =  0.5668721E-05
amuref =  0.1049554E-03
rofs =  0.5318981E-01
hrefs =  0.8291856E+05
f-edge =   0.995000


nxlim=   85
iwall= 1
iunit= 1
ak=   1.020000
epsf=  0.1000000E-03
epsh=  0.1000000E-03
iord=  5


stretching factor ak=    1.02000
zeta distribution

 0.0000000E+00  0.1324447      0.2675354      0.4053355
 0.5458827      0.6892467      0.8354774      0.9846325
  1.136768       1.291948       1.450235       1.611686
  1.776364       1.944339       2.115666       2.290430
  2.468679       2.650504       2.835959       3.025122
  3.218074       3.414877       3.615623       3.820383
  4.029237       4.242265       4.459562       4.681198
  4.907267       5.137857       5.373065       5.612971
  5.857677       6.107284       6.361872       6.621557
  6.886435       7.156616       7.432198       7.713285
  8.000000
x,wws  values

  1   0.000   0.000   2   0.005   0.000   3   0.010   0.000
```

4	0.015	0.000	5	0.020	0.000	6	0.025	0.000
7	0.030	0.000	8	0.035	0.000	9	0.040	0.000
10	0.045	0.000	11	0.050	0.000	12	0.055	0.000
13	0.060	0.000	14	0.065	0.000	15	0.070	0.000
16	0.075	0.000	17	0.080	0.000	18	0.085	0.000
19	0.090	0.000	20	0.091	0.000	21	0.092	0.000
22	0.093	0.000	23	0.094	0.000	24	0.095	0.000
25	0.096	0.000	26	0.097	0.000	27	0.098	0.000
28	0.099	0.000	29	0.100	0.000	30	0.101	0.000
31	0.102	0.000	32	0.103	0.000	33	0.104	0.000
34	0.105	0.000	35	0.106	0.000	36	0.107	0.000
37	0.108	0.000	38	0.109	0.000	39	0.110	0.000
40	0.111	0.000	41	0.112	0.000	42	0.113	0.000
43	0.114	0.000	44	0.115	0.000	45	0.116	0.000
46	0.117	0.000	47	0.118	0.000	48	0.119	0.000
49	0.120	0.000	50	0.125	0.000	51	0.130	0.000
52	0.135	0.000	53	0.140	0.000	54	0.145	0.000
55	0.150	0.000	56	0.155	0.000	57	0.160	0.000
58	0.165	0.000	59	0.170	0.000	60	0.175	0.000
61	0.180	0.000	62	0.185	0.000	63	0.190	0.000
64	0.195	0.000	65	0.200	0.000	66	0.205	0.000
67	0.210	0.000	68	0.215	0.000	69	0.220	0.000
70	0.225	0.000	71	0.230	0.000	72	0.235	0.000
73	0.240	0.000	74	0.245	0.000	75	0.250	0.000
76	0.255	0.000	77	0.260	0.000	78	0.265	0.000
79	0.270	0.000	80	0.275	0.000	81	0.280	0.000
82	0.285	0.000	83	0.290	0.000	84	0.295	0.000
85	0.300	0.000						

Partial listing of sim_bl/fort.4_case3 for wws=0 (no mass injection)

i	x	del99	cfxi	ue	vey
2	0.4999999E-02	0.1817591E-03	0.4455265E-02	1.000000	0.0000000E+00
3	0.9999998E-02	0.2570604E-03	0.3150324E-02	1.000000	0.0000000E+00
4	0.1500000E-01	0.3148592E-03	0.2572174E-02	1.000000	0.0000000E+00
5	0.2000000E-01	0.3636046E-03	0.2227505E-02	1.000000	0.0000000E+00
6	0.2499999E-01	0.4065582E-03	0.1992305E-02	1.000000	0.0000000E+00
7	0.2999999E-01	0.4453978E-03	0.1818688E-02	1.000000	0.0000000E+00
8	0.3499999E-01	0.4811205E-03	0.1683752E-02	1.000000	0.0000000E+00
9	0.3999999E-01	0.5143774E-03	0.1574979E-02	1.000000	0.0000000E+00
10	0.4499999E-01	0.5456207E-03	0.1484883E-02	1.000000	0.0000000E+00
11	0.4999999E-01	0.5751767E-03	0.1408664E-02	1.000000	0.0000000E+00
12	0.5499999E-01	0.6032949E-03	0.1343089E-02	1.000000	0.0000000E+00
13	0.5999999E-01	0.6301694E-03	0.1285888E-02	1.000000	0.0000000E+00
14	0.6500000E-01	0.6559514E-03	0.1235424E-02	1.000000	0.0000000E+00
15	0.6999999E-01	0.6807649E-03	0.1190469E-02	1.000000	0.0000000E+00
16	0.7499999E-01	0.7047101E-03	0.1150088E-02	1.000000	0.0000000E+00
17	0.7999998E-01	0.7278740E-03	0.1113549E-02	1.000000	0.0000000E+00
18	0.8499998E-01	0.7503314E-03	0.1080285E-02	1.000000	0.0000000E+00
19	0.8999997E-01	0.7721412E-03	0.1049830E-02	1.000000	0.0000000E+00
20	0.9099996E-01	0.7772278E-03	0.1042993E-02	1.000000	0.0000000E+00
21	0.9199995E-01	0.7821738E-03	0.1036781E-02	1.000000	0.0000000E+00
22	0.9299994E-01	0.7868169E-03	0.1031165E-02	1.000000	0.0000000E+00
23	0.9399992E-01	0.7913127E-03	0.1025667E-02	1.000000	0.0000000E+00
24	0.9499991E-01	0.7957316E-03	0.1020262E-02	1.000000	0.0000000E+00
25	0.9599990E-01	0.8000983E-03	0.1014941E-02	1.000000	0.0000000E+00
26	0.9699988E-01	0.8044259E-03	0.1009692E-02	1.000000	0.0000000E+00
27	0.9799987E-01	0.8087195E-03	0.1004524E-02	1.000000	0.0000000E+00
28	0.9899986E-01	0.8129824E-03	0.9994339E-03	1.000000	0.0000000E+00
29	0.9999985E-01	0.8172172E-03	0.9944150E-03	1.000000	0.0000000E+00
30	0.1009998	0.8214265E-03	0.9894725E-03	1.000000	0.0000000E+00
31	0.1019998	0.8256107E-03	0.9846005E-03	1.000000	0.0000000E+00
32	0.1029998	0.8297716E-03	0.9798002E-03	1.000000	0.0000000E+00
33	0.1039998	0.8339081E-03	0.9750684E-03	1.000000	0.0000000E+00
34	0.1049998	0.8380229E-03	0.9704088E-03	1.000000	0.0000000E+00
35	0.1059998	0.8421158E-03	0.9658064E-03	1.000000	0.0000000E+00
36	0.1069998	0.8461876E-03	0.9612725E-03	1.000000	0.0000000E+00
37	0.1079997	0.8502379E-03	0.9568008E-03	1.000000	0.0000000E+00
38	0.1089997	0.8542689E-03	0.9523893E-03	1.000000	0.0000000E+00
39	0.1099997	0.8582799E-03	0.9480407E-03	1.000000	0.0000000E+00
40	0.1109997	0.8622720E-03	0.9437504E-03	1.000000	0.0000000E+00

Partial listing of sim_bl/fort.4_case3 for wws>0 (blowing)

i	x	del99	cfxi	ue	vey
2	0.4999999E-02	0.1817591E-03	0.4455265E-02	1.000000	0.0000000E+00
3	0.9999998E-02	0.2570604E-03	0.3150324E-02	1.000000	0.0000000E+00
4	0.1500000E-01	0.3148592E-03	0.2572174E-02	1.000000	0.0000000E+00
5	0.2000000E-01	0.3636046E-03	0.2227505E-02	1.000000	0.0000000E+00
6	0.2499999E-01	0.4065582E-03	0.1992305E-02	1.000000	0.0000000E+00
7	0.2999999E-01	0.4453978E-03	0.1818688E-02	1.000000	0.0000000E+00
8	0.3499999E-01	0.4811205E-03	0.1683752E-02	1.000000	0.0000000E+00
9	0.3999999E-01	0.5143774E-03	0.1574979E-02	1.000000	0.0000000E+00
10	0.4499999E-01	0.5456207E-03	0.1484883E-02	1.000000	0.0000000E+00
11	0.4999999E-01	0.5751767E-03	0.1408664E-02	1.000000	0.0000000E+00
12	0.5499999E-01	0.6032949E-03	0.1343089E-02	1.000000	0.0000000E+00
13	0.5999999E-01	0.6301694E-03	0.1285888E-02	1.000000	0.0000000E+00
14	0.6500000E-01	0.6559514E-03	0.1235424E-02	1.000000	0.0000000E+00
15	0.6999999E-01	0.6807649E-03	0.1190469E-02	1.000000	0.0000000E+00
16	0.7499999E-01	0.7047101E-03	0.1150088E-02	1.000000	0.0000000E+00
17	0.7999998E-01	0.7278740E-03	0.1113549E-02	1.000000	0.0000000E+00
18	0.8499998E-01	0.7503314E-03	0.1080285E-02	1.000000	0.0000000E+00
19	0.8999997E-01	0.7721412E-03	0.1049830E-02	1.000000	0.0000000E+00
20	0.9099996E-01	0.7772278E-03	0.1042993E-02	1.000000	0.0000000E+00
21	0.9199995E-01	0.7821738E-03	0.1036781E-02	1.000000	0.0000000E+00
22	0.9299994E-01	0.7868169E-03	0.1031165E-02	1.000000	0.0000000E+00
23	0.9399992E-01	0.7913127E-03	0.1025667E-02	1.000000	0.0000000E+00
24	0.9499991E-01	0.7957316E-03	0.1020262E-02	1.000000	0.0000000E+00
25	0.9599990E-01	0.8352860E-03	0.6424396E-03	1.000000	0.0000000E+00
26	0.9699988E-01	0.8849534E-03	0.4485161E-03	1.000000	0.0000000E+00
27	0.9799987E-01	0.9372726E-03	0.3298856E-03	1.000000	0.0000000E+00
28	0.9899986E-01	0.9888948E-03	0.2515861E-03	1.000000	0.0000000E+00
29	0.9999985E-01	0.1042808E-02	0.1971731E-03	1.000000	0.0000000E+00
30	0.1009998	0.1095538E-02	0.1575695E-03	1.000000	0.0000000E+00
31	0.1019998	0.1147814E-02	0.1274729E-03	1.000000	0.0000000E+00
32	0.1029998	0.1202162E-02	0.1037076E-03	1.000000	0.0000000E+00
33	0.1039998	0.1255959E-02	0.8433116E-04	1.000000	0.0000000E+00
34	0.1049998	0.1313745E-02	0.6813808E-04	1.000000	0.0000000E+00
35	0.1059998	0.1374709E-02	0.5434935E-04	1.000000	0.0000000E+00
36	0.1069998	0.1439404E-02	0.4245481E-04	1.000000	0.0000000E+00
37	0.1079997	0.1512051E-02	0.3209893E-04	1.000000	0.0000000E+00
38	0.1089997	0.1596283E-02	0.2300904E-04	1.000000	0.0000000E+00
39	0.1099997	0.1698819E-02	0.1495353E-04	1.000000	0.0000000E+00
40	0.1109997	0.1840201E-02	0.7695347E-05	1.000000	0.0000000E+00
41	0.1119997	0.2083937E-02	0.9276840E-06	1.000000	0.0000000E+00

Partial listing of sim_bl/fort.4_case3 for wws<0 (suction)

i	x	del99	cfxi	ue	vey
2	0.4999999E-02	0.1817591E-03	0.4455265E-02	1.000000	0.0000000E+00
3	0.9999998E-02	0.2570604E-03	0.3150324E-02	1.000000	0.0000000E+00
4	0.1500000E-01	0.3148592E-03	0.2572174E-02	1.000000	0.0000000E+00
5	0.2000000E-01	0.3636046E-03	0.2227505E-02	1.000000	0.0000000E+00
6	0.2499999E-01	0.4065582E-03	0.1992305E-02	1.000000	0.0000000E+00
7	0.2999999E-01	0.4453978E-03	0.1818688E-02	1.000000	0.0000000E+00
8	0.3499999E-01	0.4811205E-03	0.1683752E-02	1.000000	0.0000000E+00
9	0.3999999E-01	0.5143774E-03	0.1574979E-02	1.000000	0.0000000E+00
10	0.4499999E-01	0.5456207E-03	0.1484883E-02	1.000000	0.0000000E+00
11	0.4999999E-01	0.5751767E-03	0.1408664E-02	1.000000	0.0000000E+00
12	0.5499999E-01	0.6032949E-03	0.1343089E-02	1.000000	0.0000000E+00
13	0.5999999E-01	0.6301694E-03	0.1285888E-02	1.000000	0.0000000E+00
14	0.6500000E-01	0.6559514E-03	0.1235424E-02	1.000000	0.0000000E+00
15	0.6999999E-01	0.6807649E-03	0.1190469E-02	1.000000	0.0000000E+00
16	0.7499999E-01	0.7047101E-03	0.1150088E-02	1.000000	0.0000000E+00
17	0.7999998E-01	0.7278740E-03	0.1113549E-02	1.000000	0.0000000E+00
18	0.8499998E-01	0.7503314E-03	0.1080285E-02	1.000000	0.0000000E+00
19	0.8999997E-01	0.7721412E-03	0.1049830E-02	1.000000	0.0000000E+00
20	0.9099996E-01	0.7772278E-03	0.1042993E-02	1.000000	0.0000000E+00
21	0.9199995E-01	0.7821738E-03	0.1036781E-02	1.000000	0.0000000E+00
22	0.9299994E-01	0.7868169E-03	0.1031165E-02	1.000000	0.0000000E+00
23	0.9399992E-01	0.7913127E-03	0.1025667E-02	1.000000	0.0000000E+00
24	0.9499991E-01	0.7957316E-03	0.1020262E-02	1.000000	0.0000000E+00
25	0.9599990E-01	0.7610961E-03	0.1692185E-02	1.000000	0.0000000E+00
26	0.9699988E-01	0.7425782E-03	0.1853121E-02	1.000000	0.0000000E+00
27	0.9799987E-01	0.7261650E-03	0.1997581E-02	1.000000	0.0000000E+00
28	0.9899986E-01	0.7113891E-03	0.2110409E-02	1.000000	0.0000000E+00
29	0.9999985E-01	0.6979862E-03	0.2202870E-02	1.000000	0.0000000E+00
30	0.1009998	0.6877561E-03	0.2280599E-02	1.000000	0.0000000E+00
31	0.1019998	0.6785083E-03	0.2347412E-02	1.000000	0.0000000E+00
32	0.1029998	0.6700533E-03	0.2405571E-02	1.000000	0.0000000E+00
33	0.1039998	0.6622185E-03	0.2456825E-02	1.000000	0.0000000E+00
34	0.1049998	0.6548313E-03	0.2502521E-02	1.000000	0.0000000E+00
35	0.1059998	0.6479230E-03	0.2543405E-02	1.000000	0.0000000E+00
36	0.1069998	0.6426822E-03	0.2580068E-02	1.000000	0.0000000E+00
37	0.1079997	0.6377993E-03	0.2613182E-02	1.000000	0.0000000E+00
38	0.1089997	0.6332221E-03	0.2643205E-02	1.000000	0.0000000E+00
39	0.1099997	0.6288576E-03	0.2670582E-02	1.000000	0.0000000E+00
40	0.1109997	0.6247559E-03	0.2695514E-02	1.000000	0.0000000E+00
41	0.1119997	0.6208408E-03	0.2718314E-02	1.000000	0.0000000E+00

Case 4: Program Notes

Program

File name is sym_bl/case4.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=118, [ny]=3, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile sym_bl/case4.f with f77 -o case4 case4.f ../lib/blk.o ../lib/lib.o

Run with : case4

Inputs

[pfs], [tfs], [amach], [reflen] in S/R inputs; [iunit]=0 (US units)

[iwall]=0; Adiabatic Wall

[nppi] and [ip] refer to i locations where profile output is required

[x(i),i=1,nx] distribution in S/R rinvis

[h1], [h2], [ue], [vey] are calculated in S/R rinvis for an ellipsoid at α=6° by analytical formula

Variations

[y] distribution and [vey] values depending on windward line or leeward line (see S/R rinvis).

Different x distributions and nxlim values

[nxlim] to be input < [nx] to avoid rear stagnation point

[x(1)] set to 0.0005 for windward symmetry solution and to -0.0005 for leeward symmetry solution to get a more accurate solution near the forward stagnation point.

Coefficients in S/R edgesym at i=2 calculated based on forward differencing of [ue], [h1], [h2] and [ak] to avoid differencing across the singularity point.

Program Description

This program calculates solution profiles at [nxlim] stations (at y=0 or y=1) of an ellipsoid at angle of attack. The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis. The non-dimensional free-stream velocity [ue(i,1),i=1,nx] and the crossflow velocity gradient [vey(i),i=1,nx] are calculated analytically. The metrics [h1(i,1)] and [h2(i,1)] are defined in S/R rinvis. Calculation of reference quantities are done in S/R ref in lib/blk.f.

The edge coefficients corresponding to the 3D stagnation point at i=1 are evaluated in S/R edgestg (see Table 3). The edge coefficients corresponding the symmetry flow downstream are evaluated in S/R edgesym, which are from

Table 4 with the edge cross flow velocity gradient [dvy(i)] as given in S/R rinvis. The x differencing flag [ix] is now set to 1 here which means that streamwise gradients will be calculated. The F, G, F′, G′, H, H′ profiles at i-1 and i-2 locations are stored at the end of iteration convergence in S/R nexstep.

The solution diverges as separation is approached. This occurs at i=111 for windward symmetry line and i=100 for leeward symmetry line.

The physical quantities of BL thickness [deltax], momentum thickness [thetax], displacement thickness [dispx], skin friction coefficient [cfxi] as well as profiles at selected stations are output in S/R phys.

Results

fort.2	has output of input data and reference conditions
fort.3	has profile output (S/R phys)
fort.4	wall and integrated quantities (S/R phys)
fort.1	output for plots

Windward sym. line flow Ellipsoid alpha=6, case4

Physical quantities are in U.S. units

Wall condition : Adiabatic wall

Inputs

```
nx, ny, nz = 118,   3,  41
zmax =    8.00
reflen =   2.000000
pfs =   2116.000
tfs=     520.00
mach no =      0.050
trs=       1.82
ufs =      55.88
ref Re no per unit length =  0.4765381E+09
fs  Re no per unit length =  0.3537900E+06
fs sonic vel =    1117.70
amuinf =  0.3745793E-06
amuref =  0.2780940E-09
rofs =  0.2371347E-02
hrefs =  0.3123122E+07
f-edge =   0.990000


nxlim=  111
iwall= 0
iunit= 0
ak=   1.070000
epsf=  0.5000000E-04
epsh=  0.5000000E-04
iord=  5
crs=   1.000000


stretching factor ak=    1.07000
zeta distribution

 0.0000000E+00  0.4007342E-01  0.8295202E-01  0.1288323
 0.1779239      0.2304517      0.2866566      0.3467962
 0.4111452      0.4799983      0.5536718      0.6325016
 0.7168503      0.8071027      0.9036732       1.007004
  1.117567       1.235870       1.362454       1.497898
  1.642824       1.797895       1.963820       2.141361
  2.331329       2.534595       2.752090       2.984808
  3.233817       3.500257       3.785347       4.090393
  4.416793       4.766041       5.139735       5.539589
  5.967432       6.425223       6.915060       7.439186
  8.000000

  i    x      h1      h2      ue    i    x      h1      h2      ue

  1 0.0000 5.1826 0.0000 0.0000   2 0.0010 4.0716 0.0248 0.0757
  3 0.0020 2.9607 0.0351 0.1803   4 0.0030 2.4822 0.0430 0.2554
  5 0.0040 2.2044 0.0496 0.3147   6 0.0050 2.0195 0.0554 0.3637
```

7	0.0060	1.8861	0.0607	0.4055	8	0.0070	1.7848	0.0655	0.4417
9	0.0080	1.7048	0.0700	0.4735	10	0.0090	1.6399	0.0742	0.5018
11	0.0100	1.5861	0.0781	0.5273	12	0.0110	1.5407	0.0819	0.5503
13	0.0120	1.5018	0.0855	0.5712	14	0.0130	1.4681	0.0890	0.5904
15	0.0140	1.4385	0.0923	0.6080	16	0.0150	1.4125	0.0955	0.6242
17	0.0160	1.3892	0.0985	0.6393	18	0.0170	1.3684	0.1015	0.6534
19	0.0180	1.3496	0.1044	0.6664	20	0.0190	1.3326	0.1072	0.6787
21	0.0200	1.3171	0.1100	0.6902	22	0.0300	1.2143	0.1340	0.7756
23	0.0400	1.1595	0.1539	0.8293	24	0.0500	1.1254	0.1712	0.8666
25	0.0600	1.1021	0.1865	0.8943	26	0.0700	1.0851	0.2004	0.9158
27	0.0800	1.0723	0.2131	0.9330	28	0.0900	1.0622	0.2248	0.9471
29	0.1000	1.0541	0.2356	0.9590	30	0.1100	1.0474	0.2457	0.9691
31	0.1200	1.0419	0.2552	0.9779	32	0.1300	1.0371	0.2641	0.9856
33	0.1400	1.0331	0.2725	0.9924	34	0.1500	1.0296	0.2804	0.9985
35	0.1600	1.0265	0.2879	1.0039	36	0.1700	1.0238	0.2950	1.0089
37	0.1800	1.0215	0.3017	1.0134	38	0.1900	1.0193	0.3081	1.0176
39	0.2000	1.0174	0.3142	1.0214	40	0.2100	1.0157	0.3199	1.0249
41	0.2200	1.0142	0.3253	1.0282	42	0.2300	1.0128	0.3305	1.0313
43	0.2400	1.0115	0.3354	1.0341	44	0.2500	1.0104	0.3401	1.0368
45	0.2600	1.0093	0.3445	1.0394	46	0.2700	1.0084	0.3487	1.0418
47	0.2800	1.0075	0.3526	1.0440	48	0.2900	1.0067	0.3564	1.0462
49	0.3000	1.0059	0.3599	1.0482	50	0.3100	1.0053	0.3632	1.0501
51	0.3200	1.0046	0.3664	1.0520	52	0.3300	1.0041	0.3693	1.0538
53	0.3400	1.0036	0.3721	1.0555	54	0.3500	1.0031	0.3746	1.0571
55	0.3600	1.0027	0.3770	1.0586	56	0.3700	1.0023	0.3792	1.0601
57	0.3800	1.0019	0.3812	1.0616	58	0.3900	1.0016	0.3831	1.0630
59	0.4000	1.0013	0.3848	1.0643	60	0.4100	1.0010	0.3863	1.0656
61	0.4200	1.0008	0.3876	1.0669	62	0.4300	1.0006	0.3888	1.0681
63	0.4400	1.0005	0.3899	1.0693	64	0.4500	1.0003	0.3907	1.0704
65	0.4600	1.0002	0.3914	1.0715	66	0.4700	1.0001	0.3920	1.0726
67	0.4800	1.0001	0.3924	1.0736	68	0.4900	1.0000	0.3926	1.0746
69	0.5000	1.0000	0.3927	1.0756	70	0.5100	1.0000	0.3926	1.0766
71	0.5200	1.0001	0.3924	1.0775	72	0.5300	1.0001	0.3920	1.0784
73	0.5400	1.0002	0.3914	1.0793	74	0.5500	1.0003	0.3907	1.0802
75	0.5600	1.0005	0.3899	1.0810	76	0.5700	1.0006	0.3888	1.0818
77	0.5800	1.0008	0.3876	1.0826	78	0.5900	1.0010	0.3863	1.0834
79	0.6000	1.0013	0.3848	1.0841	80	0.6100	1.0016	0.3831	1.0849
81	0.6200	1.0019	0.3812	1.0856	82	0.6300	1.0023	0.3792	1.0863
83	0.6400	1.0027	0.3770	1.0869	84	0.6500	1.0031	0.3746	1.0876
85	0.6600	1.0036	0.3720	1.0882	86	0.6700	1.0041	0.3693	1.0888
87	0.6800	1.0046	0.3664	1.0893	88	0.6900	1.0053	0.3632	1.0899
89	0.7000	1.0059	0.3599	1.0904	90	0.7100	1.0067	0.3564	1.0908
91	0.7200	1.0075	0.3526	1.0913	92	0.7300	1.0084	0.3487	1.0917
93	0.7400	1.0093	0.3445	1.0921	94	0.7500	1.0104	0.3401	1.0924
95	0.7600	1.0115	0.3354	1.0926	96	0.7700	1.0128	0.3305	1.0928
97	0.7800	1.0142	0.3253	1.0930	98	0.7900	1.0157	0.3199	1.0931
99	0.8000	1.0174	0.3142	1.0930	100	0.8100	1.0193	0.3081	1.0929
101	0.8200	1.0215	0.3017	1.0927	102	0.8300	1.0238	0.2950	1.0923
103	0.8400	1.0265	0.2879	1.0917	104	0.8500	1.0296	0.2804	1.0910
105	0.8600	1.0331	0.2725	1.0900	106	0.8700	1.0371	0.2641	1.0887
107	0.8800	1.0419	0.2552	1.0870	108	0.8900	1.0474	0.2457	1.0848
109	0.9000	1.0541	0.2356	1.0819	110	0.9100	1.0622	0.2248	1.0782
111	0.9200	1.0723	0.2131	1.0733					

Case 5: Program Notes

Program

File name is sym_bl/case5.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=99, [ny]=3, [nz]=81

Recompile lib/blk.f with f77 -c blk.f

Compile sym_bl/case5.f with f77 -o case5 case5.f ../lib/blk.o ../lib/lib.o

Run with : case5

Inputs

[pfs], [tfs], [amach], [reflen] in S/R inputs; [iunit]=0 (US units)

[iwall]=0; Adiabatic Wall; [reflen] depends on the Reynolds number

[nppi] and [ip] refer to i locations where profile output is required

[xb], [zb] (cartesian coordinates of airfoil section) are read in S/R rinvis from Navier-Stokes run data; also read in are the pressure coefficient values and skin friction results from N-S run.

The x distribution is calculated from [xb] and [zb] as arc length

[ue] and [ve] calculated from pressure coefficient values; [h1] and [h2] are unity.

Conditions at i=1 (attachment point) are obtained by quadratic extrapolation.

Variations

nxlim=44 for refs=5,000 run (input file is sym_bl/case5_dat1)

nxlim=98 for refs=1,000,000 run (input file is sym_bl/case5_dat2)

Program Description

This program calculates solution profiles at [nxlim] stations for flow past an airfoil in subsonic flow. The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis. Calculation of reference quantities are done in S/R ref in lib/blk.f. The edge conditions are as obtained from a Navier-Stokes calculation.

The edge coefficients corresponding to the 2D stagnation point at i=1 are evaluated in S/R edgestg (see Table 3). The edge coefficients corresponding the 2D flow downstream are evaluated in S/R edgesym, which are from Table 4 with the edge cross flow velocity gradient [dvy(i)] and cross-flow gradients set to zero.

The solution diverges as separation is approached. This occurs at i=98 for for the higher Re number case and i=44 for the lower Re number case. Note that the x distribution in the two cases are not the same. Laminar separation for the lower Re number case occurs farther from the leading edge.

Results

The physical quantities of BL thickness [deltax], momentum thickness [thetax], displacement thickness [dispx], skin friction coefficient [cfxi] as well as profiles at selected stations are output in S/R phys.

fort.2 has output of input data and reference conditions

fort.3 has profile output (S/R phys)

fort.4 wall and integrated quantities (S/R phys)

file sym_bl/fort.2_case5: Partial listing

```
 Flow past Airfoil Section, case5

 Physical quantities are in U.S. units

 Wall condition : Adiabatic wall

 Inputs

 nx, ny, nz =  99,   3,  81
 zmax =   10.00
 reflen =  0.2826600
 pfs =   2116.000
 tfs=     520.00
 mach no =       0.500
 trs=     182.00
 ufs =     558.85
 ref Re no per unit length =  0.9049514e+07
 fs  Re no per unit length =  0.3537902e+07
 fs sonic vel =    1117.70
 amuinf =  0.3745792e-06
 amuref =  0.1464415e-06
 rofs =  0.2371347e-02
 hrefs =  0.3123120e+07
 f-edge =   0.990000


 nxlim=   68
 iwall= 0
 iunit= 0
 ak=   1.050000
 epsf=  0.5000000e-04
 epsh=  0.5000000e-04
 iord=  5
 crs=         0.


 stretching factor ak=     1.05000
 zeta distribution

        0.       0.1029627e-01  0.2110735e-01  0.3245900e-01
 0.4437821e-01  0.5689340e-01  0.7003433e-01  0.8383231e-01
 0.9832020e-01  0.1135325       0.1295053       0.1462769
 0.1638870       0.1823776       0.2017927       0.2221786
 0.2435838       0.2660593       0.2896585       0.3144377
 0.3404558       0.3677748       0.3964598       0.4265791
 0.4582043       0.4914108       0.5262775       0.5628877
 0.6013282       0.6416910       0.6840717       0.7285714
 0.7752963       0.8243573       0.8758713       0.9299613
 0.9867554        1.046389        1.109005        1.174752
  1.243785        1.316271        1.392381        1.472296
  1.556207        1.644313        1.736825        1.833962
  1.935957        2.043051        2.155499        2.273571
  2.397545        2.527719        2.664401        2.807917
```

2.958609	3.116836	3.282973	3.457418
3.640585	3.832910	4.034852	4.246890
4.469531	4.703303	4.948764	5.206499
5.477119	5.761271	6.059630	6.372909
6.701850	7.047238	7.409895	7.790687
8.190516	8.610338	9.051150	9.514004
10.00000			

i	x	h1	h2	ue	i	x	h1	h2	ue
1	0.0000	1.0000	1.0000	0.0000	2	0.0005	1.0000	1.0000	0.0361
3	0.0015	1.0000	1.0000	0.1095	4	0.0025	1.0000	1.0000	0.1852
5	0.0036	1.0000	1.0000	0.2605	6	0.0046	1.0000	1.0000	0.3347
7	0.0058	1.0000	1.0000	0.4076	8	0.0070	1.0000	1.0000	0.4789
9	0.0082	1.0000	1.0000	0.5480	10	0.0096	1.0000	1.0000	0.6143
11	0.0110	1.0000	1.0000	0.6769	12	0.0125	1.0000	1.0000	0.7352
13	0.0141	1.0000	1.0000	0.7889	14	0.0158	1.0000	1.0000	0.8380
15	0.0177	1.0000	1.0000	0.8826	16	0.0196	1.0000	1.0000	0.9219
17	0.0218	1.0000	1.0000	0.9591	18	0.0241	1.0000	1.0000	0.9974
19	0.0265	1.0000	1.0000	1.0285	20	0.0291	1.0000	1.0000	1.0503
21	0.0319	1.0000	1.0000	1.0692	22	0.0349	1.0000	1.0000	1.0911
23	0.0381	1.0000	1.0000	1.1116	24	0.0416	1.0000	1.0000	1.1274
25	0.0452	1.0000	1.0000	1.1409	26	0.0491	1.0000	1.0000	1.1534
27	0.0533	1.0000	1.0000	1.1646	28	0.0578	1.0000	1.0000	1.1744
29	0.0625	1.0000	1.0000	1.1830	30	0.0676	1.0000	1.0000	1.1904
31	0.0730	1.0000	1.0000	1.1969	32	0.0787	1.0000	1.0000	1.2025
33	0.0848	1.0000	1.0000	1.2072	34	0.0913	1.0000	1.0000	1.2112
35	0.0982	1.0000	1.0000	1.2145	36	0.1055	1.0000	1.0000	1.2172
37	0.1133	1.0000	1.0000	1.2192	38	0.1216	1.0000	1.0000	1.2207
39	0.1303	1.0000	1.0000	1.2216	40	0.1396	1.0000	1.0000	1.2220
41	0.1494	1.0000	1.0000	1.2219	42	0.1599	1.0000	1.0000	1.2212
43	0.1709	1.0000	1.0000	1.2201	44	0.1826	1.0000	1.0000	1.2185
45	0.1949	1.0000	1.0000	1.2165	46	0.2078	1.0000	1.0000	1.2139
47	0.2210	1.0000	1.0000	1.2111	48	0.2343	1.0000	1.0000	1.2079
49	0.2478	1.0000	1.0000	1.2046	50	0.2613	1.0000	1.0000	1.2010
51	0.2748	1.0000	1.0000	1.1973	52	0.2884	1.0000	1.0000	1.1934
53	0.3021	1.0000	1.0000	1.1894	54	0.3157	1.0000	1.0000	1.1853
55	0.3294	1.0000	1.0000	1.1812	56	0.3430	1.0000	1.0000	1.1770
57	0.3566	1.0000	1.0000	1.1728	58	0.3702	1.0000	1.0000	1.1686
59	0.3838	1.0000	1.0000	1.1645	60	0.3972	1.0000	1.0000	1.1605
61	0.4107	1.0000	1.0000	1.1568	62	0.4240	1.0000	1.0000	1.1534
63	0.4373	1.0000	1.0000	1.1502	64	0.4504	1.0000	1.0000	1.1470
65	0.4635	1.0000	1.0000	1.1433	66	0.4765	1.0000	1.0000	1.1386
67	0.4894	1.0000	1.0000	1.1325	68	0.5021	1.0000	1.0000	1.1254

file sym_bl/fort.4_case5: partial listing

i	x	del99	cfxi	ue	cfxi from N.S.
2	0.4894090e-03	0.7517626e-04	0.8269813e-03	0.3607887e-01	0.8935480e-03
3	0.1480723e-02	0.7506231e-04	0.2515605e-02	0.1095190	0.2721860e-02
4	0.2500762e-02	0.7507298e-04	0.4250144e-02	0.1852039	0.4490880e-02
5	0.3554800e-02	0.7615856e-04	0.5828022e-02	0.2605470	0.6154500e-02
6	0.4648551e-02	0.7806430e-04	0.7256357e-02	0.3347190	0.7676750e-02
7	0.5788399e-02	0.8025547e-04	0.8549325e-02	0.4076206	0.9026940e-02
8	0.6981573e-02	0.8270711e-04	0.9671693e-02	0.4789198	0.1017250e-01
9	0.8236348e-02	0.8550503e-04	0.1058301e-01	0.5480332	0.1108290e-01
10	0.9562106e-02	0.8913212e-04	0.1125609e-01	0.6142756	0.1173590e-01
11	0.1096939e-01	0.9319758e-04	0.1167327e-01	0.6769152	0.1212800e-01
12	0.1246953e-01	0.9792998e-04	0.1183277e-01	0.7352426	0.1228100e-01
13	0.1407474e-01	0.1035276e-03	0.1177318e-01	0.7888664	0.1223760e-01
14	0.1579718e-01	0.1094933e-03	0.1157784e-01	0.8379766	0.1205110e-01
15	0.1764850e-01	0.1163983e-03	0.1125349e-01	0.8825843	0.1177040e-01
16	0.1963882e-01	0.1238353e-03	0.1073445e-01	0.9219008	0.1145810e-01
17	0.2177455e-01	0.1314976e-03	0.1049275e-01	0.9590654	0.1110710e-01
18	0.2405917e-01	0.1380325e-03	0.1059458e-01	0.9974195	0.1058580e-01
19	0.2650098e-01	0.1469381e-03	0.9729961e-02	1.028514	0.9927350e-02
20	0.2911451e-01	0.1579923e-03	0.8555860e-02	1.050275	0.9325380e-02
21	0.3191486e-01	0.1691054e-03	0.8012371e-02	1.069181	0.8868090e-02
22	0.3491292e-01	0.1778499e-03	0.8122587e-02	1.091137	0.8444820e-02
23	0.3811909e-01	0.1872390e-03	0.7745605e-02	1.111609	0.7975450e-02
24	0.4154657e-01	0.1991921e-03	0.7050918e-02	1.127435	0.7503290e-02
25	0.4520965e-01	0.2109887e-03	0.6585455e-02	1.140930	0.7067520e-02
26	0.4912219e-01	0.2227516e-03	0.6248122e-02	1.153384	0.6665560e-02
27	0.5329859e-01	0.2352820e-03	0.5898427e-02	1.164590	0.6287120e-02
28	0.5775428e-01	0.2479205e-03	0.5548791e-02	1.174438	0.5928200e-02
29	0.6250524e-01	0.2606935e-03	0.5215179e-02	1.183000	0.5590100e-02
30	0.6756813e-01	0.2747556e-03	0.4905238e-02	1.190438	0.5273200e-02
31	0.7296033e-01	0.2889860e-03	0.4617522e-02	1.196893	0.4976360e-02
32	0.7870024e-01	0.3032964e-03	0.4349194e-02	1.202465	0.4697720e-02
33	0.8480661e-01	0.3178189e-03	0.4097165e-02	1.207228	0.4436570e-02
34	0.9129949e-01	0.3337699e-03	0.3859424e-02	1.211236	0.4191860e-02
35	0.9819981e-01	0.3500410e-03	0.3635525e-02	1.214544	0.3960880e-02
36	0.1055291	0.3665455e-03	0.3424305e-02	1.217198	0.3742760e-02
37	0.1133100	0.3833257e-03	0.3224746e-02	1.219240	0.3537620e-02
38	0.1215670	0.4006959e-03	0.3035678e-02	1.220701	0.3342570e-02
39	0.1303241	0.4195799e-03	0.2856328e-02	1.221607	0.3156840e-02
40	0.1396073	0.4388254e-03	0.2685797e-02	1.221986	0.2980910e-02
41	0.1494446	0.4584203e-03	0.2523801e-02	1.221858	0.2813040e-02

0.00600	0.00700	0.00800	0.00900	0.01000	0.01100
0.01200	0.01300	0.01400	0.01500	0.01600	0.01700
0.01800	0.01900	0.02000	0.02100	0.02300	0.02600
0.03000	0.03400	0.03800	0.04200	0.04600	0.05000
0.05400	0.05800	0.06200	0.06600	0.07000	0.07400
0.07800	0.08200	0.08600	0.09000	0.09400	0.09800
0.10200	0.10600	0.11000	0.11400	0.11800	0.12200
0.12600	0.13000	0.13500	0.14100	0.14800	0.15600
0.16500	0.17500	0.18500	0.19500	0.20500	0.21500
0.22500	0.23500	0.24500	0.25500	0.26500	0.27500
0.28500	0.29500	0.30500	0.31500	0.32500	0.33500
0.34500	0.35500	0.36500	0.37500	0.38500	0.39500
0.40500					

y stations

0.18750	0.26250	0.33750	0.41250	0.48750	0.56250
0.63750	0.71250	0.78750	0.86250	0.93750	1.01250
1.08750	1.16250	1.23750			

Case 6: Program Notes

Program

File name is inf_bl/case6.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=101, [ny]=1, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile inf_bl/case6.f with f77 -o case6 case6.f ../lib/blk.o ../lib/lib.o

Run with : case6

Inputs

The inviscid flow on the swept ellipse is calculated by the program ellipse.f. This program generates the cross section and edge velocity ($u_{e,2D}$) for a swept ellipse at an angle of attack. The inputs to this program are the ellipse fineness ratio, angle of attack, sweep angle and the grid distribution. This program is run with,

f77 -o ellipse ellipse.f ../lib/lib.o ; ellipse

The inviscid edge conditions are output to fort.20_case6

The inputs to the BL program are: [amach], [pfs], [tfs], [reflen] and the sweep angle; [iunit] = [iwall] = 0

Program Description

This program calculates solution profiles at [nxlim] stations for flow past a swept ellipse in subsonic flow.

The inputs to the program and boundary conditions are provided in S/R inputs. BL edge conditions are specified in S/R rinvis (read in from file fort.20_case6).

The edge coefficients corresponding to an infinite swept attachment line at i=1 are evaluated in S/R edgeatt. The edge coefficients corresponding the infinite swept wing flow downstream are evaluated in S/R edgeinf.

Results

The physical quantities of BL thickness [deltax], momentum thickness [thetax], displacement thickness [dispx], skin friction coefficient [cfxi] as well as profiles at selected stations are output in S/R phys.

Swept Elliptical Cylinder Flow

Physical quantities are in U.S. units

Wall condition : Adiabatic wall

Inputs

```
nx, ny, nz = 101,   1,  41
zmax =     8.00
reflen =   1.000000
pfs =    2116.000
tfs=      520.00
mach no =       0.100
trs=        7.28
ufs =      111.77
ref Re no per unit length =  0.1223800e+09
fs  Re no per unit length =  0.7075803e+06
fs sonic vel =    1117.70
amuinf =  0.3745792e-06
amuref =  0.2165752e-08
rofs =  0.2371347e-02
hrefs =  0.3123120e+07
f-edge =   0.990000


nxlim=   79
iwall= 0
iunit= 0
ak=   1.020000
epsf=  0.5000000e-04
epsh=  0.5000000e-04


stretching factor ak=    1.02000
zeta distribution

          0.       0.1324464       0.2675414       0.4053386
   0.5458910       0.6892554       0.8354862       0.9846427
    1.136781        1.291963        1.450248        1.611700
    1.776379        1.944353        2.115686        2.290447
    2.468700        2.650521        2.835977        3.025143
    3.218091        3.414900        3.615643        3.820403
    4.029256        4.242288        4.459579        4.681217
    4.907286        5.137878        5.373082        5.612990
    5.857694        6.107295        6.361886        6.621570
    6.886445        7.156621        7.432198        7.713290
    8.000000
```

i	x	ue	ve	i	x	ue	ve	i	x	ue	ve
1	0.0000	0.0000	0.6428	2	0.0007	0.0667	0.6428	3	0.0013	0.1374	0.6428
4	0.0019	0.2120	0.6428	5	0.0025	0.2901	0.6428	6	0.0031	0.3710	0.6428
7	0.0037	0.4539	0.6428	8	0.0042	0.5379	0.6428	9	0.0048	0.6219	0.6428
10	0.0053	0.7044	0.6428	11	0.0059	0.7844	0.6428	12	0.0064	0.8606	0.6428
13	0.0070	0.9319	0.6428	14	0.0075	0.9975	0.6428	15	0.0081	1.0570	0.6428

i	x	ue	ve	i	x	ue	ve	i	x	ue	ve
16	0.0086	1.1101	0.6428	17	0.0092	1.1567	0.6428	18	0.0098	1.1971	0.6428
19	0.0104	1.2317	0.6428	20	0.0111	1.2609	0.6428	21	0.0117	1.2853	0.6428
22	0.0124	1.3053	0.6428	23	0.0131	1.3216	0.6428	24	0.0138	1.3347	0.6428
25	0.0146	1.3449	0.6428	26	0.0153	1.3528	0.6428	27	0.0161	1.3587	0.6428
28	0.0170	1.3629	0.6428	29	0.0178	1.3657	0.6428	30	0.0187	1.3673	0.6428
31	0.0196	1.3679	0.6428	32	0.0206	1.3677	0.6428	33	0.0215	1.3669	0.6428
34	0.0225	1.3655	0.6428	35	0.0236	1.3637	0.6428	36	0.0246	1.3615	0.6428
37	0.0257	1.3590	0.6428	38	0.0269	1.3563	0.6428	39	0.0280	1.3534	0.6428
40	0.0292	1.3504	0.6428	41	0.0305	1.3473	0.6428	42	0.0317	1.3441	0.6428
43	0.0330	1.3408	0.6428	44	0.0344	1.3375	0.6428	45	0.0357	1.3342	0.6428
46	0.0371	1.3309	0.6428	47	0.0385	1.3276	0.6428	48	0.0400	1.3244	0.6428
49	0.0415	1.3211	0.6428	50	0.0430	1.3179	0.6428	51	0.0446	1.3147	0.6428
52	0.0462	1.3116	0.6428	53	0.0478	1.3085	0.6428	54	0.0495	1.3055	0.6428
55	0.0512	1.3025	0.6428	56	0.0529	1.2995	0.6428	57	0.0547	1.2967	0.6428
58	0.0564	1.2938	0.6428	59	0.0583	1.2910	0.6428	60	0.0601	1.2883	0.6428
61	0.0620	1.2856	0.6428	62	0.0640	1.2830	0.6428	63	0.0659	1.2804	0.6428
64	0.0679	1.2778	0.6428	65	0.0699	1.2753	0.6428	66	0.0720	1.2729	0.6428
67	0.0741	1.2705	0.6428	68	0.0762	1.2681	0.6428	69	0.0783	1.2658	0.6428
70	0.0805	1.2636	0.6428	71	0.0827	1.2613	0.6428	72	0.0850	1.2591	0.6428
73	0.0873	1.2570	0.6428	74	0.0896	1.2549	0.6428	75	0.0919	1.2528	0.6428
76	0.0943	1.2508	0.6428	77	0.0967	1.2488	0.6428	78	0.0991	1.2468	0.6428

Case 7: Program Notes

Program

File name is att_bl/case7.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=1, [ny]=20, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile att_bl/case7.f with f77 -o case7 case7.f ../lib/lib.o ../lib/blk.o

Run with : case7

To generate the inviscid edge conditions, compile att_bl/cyl.f with

f77 -o cyl cyl.f ../lib/lib.o

and run with : cyl

The input data for cyl.f is in cp.dat (Euler results); the output from cyl.f is fort.20_case7

Inputs

The inviscid inputs are in file cp.dat. They comprise of the cartesian coordinates of the inviscid grid (x',y',z'), the cartesian components of velocity (u',v',w') and the pressure coefficient, C_p.

The inputs to the BL program are the edge velocity along the attachment line and the streamwise velocity gradient. They are calculated by the program cyl.f; Other inputs [pfs], [tfs], [amach] and [reflen] are in S/R inputs; metric coefficients are input in S/R rinvis.

Program Description

The program cyl.f reads in the inviscid grid and cartesian components of velocity. The velocities are transformed to curvilinear coordinates. The first five stations near to the symmetry plane are skipped to avoid the subsonic region in the inviscid calculation. The velocity gradient in the streamwise direction is then calculated. Results are output to a file called fort.20_case7.

The program case7.f obtains the BL solution on the swept cylinder attachment line. The edge conditions from fort.20_case7 are read in. The first station on the attachment line is calculated by solving the infinite attachment line equations. The edge conditions for this case are generated in S/R edgeinf. The solution along the attachment line is then calculated by marching in the j direction. The edge conditions for the general attachment line equations are generated in S/R edgeatt. A 2-point backward differencing scheme is used to evaluate the y gradients.

```
Att line flow swept cyl at amach=3.5, case7

Physical quantities are in U.S. units

Wall condition : Adiabatic wall

Inputs

nx, ny, nz =   1,  20,  41
zmax =    6.00
reflen =  0.8333334e-01
pfs =   188.7800
tfs=     172.75
mach no =      3.500
trs=    2962.66
ufs =    2254.76
ref Re no per unit length =  0.1240029e+07
fs  Re no per unit length =  0.1034549e+08
fs sonic vel =     644.22
amuinf =  0.1387936e-06
amuref =  0.1157947e-05
rofs =  0.6368258e-03
hrefs =  0.1037537e+07
f-edge =   0.990000


nylim=   20
iwall= 0
iunit= 0
ak=   1.030000
epsf=  0.5000000e-04
epsh=  0.5000000e-04
iord=  5
crs=          0.
gedge=  0.9900000


stretching factor ak=    1.03000
zeta distribution

        0.      0.7957422e-01  0.1615357      0.2459560
 0.3329091      0.4224705      0.5147189      0.6097348
 0.7076012      0.8084035      0.9122299       1.019171
  1.129320       1.242774       1.359632       1.479995
  1.603969       1.731662       1.863187       1.998657
  2.138190       2.281910       2.429942       2.582415
  2.739461       2.901219       3.067830       3.239439
  3.416197       3.598257       3.785779       3.978926
  4.177869       4.382779       4.593837       4.811226
  5.035137       5.265765       5.503313       5.747986
  6.000000
 j       y(j)             ve              fe

  1  0.2916670e-01  0.8124530       1.038030
  2  0.3750000e-01  0.8226570      0.9848440
  3  0.4583330e-01  0.8300120      0.9534300
```

4	0.5416670e-01	0.8350630	0.9341240
5	0.6250000e-01	0.8385820	0.9148030
6	0.7500000e-01	0.8415280	0.9011600
7	0.9166670e-01	0.8446180	0.8867070
8	0.1125000	0.8480790	0.8789700
9	0.1375000	0.8515590	0.8703080
10	0.1625000	0.8547280	0.8593810
11	0.1875000	0.8573130	0.8510940
12	0.2125000	0.8592110	0.8453430
13	0.2375000	0.8604830	0.8419720
14	0.2625000	0.8613040	0.8389780
15	0.2875000	0.8618300	0.8358840
16	0.3125000	0.8621980	0.8327700
17	0.3375000	0.8625010	0.8301450
18	0.3625000	0.8627780	0.8258100
19	0.3875000	0.8630380	0.8239340
20	0.4125000	0.8632890	0.8202360

Case 8: Program Notes

Program

File name is 3d_bl/case8.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=29, [ny]=11, [nz]=41

Recompile lib/blk.f with f77 -c blk.f

Compile 3d_bl/case8.f with f77 -o case8 case8.f ../lib/lib.o ../lib/blk.o

Run with : case8

Inputs

The inviscid inputs are generated in the program itself in S/R rinvis. The initial plane of solution and the side boundary plane of solution are read into the program in S/R initial and S/R rside respectively.

The initial profile at i=1, j=1 is generated by running the program sim_bl/case8.f corresponding to similarity profiles at the flat plate singularity point. This program is run by,

f77 -o case8 case8.f ../lib/lib.o ../lib/blk.o ; case8

The solution profile is output to file sim_bl/fort.11_case8. The initial profiles at other j stations on i=1 are obtained by copying the same profile.

The side boundary solution profiles (at j=1) are generated by running the program sym_bl/case8.f. This program is run by,

f77 -o case8 case8.f ../lib/lib.o ../lib/blk.o ; case8

Input for this run is the file sim_bl/fort.11_case8. The output is to the file sym_bl/fort.11_case8 and this corresponds to [nxlim-1] number of solution profiles on the symmetry line.

Other inputs corresponding to free-stream conditions are as given in S/R inputs

Program Description

This program solves the 3D BL equations for flow past a cylinder on a flat plate. The initial profile is read from the similarity solution and the side boundary profiles from the symmetry line solution. The η direction sweep can be done using a 'L' scheme or a 'Z' scheme as specified in S/R edge3d. The last point at j=jm always uses a 'L' scheme.

Partial listing of file 3d_bl/fort.2_case8

3d flow, cyl on fp test case #8

Physical quantities are in S.I. units

Wall condition : Adiabatic wall

Inputs

```
nx, ny, nz =  29,  11,  41
zmax =    8.00
reflen =  0.6100000e-01
pfs =   101325.5
tfs=     288.15
mach no =      0.090
trs=       3.24
ufs =      30.50
ref Re no per unit length =  0.4987959e+09
fs  Re no per unit length =  0.2087405e+07
fs sonic vel =     340.30
amuinf =  0.1789923e-04
amuref =  0.7490627e-07
rofs =  0.1224979e+01
hrefs =  0.2895064e+06
f-edge =   0.990000


nxlim=   29
nylim=   11
iwall= 0
iunit= 1
ak=   1.020000
epsf=  0.1000000e-04
epsh=  0.1000000e-04
iord=  5


stretching factor ak=    1.02000
zeta distribution

        0.      0.1324464      0.2675414      0.4053386
 0.5458910      0.6892554      0.8354862      0.9846427
  1.136781       1.291963       1.450248       1.611700
  1.776379       1.944353       2.115686       2.290447
  2.468700       2.650521       2.835977       3.025143
  3.218091       3.414900       3.615643       3.820403
  4.029256       4.242288       4.459579       4.681217
  4.907286       5.137878       5.373082       5.612990
  5.857694       6.107295       6.361886       6.621570
  6.886445       7.156621       7.432198       7.713290
  8.000000

x stations
```

```
   0.00000   0.10000   0.20000   0.30000   0.40000   0.50000
   0.60000   0.70000   0.80000   0.90000   1.00000   1.10000
   1.20000   1.30000   1.40000   1.50000   1.60000   1.70000
   1.80000   1.90000   2.00000   2.10000   2.20000   2.30000
   2.40000   2.50000   2.60000   2.70000   2.80000

y stations

   0.00000   0.10000   0.20000   0.30000   0.40000   0.50000
   0.60000   0.70000   0.80000   0.90000   1.00000
```

Case 9: Program Notes

Program

File name is 3d_bl/case9.f

Library routines are in lib/blk.f and lib/lib.f

To run Program

Edit lib/com and set [nx]=77, [ny]=38, [nz]=41

Edit lib/comint and set [ii]=53, [ji]=36

Recompile lib/blk.f with f77 -c blk.f

The interface program int_fus/case9.f is run with,

f77 -o case9 case9.f ../lib/lib.o ../lib/pack.o

The 3D BL solution is obtained with,

3d_bl/case9.f with f77 -o case9 case9.f ../lib/lib.o ../lib/blk.o ; case9

Program Description

The inviscid inputs are obtained by running the Hess code. The Hess code output data is in file int_fus/cp.dat and consists of the Cartesian coordinates, the Cartesian components of velocity and C_p corresponding to the panel centroids.

The interface program int_fus/case9.f reads the Hess output in S/R rinvis. The boundary layer grid is generated analytically with a circumferential distribution same as that of the inviscid grid (S/R blgrd). Two more points corresponding to the windward and leeward lines of symmetry are added resulting in, [ny] = [ji]+2 = 38.

The C_p values are interpolated to the BL grid in the interface program using cubic splines (lib/pack.o contains the spline routine object code). Subsequently, the surface Euler equations are solved to generate the BL edge velocities (S/R interp). The edge conditions are output to file int_fus/fort.30_case9.

The symmetry line BL solution program sym_bl/case9.f is run next similar to Case 4. The inviscid edge conditions required here can be exactly generated or read in numerical form from the interface program. The BL solution profiles are output to sym_bl/fort.11_case9.

The 3D BL solution program reads the edge conditions (exact or numerical) and the symmetry plane solution. The η direction sweep can be of the 'L' type or the 'Z' type as set in S/R edge3d. The η sweep is restricted to ny=37 i.e. one point short of the leeward symmetry line. The j=37 point is always solved by an 'L' scheme.

Solution profiles and BL parameters (t_x^*, $C_{f,x}$, $C_{f,y}$) at selected (i,j) stations are output in S/R phys.

Partial listing of file 3d_bl/fort.2_case9

3d flow, ellipsoid alpha=3 deg Case #9

Physical quantities are in U.S. units

Wall condition : Adiabatic wall

Inputs

```
nx, ny, nz =  77,  38,  41
zmax =     8.00
reflen =   2.000000
pfs =   2116.000
tfs=     520.00
mach no =       0.050
trs=        1.82
ufs =       55.88
ref Re no per unit length =  0.4765381E+09
fs  Re no per unit length =  0.3537900E+06
fs sonic vel =    1117.70
amuinf =  0.3745793E-06
amuref =  0.2780940E-09
rofs =  0.2371347E-02
hrefs =  0.3123122E+07
f-edge =   0.990000


nxlim=   76
nylim=   37
iwall= 0
iunit= 0
ak=   1.000000
epsf=  0.1000000E-04
epsh=  0.1000000E-04
iord=  5


stretching factor ak=     1.00000
zeta distribution

 0.0000000E+00  0.2000000       0.4000000       0.6000000       0.8000000
 0.9999999       1.200000        1.400000        1.600000        1.800000
  2.000000       2.200000        2.400000        2.599999        2.800000
  3.000000       3.200000        3.400000        3.599999        3.799999
  3.999999       4.199999        4.400000        4.599999        4.799999
  4.999999       5.199999        5.400000        5.599999        5.799999
  5.999999       6.199999        6.400000        6.599999        6.799999
  6.999999       7.199999        7.399999        7.599999        7.799999
  7.999999

x stations

   0.00013     0.00100     0.00110     0.00120     0.00130     0.00140
```

0.00150	0.00160	0.00170	0.00180	0.00190	0.00200
0.00210	0.00220	0.00230	0.00240	0.00250	0.00260
0.00270	0.00280	0.00290	0.00300	0.00400	0.00500
0.00600	0.00700	0.00800	0.00900	0.01000	0.01150
0.01350	0.01600	0.01900	0.02250	0.02650	0.03100
0.03600	0.04100	0.04600	0.05100	0.05600	0.06100
0.06600	0.07100	0.07600	0.08100	0.08600	0.09100
0.09600	0.10100	0.10600	0.11100	0.11600	0.12100
0.12600	0.13100	0.13600	0.14100	0.14600	0.15100
0.15600	0.16100	0.16600	0.17100	0.17600	0.18100
0.18600	0.19100	0.19600	0.20100	0.20600	0.21100
0.21600	0.22100	0.22600	0.23100		

y stations

0.00000	0.01389	0.04167	0.06944	0.09722	0.12500
0.15278	0.18056	0.20833	0.23611	0.26389	0.29167
0.31944	0.34722	0.37500	0.40278	0.43056	0.45833
0.48611	0.51389	0.54167	0.56944	0.59722	0.62500
0.65278	0.68056	0.70833	0.73611	0.76389	0.79167
0.81944	0.84722	0.87500	0.90278	0.93056	0.95833

Case 10: Program Notes

Program

The interface program:

The input file from the Hess code is int_wng/cp.dat

The corrected inviscid file is int_wng/cp_cor.dat

The corrections are necessary to eliminate the oscillations in the inviscid results and is done in the program int_wng/cor.f

The interface program is int_wng/case10.f

The dimension are set as ii=70, ji=15 in lib/comint and nx=100, ny=15 in lib/com

This program is run using f77 -o case10 case10.f ../lib/lib.o ../lib/pack.o; case10

lib/pack.f contains the spline interpolation routines

These routines are compiled with f77 -c pack.f

The edge conditions data are output to int_wng/fort.20_case10

The BL analysis programs

The attachment line program is run from sub-directory att_bl with:

f77 -o case10 case10.f ../lib/lib.o ../lib/blk.o ; case10

The locally infinite swept wing program is run from sub-directory inf_bl with:

f77 -o case10 case10.f ../lib/lib.o ../lib/blk.o ; case10

The 3d BL wing program is run from sub-directory 3d_bl with:

f77 -o case10 case10.f ../lib/lib.o ../lib/blk.o ; case10

Inputs

The input files are:

int_wng/cp_cor.dat: this file corresponds to the corrected inviscid data file; it contains 20 spanwise stations and 89 chordwise stations.

int_wng/ref.dat: this is a file which contains the reference cross sections of the wing, used in attachment line relocation

Other inputs to the interface program are given in the main routine and are as follows:

Parameters corresponding to reading the inviscid data; these include number of stations to be skipped near the root and wing sections and the surface (upper or lower) to be analysed; details are given in the program by comment statements

Mach number and error tolerances in velocity used in the surface Euler calculation

Parameters relating to the relocation iteration of the wing attachment line; details given in the program

The BL grid distribution and smoothing factors used in the spline interpolation

For the BL analysis, the inputs are as given in S/R inputs in inf_bl/case10.f, att_bl/case10.f and in 3d_bl/case10.f; the output from the interface program int_wng/fort.20_case10 is the input file for the BL analysis programs.

Program Description

The interface program first reads in the inviscid data in S/R rinv; a number of sections near the wing root and tip can be skipped if necessary to exclude regions from the BL calculation. Depending on the surface (upper or lower) of the wing being analysed, only part of the inviscid data will be read in.

The attachment line is then relocated so as to ensure that the streamwise edge velocity is zero on this line. This is done in S/R reloc. The inviscid pressures are interpolated to the BL grid with or without smoothing also in this subroutine. The surface metrics are then calculated in S/R metrix. The surface Euler equations are solved in S/R euler and the edge conditions output to a file for BL analysis.

The BL solution on the wing attachment line is obtained in the program att_bl/case10.f. The attachment line equations from Table 5 are solved here (see S/R edgeatt).

The BL solution for the locally infinite swept boundary near the wing root is obtained in the program inf_bl/case10.f (see S/R edgeinf).

The 3D BL solution on the wing is obtained in the program 3d_bl/case10.f. The initial and side boundary boundary layer solution profiles are used here from att_bl/fort.20_case10 and inf_bl/fort.20_case10.

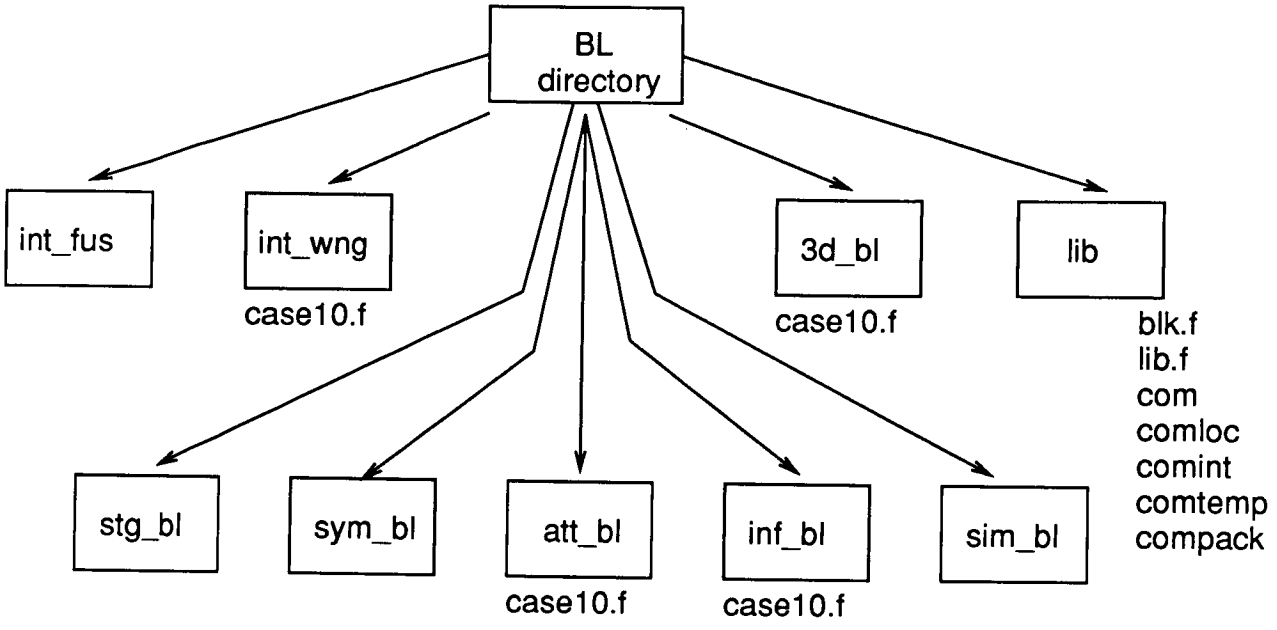
```
file 3d_bl/fort.2_case10: partial listing
-------------------------

 NACA 0012 straight wing, 20 deg swept, al=2

 Physical quantities are in U.S. units

 Wall condition : Adiabatic wall

 Inputs

 nx, ny, nz = 100,  15,  41
 zmax =    8.00
 reflen =   1.000000
 pfs =   2116.000
 tfs=     520.00
 mach no =      0.500
 trs=     182.00
 ufs =     558.85
 ref Re no per unit length =  0.9049511E+07
 fs  Re no per unit length =  0.3537899E+07
 fs sonic vel =    1117.70
 amuinf =  0.3745793E-06
 amuref =  0.1464415E-06
 rofs =  0.2371347E-02
 hrefs =  0.3123124E+07
 f-edge =   0.990000


 nxlim=   79
 nylim=   15
 iwall= 0
 iunit= 0
 ak=   1.000000
 epsf=  0.5000000E-04
 epsh=  0.5000000E-04
 iord=  5


 stretching factor ak=    1.00000
 zeta distribution

  0.0000000   0.200000  0.400000  0.600000  0.8000000
  0.9999999   1.200000  1.400000  1.600000  1.800000
   2.000000   2.200000  2.400000  2.599999  2.800000
   3.000000   3.200000  3.400000  3.599999  3.799999
   3.999999   4.199999  4.400000  4.599999  4.799999
   4.999999   5.199999  5.400000  5.599999  5.799999
   5.999999   6.199999  6.400000  6.599999  6.799999
   6.999999   7.199999  7.399999  7.599999  7.799999
   7.999999

 x stations

    0.00000    0.00100    0.00200    0.00300    0.00400    0.00500
```

0.00600	0.00700	0.00800	0.00900	0.01000	0.01100
0.01200	0.01300	0.01400	0.01500	0.01600	0.01700
0.01800	0.01900	0.02000	0.02100	0.02300	0.02600
0.03000	0.03400	0.03800	0.04200	0.04600	0.05000
0.05400	0.05800	0.06200	0.06600	0.07000	0.07400
0.07800	0.08200	0.08600	0.09000	0.09400	0.09800
0.10200	0.10600	0.11000	0.11400	0.11800	0.12200
0.12600	0.13000	0.13500	0.14100	0.14800	0.15600
0.16500	0.17500	0.18500	0.19500	0.20500	0.21500
0.22500	0.23500	0.24500	0.25500	0.26500	0.27500
0.28500	0.29500	0.30500	0.31500	0.32500	0.33500
0.34500	0.35500	0.36500	0.37500	0.38500	0.39500
0.40500					

y stations

0.18750	0.26250	0.33750	0.41250	0.48750	0.56250
0.63750	0.71250	0.78750	0.86250	0.93750	1.01250
1.08750	1.16250	1.23750			

Program Listings



The listings of the following files are given:

int_wing/case10.f	Interface program for wing, case 10
att_bl/case10.f	Attachment line solution program for wing, case 10
inf_bl/case10.f	Side boundary solution program for wing, case 10
3d_bl/case10.f	3D boundary layer solution program for wing, case 10
lib/blk.f	Subroutines to calculate reference condiions, solve momentum and energy equations etc.
lib/lib.f	Library routines
com,comloc comint,comtemp compack	'Include' files containing common blocks and parameter statements used in the different programs; array dimensions are set in these files

```
      program main
c
c     f77 -o case10 case10.f ../lib/lib.o ../lib/pack.o
c     set ii=70, ji=15 in comint; ii=70 corresponds to upper bound
c     of chordwise points in Hess results on wing upper/lower surface
c     set nx=100, ny=15 in com corresponding to BL grid
c
c     Interface procedure for flow past wing, Case10
c     Inviscid results assumed to be from Hess panel code
c
      include '../lib/comint'
      include '../lib/com'
      include '../lib/comtemp'
      parameter(nref=177)
c
c     nref is the number of points in a reference section in the
c     wrap-around direction; see s/r refcr
c
      common/a12    /xref(nref,ny),zref(nref,ny),sref(nref,ny)
      common/a11    /ntip,nroot,nxi,nyi,nxbl,nxbt
     1               ,irev,isurf,nbac
      common/a10    /nnose,mg,dst,uerlim,sufact
      common/c11    /ui(nx,ny),vi(nx,ny)
c
c     work arrays, check dimension
c
      common/a00    /ydtem(1000),temp(1000),stderr(nx)
c
c     input data
c
c        ntip----> number of root span stations to skip for bl analysis
c        nroot---> number of tip span stations to skip for bl analysis
c        nxi-----> number of chordwise inviscid points in inviscid input
c                  (total in the wrap-around direction)
c        nyi-----> number of spanwise stations
c        nxbl----> number of lines to skip before each section read
c        nxbt----> number of lines to skip after each section read
c        irev----> =1 indicates wing sections are input with outboard
c                  sections first; =0 means inboard to outboard
c        isurf---> 0 for lower surface, 1 for upper surface
c        nbac----> number of extra points to be stored beyond the
c                  attachment line for interpolation purpose, >2
c        reflen--> root chord for inviscid run
c
      data ntip,nroot,nxi,nyi,nxbl,nxbt,irev,isurf,nbac
     1    ,reflen
     1    /3,   2,    89, 20, 1,   0,   1,   1,    2
     1    ,1./
c
c     amach------> Mach number
c     gam--------> ratio of specific heats
c     iord-------> value of i at which x differencing becomes 2nd order
c     itmax------> max iterations for surface Euler solution
c     erru,errv--> error tolerances on u,v in surface Euler solution
c
      data amach,gam,iord,itmax,erru,errv
     1    /0.5,  1.4, 5  ,20   ,0.001,0.001/
c
c     nnose  --> # of points used for nose region interpolation
c     mg     --> # of points in the mini-grid used for att. line-relocation
c     mg and nnose assumed to be < 20 for dimensioning arrays in s/r reloc
c     dst    --> grid distance used in the mini-grid
c     uerlim --> error tolerance in ue for relocation iteration
c     sufact --> factor ds/due used in moving the attachment line
c
      data nnose,mg,dst,uerlim,sufact/
     1     20,   5, 0.002,0.001,0.02/
c
c     input files
c
c     'cp_cor.dat'  --> inviscid data file (from Hess code, in this case)
c     'refcr'       --> file containing reference cross sections
c                       reference sections are corresponding to the
c                       y locations and are used in att.line-relocation
c
      open(unit=1,file='cp_cor.dat')
      open(unit=3,file='ref.dat')
      open(unit=10,file='fort.10_case10')
c
c     output files
c
      open(unit=2,file='fort.2_case10')
      open(unit=20,file='fort.20_case10')
c
      rewind 20
      rewind 1
      rewind 2
      rewind 3
      rewind 10
c
c     BL grid distribution
c
      x(1)=0.
      ds1=0.001
      do 400 i=2,21
      x(i)=x(i-1)+ds1
 400  continue
      x(22)=0.021
      x(23)=0.023
      x(24)=0.026
      x(25)=0.030
      ds1=0.004
      do 401 i=26,49
      x(i)=x(i-1)+ds1
 401  continue
      x(50)=0.130
      x(51)=0.135
      x(52)=0.141
      x(53)=0.148
      x(54)=0.156
      x(55)=0.165
      ds1=0.010
      do 402 i=56,100
      x(i)=x(i-1)+ds1
 402  continue
      write(2,403)nx
      write(2,404)x
 403  format(/' BL grid x stations, nx = ',i3/)
 404  format(10(1x,f6.4))
c
c     pressure smoothing parameters
c
      do 556 i=1,10
      stderr(i)=0.01
 556  continue
      do 557 i=11,80
      stderr(i)=0.05
 557  continue
```

```
      do 558 i=81,100
      stderr(i)=0.01
 558  continue
c
c     read in inviscid data
c
      call rinv
c
      do 405 j=1,ny
      y(j)=yh(1,j)
 405  continue
      write(2,406)ny
      write(2,404)y
 406  format(/' BL grid y stations, ny = ',i3/)
c
c     read in reference cross sections
c
      call refcr
c
c     wing attachment line-relocation
c
      call reloc
c
c     calculate-metrics for BL grid
c
      call metrix
      print 1
 1    format(' metrics calculated')
c
c     solve surface Euler equations
c
      call euler
c
c     output for BL analysis
c
      do 3 i=1,nx
      do 3 j=1,ny
      write(20,4)x(i),y(j),ue(i,j),ve(i,j),h1(i,j),h2(i,j)
     1          ,g12(i,j)
 3    continue
 4    format(7(1x,g14.7))
      print 5
 5    format(' end of calculation')
 6    format(2(1x,g14.7))
 7    format(6(1x,g14.7))
      stop
      end
c-----------------------------------------------------------------
      subroutine rinv
c-----------------------------------------------------------------
c
c     read in inviscid data (Hess code results)
c
      include '../lib/comint'
      include '../lib/com'
      character*10 cha
      common/a11    /ntip,nroot,nxi,nyi,nxbl,nxbt
     1               ,irev,isurf,nbac
c
c
c     initial attachment line location from max pr line
c
c     skip ntip number of tip sections
c
      if(ntip.ne.0)then
          iskip=nxi+nxbl+nxbt
          do 20 j=1,ntip
          do 20 i=1,iskip
          read(1,3,err=110,end=120)cha
 20       continue
      endif
c
c     j loop
c
      do 1 j=1,nyi-ntip-nroot
      jj=j
c
c     reverse order of indexing if hess input
c
      if(irev.eq.1)then
          jj=nyi+1-j-ntip-nroot
      endif
c
c     skip nxbl lines before each section
c
      if(nxbl.ne.0)then
          do 650 kkk=1,nxbl
          read(1,3,err=110,end=120)cha
 650      continue
      endif
c
c     look for pr. maximum
c
      pmax=-10.
      do 2 i=1,nxi
      read(1,6555,err=110,end=120)a1,a2,a3,a4,a5,a6,a7
 6555 format(7e12.6)
      if(a7.gt.pmax)then
      pmax=a7
      ih(jj)=i
      endif
 2    continue
c
c     skip nxbt lines
c
      if(nxbt.ne.0)then
          do 651 kkk=1,nxbt
          read(1,3,err=110,end=120)cha
 651      continue
      endif
 1    continue
c
c     end j loop
c
      rewind 1
c
c     skip ntip number of tip sections
c
      if(ntip.ne.0)then
          do 21 j=1,ntip
          iskip=nxi+nxbl+nxbt
          do 21 i=1,iskip
 21       read(1,3,err=110,end=120)cha
      endif
c
c     read data corresponding to lower/upper surface of wing
c
      do 10 j=1,nyi-ntip-nroot
```

```
      jj=j
c
c     reverse order of indexing if hess input
c
      if(irev.eq.1)then
          jj=nyi+1-j-ntip-nroot
      endif
c
c     i1,i2 are the i locations in the range (1,nxi)
c
      i2=nxi
      i1=ih(jj)-nbac
      if(isurf.eq.0)then
          i1=1
          i2=ih(jj)+nbac
      endif
      ih(jj)=i2-i1+1
c
c     skip
c
      do 11 i=1,i1-1+nxbl
      read(1,3,err=110,end=120)cha
 11   continue
c
c     read
c
      do 12 i=1,ih(jj)
      iii=i
      if(isurf.eq.0)iii=ih(jj)+1-i
      read(1,6555,err=110,end=120)xh(iii,jj),yh(iii,jj),zh(iii,jj)
     1,uh(iii,jj),vh(iii,jj),wh(iii,jj),ph(iii,jj)
      xh(iii,jj)=xh(iii,jj)/reflen
      yh(iii,jj)=yh(iii,jj)/reflen
      zh(iii,jj)=zh(iii,jj)/reflen
 12   continue
c
c     skip
c
      isk=nxi+1-i1-ih(jj)+nxbt
      if(isk.ne.0)then
          do 14 i=1,isk
          read(1,3,err=110,end=120)cha
 14       continue
      endif
 10   continue
 3    format(a10)
c
c     write to unit 2
c     nbac number of points on the other side are included
c
c     do 500 j=1,ji
c     write(2,501)j,yh(1,j)
c501  format(' j= ',i2,1x,e14.7)
c     do 500 i=1,ih(j)
c     write(2,502)i,xh(i,j),zh(i,j),ph(i,j)
c500  continue
c502  format(1x,i2,3(1x,e14.7))
      write( 6,30)isurf
 30   format(//' inviscid data for surface ',i1,' read in')
      return
c
c     read error messages
c
 110  write(2,111)
      print 111
 111  format(///' error in unit 1  :  subroutine rinv ')
      stop
 120  write(2,121)
      print 121
 121  format(///' e-o-f in unit 1  :  subroutine rinv ')
      stop
c
      end
c-----------------------------------------------------------------
      subroutine refcr
c-----------------------------------------------------------------
c
c     read in the reference cross sections
c
      parameter(nref=177)
      include '../lib/comint'
      incluce '../lib/com'
      commor/a11    /ntip,nroot,nxi,nyi,nxbl,nxbt
     1               ,irev,isurf,nbac
      common/a12    /xref(nref,ny),zref(nref,ny),sref(nref,ny)
      do 1 j=1,ny
      do 1 i=1,nref
      read(3,*)xref(i,j),zref(i,j)
      if(i.eq.1)then
      sref(i,j)=0.
      else
      sref(i,j)=sref(i-1,j)+sqrt( (xref(i,j)-xref(i-1,j))**2+
     1                (zref(i,j)-zref(i-1,j))**2   )
      endif
 1    continue
      return
      end
c-----------------------------------------------------------------
      subroutine reloc
c-----------------------------------------------------------------
c
c     attachment line relocation procedure
c
      parameter(nref=177)
      include '../lib/comint'
      include '../lib/com'
      include '../lib/compack'
      include '../lib/comtemp'
      common/a10    /nnose,mg,dst,uerlim,sufact
      common/a11    /ntip,nroot,nxi,nyi,nxbl,nxbt
     1               ,irev,isurf,nbac
      common/a12    /xref(nref,ny),zref(nref,ny),sref(nref,ny)
      common/a13    /xst(ny),yst(ny),zst(ny),sst(ny),
     1               ust(ny),vst(ny),wst(ny),pst(ny)
      common/a14    /xstl(ny),ystl(ny),zstl(ny),sstl(ny),
     1               ustl(ny),vstl(ny),wstl(ny),pstl(ny)
      common/a15    /xstu(ny),ystu(ny),zstu(ny),sstu(ny),
     1               ustu(ny),vstu(ny),wstu(ny),pstu(ny)
      common/a16    /xsti(ny),ysti(ny),zsti(ny),ssti(ny),
     1               usti(ny),vsti(ny),wsti(ny),psti(ny)
      common/a00    /ydtem(1000),temp(1000),stderr(nx)
      common/b00    /pbs(nx,ny)
      common/c00    /ub(nx,ny),vb(nx,ny),wb(nx,ny)
      dimension ysm(1000)
c
c     nnose,mg assumed <20
c
      dimension sm(20,ny),xm(20,ny),zm(20,ny)
```

```
      dimension usurf(ny),vsurf(ny),wsurf(ny)
      dimension xxi(ny),zxi(ny),xetl(ny),zetl(ny),xet(ny),zet(ny)
      dimension a(3,3),b(3,3)
      print 444
      write(2,444)
  444 format(' Attachment line relocation iteration'/)
c
      iter=0
      nbac1=nbac+1
      write(2,333)
  333 format(' initial xst,yst,zst,pst,sst,sstl,sstu'/)
      do 1 j=1,ny
      xst(j)=xh(nbac1,j)
      yst(j)=yh(nbac,j)
      zst(j)=zh(nbac1,j)
      ust(j)=uh(nbac1,j)
      vst(j)=vh(nbac1,j)
      wst(j)=wh(nbac1,j)
      pst(j)=ph(nbac1,j)
c
      xsti(j)=xh(nbac1,j)
      zsti(j)=zh(nbac1,j)
      usti(j)=uh(nbac1,j)
      vsti(j)=vh(nbac1,j)
      wsti(j)=wh(nbac1,j)
      psti(j)=ph(nbac1,j)
c
      xstu(j)=xh(nbac+2,j)
      zstu(j)=zh(nbac+2,j)
      ustu(j)=uh(nbac+2,j)
      vstu(j)=vh(nbac+2,j)
      wstu(j)=wh(nbac+2,j)
      pstu(j)=ph(nbac+2,j)
c
      xstl(j)=xh(nbac,j)
      zstl(j)=zh(nbac,j)
      ustl(j)=uh(nbac,j)
      vstl(j)=vh(nbac,j)
      wstl(j)=wh(nbac,j)
      pstl(j)=ph(nbac,j)
      write(2,334)j,xst(j),yst(j),zst(j),pst(j),sst(j),sstl(j),sstu(j)
    1 continue
c
c     j loop to find s corresponding to attachment line
c
      nref2=nref/2+1
      do 2 j=1,ny
c
c     nnose points on either side of the le
c
      do 3 i=nref2-nnose,nref2+nnose
      iii=i-(nref2-nnose)+1
      xpt(iii)=zref(i,j)
      ypt(iii)=sref(i,j)
    3 continue
c
c     interpolate for stagnation s
c
      nint=2*nnose+1
      call spline(xpt,ypt,nint,zst(j),ssti(j),0)
      call spline(xpt,ypt,nint,zstu(j),sstu(j),0)
      call spline(xpt,ypt,nint,zstl(j),sstl(j),0)
  334 format(1x,i2,7(1x,f10.6))
    2 continue
      do 25 j=1,ny
      sst(j)=ssti(j)
   25 continue
c
c     relocation iteration loop
c
   31 continue
      iter=iter+1
      print23,iter
      write(2,23)iter
   23 format(' relocation iteration # ',i2)
c
c     layout a mini grid of 2*mg+1 points around the attachment line
c     with a s distribution equal to dst (equally spaced)
c
      do 22 j=1,ny
      sm(mg+1,j)=sst(j)
      do 4 i=1,mg
      sm(i,j)=sst(j)-(mg+1-i)*dst
      sm(mg+i+1,j)=sst(j)+mg*dst
    4 continue
c
c     interpolate z values for this mini-grid
c
      do 20 i=1,nref
      xpt(i)=sref(i,j)
      ypt(i)=zref(i,j)
   20 continue
      do 5 i=1,2*mg+1
      call spline(xpt,ypt,nref,sm(i,j),zm(i,j),i-1)
    5 continue
c
c     interpolate x values for this mini-grid
c
      do 6 i=1,nref
      ypt(i)=xref(i,j)
    6 continue
      do 7 i=1,2*mg+1
      call spline(xpt,ypt,nref,sm(i,j),xm(i,j),i-1)
    7 continue
      xst(j)=xm(mg+1,j)
      zst(j)=zm(mg+1,j)
c
c     calculate the two xi derivatives on attachment line
c
      xxi(j)=(xm(mg-1,j)-8*xm(mg,j)+8*xm(mg+2,j)-
     1        xm(mg+3,j))/(12*dst)
      zxi(j)=(zm(mg-1,j)-8*zm(mg,j)+8*zm(mg+2,j)-
     1        zm(mg+3,j))/(12*dst)
   22 continue
c
c     end j loop
c
c
c     calculate the two eta derivatives along attachment line
c
      do 8 j=1,ny
      xetl(j)=xm(mg+1,j)
      zetl(j)=zm(mg+1,j)
    8 continue
      do 9 j=1,ny
      xet(j)=derk(yst,xetl,ny,j)
      zet(j)=derk(yst,zetl,ny,j)
    9 continue
```

```
c
c     conversion of velocities
c
      do 10 j=1,ny
      d1=xxi(j)
      d2=0.0
      d3=zxi(j)
      d4=xet(j)
      d5=1.
      d6=zet(j)
      a(1,1)=d1
      a(2,1)=d2
      a(3,1)=d3
      a(1,2)=d4
      a(2,2)=d5
      a(3,2)=d6
      a13=d2*d6-d3*d5
      a23=d4*d3-d6*d1
      a33=d1*d5-d4*d2
      del=sqrt(a13*a13+a23*a23+a33*a33)
      a(1,3)=a13/del
      a(2,3)=a23/del
      a(3,3)=a33/del
      b(1,1)=ust(j)
      b(2,1)=vst(j)
      b(3,1)=wst(j)
      call solv3(a,b)
      rg11=sqrt(d1*d1+d2*d2+d3*d3)
      rg22=sqrt(d4*d4+d5*d5+d6*d6)
      usurf(j)=b(1,1)*rg11
      vsurf(j)=b(2,1)*rg22
      wsurf(j)=b(3,1)
 10   continue
c
c     calculate-relocated sst(j) based on usurf(j)
c
      do 21 j=1,ny
c     sufact correction
      sst(j)=sst(j)-usurf(j)*sufact
c
c     quadratic interpolation for new ust,vst,wst,pst
c
      ust(j)=quad(sstu(j),ssti(j),sstl(j),ustu(j),usti(j),ustl(j),
     1            sst(j))
      vst(j)=quad(sstu(j),ssti(j),sstl(j),vstu(j),vsti(j),vstl(j),
     1            sst(j))
      wst(j)=quad(sstu(j),ssti(j),sstl(j),wstu(j),wsti(j),wstl(j),
     1            sst(j))
      pst(j)=quad(sstu(j),ssti(j),sstl(j),pstu(j),psti(j),pstl(j),
     1            sst(j))
c
c     linear interpolation near att. line
c     for pressure to avoid +ve pressure gradient values
c
      if(isurf.eq.1)then
      pst(j)=stlin(ssti(j),sstu(j),psti(j),pstu(j),sst(j))
      else
      pst(j)=stlin(ssti(j),sstl(j),psti(j),pstl(j),sst(j))
      endif
      write(2,334)j,xst(j),yst(j),zst(j),pst(j),sst(j),sstl(j),sstu(j)
 21   continue
c
      uamax=0.0
      do 30 j=1,ny
      if(abs(usurf(j)).ge.uamax)uamax=abs(usurf(j))
 30   continue
      print33,uamax
      write(2,33)uamax
 33   format(' maximum att. line u vel error = ',e14.7)
      print5122,usurf
      write(2,5122)usurf
 5122 format(10(1x,f7.3))
      if(uamax.gt.1.)then
      print 5111
 5111 format(' no conv. in att. line-reloc'/
     1' value of sufact to be changed')
      stop
      endif
      if(uamax.ge.uerlim)goto 31
      print 32
      write(2,32)
 32   format(/' attachment line-relocation completed')
c
c     spline- interpolation for pressure
c
      do 550 j=1,ny
      xpt(1)=0.0
      ypt(1)=pst(j)
      sloc=0.
      do 208 i=2+nbac,ih(j)
      iii=i-nbac
      ypt(iii)=ph(i,j)
      if(iii.eq.2)then
      sloc=sqrt((xh(i,j)-xst(j))**2+(zh(i,j)-zst(j))**2)
      else
      sloc=sloc+sqrt((xh(i,j)-xh(i-1,j))**2+(zh(i,j)-zh(i-1,j))**2)
      endif
      xpt(iii)=sloc
 208  continue
      nih=ih(j)-nbac
      slp1=0.
      slpn=0.
      islpsw=3
      sigma=1.0
c
c     interpolation without smoothing
c     Results in pb
c
      call curv1(nih,xpt,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 209 i=1,nx
      pb(i,j)=curv2(x(i),nih,xpt,ypt,ydtem,sigma)
 209  continue
c
c     interpolation with smoothing
c     Results in pbs
c
      isw=0
      sss=nih
      eps=sqrt(2./sss)
      call curvs(nih,xpt,ypt,stderr,isw,sss,eps,
     1           ysm,ydtem,sigma,temp,ierr)
      do 909 i=1,nx
      pbs(i,j)=curv2(x(i),nih,xpt,ysm,ydtem,sigma)
 909  continue
 210  format(2(1x,g14.7))
c------------Velocity interpolation for comparison plots-------------
      do 1000 i=2+nbac,ih(j)
      iii=i-nbac
```

```
      ypt(iii)=uh(i,j)
 1000 continue
      call curv1(nih,xpt,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 1001 i=1,nx
      ub(i,j)=curv2(x(i),nih,xpt,ypt,ydtem,sigma)
 1001 continue
c
      do 2000 i=2+nbac,ih(j)
      iii=i-nbac
      ypt(iii)=vh(i,j)
 2000 continue
      call curv1(nih,xpt,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 2001 i=1,nx
      vb(i,j)=curv2(x(i),nih,xpt,ypt,ydtem,sigma)
 2001 continue
c
      do 3000 i=2+nbac,ih(j)
      iii=i-nbac
      ypt(iii)=wh(i,j)
 3000 continue
      call curv1(nih,xpt,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 3001 i=1,nx
      wb(i,j)=curv2(x(i),nih,xpt,ypt,ydtem,sigma)
 3001 continue
c---------------end velocity interpolation--------------------------
 550  continue
c
c     evaluate xb,zb for the BL grid
c
      do 260 j=1,ny
      if(isurf.eq.1)then
      do 261 i=1,nref
      xpt(i)=sref(i,j)-sst(j)
      ypt(i)=xref(i,j)
 261  continue
      else
      do 262 i=1,nref
      iii=nref+1-i
      xpt(iii)=sst(j)-sref(i,j)
      ypt(iii)=xref(i,j)
 262  continue
      endif
c
c     normalizing length
c
      clnorm=xref(nref,j)-xref(nref2,j)
      call curv1(nref,xpt,ypt,slp1,slpn,islpw,ydtem,temp,sigma,ierr)
      do 263 i=1,nx
      xb(i,j)=curv2(x(i)*clnorm,nref,xpt,ypt,ydtem,sigma)
 263  continue
 260  continue
c
      do 270 j=1,ny
      if(isurf.eq.1)then
      do 271 i=1,nref
      xpt(i)=sref(i,j)-sst(j)
      ypt(i)=zref(i,j)
 271  continue
      else
      do 272 i=1,nref
      iii=nref+1-i
      xpt(iii)=sst(j)-sref(i,j)
      ypt(iii)=zref(i,j)
 272  continue
      endif
      call curv1(nref,xpt,ypt,slp1,slpn,islpw,ydtem,temp,sigma,ierr)
      do 273 i=1,nx
      yb(i,j)=y(j)
      zb(i,j)=curv2(x(i),nref,xpt,ypt,ydtem,sigma)
 273  continue
 270  continue
      do 295 j=1,ny
      ve(1,j)=vsurf(j)
      ue(1,j)=0.
 295  continue
      return
      end
c-------------------------------------------------------------
      subroutine metrix
c-------------------------------------------------------------
      include '../lib/comint'
      include '../lib/com'
      include '../lib/compack'
      include '../lib/comtemp'
      common/all    /ntip,nroot,nxi,nyi,nxbl,nxbt
     1              ,irev,isurf,nbac
      common/a00    /ydtem(1000),temp(1000),stderr(nx)
      common/c00    /ub(nx,ny),vb(nx,ny),wb(nx,ny)
      common/c11    /ui(nx,ny),vi(nx,ny)
      dimension a(3,3),b(3,3)
      data slp1,slpn,islpsw,sigma/0.,0.,3,0./
c
c     calculate the two xi direction derivatives
c
      do 2 j=1,ny
      do 1 i=1,nx
      ypt(i)=xb(i,j)
 1    continue
      call curv1(nx,x,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 3 i=1,nx
      s(1,i,j)=curvd(x(i),nx,x,ypt,ydtem,sigma)
 3    continue
      do 4 i=1,nx
      ypt(i)=zb(i,j)
 4    continue
      call curv1(nx,x,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 5 i=1,nx
      s(2,i,j)=0.
      s(3,i,j)=curvd(x(i),nx,x,ypt,ydtem,sigma)
 5    continue
 2    continue
c
c     calculate the two eta direction derivatives
c
      do 10 i=1,nx
      do 6 j=1,ny
      ypt(j)=xb(i,j)
 6    continue
      call curv1(ny,y,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 7 j=1,ny
      s(4,i,j)=curvd(y(j),ny,y,ypt,ydtem,sigma)
 7    continue
      do 8 j=1,ny
      ypt(j)=zb(i,j)
 8    continue
      call curv1(ny,y,ypt,slp1,slpn,islpsw,ydtem,temp,sigma,ierr)
      do 9 j=1,ny
      s(5,i,j)=1.
```

```
      s(6,i,j)=curvd(y(j),ny,y,ypt,ydtem,sigma)
 9    continue
 10   continue
c
c     h1
c
      do 30 i=1,nx
      do 30 j=1,ny
      h1(i,j)=sqrt(s(1,i,j)**2+s(2,i,j)**2+s(3,i,j)**2)
 30   continue
c
c     h2
c
      do 31 i=1,nx
      do 31 j=1,ny
      h2(i,j)=sqrt(s(4,i,j)**2+s(5,i,j)**2+s(6,i,j)**2)
 31   continue
c
c     g12
c
      do 33 j=1,ny
      do 33 i=1,nx
      g12(i,j)=s(1,i,j)*s(4,i,j)+s(2,i,j)*s(5,i,j)+s(3,i,j)*s(6,i,j)
 33   continue
c
c     interpolated velocity components transformation
c
      do 100 i=1,nx
      do 100 j=1,ny
      d1=s(1,i,j)
      d2=s(2,i,j)
      d3=s(3,i,j)
      d4=s(4,i,j)
      d5=s(5,i,j)
      d6=s(6,i,j)
      a(1,1)=d1
      a(2,1)=d2
      a(3,1)=d3
      a(1,2)=d4
      a(2,2)=d5
      a(3,2)=d6
      a13=d2*d6-d3*d5
      a23=d4*d3-d6*d1
      a33=d1*d5-d4*d2
      del=sqrt(a13*a13+a23*a23+a33*a33)
      a(1,3)=a13/del
      a(2,3)=a23/del
      a(3,3)=a33/del
      b(1,1)=ub(i,j)
      b(2,1)=vb(i,j)
      b(3,1)=wb(i,j)
      call solv3(a,b)
      rg11=sqrt(d1*d1+d2*d2+d3*d3)
      rg22=sqrt(d4*d4+d5*d5+d6*d6)
      ui(i,j)=b(1,1)*rg11
      vi(i,j)=b(2,1)*rg22
 100  continue
      return
      end
c-----------------------------------------------------------------
      subroutine euler
c-----------------------------------------------------------------
c
c     solve surface euler system for the boundaries
c     of the wing (l.i.s.w. assumption)
c     and the interior (3-d surface euler)
c
      include '../lib/comint'
      include '../lib/com'
      include '../lib/compack'
      include '../lib/comtemp'
      common/a00    /ydtem(1000),temp(1000),stderr(nx)
      common/b00    /pbs(nx,ny)
c
c     avoiding -ve pressure gradients at att. line
c
      do 333 j=1,ny
      if(pbs(1,j).le.pbs(2,j))then
      pbs(1,j)=stlin(x(2),x(3),pbs(2,j),pbs(3,j),x(1))
      endif
      if(pb(1,j).le.pb(2,j))then
      pb(1,j)=stlin(x(2),x(3),pb(2,j),pb(3,j),x(1))
      endif
 333  continue
c
c     initialize u and v
c
      do 1 i=2,nx
      do 2 j=1,ny
      ue(i,j)=ue(i-1,j)
      ve(i,j)=ve(i-1,j)
      dxi=x(i)-x(i-1)
c
c     krause type y differencing coefficients
c
      if(j.ne.1.and.j.ne.ny)then
      bd1=1./(2*(y(j)-y(j-1)))
      bd2=-bd1
      bd3=1./(2*(y(j+1)-y(j)))
      bd4=-bd3
      else
      bd1=0
      bd2=0
      bd3=0
      bd4=0
      endif
c
      if(i.eq.2)then
      a1=1./dxi
      a2=-a1
      a3=0.0
      pb2=0.0
      ui2=0.0
      vi2=0.0
      else
      dxi1=x(i-1)-x(i-2)
      r=float(i-2)/(iord-2)
      if(r.gt.1)r=1.
      om=dxi/dxi1
      q1=(1-r)/dxi
      a1=q1+r*(2*om+1)/((om+1)*dxi)
      a2=-q1-r*(om+1)/dxi
      a3=r*om/((om+1)*dxi1)
      pb2=pb(i-2,j)
      pbs2=pbs(i-2,j)
      ui2=ue(i-2,j)
      vi2=ve(i-2,j)
      endif
```

```
c
c     dpx=a1*pb(i,j)+a2*pb(i-1,j)+a3*pb2
c
      dpx=a1*pbs(i,j)+a2*pbs(i-1,j)+a3*pbs2
      gl=h1(i,j)**2*h2(i,j)**2-g12(i,j)**2
c
c     density by isentropic asumption; review if flow is supersonic
c     with shocks
c
      rol=(1.+pb(i,j)*gam*amach**2*0.5)**(1./gam)
      ui1=ue(i-1,j)
      vi1=ve(i-1,j)
      g12l=g12(i,j)
      h1l=h1(i,j)
      h2l=h2(i,j)
      h2yl=derj21(i,j,nx,ny,y,h2)
      h1xl=0.
      h1yl=derj21(i,j,nx,ny,y,h1)
      h2xl=deri21(i,j,nx,ny,x,h2)
      cth1=g12l/(h1l*h2l)
      g12xl=deri21(i,j,nx,ny,x,g12)
      g12yl=derj21(i,j,nx,ny,y,g12)
      c24l=g12l/gl*(g12l*h1xl/h1l**2+h1yl-g12xl/h1l)
      c25l=(h1l*h2l*h1yl*(1.+cth1**2)-2.*g12l*h2xl)/gl
      c26l  =h1l/gl*(g12yl-h2l*h2xl-g12l*h2yl/h2l)
      c34l=h2l/gl*(g12xl-h1l*h1yl-g12l*h1xl/h1l)
      c35l=(h1l*h2l*(1+cth1**2)*h2xl-2*g12l*h1yl)/gl
      c36l=g12l/gl*(g12l*h2yl/h2l**2-g12yl/h2l+h2xl)
      rhs1=0.5*h1l*(-dpx*h2l**2+g12l*dpy)/(rol*gl)
      rhs2=0.5*h2l*(dpx*g12l-h1l**2*dpy)/(rol*gl)
      iter=0
c
c     u,v initialization for i=2 case
c
      if(i.eq.2)then
         ue(i,j)=sqrt(pb(1,j)*0.5-pb(2,j)*0.5)
         ve(i,j)=ve(1,j)-ue(i,j)*h1l/h2l
      endif
c
c     begin iteration for non-linear terms
c
 100  iter=iter+1
      if(iter.gt.itmax)goto 101
      r1=rhs1-ue(i,j)/h1l*(a1*ue(i,j)+a2*ui1+a3*ui2)-c24l*ue(i,j)**2
     1   -c25l*ue(i,j)*ve(i,j)-c26l*ve(i,j)**2
      r2=rhs2-ue(i,j)/h1l*(a1*ve(i,j)+a2*vi1+a3*vi2)-
     1   c35l*ue(i,j)*ve(i,j)-c36l*ve(i,j)**2-c34l*ue(i,j)**2
      a11=(2*a1*ue(i,j)+a2*ui1+a3*ui2)/h1l+2*ue(i,j)*c24l+ve(i,j)*c25l
      a12=ue(i,j)*c25l+2*ve(i,j)*c26l
      a21=(a1*ve(i,j)+a2*vi1+a3*vi2)/h1l+2*ue(i,j)*c34l+ve(i,j)*c35l
      a22=ue(i,j)*c35l+2*ve(i,j)*c36l+a1*ue(i,j)/h1l
c
c     extra terms in the case of the 3-d region
c
      if(j.ne.1.and.j.ne.ny)then
      a11=a11+bd1*ve(i,j)/h2l
      a12=a12+(bd1*ue(i,j)+bd2*ue(i,j-1)+bd3*ue(i-1,j+1)+bd4*ue(i-1,j))
     1    /h2l
      r1=r1-ve(i,j)/h2l*(bd1*ue(i,j)+bd2*ue(i,j-1)+bd3*ue(i-1,j+1)+
     1   bd4*ue(i-1,j))
      a22=a22+(2*bd1*ve(i,j)+bd2*ve(i,j-1)+bd3*ve(i-1,j+1)+
     1         bd4*ve(i-1,j))/h2l
      r2=r2-ve(i,j)/h2l*(bd1*ve(i,j)+bd2*ve(i,j-1)+bd3*ve(i-1,j+1)+
     1   bd4*ve(i-1,j))
      endif
      dv=(r2*a11-r1*a21)/(a11*a22-a12*a21)
      du=(r1-a12*dv)/a11
      ue(i,j)=ue(i,j)+du
      ve(i,j)=ve(i,j)+dv
c     print 151,iter,du,dv
 151  format(' iter #',i2,' du=',e14.7,' dv=',e14.7)
      if(abs(du).le.erru.and.abs(dv).le.errv)then
      write(2,150)i,j,x(i),ue(i,j),ve(i,j)
 150  format('s.e. solution at i=',i3,' j=',i2,
     1       ' x=',f10.5,' ue=',f10.5,' ve=',f10.5)
      go to 105
      else
      goto 100
      endif
 105  continue
 2    continue
      print 3,i
 3    format(' S.E. solution complete at i=',i3)
 1    continue
      return
 101  write(6,102)
 102  format(' itmax exceeded')
      stop
      end
```

```
c     f77 -o case10 case10.f ../lib/lib.o ../lib/blk.o
c     set nx=100, ny=15, nz=41
c     ATTACHMENT LINE COMPRESSIBLE BL SOLUTION (ATT_BL) FOR SPECIFIED WALL
c     TEMPERATURE OR ADIABATIC WALL CONDITION; FOURTH ORDER ACCURATE  IN
c     WALL NORMAL DIRECTION;  PROGRAM BASED ON NASA CR ..../IYER
c
c     Test case #10: Attachment line on swept wing
c-----------------------------------------------------------------------
      program att
c-----------------------------------------------------------------------
c
      include '../lib/com'
c
      open(unit=2,file='fort.2_case10')
      open(unit=3,file='fort.3_case10')
      open(unit=4,file='fort.4_case10')
      open(unit=30,file='../int_wng/fort.20_case10')
      open(unit=20,file='fort.20_case10')
c
c     fort.2 output of ref. qties., conv. history
c     fort.3 output of profiles
c     fort.4 output of global and wall qties. in physical units
c     fort.20 input from inviscid code
c
      rewind 2
      rewind 3
      rewind 4
      rewind 30
      rewind 20
c
      call inputs
      call initial
c
      do 10 j=1,nylim
      print 11,j,y(j),ve(1,j),fe(j)
      write(2,11)j,y(j),ve(1,j),fe(j)
 11   format(' iterating at j=',i3,' y=',g14.7,' ve=',g14.7,
     1       ' fe(j)=',g14.7)
      if(j.eq.1)call edgeinf
      if(j.gt.1)call edgeatt
c
      it=0
 12   it=it+1
      if(it.gt.itmax)goto 998
      call mmntm
      call cntnty
      call enrgy
      call updte
      if(istop.eq.1)goto 999
      call conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
      if(dfpmax.gt.epsf.or.dhpmax.gt.epsh)goto 12
c
      call phys
c
c     profiles at j+1 set to j
c     g profiles scaled according to edge velocity
c
      if(j.lt.nylim)then
      do 20 k=1,nz
      f(j+1,k)=f(j,k)
      g(j+1,k)=g(j,k)*ve(1,j+1)/ve(1,j)
      h(j+1,k)=h(j,k)
      fp(j+1,k)=fp(j,k)
      gp(j+1,k)=gp(j,k)*ve(1,j+1)/ve(1,j)
      hp(j+1,k)=hp(j,k)
      w(j+1,k)=w(j,k)
 20   continue
      endif
 10   continue
c
c     write attachment line solution to unit 20
c
      do 520 j=1,ny
      do 510 k=1,nz
      write(20,511)f(j,k),fp(j,k),g(j,k),gp(j,k),h(j,k),hp(j,k)
 510  continue
      do 516 k=1,nz
      write(20,512)w(j,k),al(k),alp(k),thel(k),the2(k)
 516  continue
 520  continue
 511  format(6(1x,e14.7))
 512  format(5(1x,e14.7))
      write(2,13)
      print 13
 13   format(' end of calculation')
      stop
c
 999  print 997
      write(2,997)
 997  format(' thel is -ve')
      print 996,(k,thel(k),k=1,nz)
      write (2,996)(k,thel(k),k=1,nz)
 996  format(4(i2,g14.7))
      stop
c
 998  write(2,995)
      print 995
 995  format(' itmax has been exceeded; calculation stopped')
      stop
c
      end
c-----------------------------------------------------------------------
      subroutine inputs
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      character*80 title
      data title/' Att line flow swept wing, case10'/
      data jp/1,5,10,15,96*0/
c
c     iunit=0 --> inputs assumed in U.S. Customary Units (lb, deg R, ft)
c     iunit=1 --> inputs assumed in S.I. Units (kg, deg K, m, Pa)
c
      iunit=0
      amach=0.50
c
      pfs=2116.0
      tfs=520.0
      reflen=1.
      iwall=0
c
      prl=0.72
      nxlim=1
      nylim=15
c
c     number of profile stations;  see data for ip
c
      nppj=4
```

```
      ak=1.0
      itmax=100
      epsf=5.e-05
      epsh=5.e-05
      zmax=8.0
      fedge=0.99
      gedge=0.99
      iord=5
c
      write(2,16)title
      print 16,title
 16   format(a80/)
c
      call ref
      write(2,40)nylim,iwall,iunit,ak,epsf,epsh,iord,crs,gedge
 40   format(/' nylim= ',i4/
     1        ' iwall= ',i1/
     1        ' iunit= ',i1/
     1        ' ak= ',g14.7/
     1        ' epsf= ',g14.7/
     1        ' epsh= ',g14.7/
     1        ' iord= ',i2/
     1        ' crs= ',g14.7/
     1        ' gedge= ',g14.7/)
c
c     z distribution (evenly spaced)
c
      if(ak.le.1.0001)then
      dz=zmax/(nz-1)
      do 12 k=1,nz
      z(k)=(k-1)*dz
 12   continue
      else
c
c     z distribution (stretched grid based on ak)
c
      do 13 k=1,nz
      ql=(ak**(k-1)-1)/(ak**(nz-1)-1)
      z(k)=zmax*ql
 13   continue
      endif
c
      print 14,ak
      write(2,14)ak
      print 15,z
      write(2,15)z
 14   format(/' stretching factor ak= ',f10.5/
     1' zeta distribution'/)
 15   format(4(1x,g14.7))
c
      call rinvis
c
      write(2,251)
 251  format(2x,'j',7x,'y(j)',12x,'ve',13x,'fe'/)
      do 250 j=1,ny
      write(2,202)j,y(j),ve(i,j),fe(j)
 250  continue
 202  format(1x,i3,3(1x,g14.7))
      return
      end
c-----------------------------------------------------------------------
      subroutine initial
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
c
c     initialize f,g,fp(i.e. f') and gp (g')
c
      zedge=4.0
      accel=1.0
      do 10 j=1,ny
      do 10 k=1,nz
      zr=z(k)/zedge
      if(zr.gt.1.0)then
      f(j,k)=1.0
      fp(j,k)=0.0
      g(j,k)=ve(1,j)
      gp(j,k)=0.
      else
c
c     initial profile
c
      f(j,k)=1.-(1+zr)*(1-zr)**3+accel/6.*zr*(1-zr)**3
      g(j,k)=f(j,k)*ve(1,j)
      endif
 10   continue
      do 12 j=1,ny
      do 11 k=1,nz
      teml(k)=f(j,k)
 11   continue
      do 12 k=1,nz
      fp(j,k)=derk(z,teml,nz,k)
      gp(j,k)=fp(j,k)*ve(1,j)
      w(j,k)=sumint(z,teml,nz,k)
      w(j,k)=-w(j,k)*(1.+crs)
 12   continue
c
c     specify a h distribution
c
      do 13 j=1,ny
      do 13 k=1,nz
      h(j,k)=htot
      hp(j,k)=0.0
 13   continue
      do 14 k=1,nz
      al(k)=1.0
      alp(k)=0.0
      all(k)=1./prl
      allp(k)=0.0
      all(k)=al(k)
      al2(k)=al(k)
      allp(k)=alp(k)
      al2p(k)=alp(k)
      thel(k)=1.
      the2(k)=0.
 14   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine rinvis
c-----------------------------------------------------------------------
      include '../lib/com'
      do 1 i=1,nx
      do 1 j=1,ny
      read(30,333)x(i),y(j),ue(i,j),ve(i,j),h1(i,j),h2(i,j),g12(i,j)
 1    continue
 333  format(7(1x,g14.7))
      do 3 j=1,ny
```

```
      fe(j)=deri21(1,j,nx,ny,x,ue)
c
c     fe(j) by 2-point formula
c
      fe(j)=(ue(2,j)-ue(1,j))/(x(2)-x(1))
 3    continue
c
c     calculate te assuming constant total enthalpy
c
      i=1
      do 16 j=1,ny
      te(i,j)=htot/(gam*amach*amach)-(gam-1)/(2*gam)*
     1         (ue(i,j)**2+ve(i,j)**2+
     1         2*ue(i,j)*ve(i,j)*g12(i,j)/(h1(i,j)*h2(i,j)))
 16   continue
c
      i=1
      do 17 j=1,ny
      amuel=te(i,j)**1.5*(1.+stilde)/(te(i,j)+stilde)
      amue(i,j)=amuel
c
c     roe calculated by isentropic formula
c     if free stream to bl edge is not a isentropic process
c     roe can be calculated if the bl edge pressure is known
c
      q1=1.+(gam-1.)/(2.*gam*tf)
      qt2=ue(i,j)**2+ve(i,j)**2+
     1    2*ue(i,j)*ve(i,j)*g12(i,j)/(h1(i,j)*h2(i,j))
      q2=1.+(gam-1.)*qt2/(2*gam*te(i,j))
      roel=(q1/q2)**(1./(gam-1.))
      roe(i,j)=roel
 17   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine edgeinf
c-----------------------------------------------------------------------
c
c     Locally infinite swept attachment line coefficients
c
      dimension c26(3),x1(3)
      include '../lib/com'
      j=1
      i=1
      ix=0
      a1=0.
      a2=0.
      a3=0.
      iy=0
      bd =0.
      bd1=0
      bd2=0.
      bd3=0.
      bd4=0
      bd5=0
      bd6=0
      bd7=0
      gref=1.
      grfe=gref/fe(j)
c
      aa1s=0.
      aa2s=-1.0
      aa3s=0.
      aa4s=0.0
c
c     LIS att. line
c
      h11=h1(i,j)
      h21=h2(i,j)
      g121=g12(i,j)
      h1x1=deri21(i,j,nx,ny,x,h1)
      h2x1=deri21(i,j,nx,ny,x,h2)
      h1y1=derj21(i,j,nx,ny,y,h1)
      h2y1=derj21(i,j,nx,ny,y,h2)
      g12y1=derj21(i,j,nx,ny,y,g12)
      g1=h11**2*h21**2-g121**2
      g1sq=sqrt(g1)
      c121=g1sq/h21
      c12n=sqrt(h1(i,j+1)**2*h2(i,j+1)**2-g12(i,j+1)**2)/h2(i,j+1)
      c12y1=(c12n-c121)/(y(2)-y(1))
      aa4s=-h11*c12y1/g1sq*grfe
      psi1=sqrt(fe(j)/(roe(i,j)*amue(i,j)*h1(i,j)))
c
      b1=-aa1s
      b2=0.
      b3=2.0
      b4=0
      b5=0.
      b6=-1.
c
c     LIS att. line
c
      cth1=g121/(h11*h21)
      c251=(h11*h21*h1y1*(1+cth1**2)-2*g121*h2x1)/g1
      b4=h11*c251*grfe-aa4s
c
c     calculation for c26x
c
      c26(1)=h11/g1*(g12y1-h21*h2x1-g121*h2y1/h21)
      g12=h1(2,j)/((h1(2,j)*h2(2,j))**2-g12(2,j)**2)
      g13=h1(3,j)/((h1(3,j)*h2(3,j))**2-g12(3,j)**2)
      g12y2=derj21(2,j,nx,ny,y,g12)
      g12y3=derj21(3,j,nx,ny,y,g12)
      h2x2=deri21(2,j,nx,ny,x,h2)
      h2x3=deri21(3,j,nx,ny,x,h2)
      h2y2=derj21(2,j,nx,ny,y,h2)
      h2y3=derj21(3,j,nx,ny,y,h2)
      c26(2)=h1(2,j)/g12*(g12y2-h2(2,j)*h2x2-g12(2,j)*h2y2/h2(2,j))
      c26(3)=h1(3,j)/g13*(g12y3-h2(3,j)*h2x3-g12(3,j)*h2y3/h2(3,j))
      x1(1)=x(1)
      x1(2)=x(2)
      x1(3)=x(3)
      c26x1=derk(x1,c26,3,1)
      b3s=b3+aa2s
      b4s=b4+aa4s
      b5=h11*c26x1*grfe**2
      b6=-1-b4s*g(j,nz)-b5s*g(j,nz)**2
c
      cc1=0.
      cc2=0.
      cc3=0.
      cc4=1.
      cc5=0
      cc6=0.
c
c     LIS att. line
c
      c361=g121/g1*(g121*h2y1/h21**2-g12y1/h21+h2x1)
```

```
      cc5=-aa4s+h11*c361*grfe
      cc6=-h11*c361*g(j,nz)**2
c
      dd1=-aa1s
      dd2=-aa3s
      dd3=-aa2s
      dd4=-aa4s
c
      cc4s=cc4+aa2s
      cc5s=cc5+aa4s
      dd6=1./hcon*(1-pr1)/pr1
      dd7=g121/(h11*h21)
c
      uden=0.
      vden=1.0
c
      uden2=uden**2
      vden2=vden**2
      d7uv=dd7*uden*vden
      uvden=uden*vden
      stild=stilde/te(i,j)
c
c     wall temperature as a function of s
c
      if(iwall.eq.1)then
      tws1=tws(i,j)
      tw=tws1/trs
      hw=gam*amach**2*tw
      endif
c
c     ww to be-updated in s/r updte
c
      ww=w(j,1)
c
      call diag
      return
      end
c-----------------------------------------------------------------------
      subroutine edgeatt
c-----------------------------------------------------------------------
c
c     General att. line coefficients
c
      dimension c26(3),x1(3)
      include '../lib/com'
      iy=1
      ix=0
      a1=0.
      a2=0.
      a3=0.
c
      if(iy.eq.1)then
      j1=j-1
      j2=j+1
      j3=j
      j4=j+1
      j5=j+2
      j6=j-1
      j7=j-2
      if(jb1.eq.2)j5=j+1
      if(jb1.eq.1)j7=j-1
      endif
c
      bd =1./(y(j)-y(j-1))
      bd1=-1./(y(j)-y(j-1))
      bd2=0.
      bd3=0.
      bd4=0
      bd5=0
      bd6=0
      bd7=0
      ge2y=bd*g(j,nz)**2+bd1*g(j-1,nz)**2
      h11=h1(i,j)
      h21=h2(i,j)
      g121=g12(i,j)
      h1x1=deri21(i,j,nx,ny,x,h1)
      h2x1=deri21(i,j,nx,ny,x,h2)
      h1y1=derj21(i,j,nx,ny,y,h1)
      h2y1=derj21(i,j,nx,ny,y,h2)
      g12y1=derj21(i,j,nx,ny,y,g12)
      gref=1.0
      grefx=0
      grfe=gref/fe(j)
      g1=h11**2*h21**2-g121**2
      glsq=sqrt(g1)
      phi1=sqrt(fe(j)*roe(1,j)*amue(1,j)*h11)
      psi1=sqrt(fe(j)/(roe(i,j)*amue(i,j)*h1(i,j)))
      cth1=g121/(h11*h21)
      c251=(h11*h21*h1y1*(1.+cth1**2)-2.*g121*h2x1)/g1
      ak1=glsq*phi1*grfe/h21
      ak1=sqrt(h1(i,j-1)**2*h2(i,j-1)**2-g12(i,j-1)**2)*
     1    sqrt(fe(j-1)*roe(1,j-1)*amue(1,j-1)*h1(1,j-1))
     1    *grfe/h2(1,j-1)
      aky=(ak1-ak1)/(y(j)-y(j-1))
c
      aa1s=0.0
      aa2s=-1.
      aa3s=-h1(1,j)/h2(1,j)*grfe
      aa4s=-h11/(glsq*phi1)*aky
c
      b1=-aa1s
      b2=-aa3s
      b3=2.0
      feyl=(fe(j)-fe(j-1))/(y(j)-y(j-1))
      b4=h11*gref*feyl/(h21*fe(j)**2)+c251*grfe*h11-aa4s
c     calculation for c26x
      c26(1)=h11/g1*(g12y1-h21*h2x1-g121*h2y1/h21)
      g12=h1(2,j)/((h1(2,j)*h2(2,j))**2-g12(2,j)**2)
      g13=h1(3,j)/((h1(3,j)*h2(3,j))**2-g12(3,j)**2)
      g12y2=derj21(2,j,nx,ny,y,g12)
      g12y3=derj21(3,j,nx,ny,y,g12)
      h2x2=deri21(2,j,nx,ny,x,h2)
      h2x3=deri21(3,j,nx,ny,x,h2)
      h2y2=derj21(2,j,nx,ny,y,h2)
      h2y3=derj21(3,j,nx,ny,y,h2)
      c26(2)=h1(2,j)/g12*(g12y2-h2(2,j)*h2x2-g12(2,j)*h2y2/h2(2,j))
      c26(3)=h1(3,j)/g13*(g12y3-h2(3,j)*h2x3-g12(3,j)*h2y3/h2(3,j))
      x1(1)=x(1)
      x1(2)=x(2)
      x1(3)=x(3)
      c26x1=derk(x1,c26,3,1)
      b5=h11*grfe**2*c26x1
      b3s=b3+aa2s
      b4s=b4+aa4s
      b6=-b3s-b5*g(j,nz)**2-b4s*g(j,nz)
c
      cc1=-aa1s
      cc2=-aa3s
```

```
      cc3=0.0
      cc4=1.0
      c36l=g12l/gl*(g12l*h2yl/h2l**2-g12yl/h2l+h2xl)
      cc5=-aa4s+c36l*h1l*grfe
      cc4s=cc4+aa2s
      cc5s=cc5+aa4s
      cc6=-(cc2+aa3s/2)*ge2y-cc5s*g(j,nz)**2
c
      dd1=-aa1s
      dd2=-aa3s
      dd3=-aa2s
      dd4=-aa4s
      dd6=1./hcon*(1.-pr1)/pr1
      dd7=g12l/(h1l*h2l)
c
      uden=0.0
      vden=1.0
      uden2=uden**2
      vden2=vden**2
      d7uv=dd7*uden*vden
      uvden=uden*vden
      stild=stilde/te(i,j)
c
c     wall temperature as a function of s
c
      if(iwall.eq.1)then
      tws1=tws(i,j)
      tw=tws1/trs
      hw=gam*amach**2*tw
      endif
c
c     ww to be-updated in s/r updte
c
c
      if(j.eq.2)call diag
      return
      end
c-----------------------------------------------------------------------
      subroutine diag
c-----------------------------------------------------------------------
c
c     diagnostic print-out of edge coefficients
c
      include '../lib/com'
  25  format(/1x,20(1h.),'printout from s/r diag',20(1h.))
  26  format(/1x,20(1h.),'end printout from s/r diag',20(1h.)/)
      write(2,25)
  10  format(/' i=',i3/' j=',i3/' x(i)=',g14.7/
     1       ' y(j)=',g14.7/' ue(i,j)=',g14.7/
     1       ' ve(i,j)=',g14.7)
      write(2,10)i,j,x(i),y(j),ue(i,j),ve(i,j)
  20  format(/' ix=',i1,' iy=',i1/' a1=',g14.7/
     1       ' a2=',g14.7,' a3=',g14.7/
     1       ' j1,j2,j3,j4,j5,j6,j7=',7(i2,1x)/
     1       ' bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7=',8(g13.6,1x))
      write(2,20)ix,iy,a1,a2,a3,j1,j2,j3,j4,j5,
     1           j6,j7,bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7
  11  format(/' aa1s=',g14.7/' aa2s=',g14.7/
     1       ' aa3s=',g14.7/' aa4s=',g14.7)
      write(2,11)aa1s,aa2s,aa3s,aa4s
  12  format(/' b1=',g14.7/' b2=',g14.7/' b3=',g14.7/
     1        ' b4=',g14.7/' b5=',g14.7/' b6=',g14.7/
     1        ' b3s=',g14.7/' b4s=',g14.7)
      write(2,12)b1,b2,b3,b4,b5,b6,b3s,b4s
  13  format(/' cc1=',g14.7/' cc2=',g14.7/' cc3=',g14.7/
     1       ' cc4=',g14.7/' cc5=',g14.7/' cc6=',g14.7/
     1       ' cc4s=',g14.7/' cc5s=',g14.7)
      write(2,13)cc1,cc2,cc3,cc4,cc5,cc6,cc4s,cc5s
  14  format(/' dd1=',g14.7/' dd2=',g14.7/' dd3=',g14.7/
     1       ' dd4=',g14.7)
      write(2,14)dd1,dd2,dd3,dd4
  15  format(/' dd6=',g14.7/' dd7=',g14.7/' uden=',g14.7/
     1       ' vden=',g14.7/)
      write(2,15)dd6,dd7,uden,vden
      write(2,26)
      return
      end
c-----------------------------------------------------------------------
      subroutine phys
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      common/phy    /zphys(nz),uphys(nz),tphys(nz),vphys(nz)
c
      i=1
      if(j.eq.1)write(4,20)
      yloc =y(j)
      tel=te(i,j)
      roel=roe(i,j)
      amuel=amue(i,j)
      s1bue=h1(1,j)/fe(j)
      zcon=sqrt(amuel*s1bue*reflen/(roel*rereff))
      do 10 k=1,nz
      if(k.eq.1)then
      thi=0.
      else
      thi=sumint(z,thel,nz,k)
      endif
      zphys(k)=zcon*thi
      vphys(k)=ufs*g(j,k)
      tphys(k)=thel(k)*te(i,j)*trs
  10  continue
c
c     output physical quantites
c
      do 11 ips=1,nppj
      if(j.eq.jp(ips))goto 12
  11  continue
      goto 13
  12  write(3,14)yloc
  14  format(/' profile at y= ',g14.7/
     1' k    z-normal         v            T       '/)
      do 15 k=1,nz
      write(3,16)k,zphys(k),vphys(k),tphys(k)
  15  continue
  13  continue
  16  format(1x,i2,3(1x,g14.7))
c
c     calculate cf based on free stream conditions
c
      gpthw=gp(j,1)*a1(1)
      psi=1./sqrt(roel*amuel*s1bue)
      cfyi=2./sqrt(rereff*reflen)*psi*roel*gpthw*amuel
c
c     calculate zeta value at which g reaches gedge
c
      do 17 kk=1,nz
      k=nz+1-kk
```

```
      if(g(j,k).le.gedge*ve(1,j))then
      kb=k
      kt=k+1
      goto 18
      endif
 17   continue
      print 19
 19   format(' error in bl edge calcualtion')
      stop
 18   continue
      gb=g(j,kb)
      gt=g(j,kt)
      fac=(gedge*ve(1,j)-gb)/(gt-gb)
      thib=sumint(z,the1,nz,kb)
      thit=sumint(z,the1,nz,kt)
      thi=thib+fac*(thit-thib)
      deltay=zcon*thi
c
 20   format(2x,'i',9x,'y',t25,'deltay',t40,'cfyi'
     1       ,t55,'ve',t70,'fe'/)
      write(4,21)i,yloc,deltay,cfyi,ve(1,j),fe(j)
 21   format(1x,i3,5(1x,g14.7))
      return
      end
c-----------------------------------------------------------------
      subroutine updte
c-----------------------------------------------------------------
      include '../lib/com'
      do 10 k=1,nz
      the1(k)=(h(j,k)*hcon-0.5*f(j,k)*f(j,k)*uden2-0.5*g(j,k)**2
     1        *vden2-vden*g(j,k)*f(j,k)*uden*dd7)/(te(i,j)*bet)
      if(the1(k).le.0.00)istop=1
      if(istop.eq.1)return
      the2(k)=(hp(j,k)*hcon-f(j,k)*fp(j,k)*uden2
     1         -g(j,k)*gp(j,k)*vden2-(g(j,k)*fp(j,k)+f(j,k)*
     1         gp(j,k))*dd7*uden*vden)/(te(i,j)*bet)
      thes=sqrt(the1(k))
      al(k)=thes*(1.+stild)/(the1(k)+stild)
      alp(k)=0.5*(1.+stild)*(stild-the1(k))/(the1(k)+stild)**2
     1       *the2(k)/thes
      all(k)=al(k)
      al2(k)=al(k)
      allp(k)=alp(k)
      al2p(k)=alp(k)
      all(k)=al(k)/prl
      allp(k)=alp(k)/prl
 10   continue
c
c     wws and hpw updates
c
      if(i.ne.1)then
      ww=wws(i,j)*sqrt(rereff*reflen)/(ufs*psil*the1(1)*amue(i,j))
      endif
      if(iwall.eq.2.and.i.ne.1)then
      hpw=qd(i,j)/(-amuref*ufs**2*sqrt(rereff/reflen)*bet*
     1     roe(i,j)*amue(i,j)*te(i,j)*psil*all(1))
      hpw=hpw*te(i,j)*bet/hcon
      endif
      return
      end
c-----------------------------------------------------------------
      subroutine conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
c-----------------------------------------------------------------
      include '../lib/com'
c
c     find maximum of delfp(k) absolute
c
      dfpmax=-1.e10
      do 10 k=1,nz
      if(abs(delfp(k)).gt.dfpmax)then
      kmaxm=k
      dfpmax=abs(delfp(k))
      endif
 10   continue
c
c     find maximum of delhp(k) absolute
c
      dhpmax=-1.e10
      kmaxe=1
      do 11 k=1,nz
      if(abs(delhp(k)).gt.dhpmax)then
      kmaxe=k
      dhpmax=abs(delhp(k))
      endif
 11   continue
c
c     print convergence data
c
      print 12,it,dfpmax,dhpmax,kmaxm,kmaxe
      write(2,12)it,dfpmax,dhpmax,kmaxm,kmaxe
 12   format(' iter= ',i2,' dfp max=',e14.7,
     1       ' dhp max=',e14.7,' at (',i2,',',i2,')')
      return
      end
```

```
c     f77 -o case10 case10.f ../lib/blk.o ../lib/lib.o
c     set nx=100,ny=15 nz=41
c     INFINITE SWEPT WING COMPR. BL SOLUTION (INF_BL) FOR SPECIFIED WALL
c     TEMPERATURE OR ADIABATIC WALL CONDITION; FOURTH ORDER ACCURATE IN
c     WALL NORMAL DIRECTION;  PROGRAM BASED ON NASA CR ..../IYER
c
c     Case #10: NACA 0012 swept wing, locally infinite swept wing BL
c-----------------------------------------------------------------------
      program inf
c-----------------------------------------------------------------------
c
      include '../lib/com'
c
      open(unit=1,file='fort.1_case10')
      open(unit=2,file='fort.2_case10')
      open(unit=3,file='fort.3_case10')
      open(unit=4,file='fort.4_case10')
      open(unit=10,file='fort.10_case10')
      open(unit=30,file='../int_wng/fort.20_case10')
      open(unit=20,file='fort.20_case10')
c
c     fort.1 output for plots
c     fort.2 output of ref. qties., conv. history
c     fort.3 output of profiles
c     fort.4 output of global and wall qties. in physical units
c     fort.10 output of profiles for plots
c
      rewind 1
      rewind 2
      rewind 3
      rewind 4
      rewind 30
      rewind 20
c
      call inputs
      call initial
c
      do 10 i=1,nxlim
      g(j,nz)=ve(i,1)
      print 11,i,x(i),ue(i,1),ve(i,j)
      write(2,11)i,x(i),ue(i,1),ve(i,j)
 11   format(' iterating at i=',i3,' x=',g14.7,' ue=',g14.7,
     1       ' ve=',g14.7)
      if(i.eq.1)call edgeatt
      if(i.gt.1)call edgeinf
c
      it=0
 12   it=it+1
      if(it.gt.itmax)goto 998
      call mmntm
      call cntnty
      call enrgy
      call updte
      if(istop.eq.1)goto 999
      call conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
      if(dfpmax.gt.epsf.or.dhpmax.gt.epsh)goto 12
c
      call phys
c
      call nexstep
 10   continue
c
      write(2,13)
      print 13
 13   format(' end of calculation')
      stop
c
 999  print 997
      write(2,997)
 997  format(' the1 is -ve')
      print 996,(k,the1(k),k=1,nz)
      write (2,996)(k,the1(k),k=1,nz)
 996  format(4(i2,g14.7))
      stop
c
 998  write(2,995)
      print 995
 995  format(' itmax has been exceeded; calculation stopped')
      stop
c
      end
c-----------------------------------------------------------------------
      subroutine inputs
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      character*80 title
      data title/' naca 0012 wing j=1 (locally infinite swept)'/
      data ip/5,11,21,31,41,51,61,71,81,91*0/
c
c     iunit=0 --> inputs assumed in U.S. Customary Units (lb, deg R, ft)
c     iunit=1 --> inputs assumed in S.I. Units (kg, deg K, m, Pa)
c
      iunit=0
      amach=0.50
c
      pfs=2116.00
      tfs=520.00
      do 10 i=1,nx
      tws(i,1)=520.
      wws(i,1)=0.
 10   continue
      reflen=1.
      iwall=0
c
      prl=0.72
      nxlim=99
      nylim=1
c
c     number of profile stations;  see data for ip
c
      nppi=9
      ak=1.0
      itmax=30
      epsf=5.e-05
      epsh=5.e-05
      zmax=8.0
      fedge=0.99
      gedge=0.99
      iord=5
c
      write(2,16)title
      print 16,title
 16   format(a80/)
c
      call ref
      write(2,40)nxlim,iwall,iunit,ak,epsf,epsh
 40   format(/' nxlim= ',i4/
```

```
     1        ' iwall= ',i1/
     1        ' iunit= ',i1/
     1        ' ak= ',g14.7/
     1        ' epsf= ',g14.7/
     1        ' epsh= ',g14.7/)
c
c     z distribution (evenly spaced)
c
      if(ak.le.1.0001)then
      dz=zmax/(nz-1)
      do 12 k=1,nz
      z(k)=(k-1)*dz
 12   continue
      else
c
c     z distribution (stretched grid based on ak)
c
      do 13 k=1,nz
      q1=(ak**(k-1)-1)/(ak**(nz-1)-1)
      z(k)=zmax*q1
 13   continue
      endif
c
      print 14,ak
      write(2,14)ak
      print 15,z
      write(2,15)z
 14   format(/' stretching factor ak= ',f10.5/
     1' zeta distribution'/)
 15   format(4(1x,g14.7))
c
      call rinvis
c
      write(2,203)
 203  format(/2(3x,'i ',3x,'x ',4x,'ue',5x,'ve')/)
      write(2,202)(i,x(i),ue(i,1),ve(i,1),i=1,nxlim)
 202  format(3(1x,i3,3(1x,f6.4)))
      return
      end
c-----------------------------------------------------------------------
      subroutine initial
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
c
c     initialize f,g,fp(i.e. f') and gp (g')
c
      zedge=2.5
      accel=1.0
      i=1
      j=1
      do 10 k=1,nz
      zr=z(k)/zedge
      if(zr.gt.1.0)then
      f(j,k)=1.0
      fp(j,k)=0.0
      g(j,k)=ve(1,j)
      gp(j,k)=0.
      else
c
c     initial profile
c
      f(j,k)=1.-(1+zr)*(1-zr)**3+accel/6.*zr*(1-zr)**3
      g(j,k)=f(j,k)*ve(i,j)
      endif
 10   continue
      do 11 k=1,nz
      tem1(k)=f(j,k)
 11   continue
      do 12 k=1,nz
      fp(j,k)=derk(z,tem1,nz,k)
      gp(j,k)=fp(j,k)*ve(1,j)
      w(j,k)=sumint(z,tem1,nz,k)
      w(j,k)=-2*w(j,k)
 12   continue
c
c     specify a h distribution
c
      do 13 k=1,nz
      h(j,k)=htot
      hp(j,k)=0.0
 13   continue
      do 14 k=1,nz
      al(k)=1.0
      alp(k)=0.0
      all(k)=1./prl
      allp(k)=0.0
      al1(k)=al(k)
      al2(k)=al(k)
      al1p(k)=alp(k)
      al2p(k)=alp(k)
      the1(k)=1.
      the2(k)=0.
 14   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine rinvis
c-----------------------------------------------------------------------
      include '../lib/com'
      do 11 i=1,nx
      do 11 j=1,ny
      read(30,*)x(i),y(j),ue(i,j),ve(i,j),h1(i,j),h2(i,j),g12(i,j)
 11   continue
c
      do 3 j=1,ny
      fe(j)=deri21(1,j,nx,ny,x,ue)
c
c     fe(j) by 2-point formula
c
      fe(j)=(ue(2,j)-ue(1,j))/(x(2)-x(1))
 3    continue
      j=1
      do 16 i=1,nx
      te(i,j)=htot/(gam*amach*amach)-(gam-1)/(2*gam)*
     1(ue(i,j)**2+ve(i,j)**2+2*ue(i,j)*ve(i,j)*g12(i,j)/
     1(h1(i,j)*h2(i,j)))
 16   continue
c
      j=1
      do 17 i=1,nx
      amuel=te(i,j)**1.5*(1.+stilde)/(te(i,j)+stilde)
      amue(i,j)=amuel
c
c     roe calculated by isentropic formula
c     if free stream to bl edge is not a isentropic process
c     roe can be calculated if the bl edge pressure is known
c
```

```
      ql=1.+(gam-1.)/(2.*gam*tf)
      qt2=ue(i,j)**2+ve(i,j)**2+2*ue(i,j)*ve(i,j)*g12(i,j)
     1/(h1(i,j)*h2(i,j))
      q2=1.+(gam-1.)*qt2/(2*gam*te(i,j))
      roel=(ql/q2)**(1./(gam-1.))
      roe(i,j)=roel
 17   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine edgeatt
c-----------------------------------------------------------------------
c
c     Locally infinite swept attachment line coefficients
c
      dimension c26(3),xl(3)
      include '../lib/com'
      j=1
      i=1
      ix=0
      a1=0.
      a2=0.
      a3=0.
      iy=0
      bd =0.
      bd1=0
      bd2=0.
      bd3=0.
      bd4=0
      bd5=0
      bd6=0
      bd7=0
      gref=1.
      grfe=gref/fe(j)
c
      aa1s=0.
      aa2s=-1.0
      aa3s=0.
      aa4s=0.0
c
c     LIS att. line
c
      h11=h1(i,j)
      h21=h2(i,j)
      g121=g12(i,j)
      h1x1=deri21(i,j,nx,ny,x,h1)
      h2x1=deri21(i,j,nx,ny,x,h2)
      h1y1=derj21(i,j,nx,ny,y,h1)
      h2y1=derj21(i,j,nx,ny,y,h2)
      g12y1=derj21(i,j,nx,ny,y,g12)
      gl=h11**2*h21**2-g121**2
      glsq=sqrt(gl)
      c121=glsq/h21
      c12n=sqrt(h1(i,j+1)**2*h2(i,j+1)**2-g12(i,j+1)**2)/h2(i,j+1)
      c12y1=(c12n-c121)/(y(2)-y(1))
      aa4s=-h11*c12y1/glsq*grfe
c
      b1=-aa1s
      b2=0.
      b3=2.0
      b4=0
      b5=0.
      b6=-1.
c
c     LIS att. line
c
      cth1=g121/(h11*h21)
      c251=(h11*h21*h1y1*(1+cth1**2)-2*g121*h2x1)/gl
      b4=h11*c251*grfe-aa4s
c     calculation for c26x
      c26(1)=h11/gl*(g12y1-h21*h2x1-g121*h2y1/h21)
      g12=h1(2,j)/((h1(2,j)*h2(2,j))**2-g12(2,j)**2)
      g13=h1(3,j)/((h1(3,j)*h2(3,j))**2-g12(3,j)**2)
      g12y2=derj21(2,j,nx,ny,y,g12)
      g12y3=derj21(3,j,nx,ny,y,g12)
      h2x2=deri21(2,j,nx,ny,x,h2)
      h2x3=deri21(3,j,nx,ny,x,h2)
      h2y2=derj21(2,j,nx,ny,y,h2)
      h2y3=derj21(3,j,nx,ny,y,h2)
      c26(2)=h1(2,j)/gl2*(g12y2-h2(2,j)*h2x2-g12(2,j)*h2y2/h2(2,j))
      c26(3)=h1(3,j)/gl3*(g12y3-h2(3,j)*h2x3-g12(3,j)*h2y3/h2(3,j))
      xl(1)=x(1)
      xl(2)=x(2)
      xl(3)=x(3)
      c26xl=derk(xl,c26,3,1)
      b3s=b3+aa2s
      b4s=b4+aa4s
      b5=h11*c26xl*grfe**2
      b6=-1-b4s*g(j,nz)-b5s*g(j,nz)**2
c
      cc1=0.
      cc2=0.
      cc3=0.
      cc4=1.
      cc5=0
      cc6=0.
c
c     LIS att. line
c
      c361=g121/gl*(g121*h2y1/h21**2-g12y1/h21+h2x1)
      cc5=-aa4s+h11*c361*grfe
      cc6=-h11*c361*g(j,nz)**2
c
      dd1=-aa1s
      dd2=-aa3s
      dd3=-aa2s
      dd4=-aa4s
c
      cc4s=cc4+aa2s
      cc5s=cc5+aa4s
      dd6=1./hcon*(1-pr1)/pr1
      dd7=g121/(h11*h21)
c
      uden=0.
      vden=1.0
c
      uden2=uden**2
      vden2=vden**2
      d7uv=dd7*uden*vden
      uvden=uden*vden
      stild=stilde/te(i,j)
c
c     wall temperature as a function of s
c
      if(iwall.eq.1)then
      tws1=tws(i,j)
      tw=tws1/trs
      hw=gam*amach**2*tw
```

```
      endif
c
c     ww to be-updated in s/r updte
c
      ww=w(j,1)
c
      call diag
      return
      end
c-----------------------------------------------------------------------
      subroutine diag
c-----------------------------------------------------------------------
      include '../lib/com'
 25   format(/1x,20(1h.),'printout from s/r diag',20(1h.))
 26   format(/1x,20(1h.),'end printout from s/r diag',20(1h.)/)
      write(2,25)
 10   format(/' i=',i3/' j=',i3/' x(i)=',g14.7/
     1       ' y(j)=',g14.7/' ue(i,j)=',g14.7/
     1       ' ve(i,j)=',g14.7)
      write(2,10)i,j,x(i),y(j),ue(i,j),ve(i,j)
 20   format(/' ix=',i1,' iy=',i1/' a1=',g14.7/
     1       ' a2=',g14.7,' a3=',g14.7/
     1       ' j1,j2,j3,j4,j5,j6,j7=',7(i2,1x)/
     1       ' bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7=',8(g13.6,1x))
      write(2,20)ix,iy,a1,a2,a3,j1,j2,j3,j4,j5,
     1           j6,j7,bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7
 11   format(/' aa1s=',g14.7/' aa2s=',g14.7/
     1       ' aa3s=',g14.7/' aa4s=',g14.7)
      write(2,11)aa1s,aa2s,aa3s,aa4s
 12   format(/' b1=',g14.7/' b2=',g14.7/' b3=',g14.7/
     1       ' b4=',g14.7/' b5=',g14.7/' b6=',g14.7/
     1       ' b3s=',g14.7/' b4s=',g14.7)
      write(2,12)b1,b2,b3,b4,b5,b6,b3s,b4s
 13   format(/' cc1=',g14.7/' cc2=',g14.7/' cc3=',g14.7/
     1       ' cc4=',g14.7/' cc5=',g14.7/' cc6=',g14.7/
     1       ' cc4s=',g14.7/' cc5s=',g14.7)
      write(2,13)cc1,cc2,cc3,cc4,cc5,cc6,cc4s,cc5s
 14   format(/' dd1=',g14.7/' dd2=',g14.7/' dd3=',g14.7/
     1       ' dd4=',g14.7)
      write(2,14)dd1,dd2,dd3,dd4
 15   format(/' dd6=',g14.7/' dd7=',g14.7/' uden=',g14.7/
     1       ' vden=',g14.7/)
      write(2,15)dd6,dd7,uden,vden
      write(2,26)
      return
      end
c-----------------------------------------------------------------------
      subroutine edgeinf
c-----------------------------------------------------------------------
      include '../lib/com'
      j=1
      iy=0
      ix=1
      dxi=x(i)-x(i-1)
      a1= 1./dxi
      a2=-1./dxi
      a3=0.
      if(i.ge.3)then
      dxi1=x(i-1)-x(i-2)
      r=float(i-2)/(iord-2)
      if(r.ge.1.)r=1.
      om=dxi/dxi1
      q1=(1-r)/dxi
      a1=q1+r*(2*om+1)/((om+1)*dxi)
      a2=-q1-r*(om+1)/dxi
      a3=r*om/((om+1)*dxi1)
      endif
c
      if(iy.eq.1)then
      j1=j-1
      j2=j+1
      j3=j
      j4=j+1
      j5=j+2
      j6=j-1
      j7=j-2
      if(jbl.eq.2)j5=j+1
      if(jbl.eq.1)j7=j-1
      endif
c
      bd =0.
      bd1=0
      bd2=0.
      bd3=0.
      bd4=0
      bd5=0
      bd6=0
      bd7=0
c
      h11=h1(i,j)
      h21=h2(i,j)
      g121=g12(i,j)
      h1x1=deri21(i,j,nx,ny,x,h1)
      h2x1=deri21(i,j,nx,ny,x,h2)
      h1y1=derj21(i,j,nx,ny,y,h1)
      h2y1=derj21(i,j,nx,ny,y,h2)
      g12x1=deri22(i,j,nx,ny,x,g12)
      g12y1=derj21(i,j,nx,ny,y,g12)
      g1=h11**2*h21**2-g121**2
      g1sq=sqrt(g1)
      c241=g121/g1*(g121*h1x1/h11**2+h1y1-g12x1/h11)
      cth1=g121/(h11*h21)
      c251=(h11*h21*h1y1*(1+cth1**2)-2*g121*h2x1)/g1
      c261=h11/g1*(g12y1-h21*h2x1-g121*h2y1/h21)
      c341=h21/g1*(g12x1-h11*h1y1-g121/h11*h1x1)
      c351=1./g1*(h11*h21*h2x1*(1+cth1**2)-2*g121*h1y1)
      c361=g121/g1*(g121*h2y1/h21**2-g12y1/h21+h2x1)
      s1=x(i)
c
      ue1=ue(i,j)
      ph11=sqrt(roe(i,1)*amue(i,1)*s1*ue(i,1))
      s11=x(i-1)
      ph11=sqrt(roe(i-1,1)*amue(i-1,1)*s11*ue(i-1,1))
      ge11=ve(i-1,j)
      ue11=ue(i-1,j)
      g11=h1(i-1,j)**2*h2(i-1,j)**2-g12(i-1,j)**2
      if(i.gt.2)then
      s12=x(i-2)
      ph12=sqrt(roe(i-2,1)*amue(i-2,1)*s12*ue(i-2,1))
      ge12=ve(i-2,j)
      ue12=ue(i-2,j)
      g12=h1(i-2,j)**2*h2(i-2,j)**2-g12(i-2,j)**2
      h1i2=h1(i-2,j)
      else
      ph12=0.
      ge12=0.
      ue12=0.
      g12=0
```

```
      h1i2=1
      endif
      psi1=sqrt(ue(i,1)/(roe(i,1)*amue(i,1)*s1))
      ak1=phi1
      akk1=glsq*phi1/h11
      akx=a1*phi1+a2*phi1+a3*phi2
      akkx=a1*akk1+
     1     a2*phi1*sqrt(g11)/h1(i-1,j)+
     1     a3*phi2*sqrt(g12)/h1i2
      uex=a1*ue(i,j)+a2*uei1+a3*uei2
      gex=a1*g(j,nz)+a2*gei1+a3*gei2
c
      gref=1.0
      grefx=0.
c
      aa1s=-s1/h11
c     aa2s=aa1s*akx/ak1
      aa2s=aa1s*akkx/akk1
c
      aa3s=0.
      aa4s=0.
      c131=glsq
      c132=sqrt(h1(i,j+1)**2*h2(i,j+1)**2-g12(i,j+1)**2)
      c13y=(c132/h2(i,j+1)-c131/h21)/(y(2)-y(1))
      aa4s=-s1*gref*c13y/(c131*ue(i,j))
c
      b1=-aa1s
      b2=0.
      b3=-aa2s+s1*uex/(h11*uel)+c241*s1
      b4=-aa4s+c251*s1*gref/ue(i,j)
      b5=c261*s1*gref**2/ue(i,j)**2
      b3s=b3+aa2s
      b4s=b4+aa4s
      b6=-b3s-b4s*g(j,nz)-b5*g(j,nz)**2
c
      cc1=-aa1s
      cc2=0.
      cc3=c341*s1*ue(i,j)/gref
      cc4=-aa2s+s1*c351
      cc5=-aa4s+c361*s1*gref/ue(i,j)
      cc4s=cc4+aa2s
      cc5s=cc5+aa4s
      cc6=-cc1*gex-cc3-cc4s*g(j,nz)-cc5s*g(j,nz)**2
c
      dd1=-aa1s
      dd2=-aa3s
      dd3=-aa2s
      dd4=-aa4s
      dd6=1./hcon*(1.-pr1)/pr1
      dd7=g121/(h11*h21)
c
      uden=ue(i,j)
      vden=1.0
      uden2=uden**2
      vden2=vden**2
      d7uv=dd7*uden*vden
      uvden=uden*vden
      stild=stilde/te(i,j)
c
c     wall temperature as a function of s
c
      if(iwall.eq.1)then
      tws1=tws(i,j)
      tw=tws1/trs
      hw=gam*amach**2*tw
      endif
c
c     ww to be-updated in s/r updte
c
      ww=w(j,1)
c
      if(i.eq.2)call diag
      return
      end
c-----------------------------------------------------------------------
      subroutine phys
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      common/phy    /zphys(nz),uphys(nz),tphys(nz)
c
      if(i.eq.1)return
      xloc =x(i)
      tel=te(i,j)
      roel=roe(i,j)
      amuel=amue(i,j)
      zcon=sqrt(amuel*s1*reflen/(roel*ue(i,j)*rereff))
      do 10 k=1,nz
      if(k.eq.1)then
      thi=0.
      else
      thi=sumint(z,thel,nz,k)
      endif
      zphys(k)=zcon*thi
      uphys(k)=ufs*f(j,k)*ue(i,j)
      tphys(k)=thel(k)*te(i,j)*trs
 10   continue
      refs=refsf*xloc*reflen
c
c     output physical quantites
c
      do 11 ips=1,nppi
      if(i.eq.ip(ips))goto 12
 11   continue
      goto 13
 12   write(3,14)xloc*reflen,refs
 14   format(/' profile at x = ',g14.7,' local Re no=',g14.7/
     1' k    z-normal         u            T       '/)
      do 15 k=1,nz
      write(3,16)k,zphys(k),uphys(k),g(1,k)*ufs
      write(10,23)z(k),f(1,k),g(1,k)
 15   continue
 13   continue
 16   format(1x,i2,3(1x,g14.7))
 23   format(3(1x,e14.7))
c
c     calculate cf based on free stream conditions
c
      fpthw=fp(j,1)*al1(1)
      psi=phi1/(roel*amuel*s1)
      cfxi=2./sqrt(rereff*reflen)*psi*ue(i,j)*roel*fpthw*amuel
      gpthw=gp(j,1)*al2(1)
      cfyi=2./sqrt(rereff*reflen)*psi*roel*gpthw*amuel
c
c     calculate zeta value at which f reaches fedge
c
      do 17 kk=1,nz
      k=nz+1-kk
```

```
      if(f(1,k).le.fedge)then
      kb=k
      kt=k+1
      goto 18
      endif
 17   continue
      print 19
 19   format(' error in bl edge calcualtion')
      stop
 18   zbc=z(kb)
      zt=z(kt)
      fb=f(j,kb)
      ft=f(j,kt)
      fac=(fedge-fb)/(ft-fb)
      thib=sumint(z,thel,nz,kb)
      thit=sumint(z,thel,nz,kt)
      thi=thib+fac*(thit-thib)
      deltax=zcon*thi
c
c     calculate zeta value at which g reaches gedge
c
      do 517 kk=1,nz
      k=nz+1-kk
      if(g(1,k).le.gedge*ve(i,1))then
      kb=k
      kt=k+1
      goto 518
      endif
 517  continue
      print 19
      stop
 518  zbc=z(kb)
      zt=z(kt)
      fb=g(j,kb)
      ft=g(j,kt)
      fac=(gedge*ve(i,1)-fb)/(ft-fb)
      thib=sumint(z,thel,nz,kb)
      thit=sumint(z,thel,nz,kt)
      thi=thib+fac*(thit-thib)
      deltay=zcon*thi
c
      do 26 k=1,nz
      teml(k)=thel(k)-f(j,k)
 26   continue
      dispx=sumint(z,teml,nz,nz)
      dispx=zcon*dispx
c
 20   format(2x,'i',9x,'x',t25,'deltax',t40,'cfxi'
     1       ,t55,'deltay',t70,'cfy'/)
      if(i.eq.2)write(4,20)
      write(4,21)i,x(i),deltax,cfxi,deltay,cfyi
 21   format(1x,i3,5(1x,g14.7))
c
c     write to unit 22
c
      do 501 k=1,nz
      write(20,503)f(1,k),fp(1,k),g(1,k),gp(1,k),h(1,k),hp(1,k)
 501  continue
      do 502 k=1,nz
      write(20,504)w(1,k),al(k),alp(k),thel(k),the2(k)
 502  continue
      write(20,*)ii,deltax,deltay,cfxi,cfyi
 503  format(6(1x,e14.7))
 504  format(5(1x,e14.7))
      return
      end
c-----------------------------------------------------------------
      subroutine updte
c-----------------------------------------------------------------
      include '../lib/com'
      do 10 k=1,nz
      thel(k)=(h(j,k)*hcon-0.5*f(j,k)*f(j,k)*uden2-0.5*g(j,k)**2
     1        *vden2-vden*g(j,k)*f(j,k)*uden*dd7)/(te(i,j)*bet)
      if(thel(k).le.0.00)istop=1
      if(istop.eq.1)return
      the2(k)=(hp(j,k)*hcon-f(j,k)*fp(j,k)*uden2
     1         -g(j,k)*gp(j,k)*vden2-(g(j,k)*fp(j,k)+f(j,k)*
     1         gp(j,k))*dd7*uden*vden)/(te(i,j)*bet)
      thes=sqrt(thel(k))
      al(k)=thes*(1.+stild)/(thel(k)+stild)
      alp(k)=0.5*(1.+stild)*(stild-thel(k))/(thel(k)+stild)**2
     1       *the2(k)/thes
      all(k)=al(k)
      al2(k)=al(k)
      allp(k)=alp(k)
      al2p(k)=alp(k)
      all(k)=al(k)/prl
      allp(k)=alp(k)/prl
 10   continue
c
c     wall mass transfer update
c     and heat flux update not included
c
      return
      end
c-----------------------------------------------------------------
      subroutine conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
c-----------------------------------------------------------------
      include '../lib/com'
c
c     find maximum of delfp(k) absolute
c
      dfpmax=-1.e10
      do 10 k=1,nz
      if(abs(delfp(k)).gt.dfpmax)then
      kmaxm=k
      dfpmax=abs(delfp(k))
      endif
 10   continue
c
c     find maximum of delhp(k) absolute
c
      dhpmax=-1.e10
      kmaxe=1
      do 11 k=1,nz
      if(abs(delhp(k)).gt.dhpmax)then
      kmaxe=k
      dhpmax=abs(delhp(k))
      endif
 11   continue
c
c     print convergence data
c
      print 12,it,dfpmax,dhpmax,kmaxm,kmaxe
      write(2,12)it,dfpmax,dhpmax,kmaxm,kmaxe
 12   format(' iter= ',i2,' dfp max=',e14.7,
     1       ' dhp max=',e14.7,' at (',i2,',',i2,')')
      return
```

```
      end
c-----------------------------------------------------------------------
      subroutine nexstep
c-----------------------------------------------------------------------
      include '../lib/com'
      do 2 k=1,nz
      f2(1,k)=f1(1,k)
      g2(1,k)=g1(1,k)
      fp2(1,k)=fp1(1,k)
      gp2(1,k)=gp1(1,k)
      hh2(1,k)=hh1(1,k)
      hp2(1,k)=hp1(1,k)
 2    continue
      do 1 k=1,nz
      f1(1,k)=f(1,k)
      g1(1,k)=g(1,k)
      fp1(1,k)=fp(1,k)
      gp1(1,k)=gp(1,k)
      hh1(1,k)=h(1,k)
      hp1(1,k)=hp(1,k)
 1    continue
      end
```

```
c     f77  -o case10 case10.f ../lib/lib.o ../lib/blk.o
c     dimensions: nx=100, ny=15, nz=21
c
c     COMPRESSIBLE BL SOLUTION FOR SPECIFIED WALL TEMPERATURE
c     HEAT FLUX   OR ADIABATIC WALL CONDITION; FOURTH ORDER ACCURATE  IN
c     WALL NORMAL DIRECTION;  PROGRAM BASED ON NASA CR ..../IYER
c
c     Boundary Layer Calculation on Swept Wing, Case 10
c
c     INPUTS ARE:
c
c     ../int_wng/fort.20_case10 : Bl edge (suraface Euler) conditions
c     ../att_bl/fort.20_case10  : attachment line solution
c     ../inf_bl/fort.20_case10  : side boundary solution
c-----------------------------------------------------------------------
      program case10
c-----------------------------------------------------------------------
c
      include '../lib/com'
c
      open(unit=2,file='fort.2_case10')
      open(unit=3,file='fort.3_case10')
      open(unit=4,file='fort.4_case10')
      open(unit=10,file='fort.10_case10')
      open(unit=11,file='fort.11_case10')
      open(unit=50,file='fort.50_case10')
      rewind 50
      open(unit=20,file='../int_wng/fort.20_case10')
      open(unit=21,file='../att_bl/fort.20_case10')
      open(unit=22,file='../inf_bl/fort.20_case10')

c
c     fort.2  echo print for all inputs ; reference qties.
c     fort.3  output of profiles
c     fort.4  output of global and wall qties. in physical units
c     fort.10 output of profiles for plots
c
      rewind 2
      rewind 3
      rewind 4
      rewind 10
      rewind 20
      rewind 21
      rewind 22
c
      call inputs
      call initial
c
      do 50 i=2,nxlim
c
c     read side boundary
c
      call rside
      do 10 j=2,15
      print 11,i,j,x(i),y(j),ue(i,j),ve(i,j)
      write(2,11)i,j,x(i),y(j),ue(i,j),ve(i,j)
 11   format(' iterating at i=',i3,' j=',i3,' x=',g14.7,' y=',g14.7,
     1       ' ue=',g14.7,' ve=',g14.7)
      call edge3d
c
      it=0
 12   it=it+1
      if(it.gt.itmax)goto 998
      call mmntm
      call cntnty
      call enrgy
      call updte
      if(istop.eq.1)goto 999
      call conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
      if(dfpmax.gt.epsf.or.dhpmax.gt.epsh)goto 12
c
      call phys
c
 10   continue
      call nexstep
 50   continue
c
      write(2,13)
      print 13
 13   format(' end of calculation')
      stop
c
 999  print 997
      write(2,997)
 997  format(' thel is -ve')
      print 996,(k,thel(k),k=1,nz)
      write (2,996)(k,thel(k),k=1,nz)
 996  format(4(i2,g14.7))
      stop
c
 998  write(2,995)
      print 995
 995  format(' itmax has been exceeded; calculation stopped')
      stop
c
      end
c-----------------------------------------------------------------------
      subroutine inputs
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      character*80 title
      data title/' NACA 0012 straight wing, 20 deg swept, al=2'/
c
c     i and j stations at which profile output is required
c
      data ip/5,11,21,41,51,61,71,81,91,91*0/
      data jp/5,10,15,97*0/
c
c     iunit=0 --> inputs assumed in U.S. Customary Units (lb, deg R, ft)
c     iunit=1 --> inputs assumed in S.I. Units (kg, deg K, m, Pa)
c
      nxlim=10
      nylim=15
      iunit=0
      amach=0.5
c
      pfs=2116.0
      tfs=520.0
      reflen=1.0
c
c     iwall=0 adiabatic wall
c     iwall=1 wall temperature specified
c     iwall=2 wall heat flux specified as non-zero value
c
      iwall=0
c
      prl=0.72
```

```
c
c     number of profile stations;  see data for ip
c
      nppi=7
      nppj=3
      ak=1.00
      itmax=100
      epsf=5.e-05
      epsh=5.e-05
      zmax=8.0
      fedge=0.99
      gedge=0.99
      iord=5
c
      write(2,16)title
      print 16,title
 16   format(a80/)
c
      call ref
      write(2,40)nxlim,nylim,iwall,iunit,ak,epsf,epsh,iord
 40   format(/' nxlim= ',i4/
     1        ' nylim= ',i4/
     1        ' iwall= ',i1/
     1        ' iunit= ',i1/
     1        ' ak= ',g14.7/
     1        ' epsf= ',g14.7/
     1        ' epsh= ',g14.7/
     1        ' iord= ',i2/)
c
c     z distribution (evenly spaced)
c
      if(ak.le.1.0001)then
      dz=zmax/(nz-1)
      do 12 k=1,nz
      z(k)=(k-1)*dz
 12   continue
      else
c
c     z distribution (stretched grid based on ak)
c
      do 13 k=1,nz
      q1=(ak**(k-1)-1)/(ak**(nz-1)-1)
      z(k)=zmax*q1
 13   continue
      endif
c
      print 14,ak
      write(2,14)ak
      print 15,z
      write(2,15)z
 14   format(/' stretching factor ak= ',f10.5/
     1' zeta distribution'/)
 15   format(5(1x,g14.7))
c
      call rinvis
c
      write(2,200)
      write(2,201)(x(i),i=1,nxlim)
      write(2,202)
      write(2,201)(y(j),j=1,nylim)
 200  format(/' x stations'/)
 201  format(6(1x,f10.5))
 202  format(/' y stations'/)
      return
      end
c-----------------------------------------------------------------------
      subroutine initial
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
c
c     read attachment line profile
c
      do 20 j=1,ny
      do 10 k=1,nz
      read(21,11)f(j,k),fp(j,k),g(j,k),gp(j,k),h(j,k),hp(j,k)
 10   continue
      do 16 k=1,nz
      read(21,12)w(j,k),al(k),alp(k),thel(k),the2(k)
 16   continue
 20   continue
c
      do 15 j=1,ny
      do 15 k=1,nz
      f1(j,k)=f(j,k)
      g1(j,k)=g(j,k)
      fp1(j,k)=fp(j,k)
      gp1(j,k)=gp(j,k)
      hh1(j,k)=h(j,k)
      hp1(j,k)=hp(j,k)
 15   continue
 11   format(6(1x,e14.7))
 12   format(5(1x,e14.7))
      do 14 k=1,nz
      all(k)=al(k)/pr1
      allp(k)=alp(k)/pr1
      al1(k)=al(k)
      al2(k)=al(k)
      allp(k)=alp(k)
      al2p(k)=alp(k)
 14   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine rinvis
c-----------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      do 1 i=1,nx
      do 1 j=1,ny
      read(20,2)x(i),y(j),ue(i,j),ve(i,j),h1(i,j),h2(i,j),g12(i,j)
 1    continue
 2    format(7(1x,e14.7))
c
c     calculate te assuming constant total enthalpy
c
      do 16 j=1,nylim
      do 16 i=1,nx
      te(i,j)=htot/(gam*amach*amach)-(gam-1)/(2*gam)*
     1(ue(i,j)**2+ve(i,j)**2+2*ue(i,j)*ve(i,j)*g12(i,j)/
     1(h1(i,j)*h2(i,j)))
 16   continue
c
      do 17 j=1,nylim
      do 17 i=1,nx
      amuel=te(i,j)**1.5*(1.+stilde)/(te(i,j)+stilde)
      amue(i,j)=amuel
c
```

```
c     roe calculated by isentropic formula
c     if free stream to bl edge is not a isentropic process
c     roe can be calculated if the bl edge pressure is known
c
      q1=1.+(gam-1.)/(2.*gam*tf)
      qt2=ue(i,j)**2+ve(i,j)**2+2*ue(i,j)*ve(i,j)*g12(i,j)/
     1    (h1(i,j)*h2(i,j))
      q2=1.+(gam-1.)*qt2/(2*gam*te(i,j))
      roe1=(q1/q2)**(1./(gam-1.))
      roe(i,j)=roe1
 17   continue
      return
      end
c-----------------------------------------------------------------------
      subroutine rside
c-----------------------------------------------------------------------
c
c     read the j=1 side boundary profiles
c
      include '../lib/com'
      common/dsds   /delxs(nx),delys(nx),cfxs(nx),cfys(nx)
      do 1 k=1,nz
      read(22,3)f(1,k),fp(1,k),g(1,k),gp(1,k),h(1,k),hp(1,k)
 1    continue
      do 2 k=1,nz
      read(22,4)w(1,k),al(k),alp(k),the1(k),the2(k)
 2    continue
      read(22,*)ii,delxs(i),delys(i),cfxs(i),cfys(i)
 3    format(6(1x,e14.7))
 4    format(5(1x,e14.7))
      return
      end
c-----------------------------------------------------------------------
      subroutine edge3d
c-----------------------------------------------------------------------
      include '../lib/com'
c     calculate bl equation coefficients arising from edge conditions
c     general 3d case; interior point (i,j)
c
c     redefine x(i) dep on j
c
      g(j,nz)=ve(i,j)
      iy=1
      ix=1
      dxi=x(i)-x(i-1)
      a1= 1./dxi
      a2=-1./dxi
      a3=0.
      if(i.ge.3)then
      dxi1=x(i-1)-x(i-2)
      r=float(i-2)/(iord-2)
      if(r.ge.1.)r=1.
      om=dxi/dxi1
      q1=(1-r)/dxi
      a1=q1+r*(2*om+1)/((om+1)*dxi)
      a2=-q1-r*(om+1)/dxi
      a3=r*om/((om+1)*dxi1)
      endif
c
      if(iy.eq.1)then
      detaj=y(j)-y(j-1)
      detaj1=y(j+1)-y(j)
      jbl=0
      if(j.eq.2)jbl=1
      if(j.eq.nylim-1)jbl=2
      if(j.eq.nylim)jbl=3
      j1=j-1
      j2=j+1
      j3=j
      j4=j+1
      j5=j+2
      j6=j-1
      j7=j-2
      if(jbl.eq.2)j5=j+1
      if(jbl.eq.1)j7=j-1
      if(jbl.eq.3)then
      j4=j
      j2=j
      j5=j4
      j7=j6
      endif
      if(jbl.ne.3)then
c
c     'Z' scheme coefficients
c
      bd =0.5/detaj
      bd1=-bd
      bd2=0.
      bd3=-0.5/detaj1
      bd4=-bd3
      bd5=0
      bd6=0
      bd7=0
c
c     plain left differencing
c
c     bd=1./detaj
c     bd1=-bd
c     bd2=0.
c     bd3=0.
c     bd4=0
c     bd6=0.
c     bd7=0.
      else
c
c     plain left differencing
c
      bd=1./detaj
      bd1=-bd
      bd3=0.
      bd4=0.
      bd5=0.
      bd6=0.
      bd7=0.
      endif
      endif
c
      h11=h1(i,j)
      h21=h2(i,j)
      g121=g12(i,j)
      g1=(h11*h21)**2-g121**2
      g1sq=sqrt(g1)
      cth1=g121/(h11*h21)
      h1x1=deri21(i,j,nx,ny,x,h1)
      h1y1=derj21(i,j,nx,ny,y,h1)
      h2x1=deri21(i,j,nx,ny,x,h2)
      h2y1=derj21(i,j,nx,ny,y,h2)
      g12x1=deri21(i,j,nx,ny,x,g12)
```

```
      g12y1=derj21(i,j,nx,ny,y,g12)
      c111=glsq/h11
      s11=sumx21(i,j,nx,ny,x,h1)
c
c     for wing grid
c
      s11=x(i)
      s1=s11
      phi1=sqrt(roe(i,j)*amue(i,j)*s11*ue(i,j))
      psi1=phi1/(roe(i,j)*amue(i,j)*s11)
      ak1=sqrt(roe(i,j)*amue(i,j)*ue(i,j))
      if(i.eq.2)then
      uei2=0.
      vei2=0.
      ak12=0.
      c112=0.
      else
      ak12=sqrt(roe(i-2,j)*amue(i-2,j)*ue(i-2,j))
      uei2=ue(i-2,j)
      vei2=ve(i-2,j)
      c112=sqrt(h1(i-2,j)**2*h2(i-2,j)**2-g12(i-2,j)**2)/h1(i-2,j)
      endif
c
      uei1=ue(i-1,j)
      vei1=ve(i-1,j)
      ak11=sqrt(roe(i-1,j)*amue(i-1,j)*ue(i-1,j))
      c111=sqrt(h1(i-1,j)**2*h2(i-1,j)**2-g12(i-1,j)**2)/h1(i-1,j)
c
      uex=a1*ue(i,j)+a2*uei1+a3*uei2
      vex=a1*ve(i,j)+a2*vei1+a3*vei2
      akx=a1*ak1+a2*ak11+a3*ak12
      c11x=a1*c111+a2*c111+a3*c112
      uey=bd*ue(i,j)+bd1*ue(i,j1)+bd2*ue(i,j2)+
     1    bd3*ue(i-1,j3)+bd4*ue(i-1,j4)+bd5*ue(i-1,j5)+
     1    bd6*ue(i-1,j6)+bd7*ue(i-1,j7)
      vey1=bd*ve(i,j)+bd1*ve(i,j1)+bd2*ve(i,j2)+
     1    bd3*ve(i-1,j3)+bd4*ve(i-1,j4)+bd5*ve(i-1,j5)+
     1    bd6*ve(i-1,j6)+bd7*ve(i-1,j7)
      ve2y=2*ve(i,j)*vey1
      bdi=1./(y(j)-y(j1))
      bdi1=-bdi
      c13fy=bdi*glsq*phi1/(h21*ue(i,j))+
     1      bdi1*sqrt(h1(i,j-1)**2*h2(i,j-1)**2-g12(i-1,j)**2)*
     1      sqrt(roe(i,j-1)*amue(i,j-1)*ue(i,j-1)*
     1      x(i))/(h2(i,j-1)*ue(i,j-1))
c
      c241=g121/gl*(g121*h1x1/h11**2+h1y1-g12x1/h11)
      c251=1./gl*(h11*h21*(1+cth1**2)*h1y1-2*g121*h2x1)
      c261=h11/gl*(g12y1-h21*h2x1-g121*h2y1/h21)
      c341=h21/gl*(g12x1-h11*h1y1-g121/h11*h1x1)
      c351=1./gl*(h11*h21*h2x1*(1+cth1**2)-2*g121*h1y1)
      c361=g121/gl*(g121*h2y1/h21**2-g12y1/h21+h2x1)
c
      gref=1.0
      grefx=0.
      grefy=0.
c
      aa1s=-s11/h11
      aa2s=aa1s*c11x/c111+aa1s*akx/(ak1*c111)-0.5/(h11*c111)
      aa3s=-s11*gref/(ue(i,j)*h21)
      aa4s=c13fy*aa1s/(c111*phi1)
c
      b1=-aa1s
      b2=-aa3s
      b3=s11*uex/(h11*ue(i,j))+c241*s11-aa2s
      b4=s11*gref/(h21*ue(i,j)**2)*uey+c251*s11*gref/ue(i,j)-aa4s
      b5=c261*s11*gref**2/(ue(i,j)**2)
c
      cc1=-aa1s
      cc2=-aa3s
      cc3=c341*s11*ue(i,j)/gref
      cc4=c351*s11-aa2s-aa1s/gref*grefx
      cc5=c361*s11*gref/ue(i,j)-aa4s+s11/(h21*ue(i,j))*grefy
c
      b6=-g(j,nz)*(b4+aa4s)-b3-aa2s-b5*g(j,nz)**2-(b2+aa3s)*vey1
      cc6=-(cc5-aa4s)*g(j,nz)**2-(aa2s+cc4)*g(j,nz)-cc3-cc1*vex
     1    -(cc2-aa3s/2)*ve2y
      b3s=b3+aa2s
      b4s=b4+aa4s
      cc4s=cc4+aa2s
      cc5s=cc5+aa4s
c
      dd1=-aa1s
      dd2=-aa3s
      dd3=-aa2s
      dd4=-aa4s
c
      dd6=1./hcon*(1.-pr1)/pr1
      dd7=g121/(h11*h21)
c
      uden=ue(i,j)
      vden=1.0
      uden2=uden**2
      vden2=vden**2
      d7uv=dd7*uden*vden
      uvden=uden*vden
      stild=stilde/te(i,j)
c
c     wall temperature as a function of s
c
      if(iwall.eq.1)then
      tws1=tws(i,j)
      tw=tws1/trs
      hw=gam*amach**2*tw
      endif
c
c     ww to be-updated in s/r updte
c
      ww=w(j,1)
c
      if(i.eq.2.and.j.eq.2)call diag
      return
      end
c-----------------------------------------------------------------------
      subroutine diag
c-----------------------------------------------------------------------
      include '../lib/com'
 25   format(/1x,20(1h.),'printout from s/r diag',20(1h.))
 26   format(/1x,20(1h.),'end printout from s/r diag',20(1h.)/)
      write(2,25)
 10   format(/' i=',i3/' j=',i3/' x(i)=',g14.7/
     1        ' y(j)=',g14.7/' ue(i,j)=',g14.7/
     1        ' ve(i,j)=',g14.7)
      write(2,10)i,j,x(i),y(j),ue(i,j),ve(i,j)
 20   format(/' ix=',i1,' iy=',i1/' a1=',g14.7/
     1        ' a2=',g14.7,' a3=',g14.7/
     1        ' j1,j2,j3,j4,j5,j6,j7=',7(i2,1x)/
     1        ' bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7=',8(g13.6,1x))
```

```
      write(2,20)ix,iy,a1,a2,a3,j1,j2,j3,j4,j5,
     1           j6,j7,bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7
 11   format(/' aa1s=',g14.7/' aa2s=',g14.7/
     1       ' aa3s=',g14.7/' aa4s=',g14.7)
      write(2,11)aa1s,aa2s,aa3s,aa4s
 12   format(/' b1=',g14.7/' b2=',g14.7/' b3=',g14.7/
     1        ' b4=',g14.7/' b5=',g14.7/' b6=',g14.7/
     1        ' b3s=',g14.7/' b4s=',g14.7)
      write(2,12)b1,b2,b3,b4,b5,b6,b3s,b4s
 13   format(/' cc1=',g14.7/' cc2=',g14.7/' cc3=',g14.7/
     1        ' cc4=',g14.7/' cc5=',g14.7/' cc6=',g14.7/
     1        ' cc4s=',g14.7/' cc5s=',g14.7)
      write(2,13)cc1,cc2,cc3,cc4,cc5,cc6,cc4s,cc5s
 14   format(/' dd1=',g14.7/' dd2=',g14.7/' dd3=',g14.7/
     1        ' dd4=',g14.7)
      write(2,14)dd1,dd2,dd3,dd4
 15   format(/' dd6=',g14.7/' dd7=',g14.7/' uden=',g14.7/
     1        ' vden=',g14.7/)
      write(2,15)dd6,dd7,uden,vden
      write(2,26)
      return
      end
c-------------------------------------------------------------------------
      subroutine phys
c-------------------------------------------------------------------------
      include '../lib/com'
      include '../lib/comtemp'
      common/dsds   /delxs(nx),delys(nx),cfxs(nx),cfys(nx)
      common/phy    /zphys(nz),uphys(nz),tphys(nz)
c
      xloc=x(i)
      tel=te(i,j)
      roel=roe(i,j)
      amuel=amue(i,j)
      zcon=sqrt(amuel*s1*reflen/(roel*ue(i,j)*rereff))
      do 10 k=1,nz
      if(k.eq.1)then
      thi=0.
      else
      thi=sumint(z,thel,nz,k)
      endif
      zphys(k)=zcon*thi
      uphys(k)=ufs*f(j,k)*ue(i,j)
      tphys(k)=thel(k)*te(i,j)*trs
 10   continue
      refs=refsf*xloc*reflen
c
c     output physical quantites
c
      do 11 ips=1,nppi
      do 11 jps=1,nppj
      if(i.eq.ip(ips).and.j.eq.jp(jps))goto 12
 11   continue
      goto 13
 12   write(3,14)xloc*reflen,y(j)*reflen,refs
 14   format(/' profile at x = ',g14.7,
     1' j = ',g14.7,
     1' local Re no=',g14.7/
     1' k    z-normal         u            v       '/)
      do 15 k=1,nz
      write(3,16)k,zphys(k),uphys(k),g(j,k)*ufs
 15   continue
 13   continue
 16   format(1x,i2,3(1x,g14.7))
 23   format(3(1x,e14.7))
c
c     calculate cf based on free stream conditions
c
      fpthw=fp(j,1)*al1(1)
      gpthw=gp(j,1)*al2(1)
      psi=phi1/(roel*amuel*s1)
      cfxi=2./sqrt(rereff*reflen)*psi*ue(i,j)*roel*fpthw*amuel
      cfyi=2./sqrt(rereff*reflen)*psi*roel*gpthw*amuel
c
c     calculate zeta value at which f reaches fedge
c
      do 17 kk=1,nz
      k=nz+1-kk
      if(f(j,k).le.fedge)then
      kb=k
      kt=k+1
      goto 18
      endif
 17   continue
      print 19
 19   format(' error in bl edge calcualtion')
      stop
 18   continue
      fb=f(j,kb)
      ft=f(j,kt)
      fac=(fedge-fb)/(ft-fb)
      thib=sumint(z,thel,nz,kb)
      thit=sumint(z,thel,nz,kt)
      thi=thib+fac*(thit-thib)
      deltax=zcon*thi
c
c     calculate zeta value at which g reaches gedge
c
      do 517 kk=1,nz
      k=nz+1-kk
      if(g(1,k).le.gedge*ve(i,1))then
      kb=k
      kt=k+1
      goto 518
      endif
 517  continue
      print 19
      stop
 518  zbc=z(kb)
      zt=z(kt)
      fb=g(j,kb)
      ft=g(j,kt)
      fac=(gedge*ve(i,1)-fb)/(ft-fb)
      thib=sumint(z,thel,nz,kb)
      thit=sumint(z,thel,nz,kt)
      thi=thib+fac*(thit-thib)
      deltay=zcon*thi
      do 26 k=1,nz
      teml(k)=thel(k)-f(j,k)
 26   continue
      dispx=sumint(z,teml,nz,nz)
      dispx=zcon*dispx
c
c     momentum thickness calculation
c
      do 27 k=1,nz
      teml(k)=f(j,k)*(1.-f(j,k))
 27   continue
```

```
      thetax=sumint(z,teml,nz,nz)
      thetax=zcon*thetax
c
c     heat flux calculation
c
      qdot=-amuref*ufs**2*sqrt(rereff/reflen)*bet*
     1     roe(i,j)*amue(i,j)*te(i,j)*psil*all(1)*the2(1)
c
      if(j.eq.2)then
      write(4,21)i,j-1,xloc,y(1),delxs(i),cfxs(i),delys(i),
     1           cfys(i)
      endif
      write(4,21)i,j,xloc,y(j),deltax,cfxi,deltay,cfyi
 21   format(2(1x,i3),6(1x,g14.7))
      return
      end
c-----------------------------------------------------------------
      subroutine updte
c-----------------------------------------------------------------
      include '../lib/com'
      do 10 k=1,nz
      thel(k)=(h(j,k)*hcon-0.5*f(j,k)*f(j,k)*uden2-0.5*g(j,k)**2
     1        *vden2-vden*g(j,k)*f(j,k)*uden*dd7)/(te(i,j)*bet)
      if(thel(k).le.0.00)istop=1
      if(istop.eq.1)return
      the2(k)=(hp(j,k)*hcon-f(j,k)*fp(j,k)*uden2
     1         -g(j,k)*gp(j,k)*vden2-(g(j,k)*fp(j,k)+f(j,k)*
     1         gp(j,k))*dd7*uden*vden)/(te(i,j)*bet)
      thes=sqrt(thel(k))
      al(k)=thes*(1.+stild)/(thel(k)+stild)
      alp(k)=0.5*(1.+stild)*(stild-thel(k))/(thel(k)+stild)**2
     1       *the2(k)/thes
      all(k)=al(k)
      al2(k)=al(k)
      allp(k)=alp(k)
      al2p(k)=alp(k)
      all(k)=al(k)/prl
      allp(k)=alp(k)/prl
 10   continue
c
c     wall mass transfer update
c
      slbue=0.
      if(i.gt.1)slbue=sl/ue(i,j)
      ww=wws(i,j)*thel(1)*sqrt(rereff*reflen*slbue*roe(i,j)/
     1                 amue(i,j))/ufs
      if(iwall.eq.2.and.i.ne.1)then
      hpw=qd(i,j)/(-amuref*ufs**2*sqrt(rereff/reflen)*bet*
     1     roe(i,j)*amue(i,j)*te(i,j)*psil*all(1))
      hpw=hpw*te(i,j)*bet/hcon
      endif
      return
      end
c-----------------------------------------------------------------
      subroutine conv(it,dfpmax,dhpmax,kmaxm,kmaxe)
c-----------------------------------------------------------------
      include '../lib/com'
c
c     find maximum of delfp(k) absolute
c
      dfpmax=-1.e10
      do 10 k=1,nz
      if(abs(delfp(k)).gt.dfpmax)then
      kmaxm=k
      dfpmax=abs(delfp(k))
      endif
 10   continue
c
c     find maximum of delhp(k) absolute
c
      dhpmax=-1.e10
      kmaxe=1
      do 11 k=1,nz
      if(abs(delhp(k)).gt.dhpmax)then
      kmaxe=k
      dhpmax=abs(delhp(k))
      endif
 11   continue
c
c     find |f' max| and |h' max| in the profile
c
c     hpmax=-1.e10
c     fpmax=-1.e10
c     do 20 k=1,nz
c     if(abs(hp(j,k)).gt.hpmax)then
c     hpmax=abs(hp(j,k))
c     endif
c     if(abs(fp(j,k)).gt.fpmax)then
c     fpmax=abs(fp(j,k))
c     endif
c20   continue
c     dhpmax=dhpmax/hpmax
c     dfpmax=dfpmax/fpmax
c
c     print convergence data
c
      print 12,it,dfpmax,dhpmax,kmaxm,kmaxe
      write(2,12)it,dfpmax,dhpmax,kmaxm,kmaxe
 12   format(' iter= ',i2,' dfp max=',e14.7,
     1       ' dhp max=',e14.7,' at (',i2,',',i2,')')
      return
      end
c-----------------------------------------------------------------
      subroutine nexstep
c-----------------------------------------------------------------
      include '../lib/com'
      do 2 j=1,nylim
      do 2 k=1,nz
      f2(j,k)=f1(j,k)
      g2(j,k)=g1(j,k)
      fp2(j,k)=fp1(j,k)
      gp2(j,k)=gp1(j,k)
      hh2(j,k)=hh1(j,k)
      hp2(j,k)=hp1(j,k)
 2    continue
      do 1 j=1,nylim
      do 1 k=1,nz
      f1(j,k)=f(j,k)
      g1(j,k)=g(j,k)
      fp1(j,k)=fp(j,k)
      gp1(j,k)=gp(j,k)
      hh1(j,k)=h(j,k)
      hp1(j,k)=hp(j,k)
 1    continue
      end
```

```
c     f77 -c blk.f
c-----------------------------------------------------------------------
      subroutine ref
c-----------------------------------------------------------------------
      include 'com'
      if(iunit.eq.0)then
      rstar=1716.
      amustd=2.27e-08
      sstar=198.6
      write(2,10)
 10   format(' Physical quantities are in U.S. units'/)
      endif
      if(iunit.eq.1)then
      rstar=287.05925
      amustd=1.4581972e-6
      sstar=110.333
      write(2,11)
 11   format(' Physical quantities are in S.I. units'/)
      endif
c
      if(iwall.eq.0)then
      write(2,12)
      print 12
 12   format(' Wall condition : Adiabatic wall'/)
      endif
      if(iwall.eq.1)then
      write(2,13)
      print 13
 13   format(' Wall condition : Temperature specified'/)
      endif
      if(iwall.eq.2)then
      write(2,15)
      print 15
 15   format(' Wall condition : Wall heat flux specified'/)
      endif
      pi=3.1415927
      c1=0.5
      c2=0.5
      c3=-1./12
      c4= 1./12.
      nz1=nz-1
      istop=0
      gam=1.4
      bet=gam/(gam-1.)
      velson=(gam*rstar*tfs)**0.5
      ufs=amach*velson
      trs=ufs**2/rstar
      stilde=sstar/trs
      tf=tfs/trs
      rofs=pfs/(tfs*rstar)
      amuref=amustd*trs**1.5/(trs+sstar)
      amuinf=amustd*tfs**1.5/(tfs+sstar)
      amui=amuinf/amuref
      rereff=rofs*amach*velson/amuref
      refsf=rofs*amach*velson/amuinf
      hcon=1./((gam-1.)*amach*amach)
      hrefs=ufs*ufs*hcon
      htot=1.+(gam-1.)*0.5*amach*amach
      write(2,14) nx,ny,nz,zmax,reflen,pfs,tfs,amach,trs,ufs,rereff
     1,refsf,velson,amuinf,amuref,rofs,hrefs,fedge
 14   format(' Inputs'//
     1 ' nx, ny, nz = ',2(i3,', '),i3/
     1 ' zmax = ',f7.2/
     1 ' reflen = ',g14.7/
     1 ' pfs = ',g14.7/
     1 ' tfs= ',f10.2/' mach no = ',f10.3/
     1 ' trs= ',f10.2/' ufs = ',f10.2/
     1 ' ref Re no per unit length = ',e14.7/
     1 ' fs  Re no per unit length = ',e14.7/
     1 ' fs sonic vel = ',f10.2/
     1 ' amuinf = ',e14.7/
     1 ' amuref = ',e14.7/
     1 ' rofs = ',e14.7/
     1 ' hrefs = ',e14.7/
     1 ' f-edge = ',f10.6/)
      return
      end
c-----------------------------------------------------------------------
      subroutine mmntm
c-----------------------------------------------------------------------
      include 'com'
      include 'comloc'
      dimension      a(4,4,nz),b(4,4,nz),c(4,4,nz),r(4,nz),
     1               ir(4,nz),ic(4,nz)
c
c     set up system for solving the momentum equations (3d bl)
c
c     initialize all coefft.s to zero
c
      do 1 k=1,nz
      do 1 l=1,4
      r(l,k)=0.
      do 1 m=1,4
      a(l,m,k)=0.
      b(l,m,k)=0.
      c(l,m,k)=0.
 1    continue
c
c     k=1
c
      k=1
      call getfg(k)
      bfk =bd*fjk +bd1*fj1k +bd2*fj2k +bd3*fj3k +bd4*fj4k +
     1     bd5*fj5k +bd6*fj6k +bd7*fj7k
      bfk2=bd*fjk2+bd1*fj1k2+bd2*fj2k2+bd3*fj3k2+bd4*fj4k2+
     1     bd5*fj5k2+bd6*fj6k2+bd7*fj7k2
      bggk =bd*gjk **2+bd1*gj1k **2+bd2*gj2k **2+
     1     bd3*gj3k **2+bd4*gj4k **2+
     1     bd5*gj5k **2+bd6*gj6k **2+bd7*gj7k **2
      bggk2=bd*gjk2**2+bd1*gj1k2**2+bd2*gj2k2**2+
     1     bd3*gj3k2**2+bd4*gj4k2**2+
     1     bd5*gj5k2**2+bd6*gj6k2**2+bd7*gj7k2**2
c
c     block b(1), with wall b.c.
c
      b(1,1,k)=1.                                                    b.c.
      b(2,2,k)=1.                                                    b.c.
      b(3,3,k)=-c2dz-c4dz2/al1(k)*(wk-al1p(k))                       a33
      b(3,4,k)=0.                                                    a34
      b(4,3,k)=0.                                                    a43
      b(4,4,k)=-c2dz-c4dz2/al2(k)*(wk-al2p(k))                       a44
c
c     block c1
c
      c(3,1,k)=1.-c3dz2/al1(k2)*(2*b3s*fk2+b4s*gk2+a1*b1*fik2+       b31
     1         b2*bd*gjk2)
      c(3,2,k)=-c3dz2/al1(k2)*(b2*bfk2+                              b32
     1         b4s*fk2+2*b5*gk2)
```

```
      c(3,3,k)=-c1dz-c3dz2/a11(k2)*(wk2-a11p(k2))                        b33
      c(3,4,k)=0.                                                       b34
      c(4,1,k)=-c3dz2/a12(k2)*(cc4s*gk2+2*cc3*fk2+                      b41
     1         cc1*(a1*gik2+a2*gi1k2+a3*gi2k2))
      c(4,2,k)=1.-c3dz2/a12(k2)*(a1*cc1*fik2+cc4s*fk2+                  b42
     1         bd*cc2*gjk2+2*cc5s*gk2)
      c(4,3,k)=0.                                                       b43
      c(4,4,k)=-c1dz-c3dz2/a12(k2)*(wk2-a12p(k2))                       b44
c
c     k=1 rhs vector
c
      r(1,k)=0.                                                         b.c.
      r(2,k)=0.                                                         b.c.
c
      r(3,k)= fk-fk2+c1dz*fpk2+c2dz*fpk+                                r3
     1        c3dz2/a11(k2)*(0.5*b1*(a1*fik2**2+a2*fi1k2**2+
     1        a3*fi2k2**2)+b2*gk2*bfk2+
     1        b3s*fk2**2+b4s*fk2*gk2+b5*gk2**2+b6*the1(k2)+
     1        fpk2*(wk2-a11p(k2)))+
     1        c4dz2/a11(k )*(0.5*b1*(a1*fik **2+a2*fi1k **2+
     1        a3*fi2k **2)+b2*gk *bfk+
     1        b3s*fk **2+b4s*fk *gk +b5*gk **2+b6*the1(k )+
     1        fpk *(wk -a11p(k )))
c
      r(4,k)= gk-gk2+c1dz*gpk2+c2dz*gpk+                                r4
     1        c3dz2/a12(k2)*(cc1*fk2*(a1*gik2+a2*gi1k2+a3*gi2k2)+
     1        0.5*cc2*bggk2+
     1        cc3*fk2**2+cc4s*fk2*gk2+cc5s*gk2**2+cc6*the1(k2)+
     1        gpk2*(wk2-a12p(k2)))+
     1        c4dz2/a12(k )*(cc1*fk *(a1*gik +a2*gi1k +a3*gi2k )+
     1        0.5*cc2*bggk+
     1        cc3*fk **2+cc4s*fk *gk +cc5s*gk **2+cc6*the1(k )+
     1        gpk *(wk -a12p(k )))
c
c     k=nz
c
c
c     block b(nz), with free-sream b.c.
c
      k=nz
      call getfg(k)
      b(1,3,k)=a11(k)-c3dz2*(2*b3*fk+b4*gk+2*b1*a1*fik+b2*bd*gjk)        b13
      b(1,4,k)=-c3dz2*(b4*fk+2*b5*gk+b2*bd*fjk)                         b14
      b(2,3,k)=-c3dz2*(2*cc3*fk+cc4*gk+cc1*a1*gik)                      b23
      b(2,4,k)=a12(k)-c3dz2*(cc4*fk+2*cc5*gk+cc1*a1*fik+2*cc2*bd*gjk)   b24
      b(3,1,k)=1.                                                       b.c.
      b(4,2,k)=1.0                                                      b.c.
c
c     block a(nz)
c
      a(1,1,k)=wk1-c2dz*(2*b3*fk1+b4*gk1+2*b1*a1*fik1+b2*bd*gjk1)-      a11
     1         c4dz2*(2*b3*fpk1+b4*gpk1+2*b1*a1*fpik1+b2*bd*gpjk1)
      a(1,2,k)=-c2dz*(b4*fk1+2*b5*gk1+b2*bd*fjk1)                       a12
     1         -c4dz2*(b4*fpk1+2*b5*gpk1+b2*bd*fpjk1)
      a(1,3,k)=-a11(k1)-c4dz2*(2*b3*fk1+b4*gk1+                         a13
     1         2*b1*a1*fik1+b2*bd*gjk1)
      a(1,4,k)=-c4dz2*(b4*fk1+2*b5*gk1+b2*bd*fjk1)                      a14
      a(2,1,k)=-c2dz*(2*cc3*fk1+cc4*gk1+cc1*a1*gik1)                    a21
     1         -c4dz2*(2*cc3*fpk1+cc4*gpk1+cc1*a1*gpik1)
      a(2,2,k)=wk1-c2dz*(cc4*fk1+2*cc5*gk1+cc1*a1*fik1+                 a22
     1         2*cc2*bd*gjk1)-c4dz2*(cc4*fpk1+2*cc5*gpk1+
     1         cc1*a1*fpik1+2*cc2*bd*gpjk1)
      a(2,3,k)=-c4dz2*(2*cc3*fk1+cc4*gk1+cc1*a1*gik1)                   a23
      a(2,4,k)=-a12(k1)-c4dz2*(cc4*fk1+2*cc5*gk1+cc1*a1*fik1+           a24
```

```
     1                  2*cc2*bd*gjk1)
c
c     rhs vector at nz
c
      r(3,k)=0.                                                         b.c.
      r(4,k)=0.                                                         b.c.
      bfgk =bd*fjk *gjk +bd1*fj1k *gj1k +bd2*fj2k *gj2k +
     1      bd3*fj3k *gj3k +bd4*fj4k *gj4k +bd5*fj5k *gj5k +
     1      bd6*fj6k *gj6k +bd7*fj7k *gj7k
      bfgk1=bd*fjk1*gjk1+bd1*fj1k1*gj1k1+bd2*fj2k1*gj2k1+
     1      bd3*fj3k1*gj3k1+bd4*fj4k1*gj4k1+bd5*fj5k1*gj5k1+
     1      bd6*fj6k1*gj6k1+bd7*fj7k1*gj7k1
      bfgpk1=bd *(fjk1*gpjk1+fpjk1*gjk1)+
     1      bd1*(fj1k1*gpj1k1+fpj1k1*gj1k1)+
     1      bd2*(fj2k1*gpj2k1+fpj2k1*gj2k1)+
     1      bd3*(fj3k1*gpj3k1+fpj3k1*gj3k1)+
     1      bd4*(fj4k1*gpj4k1+fpj4k1*gj4k1)+
     1      bd5*(fj5k1*gpj5k1+fpj5k1*gj5k1)+
     1      bd6*(fj6k1*gpj6k1+fpj6k1*gj6k1)+
     1      bd7*(fj7k1*gpj7k1+fpj7k1*gj7k1)
      bfgpk =bd *(fjk *gpjk +fpjk *gjk )+
     1      bd1*(fj1k *gpj1k +fpj1k *gj1k )+
     1      bd2*(fj2k *gpj2k +fpj2k *gj2k )+
     1      bd3*(fj3k *gpj3k +fpj3k *gj3k )+
     1      bd4*(fj4k *gpj4k +fpj4k *gj4k )+
     1      bd5*(fj5k *gpj5k +fpj5k *gj5k )+
     1      bd6*(fj6k *gpj6k +fpj6k *gj6k )+
     1      bd7*(fj7k *gpj7k +fpj7k *gj7k )
      r(1,k)=(a11(k1)*fpk1-wk1*fk1)-(a11(k)*fpk-wk*fk)+                 r1
     1       c2dz*(b3*fk1**2+b4*fk1*gk1+b5*gk1**2+b6*the1(k1)+
     1       b1*(a1*fik1**2+a2*fi1k1**2+a3*fi2k1**2)+
     1       b2*bfgk1)+
     1       c1dz*(b3*fk **2+b4*fk *gk +b5*gk **2+b6*the1(k )+
     1       b1*(a1*fik **2+a2*fi1k **2+a3*fi2k **2)+
     1       b2*bfgk)+
     1       c4dz2*(2*b3*fk1*fpk1+b4*(fk1*gpk1+gk1*fpk1)+
     1       2*b5*gk1*gpk1+b6*the2(k1)+2*b1*(a1*fik1*fpik1+
     1       a2*fi1k1*fpi1k1+a3*fi2k1*fpi2k1)+
     1       b2*bfgpk1)+
     1       c3dz2*(2*b3*fk *fpk +b4*(fk *gpk +gk *fpk )+
     1       2*b5*gk *gpk +b6*the2(k )+2*b1*(a1*fik *fpik +
     1       a2*fi1k *fpi1k +a3*fi2k *fpi2k )+
     1       b2*bfgpk)
c
      bggk =bd*gjk **2+bd1*gj1k **2+bd2*gj2k **2+
     1     bd3*gj3k **2+bd4*gj4k **2+
     1     bd5*gj5k **2+bd6*gj6k **2+bd7*gj7k **2
      bggk1=bd*gjk1**2+bd1*gj1k1**2+bd2*gj2k1**2+
     1     bd3*gj3k1**2+bd4*gj4k1**2+
     1     bd5*gj5k1**2+bd6*gj6k1**2+bd7*gj7k1**2
      r(2,k)=a12(k1)*gpk1-wk1*gk1-(a12(k)*gpk-wk*gk)+                   r2
     1       c2dz*(cc4*fk1*gk1+cc5*gk1**2+cc3*fk1**2+
     1       cc6*the1(k1)+cc1*(a1*fik1*gik1+a2*fi1k1*gi1k1+
     1       a3*fi2k1*gi2k1)+cc2*bggk1)+
     1       c1dz*(cc4*fk *gk +cc5*gk **2+cc3*fk **2+
     1       cc6*the1(k )+cc1*(a1*fik *gik +a2*fi1k *gi1k +
     1       a3*fi2k *gi2k )+cc2*bggk)+
     1       c4dz2*(cc4*(fk1*gpk1+gk1*fpk1)+2*cc5*gk1*gpk1+
     1       2*cc3*fk1*fpk1+cc6*the2(k1)+cc1*(a1*
     1       (fik1*gpik1+gik1*fpik1)+a2*(fi1k1*gpi1k1+
     1       gi1k1*fpi1k1)+a3*(fi2k1*gpi2k1+gi2k1*fpi2k1)))+
     1       c3dz2*(cc4*(fk *gpk +gk *fpk )+2*cc5*gk *gpk +
     1       2*cc3*fk *fpk +cc6*the2(k )+cc1*(a1*
     1       (fik *gpik +gik *fpik )+a2*(fi1k *gpi1k +
```

```
     1       gi1k *fpi1k )+a3*(fi2k *gpi2k +gi2k *fpi2k )))
c
c     a,b,c,r for an interior point k
c
      do 101 k=2,nz1
      call getfg(k)
      bfk =bd*fjk +bd1*fj1k +bd2*fj2k +bd3*fj3k +bd4*fj4k +
     1     bd5*fj5k +bd6*fj6k +bd7*fj7k
      bfk2=bd*fjk2+bd1*fj1k2+bd2*fj2k2+bd3*fj3k2+bd4*fj4k2+
     1     bd5*fj5k2+bd6*fj6k2+bd7*fj7k2
      a(1,1,k)=wk1-c2dz*(2*b3*fk1+b4*gk1+2*b1*a1*fik1+b2*bd *gjk1)-      a11
     1         c4dz2*(2*b3*fpk1+b4*gpk1+2*b1*a1*fpik1+b2*bd *gpjk1)
      a(1,2,k)=-c2dz*(b4*fk1+2*b5*gk1+b2*bd *fjk1)                       a12
     1         -c4dz2*(b4*fpk1+2*b5*gpk1+b2*bd *fpjk1)
      a(1,3,k)=-a11(k1)-c4dz2*(2*b3*fk1+b4*gk1+                          a13
     1         2*b1*a1*fik1+b2*bd *gjk1)
      a(1,4,k)=-c4dz2*(b4*fk1+2*b5*gk1+b2*bd *fjk1)                      a14
      a(2,1,k)=-c2dz*(2*cc3*fk1+cc4*gk1+cc1*a1*gik1)                     a21
     1         -c4dz2*(2*cc3*fpk1+cc4*gpk1+cc1*a1*gpik1)
      a(2,2,k)=wk1-c2dz*(cc4*fk1+2*cc5*gk1+cc1*a1*fik1+                  a22
     1         2*cc2*bd *gjk1)-c4dz2*(cc4*fpk1+2*cc5*gpk1+
     1         cc1*a1*fpik1+2*cc2*bd *gpjk1)
      a(2,3,k)=-c4dz2*(2*cc3*fk1+cc4*gk1+cc1*a1*gik1)                    a23
      a(2,4,k)=-a12(k1)-c4dz2*(cc4*fk1+2*cc5*gk1+cc1*a1*fik1+            a24
     1                         2*cc2*bd *gjk1)
      b(1,1,k)=-wk-c1dz*(2*b3*fk+b4*gk+2*a1*b1*fik+b2*bd *gjk)-          b11
     1         c3dz2*(2*b3*fpk+b4*gpk+2*a1*b1*fpik+b2*bd *gpjk)
      b(1,2,k)=-c1dz*(b4*fk+2*b5*gk+b2*bd *fjk)-                         b12
     1         c3dz2*(b4*fpk+2*b5*gpk+b2*bd *fpjk)
      b(1,3,k)=a11(k)-c3dz2*(2*b3*fk+b4*gk+2*b1*a1*fik+b2*bd *gjk)       b13
      b(1,4,k)=-c3dz2*(b4*fk+2*b5*gk+b2*bd *fjk)                         b14
      b(2,1,k)=-c1dz*(cc4*gk+2*cc3*fk+a1*cc1*gik)-                       b21
     1         c3dz2*(cc4*gpk+2*cc3*fpk+a1*cc1*gpik)
      b(2,2,k)=-wk-c1dz*(cc4*fk+2*cc5*gk+a1*cc1*fik+2*bd *cc2*gjk)       b22
     1         -c3dz2*(cc4*fpk+2*cc5*gpk+a1*cc1*fpik+2*bd *cc2*gpjk)
      b(2,3,k)=-c3dz2*(2*cc3*fk+cc4*gk+cc1*a1*gik)                       b23
      b(2,4,k)=a12(k)-c3dz2*(cc4*fk+2*cc5*gk+cc1*a1*fik+2*cc2*bd *gjk)   b24
      b(3,1,k)=-1.-d4dz2/a11(k)*(2*b3s*fk+b4s*gk+a1*b1*fik+b2*bd *gjk)   a31
      b(3,2,k)=-d4dz2/a11(k)*(b2*bfk+b4s*fk+2*b5*gk)                     a32
      b(3,3,k)=-d2dz-d4dz2/a11(k)*(wk-a11p(k))                           a33
      b(3,4,k)=0.                                                        a34
      b(4,1,k)=-d4dz2/a12(k)*(cc4s*gk+2*cc3*fk+cc1*(a1*gik+              a41
     1         a2*gi1k+a3*gi2k))
      b(4,2,k)=-1.-d4dz2/a12(k)*(cc4s*fk+2*cc5s*gk+a1*cc1*fik+           a42
     1         bd *cc2*gjk)
      b(4,3,k)=0.                                                        a43
      b(4,4,k)=-d2dz-d4dz2/a12(k)*(wk-a12p(k))                           a44
      c(3,1,k)=1.-d3dz2/a11(k2)*(2*b3s*fk2+b4s*gk2+a1*b1*fik2+           b31
     1         b2*bd *gjk2)
      c(3,2,k)=-d3dz2/a11(k2)*(b2*bfk2+b4s*fk2+2*b5*gk2)                 b32
      c(3,3,k)=-d1dz-d3dz2/a11(k2)*(wk2-a11p(k2))                        b33
      c(3,4,k)=0.                                                        b34
      c(4,1,k)=-d3dz2/a12(k2)*(cc4s*gk2+2*cc3*fk2+                       b41
     1         cc1*(a1*gik2+a2*gi1k2+a3*gi2k2))
      c(4,2,k)=1.-d3dz2/a12(k2)*(a1*cc1*fik2+cc4s*fk2+                   b42
     1         bd *cc2*gjk2+2*cc5s*gk2)
      c(4,3,k)=0.                                                        b43
      c(4,4,k)=-d1dz-d3dz2/a12(k2)*(wk2-a12p(k2))                        b44
      bfgpk1=bd *(fjk1*gpjk1+fpjk1*gjk1)+
     1      bd1*(fj1k1*gpj1k1+fpj1k1*gj1k1)+
     1      bd2*(fj2k1*gpj2k1+fpj2k1*gj2k1)+
     1      bd3*(fj3k1*gpj3k1+fpj3k1*gj3k1)+
     1      bd4*(fj4k1*gpj4k1+fpj4k1*gj4k1)+
     1      bd5*(fj5k1*gpj5k1+fpj5k1*gj5k1)+
     1      bd6*(fj6k1*gpj6k1+fpj6k1*gj6k1)+
     1      bd7*(fj7k1*gpj7k1+fpj7k1*gj7k1)
      bfgpk =bd *(fjk *gpjk +fpjk *gjk )+
     1      bd1*(fj1k *gpj1k +fpj1k *gj1k )+
     1      bd2*(fj2k *gpj2k +fpj2k *gj2k )+
     1      bd3*(fj3k *gpj3k +fpj3k *gj3k )+
     1      bd4*(fj4k *gpj4k +fpj4k *gj4k )+
     1      bd5*(fj5k *gpj5k +fpj5k *gj5k )+
     1      bd6*(fj6k *gpj6k +fpj6k *gj6k )+
     1      bd7*(fj7k *gpj7k +fpj7k *gj7k )
      bfgk =bd*fjk *gjk +bd1*fj1k *gj1k +bd2*fj2k *gj2k +
     1      bd3*fj3k *gj3k +bd4*fj4k *gj4k +bd5*fj5k *gj5k +
     1      bd6*fj6k *gj6k +bd7*fj7k *gj7k
      bfgk1=bd*fjk1*gjk1+bd1*fj1k1*gj1k1+bd2*fj2k1*gj2k1+
     1      bd3*fj3k1*gj3k1+bd4*fj4k1*gj4k1+bd5*fj5k1*gj5k1+
     1      bd6*fj6k1*gj6k1+bd7*fj7k1*gj7k1
      r(1,k)=(a11(k1)*fpk1-wk1*fk1)-(a11(k)*fpk-wk*fk)+                  r1
     1       c2dz*(b3*fk1**2+b4*fk1*gk1+b5*gk1**2+b6*the1(k1)+
     1       b1*(a1*fik1**2+a2*fi1k1**2+a3*fi2k1**2)+
     1       b2*bfgk1)+
     1       c1dz*(b3*fk **2+b4*fk *gk +b5*gk **2+b6*the1(k )+
     1       b1*(a1*fik **2+a2*fi1k **2+a3*fi2k **2)+
     1       b2*bfgk)+
     1       c4dz2*(2*b3*fk1*fpk1+b4*(fk1*gpk1+gk1*fpk1)+
     1       2*b5*gk1*gpk1+b6*the2(k1)+2*b1*(a1*fik1*fpik1+
     1       a2*fi1k1*fpi1k1+a3*fi2k1*fpi2k1)+
     1       b2*bfgpk1)+
     1       c3dz2*(2*b3*fk *fpk +b4*(fk *gpk +gk *fpk )+
     1       2*b5*gk *gpk +b6*the2(k )+2*b1*(a1*fik *fpik +
     1       a2*fi1k *fpi1k +a3*fi2k *fpi2k )+
     1       b2*bfgpk )
c
      bggk =bd*gjk **2+bd1*gj1k **2+bd2*gj2k **2+
     1      bd3*gj3k **2+bd4*gj4k **2+
     1      bd5*gj5k **2+bd6*gj6k **2+bd7*gj7k **2
      bggk1=bd*gjk1**2+bd1*gj1k1**2+bd2*gj2k1**2+
     1      bd3*gj3k1**2+bd4*gj4k1**2+
     1      bd5*gj5k1**2+bd6*gj6k1**2+bd7*gj7k1**2
      r(2,k)=a12(k1)*gpk1-wk1*gk1-(a12(k)*gpk-wk*gk)+                    r2
     1       c2dz*(cc4*fk1*gk1+cc5*gk1**2+cc3*fk1**2+
     1       cc6*the1(k1)+cc1*(a1*fik1*gik1+a2*fi1k1*gi1k1+
     1       a3*fi2k1*gi2k1)+cc2*bggk1)+
     1       c1dz*(cc4*fk *gk +cc5*gk **2+cc3*fk **2+
     1       cc6*the1(k )+cc1*(a1*fik *gik +a2*fi1k *gi1k +
     1       a3*fi2k *gi2k )+cc2*bggk )+
     1       c4dz2*(cc4*(fk1*gpk1+gk1*fpk1)+2*cc5*gk1*gpk1+
     1       2*cc3*fk1*fpk1+cc6*the2(k1)+cc1*(a1*
     1       (fik1*gpik1+gik1*fpik1)+a2*(fi1k1*gpi1k1+
     1       gi1k1*fpi1k1)+a3*(fi2k1*gpi2k1+gi2k1*fpi2k1)))+
     1       c3dz2*(cc4*(fk *gpk +gk *fpk )+2*cc5*gk *gpk +
     1       2*cc3*fk *fpk +cc6*the2(k )+cc1*(a1*
     1       (fik *gpik +gik *fpik )+a2*(fi1k *gpi1k +
     1       gi1k *fpi1k )+a3*(fi2k *gpi2k +gi2k *fpi2k )))
      bfk =bd*fjk +bd1*fj1k +bd2*fj2k +bd3*fj3k +bd4*fj4k +
     1     bd5*fj5k +bd6*fj6k +bd7*fj7k
      bfk2=bd*fjk2+bd1*fj1k2+bd2*fj2k2+bd3*fj3k2+bd4*fj4k2+
     1     bd5*fj5k2+bd6*fj6k2+bd7*fj7k2
      bggk =bd*gjk **2+bd1*gj1k **2+bd2*gj2k **2+
     1      bd3*gj3k **2+bd4*gj4k **2+
     1      bd5*gj5k **2+bd6*gj6k **2+bd7*gj7k **2
      bggk2=bd*gjk2**2+bd1*gj1k2**2+bd2*gj2k2**2+
     1      bd3*gj3k2**2+bd4*gj4k2**2+
     1      bd5*gj5k2**2+bd6*gj6k2**2+bd7*gj7k2**2
      r(3,k)= fk-fk2+d1dz*fpk2+d2dz*fpk+                                 r3
```

```
     1        d3dz2/al1(k2)*(0.5*b1*(a1*fik2**2+a2*fi1k2**2+
     1        a3*fi2k2**2)+b2*gk2*bfk2+
     1        b3s*fk2**2+b4s*fk2*gk2+b5*gk2**2+b6*the1(k2)+
     1        fpk2*(wk2-a1lp(k2)))+
     1        d4dz2/al1(k )*(0.5*b1*(a1*fik **2+a2*fi1k **2+
     1        a3*fi2k **2)+b2*gk *bfk +
     1        b3s*fk **2+b4s*fk *gk +b5*gk **2+b6*the1(k )+
     1        fpk *(wk -a1lp(k )))
c
      r(4,k)= gk-gk2+d1dz*gpk2+d2dz*gpk+                                    r4
     1        d3dz2/al2(k2)*(cc1*fk2*(a1*gik2+a2*gi1k2+a3*gi2k2)+
     1        0.5*cc2*bggk2+
     1        cc3*fk2**2+cc4s*fk2*gk2+cc5s*gk2**2+cc6*the1(k2)+
     1        gpk2*(wk2-al2p(k2)))+
     1        d4dz2/al2(k )*(cc1*fk *(a1*gik +a2*gi1k +a3*gi2k )+
     1        0.5*cc2*bggk+
     1        cc3*fk **2+cc4s*fk *gk +cc5s*gk **2+cc6*the1(k )+
     1        gpk *(wk -al2p(k )))

 101  continue
c
c     call tridiagonal solver; solution in r
c
      call blklu(a,b,c,nz,4,16,ir,ic,r)
      call blksolv(a,b,c,nz,4,16,ir,ic,r)
c
c     update the solution vector
c
      do 102 k=1,nz
      f (j,k)=f (j,k)+r(1,k)
      g (j,k)=g (j,k)+r(2,k)
      fp(j,k)=fp(j,k)+r(3,k)
      gp(j,k)=gp(j,k)+r(4,k)
      delfp(k)=r(3,k)
 102  continue
      return
      end
c-----------------------------------------------------------------------
      subroutine cntnty
c-----------------------------------------------------------------------
      include 'com'
      include 'comloc'
      w(j,1)=ww
      do 1 k=2,nz
      call getfg(k)
      if(k.eq.2)wk1=ww
      bgk =bd*gjk +bd1*gj1k +bd2*gj2k +bd3*gj3k +bd4*gj4k +
     1     bd5*gj5k +bd6*gj6k +bd7*gj7k
      bgk1=bd*gjk1+bd1*gj1k1+bd2*gj2k1+bd3*gj3k1+bd4*gj4k1+
     1     bd5*gj5k1+bd6*gj6k1+bd7*gj7k1
      bgpk =bd*gpjk +bd1*gpj1k +bd2*gpj2k +bd3*gpj3k +bd4*gpj4k +
     1     bd5*gpj5k +bd6*gpj6k +bd7*gpj7k
      bgpk1=bd*gpjk1+bd1*gpj1k1+bd2*gpj2k1+bd3*gpj3k1+bd4*gpj4k1+
     1     bd5*gpj5k1+bd6*gpj6k1+bd7*gpj7k1
      w(j,k)=wk1+c1dz*(aa1s*(a1*fik+a2*fi1k+a3*fi2k)+
     1       aa2s*fk +aa4s*gk  +
     1       aa3s*bgk)+
     1       c2dz*(aa1s*(a1*fik1+a2*fi1k1+a3*fi2k1)+
     1       aa2s*fk1+aa4s*gk1 +
     1       aa3s*bgk1)
     1       +c3dz2*(aa1s*(a1*fpik+a2*fpi1k+a3*fpi2k)+
     1       aa2s*fpk+aa4s*gpk+
     1       aa3s*bgpk)
     1       +c4dz2*(aa1s*(a1*fpik1+a2*fpi1k1+a3*fpi2k1)+
     1       aa2s*fpk1+aa4s*gpk1+
     1       aa3s*bgpk1)
 1    continue
      return
      end
c-----------------------------------------------------------------------
      subroutine enrgy
c-----------------------------------------------------------------------
      include 'com'
      include 'comloc'
      dimension      a(2,2,nz),b(2,2,nz),c(2,2,nz),r(2,nz),
     1               ir(2,nz),ic(2,nz)
c
c     set up system for solving the energy equation
c     iwall=0 adiabatic case; iwall=1 wall temp. specified
c     iwall=2 heat flux at wall specified as non-zero value
c
c     initialize all coefft.s to zero first
c
      do 1 k=1,nz
      do 1 l=1,2
      r(l,k)=0.
      do 1 m=1,2
      a(l,m,k)=0.
      b(l,m,k)=0.
      c(l,m,k)=0.
 1    continue
c
c     k=1 diagonal block
c
      k=1
      call gethi(k)
      call getfg(k)
c
      if(iwall.eq.1)b(1,1,k)=1.                                          b.c.
      if(iwall.eq.0.or.iwall.eq.2)b(1,2,k)=1.                            b.c.
      b(2,1,k)=-1.                                                       a21
      b(2,2,k)=-c2dz-c4dz2*((wk-a1lp(k))/a11(k))                         a22
c
c     k=1, c block
c
      c(2,1,k)=1.-c3dz2/a11(k2)*(dd1*fik2   *a1+dd2*bd *gjk2   )         b21
      c(2,2,k)=-c1dz-c3dz2*((wk2    -a1lp(k2))/a11(k2))                  b22
c
c     k=1 rhs vector
c
      if(iwall.eq.1)r(1,k)=hw                                            b.c.
      if(iwall.eq.2)r(1,k)=hpw                                           b.c.
      if(iwall.eq.0)r(1,k)=0.                                            b.c.
      bhk =bd1*hj1k +bd2*hj2k +bd3*hj3k +bd4*hj4k +
     1     bd5*hj5k +bd6*hj6k +bd7*hj7k
      bhk2=bd1*hj1k2+bd2*hj2k2+bd3*hj3k2+bd4*hj4k2+
     1     bd5*hj5k2+bd6*hj6k2+bd7*hj7k2
      r(2,k)=c3dz2/a11(k2)*(dd5k2  +dd1*fk2    *(a2*hi1k2    +          r(2)
     1       a3*hi2k2    )+dd2*gk2    *bhk2)+
     1       c4dz2/a11(k )*(dd5k   +dd1*fk     *(a2*hi1k     +
     1       a3*hi2k     )+dd2*gk     *bhk )
c
c     k=nz diagonal block
c
      k=nz
      call getfg(k)
      call gethi(k)
c
```

```
      b(1,1,k)=-wk-c1dz*(dd1*a1*fik+dd3*fk+dd4*gk+dd2*bd *gjk)-          b11
     1         c3dz2*( dd1*a1*fpik+dd3*fpk + dd4*gpk+bd *dd2*gpjk )
      b(1,2,k)=all(k)-c3dz2*(fik*dd1*a1+dd3*fk+dd4*gk+dd2*bd *gjk)      b12
      b(2,1,k)=1.                                                       b.c.
c
c     k=nz, block a
c
      a(1,1,k)=wk1-c2dz*(dd3*fk1+dd1*a1*fik1+dd4*gk1+dd2*bd *gjk1)-     a11
     1         c4dz2*(dd3*fpk1+dd1*a1*fpik1+dd4*gpk1+dd2*bd *gpjk1)
      a(1,2,k)=-all(k1)-c4dz2*(dd1*a1*fik1+dd3*fk1+                     a12
     1         dd2*bd *gjk1+dd4*gk1)
c
c     k=nz, rhs vector
c
c
c
      bghk =bd1*gj1k *hj1k +bd2*gj2k *hj2k +bd3*gj3k *hj3k +
     1      bd4*gj4k *hj4k +bd5*gj5k *hj5k +bd6*gj6k *hj6k +
     1      bd7*gj7k *hj7k
      bghk1=bd1*gj1k1*hj1k1+bd2*gj2k1*hj2k1+bd3*gj3k1*hj3k1+
     1      bd4*gj4k1*hj4k1+bd5*gj5k1*hj5k1+bd6*gj6k1*hj6k1+
     1      bd7*gj7k1*hj7k1
      bhgpk1=bd1*(hj1k1*gpj1k1+hpj1k1*gj1k1)+
     1      bd2*(hj2k1*gpj2k1+hpj2k1*gj2k1)+
     1      bd3*(hj3k1*gpj3k1+hpj3k1*gj3k1)+
     1      bd4*(hj4k1*gpj4k1+hpj4k1*gj4k1)+
     1      bd5*(hj5k1*gpj5k1+hpj5k1*gj5k1)+
     1      bd6*(hj6k1*gpj6k1+hpj6k1*gj6k1)+
     1      bd7*(hj7k1*gpj7k1+hpj7k1*gj7k1)
      bhgpk=bd1*(hj1k *gpj1k +hpj1k *gj1k )+
     1      bd2*(hj2k *gpj2k +hpj2k *gj2k )+
     1      bd3*(hj3k *gpj3k +hpj3k *gj3k )+
     1      bd4*(hj4k *gpj4k +hpj4k *gj4k )+
     1      bd5*(hj5k *gpj5k +hpj5k *gj5k )+
     1      bd6*(hj6k *gpj6k +hpj6k *gj6k )+
     1      bd7*(hj7k *gpj7k +hpj7k *gj7k )
      r(1,k)=c1dz*(dd5k  +dd1*(a2*hi1k    *fi1k   +a3*hi2k    *          r(1)
     1             fi2k   )+dd2*bghk)+
     1       c2dz*(dd5k1  +dd1*(a2*hi1k1    *fi1k1   +a3*hi2k1    *
     1             fi2k1   )+dd2*bghk1)+
     1       c3dz2*(dd5pk  +dd1*(a2*hi1k    *fpi1k   +a3*hi2k    *
     1              fpi2k   +a2*hpi1k    *fi1k   +a3*hpi2k    *fi2k   )
     1             +dd2*bhgpk)+
     1       c4dz2*(dd5pk1 +dd1*(a2*hi1k1   *fpi1k1  +a3*hi2k1   *
     1              fpi2k1  +a2*hpi1k1   *fi1k1  +a3*hpi2k1   *fi2k1  )
     1             +dd2*bhgpk1)
      r(2,k)=htot                                                       b.c.
c
c     a,b,c,r for an interior point k
c
      do 101 k=2,nz1
      call getfg(k)
      call gethi(k)
      a(1,1,k)=wk1-c2dz*(dd3*fk1+dd1*a1*fik1+dd4*gk1+dd2*bd *gjk1)-     a11
     1         c4dz2*(dd3*fpk1+dd1*a1*fpik1+dd4*gpk1+dd2*bd *gpjk1)
      a(1,2,k)=-all(k1)-c4dz2*(dd1*a1*fik1+dd3*fk1+                     a12
     1         dd2*bd *gjk1+dd4*gk1)
      c(2,1,k)=1.-d3dz2/all(k2)*(dd1*fik2   *a1+dd2*bd *gjk2   )       b21
      c(2,2,k)=-d1dz-d3dz2*((wk2    -allp(k2))/all(k2))                 b22
c
c     diagonal elements, b
c
      b(1,1,k)=-wk-c1dz*(dd1*a1*fik+dd3*fk+dd4*gk+dd2*bd *gjk)-          b11
     1         c3dz2*( dd1*a1*fpik+dd3*fpk + dd4*gpk+bd *dd2*gpjk )
      b(1,2,k)=all(k)-c3dz2*(fik*dd1*a1+dd3*fk+dd4*gk+dd2*bd *gjk)      b12
      b(2,1,k)=-1.-d4dz2/all(k)*(dd1*fik*a1+dd2*bd *gjk)                a21
      b(2,2,k)=-d2dz-d4dz2*((wk-allp(k))/all(k))                        a22
c
c     rhs vector
      bghk =bd1*gj1k *hj1k +bd2*gj2k *hj2k +bd3*gj3k *hj3k +
     1      bd4*gj4k *hj4k +bd5*gj5k *hj5k +bd6*gj6k *hj6k +
     1      bd7*gj7k *hj7k
      bghk1=bd1*gj1k1*hj1k1+bd2*gj2k1*hj2k1+bd3*gj3k1*hj3k1+
     1      bd4*gj4k1*hj4k1+bd5*gj5k1*hj5k1+bd6*gj6k1*hj6k1+
     1      bd7*gj7k1*hj7k1
      bhgpk1=bd1*(hj1k1*gpj1k1+hpj1k1*gj1k1)+
     1      bd2*(hj2k1*gpj2k1+hpj2k1*gj2k1)+
     1      bd3*(hj3k1*gpj3k1+hpj3k1*gj3k1)+
     1      bd4*(hj4k1*gpj4k1+hpj4k1*gj4k1)+
     1      bd5*(hj5k1*gpj5k1+hpj5k1*gj5k1)+
     1      bd6*(hj6k1*gpj6k1+hpj6k1*gj6k1)+
     1      bd7*(hj7k1*gpj7k1+hpj7k1*gj7k1)
      bhgpk=bd1*(hj1k *gpj1k +hpj1k *gj1k )+
     1      bd2*(hj2k *gpj2k +hpj2k *gj2k )+
     1      bd3*(hj3k *gpj3k +hpj3k *gj3k )+
     1      bd4*(hj4k *gpj4k +hpj4k *gj4k )+
     1      bd5*(hj5k *gpj5k +hpj5k *gj5k )+
     1      bd6*(hj6k *gpj6k +hpj6k *gj6k )+
     1      bd7*(hj7k *gpj7k +hpj7k *gj7k )
      r(1,k)=c1dz*(dd5k  +dd1*(a2*hi1k    *fi1k   +a3*hi2k    *          r(1)
     1             fi2k   )+dd2*bghk)+
     1       c2dz*(dd5k1  +dd1*(a2*hi1k1    *fi1k1   +a3*hi2k1    *
     1             fi2k1   )+dd2*bghk1)
     1       +c3dz2*(dd5pk  +dd1*(a2*hi1k    *fpi1k   +a3*hi2k    *
     1              fpi2k   +a2*hpi1k    *fi1k   +a3*hpi2k    *fi2k   )
     1             +dd2*bhgpk )
     1       +c4dz2*(dd5pk1 +dd1*(a2*hi1k1   *fpi1k1  +a3*hi2k1   *
     1              fpi2k1  +a2*hpi1k1   *fi1k1  +a3*hpi2k1   *fi2k1  )
     1             +dd2*bhgpk1)
      bhk =bd1*hj1k +bd2*hj2k +bd3*hj3k +bd4*hj4k +
     1     bd5*hj5k +bd6*hj6k +bd7*hj7k
      bhk2=bd1*hj1k2+bd2*hj2k2+bd3*hj3k2+bd4*hj4k2+
     1     bd5*hj5k2+bd6*hj6k2+bd7*hj7k2
      r(2,k)=d3dz2/all(k2)*(dd5k2  +dd1*fk2    *(a2*hi1k2    +          r(2)
     1       a3*hi2k2    )+dd2*gk2    *bhk2)+
     1       d4dz2/all(k )*(dd5k   +dd1*fk     *(a2*hi1k     +
     1       a3*hi2k     )+dd2*gk     *bhk )
  101 continue
c
c     call tridiagonal solver; solution in r
c
      call blklu(a,b,c,nz,2, 4,ir,ic,r)
      call blksolv(a,b,c,nz,2, 4,ir,ic,r)
c
c     update the solution vector
c
      do 102 k=1,nz
      h (j,k)=r(1,k)
      hppk=hp(j,k)
      hp(j,k)=r(2,k)
      delhp(k)=r(2,k)-hppk
  102 continue
      return
      end
c-----------------------------------------------------------------------
      subroutine getfg(k)
c-----------------------------------------------------------------------
      include 'com'
```

```
      include 'comloc'
c
      if(k.eq.1)then
      k2=2
      dz=z(k+1)-z(k)
      c1dz=c1*dz
      c2dz=c2*dz
      c3dz2=c3*dz*dz
      c4dz2=c4*dz*dz
      endif
      if(k.eq.nz)then
      dz=z(nz)-z(nz1)
      k1=nz-1
      c1dz=c1*dz
      c2dz=c2*dz
      c3dz2=c3*dz*dz
      c4dz2=c4*dz*dz
      endif
      if(k.ne.1.and.k.ne.nz)then
      k1=k-1
      k2=k+1
      dz=z(k )-z(k1)
      c1dz=c1*dz
      c2dz=c2*dz
      c3dz2=c3*dz*dz
      c4dz2=c4*dz*dz
      dz=z(k2)-z(k )
      d1dz=c1*dz
      d2dz=c2*dz
      d3dz2=c3*dz*dz
      d4dz2=c4*dz*dz
      endif
c
      if(k.ne.1)then
      if(iy.eq.1)then
      fjk1=f(j,k1)                                          k1
      fpjk1=fp(j,k1)                                        k1
      gjk1=g(j,k1)                                          k1
      gpjk1=gp(j,k1)                                        k1
c
      fj1k1=f(j1,k1)                                        k1
      fpj1k1=fp(j1,k1)                                      k1
      gj1k1=g(j1,k1)                                        k1
      gpj1k1=gp(j1,k1)                                      k1
c
      fj2k1=f(j2,k1)                                        k1
      fpj2k1=fp(j2,k1)                                      k1
      gj2k1=g(j2,k1)                                        k1
      gpj2k1=gp(j2,k1)                                      k1
c
      fj3k1=f1(j3,k1)                                        k1
      fpj3k1=fp1(j3,k1)                                      k1
      gj3k1=g1(j3,k1)                                        k1
      gpj3k1=gp1(j3,k1)                                      k1
c
      fj4k1=f1(j4,k1)                                        k1
      fpj4k1=fp1(j4,k1)                                      k1
      gj4k1=g1(j4,k1)                                        k1
      gpj4k1=gp1(j4,k1)                                      k1
c
      fj5k1=f1(j5,k1)                                        k1
      fpj5k1=fp1(j5,k1)                                      k1
      gj5k1=g1(j5,k1)                                        k1
      gpj5k1=gp1(j5,k1)                                      k1
c
      fj6k1=f1(j6,k1)                                        k1
      fpj6k1=fp1(j6,k1)                                      k1
      gj6k1=g1(j6,k1)                                        k1
      gpj6k1=gp1(j6,k1)                                      k1
c
      fj7k1=f1(j7,k1)                                        k1
      fpj7k1=fp1(j7,k1)                                      k1
      gj7k1=g1(j7,k1)                                        k1
      gpj7k1=gp1(j7,k1)                                      k1
c
      endif
      if(ix.eq.1)then
      fik1=f(j,k1)                                          k1
      fi1k1=f1(j,k1)                                        k1
      fpik1=fp(j,k1)                                        k1
      fpi1k1=fp1(j,k1)                                      k1
      gik1=g(j,k1)                                          k1
      gi1k1=g1(j,k1)                                        k1
      gpik1=gp(j,k1)                                        k1
      gpi1k1=gp1(j,k1)                                      k1
      if(i.eq.2)then
      fi2k1=0                                               k1
      gi2k1=0                                               k1
      fpi2k1=0                                              k1
      gpi2k1=0                                              k1
      else
      fi2k1=f2(j,k1)                                        k1
      gi2k1=g2(j,k1)                                        k1
      fpi2k1=fp2(j,k1)                                      k1
      gpi2k1=gp2(j,k1)                                      k1
      endif
      endif
      fk1=f(j,k1)                                           k1
      gk1=g(j,k1)                                           k1
      wk1=w(j,k1)                                           k1
      fpk1=fp(j,k1)                                         k1
      gpk1=gp(j,k1)                                         k1
      endif
c
      if(k.ne.nz)then
      if(iy.eq.1)then
      fjk2=f(j,k2)                                                 k2
      fpjk2=fp(j,k2)                                               k2
      gjk2=g(j,k2)                                                 k2
      gpjk2=gp(j,k2)                                               k2
c
      fj1k2=f(j1,k2)                                               k2
      fpj1k2=fp(j1,k2)                                             k2
      gj1k2=g(j1,k2)                                               k2
      gpj1k2=gp(j1,k2)                                             k2
c
      fj2k2=f(j2,k2)                                               k2
      fpj2k2=fp(j2,k2)                                             k2
      gj2k2=g(j2,k2)                                               k2
      gpj2k2=gp(j2,k2)                                             k2
c
      fj3k2=f1(j3,k2)                                               k2
      fpj3k2=fp1(j3,k2)                                             k2
      gj3k2=g1(j3,k2)                                               k2
      gpj3k2=gp1(j3,k2)                                             k2
c
      fj4k2=f1(j4,k2)                                               k2
      fpj4k2=fp1(j4,k2)                                             k2
```

```
      gj4k2=gl(j4,k2)                                                  k2
      gpj4k2=gpl(j4,k2)                                                k2
c
      fj5k2=fl(j5,k2)                                                  k2
      fpj5k2=fpl(j5,k2)                                                k2
      gj5k2=gl(j5,k2)                                                  k2
      gpj5k2=gpl(j5,k2)                                                k2
c
      fj6k2=fl(j6,k2)                                                  k2
      fpj6k2=fpl(j6,k2)                                                k2
      gj6k2=gl(j6,k2)                                                  k2
      gpj6k2=gpl(j6,k2)                                                k2
c
      fj7k2=fl(j7,k2)                                                  k2
      fpj7k2=fpl(j7,k2)                                                k2
      gj7k2=gl(j7,k2)                                                  k2
      gpj7k2=gpl(j7,k2)                                                k2
c
      endif
      if(ix.eq.1)then
      fik2=f(j,k2)                                                     k2
      filk2=fl(j,k2)                                                   k2
      fpik2=fp(j,k2)                                                   k2
      fpilk2=fpl(j,k2)                                                 k2
      gik2=g(j,k2)                                                     k2
      gilk2=gl(j,k2)                                                   k2
      gpik2=gp(j,k2)                                                   k2
      gpilk2=gpl(j,k2)                                                 k2
      if(i.eq.2)then
      fi2k2=0                                                          k2
      gi2k2=0                                                          k2
      fpi2k2=0                                                         k2
      gpi2k2=0                                                         k2
      else
      fi2k2=f2(j,k2)                                                   k2
      gi2k2=g2(j,k2)                                                   k2
      fpi2k2=fp2(j,k2)                                                 k2
      gpi2k2=gp2(j,k2)                                                 k2
      endif
      endif
      fk2=f(j,k2)                                                      k2
      gk2=g(j,k2)                                                      k2
      wk2=w(j,k2)                                                      k2
      fpk2=fp(j,k2)                                                    k2
      gpk2=gp(j,k2)                                                    k2
      endif
c
      if(iy.eq.1)then
      fjk=f(j,k)                                                       k
      fpjk=fp(j,k)                                                     k
      gjk=g(j,k)                                                       k
      gpjk=gp(j,k)                                                     k
c
      fj1k=f(j1,k)                                                     k
      fpj1k =fp(j1,k)                                                  k
      gj1k=g(j1,k)                                                     k
      gpj1k =gp(j1,k)                                                  k
c
      fj2k=f(j2,k)                                                     k
      fpj2k =fp(j2,k)                                                  k
      gj2k=g(j2,k)                                                     k
      gpj2k =gp(j2,k)                                                  k
c
      fj3k=fl(j3,k)                                                     k
      fpj3k =fpl(j3,k)                                                 k
      gj3k=gl(j3,k)                                                    k
      gpj3k =gpl(j3,k)                                                 k
c
      fj4k=fl(j4,k)                                                    k
      fpj4k =fpl(j4,k)                                                 k
      gj4k=gl(j4,k)                                                    k
      gpj4k =gpl(j4,k)                                                 k
c
      fj5k=fl(j5,k)                                                    k
      fpj5k =fpl(j5,k)                                                 k
      gj5k=gl(j5,k)                                                    k
      gpj5k =gpl(j5,k)                                                 k
c
      fj6k=fl(j6,k)                                                    k
      fpj6k =fpl(j6,k)                                                 k
      gj6k=gl(j6,k)                                                    k
      gpj6k =gpl(j6,k)                                                 k
c
      fj7k=fl(j7,k)                                                    k
      fpj7k =fpl(j7,k)                                                 k
      gj7k=gl(j7,k)                                                    k
      gpj7k =gpl(j7,k)                                                 k
c
      endif
      if(ix.eq.1)then
      fik =f(j,k )                                                     k
      filk =fl(j,k )                                                   k
      fpik =fp(j,k )                                                   k
      fpilk =fpl(j,k )                                                 k
      gik =g(j,k )                                                     k
      gilk =gl(j,k )                                                   k
      gpik =gp(j,k )                                                   k
      gpilk =gpl(j,k )                                                 k
      if(i.eq.2)then
      fi2k =0                                                          k
      gi2k =0                                                          k
      fpi2k =0                                                         k
      gpi2k =0                                                         k
      else
      fi2k =f2(j,k )                                                   k
      gi2k =g2(j,k )                                                   k
      fpi2k =fp2(j,k )                                                 k
      gpi2k =gp2(j,k )                                                 k
      endif
      endif
      fk =f(j,k )                                                      k
      gk =g(j,k )                                                      k
      wk =w(j,k )                                                      k
      fpk =fp(j,k )                                                    k
      gpk =gp(j,k )                                                    k
c
      return
      end
c-----------------------------------------------------------------------
      subroutine gethi(k)
c-----------------------------------------------------------------------
      include 'com'
      include 'comloc'
      include 'comtemp'
      if(k.eq.1)then
c
c     calculate the heat dissipation terms and store in array
c
```

```
      do 1 kk=1,nz
      call getfg(kk)
      d6l=dd6*al(kk)
      d6lp=dd6*alp(kk)
      bfk =bd*fjk +bd1*fj1k +bd2*fj2k +bd3*fj3k +bd4*fj4k +
     1     bd5*fj5k +bd6*fj6k +bd7*fj7k
      bggk =bd*gjk **2+bd1*gj1k **2+bd2*gj2k **2+
     1     bd3*gj3k **2+bd4*gj4k **2+
     1     bd5*gj5k **2+bd6*gj6k **2+bd7*gj7k **2
      fpp=(0.5*b1*(a1*f(j,kk)**2+a2*f1(j,kk)**2+a3*f2(j,kk)**2)+
     1    b2*g(j,kk)*bfk+
     1    b3s*f(j,kk)**2+b4s*f(j,kk)*g(j,kk)+b5*g(j,kk)**2+
     1    b6*the1(kk)+fp(j,kk)*(w(j,kk)-al1p(kk)))/al1(kk)
      gpp=(cc1*f(j,kk)*(a1*g(j,kk)+a2*g1(j,kk)+a3*g2(j,kk))+
     1    0.5*cc2*bggk+
     1    cc3*f(j,kk)**2+cc4s*f(j,kk)*g(j,kk)+cc5s*g(j,kk)**2+
     1    cc6*the1(kk)+gp(j,kk)*(w(j,kk)-al2p(kk)))/al2(kk)
      fpps(kk)=fpp
      gpps(kk)=gpp
c
c     calculate dd5 and dd5p arrays
c
      tem1(kk)=d6lp*(uden2*f(j,kk)*fp(j,kk)+vden2*g(j,kk)*gp(j,kk)+
     1         d7uv*(f(j,kk)*gp(j,kk)+fp(j,kk)*g(j,kk)))+
     1         d6l*(uden2*(fp(j,kk)**2+f(j,kk)*fpp)+
     1         vden2*(gp(j,kk)**2+g(j,kk)*gpp)+d7uv*(f(j,kk)*gpp+
     1         gp(j,kk)*fp(j,kk)+g(j,kk)*fpp+fp(j,kk)*gp(j,kk)))
 1    continue
      do 2 kk=1,nz
      tem2(kk)=derk(z,tem1,nz,kk)
 2    continue
c
c     TEMP
c
c     Replace by a 4th order accurate scheme for equispaced formula
c
c     dz=zmax/nz1
c     do 2 kk=1,nz
c     if(kk.eq.1.or.kk.eq.2)then
c     tem2(kk)=-1./(6*dz)*(11*tem1(kk)-18*tem1(kk+1)
c    1                     +9*tem1(kk+2)-2*tem1(kk+3))
c     endif
c     if(kk.eq.nz.or.kk.eq.nz1)then
c     tem2(kk)=-1./(6*dz)*(-11*tem1(kk)+18*tem1(kk-1)
c    1                     -9*tem1(kk-2)+2*tem1(kk-3))
c     endif
c     if(kk.gt.2.and.kk.lt.nz1)then
c     tem2(kk)=1./(12*dz)*(tem1(kk-2)-8*tem1(kk-1)
c    1                   +8*tem1(kk+1)-tem1(kk+2))
c     endif
c2    continue
      endif
c
      if(k.ne.1)then
      if(iy.eq.1)then
      hjk1=h(j,k1)                                              k1
      hpjk1=hp(j,k1)                                            k1
      hj1k1=h(j1,k1)                                            k1
      hpj1k1=hp(j1,k1)                                          k1
      hj2k1=h(j2,k1)                                            k1
      hpj2k1=hp(j2,k1)                                          k1
      hj3k1=hh1(j3,k1)                                          k1
      hpj3k1=hp1(j3,k1)                                         k1
      hj4k1=hh1(j4,k1)                                          k1
      hpj4k1=hp1(j4,k1)                                         k1
      hj5k1=hh1(j5,k1)                                          k1
      hpj5k1=hp1(j5,k1)                                         k1
      hj6k1=hh1(j6,k1)                                          k1
      hpj6k1=hp1(j6,k1)                                         k1
      hj7k1=hh1(j7,k1)                                          k1
      hpj7k1=hp1(j7,k1)                                         k1
      endif
      if(ix.eq.1)then
      hik1=h(j,k1)                                              k1
      hi1k1=hh1(j,k1)                                           k1
      hpik1=hp(j,k1)                                            k1
      hpi1k1=hp1(j,k1)                                          k1
      if(i.eq.2)then
      hi2k1=0                                                   k1
      hpi2k1=0                                                  k1
      else
      hi2k1=hh2(j,k1)                                           k1
      hpi2k1=hp2(j,k1)                                          k1
      endif
      endif
      hk1=h(j,k1)                                               k1
      hpk1=hp(j,k1)                                             k1
      dd5k1=tem1(k1)
      dd5pk1=tem2(k1)
      endif
c
      if(k.ne.nz)then
      if(iy.eq.1)then
      hjk2=h(j,k2)                                              k2
      hpjk2=hp(j,k2)                                            k2
      hj1k2=h(j1,k2)                                            k2
      hpj1k2=hp(j1,k2)                                          k2
      hj2k2=h(j2,k2)                                            k2
      hpj2k2=hp(j2,k2)                                          k2
      hj3k2=hh1(j3,k2)                                          k2
      hpj3k2=hp1(j3,k2)                                         k2
      hj4k2=hh1(j4,k2)                                          k2
      hpj4k2=hp1(j4,k2)                                         k2
      hj5k2=hh1(j5,k2)                                          k2
      hpj5k2=hp1(j5,k2)                                         k2
      hj6k2=hh1(j6,k2)                                          k2
      hpj6k2=hp1(j6,k2)                                         k2
      hj7k2=hh1(j7,k2)                                          k2
      hpj7k2=hp1(j7,k2)                                         k2
      endif
      if(ix.eq.1)then
      hik2=h(j,k2)                                              k2
      hi1k2=hh1(j,k2)                                           k2
      hpik2=hp(j,k2)                                            k2
      hpi1k2=hp1(j,k2)                                          k2
      if(i.eq.2)then
      hi2k2=0                                                   k2
      hpi2k2=0                                                  k2
      else
      hi2k2=hh2(j,k2)                                           k2
      hpi2k2=hp2(j,k2)                                          k2
      endif
      endif
      hk2=h(j,k2)                                               k2
      hpk2=hp(j,k2)                                             k2
      dd5k2=tem1(k2)
      dd5pk2=tem2(k2)
      endif
```

```
      if(iy.eq.1)then
      hjk=h(j,k)
      hpjk=hp(j,k)
      hj1k=h(j1,k)
      hpj1k =hp(j1,k)
      hj2k=h(j2,k)
      hpj2k =hp(j2,k)
      hj3k=hh1(j3,k)
      hpj3k =hp1(j3,k)
      hj4k=hh1(j4,k)
      hpj4k =hp1(j4,k)
      hj5k=hh1(j5,k)
      hpj5k =hp1(j5,k)
      hj6k=hh1(j6,k)
      hpj6k =hp1(j6,k)
      hj7k=hh1(j7,k)
      hpj7k =hp1(j7,k)
      endif
      if(ix.eq.1)then
      hik =h(j,k )
      hi1k =hh1(j,k )
      hpik =hp(j,k )
      hpi1k =hp1(j,k )
      if(i.eq.2)then
      hi2k =0
      hpi2k =0
      else
      hi2k =hh2(j,k )
      hpi2k =hp2(j,k )
      endif
      endif
      hk =h(j,k )
      hpk =hp(j,k )
      dd5k =tem1(k )
      dd5pk =tem2(k)

      return
      end
```

```
c-----------------------------------------------------------------------
      function deri22(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     calculates the i-derivative of yp w.r.t. xp at point i,j
c     unequally spaced,second order scheme
      dimension xp(im,jm),yp(im,jm)
c
c     first check if a boundary point; if yes, use one-sided diff.
c     if not, use central differences
c
      if(i.ne.1.and.i.ne.im)then
      dl=xp(i,j)-xp(i-1,j)
      dr=xp(i+1,j)-xp(i,j)
      dl2=dl*dl
      dr2=dr*dr
      deri22=(dl2*yp(i+1,j)+(dr2-dl2)*yp(i,j)
     1          -dr2*yp(i-1,j))/(dl2*dr+dr2*dl)
      return
      endif
      if(i.eq.1)then
      r1=xp(2,j)-xp(1,j)
      r2=xp(3,j)-xp(2,j)
      f=r1/(r1+r2)
      deri22=(yp(1,j)*(1.-f*f)-yp(2,j)+yp(3,j)*f*f)/(r1*(f-1.))
      return
      endif
      rn1=xp(im,j)-xp(im-1,j)
      rn2=xp(im-1,j)-xp(im-2,j)
      f=rn1/(rn1+rn2)
      deri22=(yp(im,j)*(f*f-1.)+yp(im-1,j)-yp(im-2,j)*f*f)/(rn1*(f-1.))
      return
      end
c-----------------------------------------------------------------------
      function derj22(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     calculates the j-derivative of yp w.r.t. xp at point i,j
c     xp and yp are 2-d arrays
c     unequally spaced,second order scheme
      dimension xp(im,jm),yp(im,jm)
c
c     first check if a boundary point; if yes, use one-sided diff.
c     if not, use central differences
c
      if(j.ne.1.and.j.ne.jm)then
      dl=xp(i,j)-xp(i,j-1)
      dr=xp(i,j+1)-xp(i,j)
      dl2=dl*dl
      dr2=dr*dr
      derj22=(dl2*yp(i,j+1)+(dr2-dl2)*yp(i,j)
     1          -dr2*yp(i,j-1))/(dl2*dr+dr2*dl)
      return
      endif
      if(j.eq.1)then
      r1=xp(i,2)-xp(i,1)
      r2=xp(i,3)-xp(i,2)
      f=r1/(r1+r2)
      derj22=(yp(i,1)*(1.-f*f)-yp(i,2)+yp(i,3)*f*f)/(r1*(f-1.))
      return
      endif
      rn1=xp(i,jm)-xp(i,jm-1)
      rn2=xp(i,jm-1)-xp(i,jm-2)
      f=rn1/(rn1+rn2)
      derj22= (yp(i,jm)*(f*f-1.)+yp(i,jm-1)-yp(i,jm-2)*f*f)/(rn1*(f-1.))
      return
      end
c-----------------------------------------------------------------------
      function derj21(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     xp is 1-d array; yp is 2-d array
c     calculates the j-derivative of yp w.r.t. xp at point i,j
c     unequally spaced,second order scheme
      dimension xp(jm),yp(im,jm)
c
c     first check if a boundary point; if yes, use one-sided diff.
c     if not, use central differences
c
      if(j.ne.1.and.j.ne.jm)then
      dl=xp(j)-xp(j-1)
      dr=xp(j+1)-xp(j)
      dl2=dl*dl
      dr2=dr*dr
      derj21=(dl2*yp(i,j+1)+(dr2-dl2)*yp(i,j)
     1          -dr2*yp(i,j-1))/(dl2*dr+dr2*dl)
      return
      endif
      if(j.eq.1)then
      r1=xp(2)-xp(1)
      r2=xp(3)-xp(2)
      f=r1/(r1+r2)
      derj21=(yp(i,1)*(1.-f*f)-yp(i,2)+yp(i,3)*f*f)/(r1*(f-1.))
      return
      endif
      rn1=xp(jm)-xp(jm-1)
      rn2=xp(jm-1)-xp(jm-2)
      f=rn1/(rn1+rn2)
      derj21=(yp(i,jm)*(f*f-1.)+yp(i,jm-1)-yp(i,jm-2)*f*f)/(rn1*(f-1.))
      return
      end
c-----------------------------------------------------------------------
      function deri21(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     xp is 1-d array; yp is 2-d array
c     calculates the i-derivative of yp w.r.t. xp at point i,j
c     unequally spaced,second order scheme
      dimension xp(im),yp(im,jm)
c
c     first check if a boundary point; if yes, use one-sided diff.
c     if not, use central differences
c
      if(i.ne.1.and.i.ne.im)then
      dl=xp(i)-xp(i-1)
      dr=xp(i+1)-xp(i)
      dl2=dl*dl
      dr2=dr*dr
      deri21=(dl2*yp(i+1,j)+(dr2-dl2)*yp(i,j)
     1          -dr2*yp(i-1,j))/(dl2*dr+dr2*dl)
      return
      endif
      if(i.eq.1)then
      r1=xp(2)-xp(1)
      r2=xp(3)-xp(2)
      f=r1/(r1+r2)
      deri21=(yp(1,j)*(1.-f*f)-yp(2,j)+yp(3,j)*f*f)/(r1*(f-1.))
      return
      endif
      rn1=xp(im)-xp(im-1)
      rn2=xp(im-1)-xp(im-2)
      f=rn1/(rn1+rn2)
```

```
      deri21=(yp(im,j)*(f*f-1.)+yp(im,j-1)-yp(im,j-2)*f*f)/(rn1*(f-1.))
      return
      end
c-----------------------------------------------------------------------
      function derk(z,f,nz,k)
c-----------------------------------------------------------------------
c     calculates the k-derivative of f w.r.t. z at point k
c     z and f are 1-d arrays
c     unequally spaced,second order scheme
      dimension z(nz),f(nz)
c
c     first check if a boundary point; if yes, use one-sided diff.
c     if not, use central differences
c
      if(k.ne.1.and.k.ne.nz)then
      dl=z(k)-z(k-1)
      dr=z(k+1)-z(k)
      dl2=dl*dl
      dr2=dr*dr
      derk =(dl2*f(k+1)+(dr2-dl2)*f(k)
     1          -dr2*f(k-1))/(dl2*dr+dr2*dl)
      return
      endif
      if(k.eq.1)then
      r1=z(2)-z(1)
      r2=z(3)-z(2)
      ff=r1/(r1+r2)
      derk =(f(1)*(1.-ff*ff)-f(2)+f(3)*ff*ff)/(r1*(ff-1.))
      return
      endif
      rn1=z(nz)-z(nz-1)
      rn2=z(nz-1)-z(nz-2)
      ff=rn1/(rn1+rn2)
      derk = (f(nz)*(ff*ff-1.)+f(nz-1)-f(nz-2)*ff*ff)/(rn1*(ff-1.))
      return
      end
c-----------------------------------------------------------------------
      function sumint(xp,yp,im,i)
c-----------------------------------------------------------------------
c
c     integrates the function yp wrt xp
c     from 1 to i (integration in the i direction)
c     simple trapezoidal formula used
c     xp and yp are 1-d arrays
c
      dimension xp(im), yp(im)
      if(i.eq.1)then
      sumint=0.
      return
      endif
c
      sum=0.
      do 1 ii=2,i
      ii1=ii-1
      sum=sum+0.5*(yp(ii  )+yp(ii1  ))*(xp(ii  )-xp(ii1  ))
 1    continue
      sumint=sum
      return
      end
c-----------------------------------------------------------------------
      function sumx21(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     xp s 1D array and yp is a 2-d array
c
c     integrates the function yp wrt xp
c     from 1 to i (integration in the i direction)
c     simple trapezoidal formula used
c
      dimension xp(im), yp(im,jm)
      if(i.eq.1)then
      sumx21=0.
      return
      endif
c
      sum=0.
      do 1 ii=2,i
      ii1=ii-1
      sum=sum+0.5*(yp(ii,j)+yp(ii1,j))*(xp(ii)-xp(ii1))
 1    continue
      sumx21=sum
      return
      end
c-----------------------------------------------------------------------
      function sumx22(i,j,im,jm,xp,yp)
c-----------------------------------------------------------------------
c     xp and yp 2-d arrays
c
c     integrates the function yp wrt xp
c     from 1 to i (integration in the i direction)
c     simple trapezoidal formula used
c
      dimension xp(im,jm), yp(im,jm)
      if(i.eq.1)then
      sumx22=0.
      return
      endif
c
      sum=0.
      do 1 ii=2,i
      ii1=ii-1
      sum=sum+0.5*(yp(ii,j)+yp(ii1,j))*(xp(ii,j)-xp(ii1,j))
 1    continue
      sumx22=sum
      return
      end
c-----------------------------------------------------------------------
      subroutine blklu(a,b,c,n,m,m2,ir,ic,f)
c-----------------------------------------------------------------------
c
c     this routine solves n block tridiagonal equations with mxm blocks
c     ( m2 = m**2 ).
c
      real a(m2,*),b(m2,*),c(m2,*)
      real d(200),f(m,1)
c
c     if m is greater than 10, d array must be redimensioned
c
      dimension ir(m,1),ic(m,1)
      call lu(b,m,m,ir,ic)
      do 1 i=2,n
      call solve(c(1,i-1),b(1,i-1),m,m,m,ir(1,i-1),ic(1,i-1))
      call cmult(a(1,i),c(1,i-1),d,m,m)
      do 2 j=1,m2
2     b(j,i)=b(j,i)-d(j)
1     call lu(b(1,i),m,m,ir(1,i),ic(1,i))
      return
      entry blksolv(a,b,c,n,m,m2,ir,ic,f)
      call solve(f,b,1,m,m,ir,ic)
```

```
      do 3 i=2,n
      call ax(a(1,i),f(1,i-1),d,m,m)
      do 4 k=1,m
4     f(k,i)=f(k,i)-d(k)
3     call solve(f(1,i),b(1,i),1,m,m,ir(1,i),ic(1,i))
      do 5 ii=2,n
      i=n+1-ii
      call ax(c(1,i),f(1,i+1),d,m,m)
      do 5 j=1,m
5     f(j,i)=f(j,i)-d(j)
      return
      end
c-----------------------------------------------------------------------
      subroutine cmult(a,b,c,m,ndim)
c-----------------------------------------------------------------------
      real a(ndim,*),b(ndim,*),c(ndim,*)
      do 1 i=1,m
      do 1 j=1,m
      c(i,j)=0.0
      do 1 k=1,m
1     c(i,j)=c(i,j)+a(i,k)*b(k,j)
      return
      end
c-----------------------------------------------------------------------
      subroutine lu (a,n,ndim,ir,ic)
c-----------------------------------------------------------------------
c
c     this routine does fully pivoted lu decomp. on a
c
      real a(ndim,*),b,c
      dimension ir(1), ic(1)
      do 1 i=1,n
      ir(i)=i
1     ic(i)=i
      k=1
      l=k
      m=k
      xmax=abs(a(k,k))
      do 2 i=k,n
      do 2 j=k,n
      y=abs(a(i,j))
      if (xmax.ge.y) go to 2
      xmax=y
      l=i
      m=j
2     continue
      do 9 k=1,n
      irl=ir(l)
      ir(l)=ir(k)
      ir(k)=irl
      icm=ic(m)
      ic(m)=ic(k)
      ic(k)=icm
      if (l.eq.k) go to 4
      do 3 j=1,n
      b=a(k,j)
      a(k,j)=a(l,j)
3     a(l,j)=b
4     if (m.eq.k) go to 6
      do 5 i=1,n
      b=a(i,k)
      a(i,k)=a(i,m)
5     a(i,m)=b
6     c=1./a(k,k)
      a(k,k)=c
      if (k.eq.n) go to 9
      k1=k+1
      xmax=(a(k1,k1))
      l=k1
      m=k1
      do 7 i=k1,n
7     a(i,k)=c*a(i,k)
      do 8 i=k1,n
      b=a(i,k)
      do 8 j=k1,n
      a(i,j)=a(i,j)-b*a(k,j)
      y=abs(a(i,j))
      if (xmax.ge.y) go to 8
      xmax=y
      l=i
      m=j
8     continue
9     continue
      return
      end
c-----------------------------------------------------------------------
      subroutine solve (f,a,k,n,ndim,ir,ic)
c-----------------------------------------------------------------------
c
c     this routine solves k equations ax = f (dimensioned ndim)
c     assume previous call to routine lu has been made
c
      real a(ndim,1),f(ndim,1),g(100),b
      dimension ir(1), ic(1)
      n1=n+1
      do 7 kk=1,k
      do 1 i=1,n
      iri=ir(i)
1     g(i)=f(iri,kk)
      do 3 i=2,n
      i1=i-1
      b=g(i)
      do 2 j=1,i1
2     b=b-a(i,j)*g(j)
3     g(i)=b
      do 5 it=1,n
      i=n1-it
      i1=i+1
      b=g(i)
      if (i.eq.n) go to 5
      do 4 j=i1,n
4     b=b-a(i,j)*g(j)
5     g(i)=b*a(i,i)
      do 6 i=1,n
      ici=ic(i)
6     f(ici,kk)=g(i)
7     continue
      return
      end
c-----------------------------------------------------------------------
      subroutine ax(a,x,y,n,ndim)
c-----------------------------------------------------------------------
      real a(ndim,1),x(1),y(1)
      do 2 i=1,n
      y(i)=0.
      do 1 j=1,n
1     y(i)=y(i)+a(i,j)*x(j)
2     continue
```

```
      return
      end
c-----------------------------------------------------------------
      function quad(x1,x2,x3,y1,y2,y3,x)
c-----------------------------------------------------------------
c     performs a simple quadratic interpolation/extrapolation (lagrange)
c     (x1,y1), (x2,y2), (x3,y3)  given;  finds y at x  by 2nd order poly
      h1=x2-x1
      h2=x3-x2
      h3=x3-x1
      f1=(x-x2)*(x-x3)/(h1*h3)
      f2=(x-x1)*(x-x3)/(h1*h2)
      f3=(x-x1)*(x-x2)/(h2*h3)
      quad=f1*y1-f2*y2+f3*y3
      return
      end
c-----------------------------------------------
      function stlin(x1,xr,v1,vr,xm)
c-----------------------------------------------
c     perform linear extra/inter polation
c     (x1,v1), (xr,vr)  given; finds ym at a certain xm
      ff=(xm-x1)/(xr-x1)
      f=1.-ff
      stlin=ff*vr+f*v1
      return
      end
c-----------------------------------------------------------------
      subroutine solv3(a,b)
c-----------------------------------------------------------------
c     solves the 3x3 system a.x=b; solution returned in b
c
      dimension a(3,3),b(3,1),c(3,3),d(3,1)
      c(1,1)=a(2,2)*a(3,3)-a(3,2)*a(2,3)
      c(1,2)=a(3,2)*a(1,3)-a(1,2)*a(3,3)
      c(1,3)=a(1,2)*a(2,3)-a(2,2)*a(1,3)
      c(2,1)=a(3,1)*a(2,3)-a(2,1)*a(3,3)
      c(2,2)=a(1,1)*a(3,3)-a(3,1)*a(1,3)
      c(2,3)=a(2,1)*a(1,3)-a(1,1)*a(2,3)
      c(3,1)=a(2,1)*a(3,2)-a(3,1)*a(2,2)
      c(3,2)=a(3,1)*a(1,2)-a(1,1)*a(3,2)
      c(3,3)=a(1,1)*a(2,2)-a(2,1)*a(1,2)
      del=a(1,1)*(a(2,2)*a(3,3)-a(2,3)*a(3,2))
     #   +a(1,2)*(a(2,3)*a(3,1)-a(2,1)*a(3,3))
     #   +a(1,3)*(a(2,1)*a(3,2)-a(2,2)*a(3,1))
      do 1 i=1,3
      do 1 j=1,3
      c(i,j)=c(i,j)/del
 1    continue
      do 2 i=1,3
      sum=0.
      do 3 j=1,3
      sum=sum+c(i,j)*b(j,1)
 3    continue
      d(i,1)=sum
 2    continue
      do 4 i=1,3
 4    b(i,1)=d(i,1)
      return
      end
c-----------------------------------------------------
      function sind(x)
c-----------------------------------------------------
      y=x*3.1415927/180.
      sind=sin(y)
      return
      end
c-----------------------------------------------------
      function cosd(x)
c-----------------------------------------------------
      y=x*3.1415927/180.
      cosd=cos(y)
      return
      end
c-----------------------------------------------------
      function tand(x)
c-----------------------------------------------------
      y=x*3.1415927/180.
      tand=tan(y)
      return
      end
```

```
      parameter(nx=100,ny=15,nz=41)
      common/vgrad  /vey(nx)
      common/duedx  /fe(ny)
      common/qflux  /qd(nx,ny)
      common/prsta  /nppi,nppj,ip(100),jp(100)
      common/fgpp   /fpps(nz),gpps(nz)
      common/ueve   /ue(nx,ny),ve(nx,ny)
      common/teroe  /te(nx,ny),roe(nx,ny),amue(nx,ny)
      common/wall   /tws(nx,ny),wws(nx,ny)
      common/pres   /pb(nx,ny)
      common/xyb    /xb(nx,ny),yb(nx,ny),zb(nx,ny)
      common/xy     /x(nx),y(ny)
      common/met    /h1(nx,ny),h2(nx,ny),g12(nx,ny)
      common/zz     /z(nz)
      common/fg     /f(ny,nz),g(ny,nz),fp(ny,nz),gp(ny,nz),w(ny,nz)
      common/fg1    /f1(ny,nz),g1(ny,nz),fp1(ny,nz),gp1(ny,nz)
      common/fg2    /f2(ny,nz),g2(ny,nz),fp2(ny,nz),gp2(ny,nz)
      common/hi     /h(ny,nz),hp(ny,nz)
      common/hi1    /hh1(ny,nz),hp1(ny,nz)
      common/hi2    /hh2(ny,nz),hp2(ny,nz)
      common/all2   /all(nz),al2(nz),all2p(nz),al2p(nz)
      common/ale    /all(nz),allp(nz),al(nz),alp(nz)
      common/the    /the1(nz),the2(nz)
      common/misc   /amach,zmax,gam,prl,itmax,epsf,epsh,
     1               pfs,tfs,c1,c2,c3,c4,rstar,sstar,nz1,
     1               amustd,rofs,velson,amuinf,reinff,ufs,trs,
     1               amuref,reref,tf,hcon,hrefs,htot,stilde,
     1               pi,iwall,ak,refsf,amui,bet,crs,reflen,
     1               fedge,gedge,iunit,iord
      common/edge   /aa1s,aa2s,aa3s,aa4s,
     1               b1,b2,b3,b4,b5,b6,b3s,b4s,
     1               cc1,cc2,cc3,cc4,cc5,cc6,cc4s,cc5s,
     1               dd1,dd2,dd3,dd4,dd6,dd7,stild,ww,hpw,
     1               uden,vden,hw,uden2,vden2,d7uv,uvden,s1,phil,psil
      common/uc     /a1,a2,a3
      common/vc     /bd,bd1,bd2,bd3,bd4,bd5,bd6,bd7
      common/ij     /i,j,j1,j2,j3,j4,j5,j6,j7
      common/ixy    /ix,iy,jbl
      common/convm  /delfp(nz)
      common/conve  /delhp(nz)
      common/stop   /istop
      common/limits /nxlim,nylim
```

```
common/k1k2    /k1,k2
common/dzm     /c1dz,c2dz,c3dz2,c4dz2,d1dz,d2dz,d3dz2,d4dz2
common/fgk     /fk,gk,wk,fk1,gk1,wk1,fk2,gk2,wk2
common/fgpk    /fpk,gpk,fpk1,gpk1,fpk2,gpk2
common/fgik    /fik  ,gik  ,fpik  ,gpik
common/fgik1   /fik1 ,gik1 ,fpik1 ,gpik1
common/fgik2   /fik2 ,gik2 ,fpik2 ,gpik2
common/fgi1k   /fi1k ,gi1k ,fpi1k ,gpi1k
common/fgi1k1  /fi1k1,gi1k1,fpi1k1,gpi1k1
common/fgi1k2  /fi1k2,gi1k2,fpi1k2,gpi1k2
common/fgi2k   /fi2k ,gi2k ,fpi2k ,gpi2k
common/fgi2k1  /fi2k1,gi2k1,fpi2k1,gpi2k1
common/fgi2k2  /fi2k2,gi2k2,fpi2k2,gpi2k2
common/fgjk    /fjk  ,gjk  ,fpjk  ,gpjk
common/fgjk1   /fjk1 ,gjk1 ,fpjk1 ,gpjk1
common/fgjk2   /fjk2 ,gjk2 ,fpjk2 ,gpjk2
common/fgj1k   /fj1k ,gj1k ,fpj1k ,gpj1k
common/fgj1k1  /fj1k1,gj1k1,fpj1k1,gpj1k1
common/fgj1k2  /fj1k2,gj1k2,fpj1k2,gpj1k2
common/fgj2k   /fj2k ,gj2k ,fpj2k ,gpj2k
common/fgj2k1  /fj2k1,gj2k1,fpj2k1,gpj2k1
common/fgj2k2  /fj2k2,gj2k2,fpj2k2,gpj2k2
common/fgj3k   /fj3k ,gj3k ,fpj3k ,gpj3k
common/fgj3k1  /fj3k1,gj3k1,fpj3k1,gpj3k1
common/fgj3k2  /fj3k2,gj3k2,fpj3k2,gpj3k2
common/fgj4k   /fj4k ,gj4k ,fpj4k ,gpj4k
common/fgj4k1  /fj4k1,gj4k1,fpj4k1,gpj4k1
common/fgj4k2  /fj4k2,gj4k2,fpj4k2,gpj4k2
common/fgj5k   /fj5k ,gj5k ,fpj5k ,gpj5k
common/fgj5k1  /fj5k1,gj5k1,fpj5k1,gpj5k1
common/fgj5k2  /fj5k2,gj5k2,fpj5k2,gpj5k2
common/fgj6k   /fj6k ,gj6k ,fpj6k ,gpj6k
common/fgj6k1  /fj6k1,gj6k1,fpj6k1,gpj6k1
common/fgj6k2  /fj6k2,gj6k2,fpj6k2,gpj6k2
common/fgj7k   /fj7k ,gj7k ,fpj7k ,gpj7k
common/fgj7k1  /fj7k1,gj7k1,fpj7k1,gpj7k1
common/fgj7k2  /fj7k2,gj7k2,fpj7k2,gpj7k2
common/hhpk    /hk,hpk,hk1,hpk1,hk2,hpk2
common/hhpjk   /hjk,hpjk,hjk1,hpjk1,hjk2,hpjk2
common/hhpj1k  /hj1k,hpj1k,hj1k1,hpj1k1,hj1k2,hpj1k2
common/hhpj2k  /hj2k,hpj2k,hj2k1,hpj2k1,hj2k2,hpj2k2
common/hhpj3k  /hj3k,hpj3k,hj3k1,hpj3k1,hj3k2,hpj3k2
common/hhpj4k  /hj4k,hpj4k,hj4k1,hpj4k1,hj4k2,hpj4k2
common/hhpj5k  /hj5k,hpj5k,hj5k1,hpj5k1,hj5k2,hpj5k2
common/hhpj6k  /hj6k,hpj6k,hj6k1,hpj6k1,hj6k2,hpj6k2
common/hhpj7k  /hj7k,hpj7k,hj7k1,hpj7k1,hj7k2,hpj7k2
common/hikc    /hik,hi1k,hi2k,hik1,hi1k1,hi2k1,hik2,hi1k2,hi2k2
```

```
common/hpikc  /hpik,hpi1k,hpi2k
common/hpik1c /hpik1,hpi1k1,hpi2k1
common/hpik2c /hpik2,hpi1k2,hpi2k2
common/dd5v   /dd5k1,dd5pk1,dd5k2,dd5pk2,dd5k,dd5pk
```

```
      parameter(ii=70,ji=15)
      common/erri   /erru,errv
      common/xyi    /xh(ii,ji),yh(ii,ji),zh(ii,ji)
      common/uve    /uh(ii,ji),vh(ii,ji),wh(ii,ji)
      common/pinv   /ph(ii,ji)
      common/wl     /xww(ii),yww(ii),zww(ii),xlw(ii),ylw(ii),zlw(ii)
      common/gri    /ds(ji),ys(ii,ji),xs(ii,ji)
      common/uvs    /us(ii,ji),vs(ii,ji),ws(ii,ji)
      common/index  /ih(ji)
```

```
      common/temp   /tem1(nz),tem2(nz)
      common/tem    /xpt(10*nx),ypt(10*nx)
      common/metcf  /s(6,nx,ny)
```

```
parameter(nnmax=1000)
common/splcom/yss(nnmax),wk(nnmax)
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-4269		

4. Title and Subtitle

Computation of Three-Dimensional Compressible Boundary Layers to Fourth-Order Accuracy on Wings and Fuselages

5. Report Date

January 1990

6. Performing Organization Code

7. Author(s)

Venkit Iyer

8. Performing Organization Report No.

10. Work Unit No.

505-60-31-05

9. Performing Organization Name and Address

Vigyan, Inc.
Hampton, VA 23666

11. Contract or Grant No.

NAS1-18585

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

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 40

16. Abstract

A solution method, fourth-order accurate in the body-normal direction and second-order accurate in the stream surface directions, to solve the compressible 3D boundary layer equations is presented. The report describes the transformation used, the discretization details and the solution procedure. Ten validation cases of varying complexity are presented and results of calculation given. They range from subsonic flow to supersonic flow and involve 2D or 3D geometries. Applications to laminar flow past wing and fuselage-type bodies are discussed. An interface procedure is used to solve the surface Euler equations with the inviscid flow pressure field as the input to assure accurate boundary conditions at the boundary layer edge. Complete details of the computer programs used and information necessary to run each of the test cases are given in the Appendix.

17. Key Words (Suggested by Author(s))

Boundary layer
Computer program
Compressible
Laminar
Three-dimensional
Wing flows
Fuselage flows

18. Distribution Statement

[redacted]
(Review for general release January 31, 1994.)
Subject Category 34

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	276	

NASA FORM 1626 OCT 86