A NEW TRANSFORMATION AND INTEGRATION SCHEME FOR THE COMPRESSIBLE BOUNDARY LAYER EQUATIONS, AND SOLUTION BEHAVIOR AT SEPARATION

by

MARK DRELA

S.B. Massachusetts Institute of Technology
(1982)

Submitted to the Department of
Aeronautics and Astronautics
in Partial Fulfillment of the
Requirements of the Degree of

MASTER OF SCIENCE IN
AERONAUTICS AND ASTRONAUTICS
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 1983

Signature of Author ______________________________
Department of Aeronautics and Astronautics, May 1983

Certified by [illegible]
W.T. Thompkins, Jr. Thesis Supervisor

Accepted by ______________________________
Harold Y. Wachman Chairman, Aeronautics and Astronautics Graduate Committee

A NEW TRANSFORMATION AND INTEGRATION SCHEME FOR THE COMPRESSIBLE BOUNDARY LAYER EQUATIONS, AND SOLUTION BEHAVIOR AT SEPARATION

by

MARK DRELA

Submitted to the
Department of Aeronautics and Astronautics
on May 6, 1983 in partial fulfillment of the
requirements for the Degree of Master of Science in
Aeronautics and Astronautics.

ABSTRACT

A new coordinate and variable transformation for the two-dimensional boundary layer equations is presented. The normal coordinate is stretched with a scaling length determined by the local solution. The boundary layer thickness is then essentially constant in computational space for most types of flows, including separation bubbles and rapidly growing turbulent boundary layers. Similarity solutions can be obtained for all wedge flows.

Two finite difference schemes are presented: the Shifted Box Scheme and the Double-Shifted Box Scheme. Both schemes are more resistant to streamwise profile oscillations than the standard Keller's Box Scheme. All governing equations, including the turbulence model, are solved simultaneously as a fully coupled system. This is faster and more robust than conventional weak-coupling iteration schemes. The solution scheme implementation presented makes no restriction on one boundary condition. Any point or integral quantity such as edge velocity, wall shear, displacement thickness, or some functional relationship between two or more of such quantities can be prescribed.

The behavior of the boundary layer solution near separation is investigated. It is demonstrated that non-unique solutions always exist whenever an adverse pressure gradient is specified. This bifurcation of the solution is responsible for inability of calculations with prescribed pressure or edge velocity to be carried past separation.

Thesis Supervisor: William T. Thompkins Jr.

Title: Associate Professor of Aeronautics and Astronautics.

ACKNOWLEDGEMENTS

I would like to thank Professor William T. Thompkins Jr. for his advice and encouragement throughout this research. I also give thanks to all my fellow students who provided ample diversion in addition to their valuable discussion.

I am very grateful for the financial support provided by the NASA Training Grant NGT-22-009-901 under the direction of Randy Graves and also by NASA Lewis Research Center grant no. NGL-22-009-383 under the direction of A. J. Strazisar.

INTRODUCTION

The primary purpose of this thesis is to develop a new, efficient, versatile finite-difference method for the solution of the compressible boundary layer equations. The method differs in several ways from the other methods which currently exist, such as those of Carter [2] and Cebeci and Smith [6]. Most of these methods use some form of the unnecessarily complicated Levy-Lees transformation, in which the streamwise node locations usually depend on the solution. To simplify the application of the present method to viscous-inviscid coupling, the streamwise coordinate is not transformed. The normal coordinate is simply scaled by a length which is roughly proportional to the boundary layer thickness for virtually all types of flow found in practice. Thus the boundary layer always remains within the computational grid.

It is found that the popular Keller's Box Scheme discretization as found in Cebeci and Bradshaw [4] is not suitable for solving the governing equations with the present transformation, since it is susceptible to streamwise profile and wall shear stress oscillations. The reason for this behavior is investigated and two new discretization schemes are introduced to eliminate the problem.

Most real flow situations involve turbulence, and hence some form of turbulence modeling is necessary for practical calculations. For simplicity, the popular Cebeci-Smith two-layer algebraic eddy viscosity model obtained from Cebeci and Smith [6] is used in this thesis.

In the Newton-Raphson procedure used to solve the non-linear finite difference equations most methods found in literature neglect the coupling between some of the governing equations. In particular, the eddy viscosity formulas are not linearized, possibly in the belief that it is not important or just to simplify programming. The solution method in this thesis solves all governing equations simultaneously. This is demonstrated to produce large reductions in computation time.

The final unique feature of this method is versatility. With most other methods one is restricted to either a so-called direct mode, where the edge velocity is prescribed, or an inverse mode, where the displacement thickness is prescribed. This method makes no particular distinction between direct and inverse modes. Any quantity can be prescribed in lieu of the edge velocity or displacement thickness. This feature is very useful for design work. For instance, by specifing a zero wall shear everywhere one can determine the fastest pressure recovery possible without separation. Efficient viscous-inviscid coupling can be achieved by prescribing a functional relationship between edge velocity and displacement thickness. Four different types of prescribed quantities are programmed demonstrating the flexibility of the solution scheme.

A secondary purpose of this thesis is ~~is~~ to investigate the well-known inability of all direct solution schemes to calculate a solution past a separation point. Using the developed program it is shown that there are always two solutions to the finite difference equations whenever a decelerating edge velocity is prescribed and that near separation these two solutions approach each other causing the failure of the Newton-Raphson algorithm. It is also shown that it is possible to prescribe an edge velocity for which there is no solution to the finite difference equations.

ANALYSIS

Equations (1-5) are the two-dimensional, compressible, boundary layer equations written as a first-order system. An eddy viscosity and turbulent Prandtl number have been included to allow for turbulence modeling. Bars denote dimensioned quantites. The "e" subscript denotes edge, or freestream quantities.

continuity:
$$\frac{\partial(\bar{\rho}\bar{u})}{\partial\bar{x}} + \frac{\partial(\bar{\rho}\bar{v})}{\partial\bar{y}} = 0 \tag{1}$$

$\bar{x}$-momentum:
$$\bar{\rho}\bar{u}\,\frac{\partial\bar{u}}{\partial\bar{x}} + \bar{\rho}\bar{v}\,\frac{\partial\bar{u}}{\partial\bar{y}} = \frac{\partial\bar{\tau}}{\partial\bar{y}} + \bar{\rho}_e\bar{u}_e\,\frac{d\bar{u}_e}{d\bar{x}} \tag{2}$$

total enthalpy:
$$\bar{\rho}\bar{u}\,\frac{\partial\bar{h}}{\partial\bar{x}} + \bar{\rho}\bar{v}\,\frac{\partial\bar{h}}{\partial\bar{y}} = \frac{\partial\bar{q}}{\partial\bar{y}} \tag{3}$$

shear:
$$\bar{\tau} = (\bar{\mu} + \bar{\mu}_t)\,\frac{\partial\bar{u}}{\partial\bar{y}} \tag{4}$$

enthalpy flux:
$$\bar{q} = \left(\frac{\bar{\mu}}{Pr} + \frac{\bar{\mu}_t}{Pr_t}\right)\frac{\partial\bar{h}}{\partial\bar{y}} + \bar{\mu}\left(1 - \frac{1}{Pr}\right)\bar{u}\,\frac{\partial\bar{u}}{\partial\bar{y}} \tag{5}$$

With the reference quantities L, ρ_o, μ_o, T_o, $a_o = \sqrt{\gamma R T_o}$, and $Re_o = \rho_o a_o L/\mu_o$, non-dimensional variables are defined as follows:

$$x = \frac{\bar{x}}{L} \qquad y = \frac{\bar{y}}{L}\sqrt{Re_o} \tag{6a-b}$$

$$f = \frac{\bar{\Psi}}{\rho_o a_o L}\sqrt{Re_o} \qquad u = \frac{\bar{u}}{a_o} \qquad h = \frac{\bar{h}}{a_o^2} \tag{6c-e}$$

$$\tau = \frac{\bar{\tau}}{\rho_o a_o^2}\sqrt{Re_o} \qquad q = \frac{\bar{q}}{\rho_o a_o^3}\sqrt{Re_o} \tag{6f-g}$$

$$\mu = \frac{\bar{\mu}}{\mu_o} \qquad \mu_t = \frac{\bar{\mu}_t}{\mu_o} \tag{6h-i}$$

where $\bar{\Psi}$ represents the usual dimensioned stream function.

The computational coordinates x and η used in this analysis are defined as:

$$x = x \qquad \eta = \frac{y}{\Delta} \tag{7a-b}$$

$\Delta = \Delta(x)$ is a scaling length which depends on the solution itself. It will be defined later.

With the above definitions, equations (1-5) become:

$$\rho u \Delta = \frac{\partial f}{\partial \eta} \tag{8}$$

$$\frac{\partial f}{\partial \eta}\frac{\partial u}{\partial x} - \frac{\partial f}{\partial x}\frac{\partial u}{\partial \eta} = \frac{\partial \tau}{\partial \eta} + \rho_e u_e \Delta \frac{du_e}{dx} \tag{9}$$

$$\frac{\partial f}{\partial \eta}\frac{\partial h}{\partial x} - \frac{\partial f}{\partial x}\frac{\partial h}{\partial \eta} = \frac{\partial q}{\partial \eta} \tag{10}$$

$$\tau \Delta = (\mu + \mu_t)\frac{\partial u}{\partial \eta} \tag{11}$$

$$q\Delta = \left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial h}{\partial \eta} + \mu\left(1 - \frac{1}{Pr}\right) u \frac{\partial u}{\partial \eta} \tag{12}$$

Equations (8-12) are singular at a leading edge, and therefore cannot be used to generate a similarity solution to start streamwise marching. To remove this singularity, the dependent variables are scaled with appropriate local reference values, giving the following transformed variables (in uppercase):

$$F = \frac{f}{n} \qquad \text{where} \qquad n = \rho_e u_e \Delta \tag{13a-b}$$

$$U = \frac{u}{u_e} \qquad H = \frac{h}{h_e} \qquad R = \frac{\rho}{\rho_e} \tag{13c-e}$$

$$S = \frac{1}{n}\frac{x}{u_e}\tau \qquad Q = \frac{1}{n}\frac{x}{h_e} q \tag{13f-g}$$

$$\beta_u = \frac{x}{u_e}\frac{du_e}{dx} \qquad \beta_h = \frac{x}{h_e}\frac{dh_e}{dx} \qquad \beta_n = \frac{x}{n}\frac{dn}{dx} \tag{14a-c}$$

The resulting equation set with relevant boundary conditions is:

$$RU = \frac{\partial F}{\partial \eta} \tag{15}$$

$$\frac{\partial S}{\partial \eta} + \beta_n F \frac{\partial U}{\partial \eta} + \beta_u\left(1 - U\frac{\partial F}{\partial \eta}\right) = x\left(\frac{\partial F}{\partial \eta}\frac{\partial U}{\partial x} - \frac{\partial F}{\partial x}\frac{\partial U}{\partial \eta}\right) \tag{16}$$

$$\frac{\partial Q}{\partial \eta} + \beta_n F \frac{\partial H}{\partial \eta} - \beta_h H \frac{\partial F}{\partial \eta} = x \left(\frac{\partial F}{\partial \eta}\frac{\partial H}{\partial x} - \frac{\partial F}{\partial x}\frac{\partial H}{\partial \eta} \right) \tag{17}$$

$$S = \frac{\rho_e u_e x}{n^2} \left(\mu + \mu_t \right) \frac{\partial U}{\partial \eta} \tag{18}$$

$$Q = \frac{\rho_e u_e x}{n^2} \left(\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t} \right) \frac{\partial H}{\partial \eta} + \mu \left(1 - \frac{1}{Pr} \right) \frac{u_e^2}{h_e} U \frac{\partial U}{\partial \eta} \right) \tag{19}$$

Boundary conditions:

$\eta = 0$:	1) $U = 0$	(20a)
	2) $F = 0$	(20b)
	3) $H = H_w$ or $Q = Q_w$	(20c)
$\eta = \eta_e$:	4) $U = 1$	(20d)
	5) $H = 1$	(20e)

In virtually all practical situations, the outer flow is adiabatic, and hence β_h is zero. This quantity will therefore be ignored in the ensuing discussion.

Using equations (15-20), the calculation of Falkner-Skan type similarity solutions is straightforward, provided the requirements for similarity are satisfied. For similarity, the lefthand sides of equations (16) and (17) must be independent of x, and therefore β_u and β_n must be constants. By integrating equations (14a) and (14c), one concludes that $u_e(x)$ and $n(x)$ must be of the form:

$$u_e(x) \sim x^{\beta_u} \qquad n(x) \sim x^{\beta_n} \tag{21a-b}$$

To make the grouping $\rho_e u_e x/n^2$ in equations (18) and (19) independent of x, β_n must be related to β_u by

$$\beta_n = \frac{1 + \beta_u}{2} \tag{22}$$

Finally, of the remaining x-dependent quantities, ρ_e must be constant, and $u_e{}^2/h_e$ and μ_t must be either constant or negligibly small near the leading edge.

Fortunately, all these requirements are satisfied for laminar wedge flows in the vicinity of the leading edge, provided that $\Delta(x)$ varies with x as follows:

$$\Delta(x) \sim x^{\beta_\Delta} \qquad \text{where} \qquad \beta_\Delta = \frac{1 - \beta_u}{2} \qquad (23a\text{-}b)$$

For the zero pressure gradient case ($\beta_u = 0$), ρ_e and $u_e{}^2/h_e$ are indeed constant, assuring similarity. For ($\beta_u > 0$), near-stagnation conditions exist in the vicinity of the leading edge. In this case, ρ_e is nearly equal to its constant stagnation value, and $u_e{}^2/h_e$ is negligible, again producing similarity within some small interval close to the leading edge.

It only remains to specify the scaling length Δ to close equations (15-19). Although Δ is arbitrary, it is desirable that it satisfy equations (23a-b) so that similarity solutions can be obtained. Ideally, Δ is proportional to some nominal boundary layer thickness δ for nonsimilar as well as similar flows. If δ/Δ is constant, then the boundary layer thickness in the computational x-η space is constant, and grid extension is never necessary during marching calculations.

Several various definitions of Δ have been tried, including the displacement thickness and the momentum thickness. The definition selected as most suitable is:

$$\Delta(x) = \int_0^{y_e} U(1 - U)\, dy \qquad \text{implying} \qquad 1 = \int_0^{\eta_e} U(1 - U)\, d\eta \qquad (24a\text{-}b)$$

This corresponds to the momentum thickness in the incompressible limit. With this definition, the ratio δ/Δ varies by no more than 10% for such diverse flows as laminar separation bubbles and rapidly growing turbulent boundary layers.

SOLUTION SCHEMES

To solve equations (15-19), three finite difference schemes were tried (Figures 1-3):

1) Standard Keller's Box Scheme (KBS)
2) Shifted Box Scheme (SBS)
3) Double-Shifted Box Scheme (DBS)

When KBS is used to solve equations (15-19), the gradient parameters (β's) must be defined midway between the profiles if second-order accuracy is to be maintained. This formulation has a serious drawback in that it permits the occurence of streamwise profile oscillations with little tendency to damp out (see Figure 4). This behavior is readily explained by noting that equations (16) and (18) at the wall reduce to

$$\beta_u = k(x) \frac{\partial^2 U}{\partial \eta^2} \tag{25}$$

where k(x) is a weak function of x. Since β_u is defined at the box midpoints, equation (25) constrains the average of $\partial^2 U/\partial\eta^2$ between any two successive streamwise stations:

$$\beta_{u_{i+\frac{1}{2}}} = \frac{k}{2}\left(\left(\frac{\partial^2 U}{\partial \eta^2}\right)_{i+1} + \left(\frac{\partial^2 U}{\partial \eta^2}\right)_i\right) \tag{26}$$

Hence, at the wall, $\partial^2 U/\partial\eta^2$ can have large amplitude excursions with alternating signs and still satisfy the finite difference equations. Figure 4 shows that the velocity profiles do indeed exhibit these fluctuations following a disturbance. SBS and DBS eliminate this problem by calculating the profiles midway between the x stations. This permits β_u to be defined at the same position as the profiles:

$$\beta_{u_{i+\frac{1}{2}}} = k \left(\frac{\partial^2 U}{\partial \eta^2}\right)_{i+\frac{1}{2}} \tag{27}$$

Thus, the velocity profiles cannot oscillate at the wall because each one is individually constrained (see Figure 5).

Both KBS and SBS require the solution of block tridiagonal systems with 5x5 blocks. In contrast, DBS has only 3x3 blocks. As a result, it requires roughly one-half the calculation time of the other two schemes--a substantial savings. Furthermore, it has the same high resistance to streamwise oscillations that SBS has.

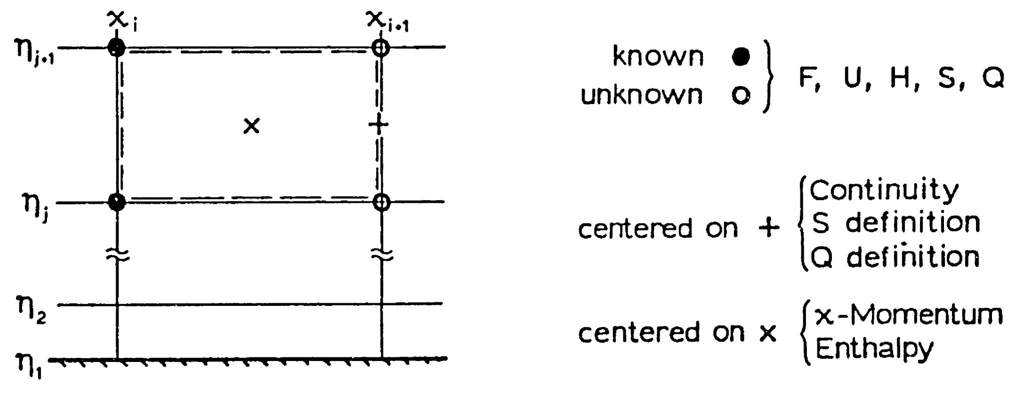

Figure 1. Keller's Box Scheme

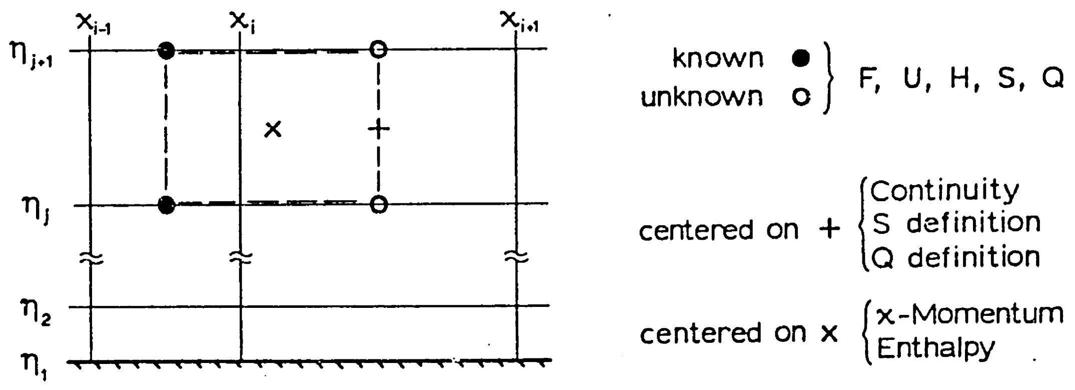

Figure 2. Shifted Box Scheme

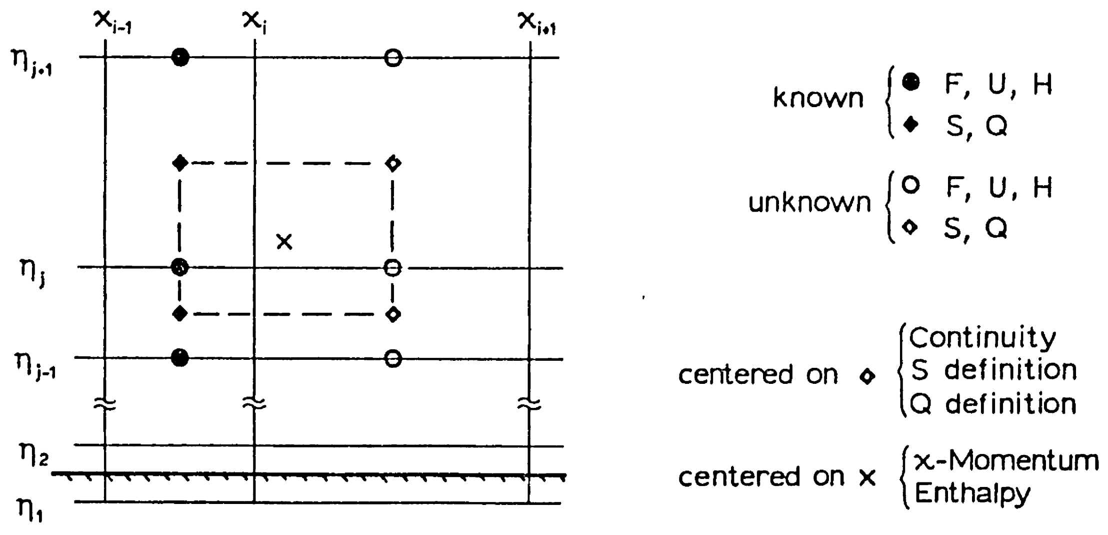

Figure 3. Double-Shifted Box Scheme

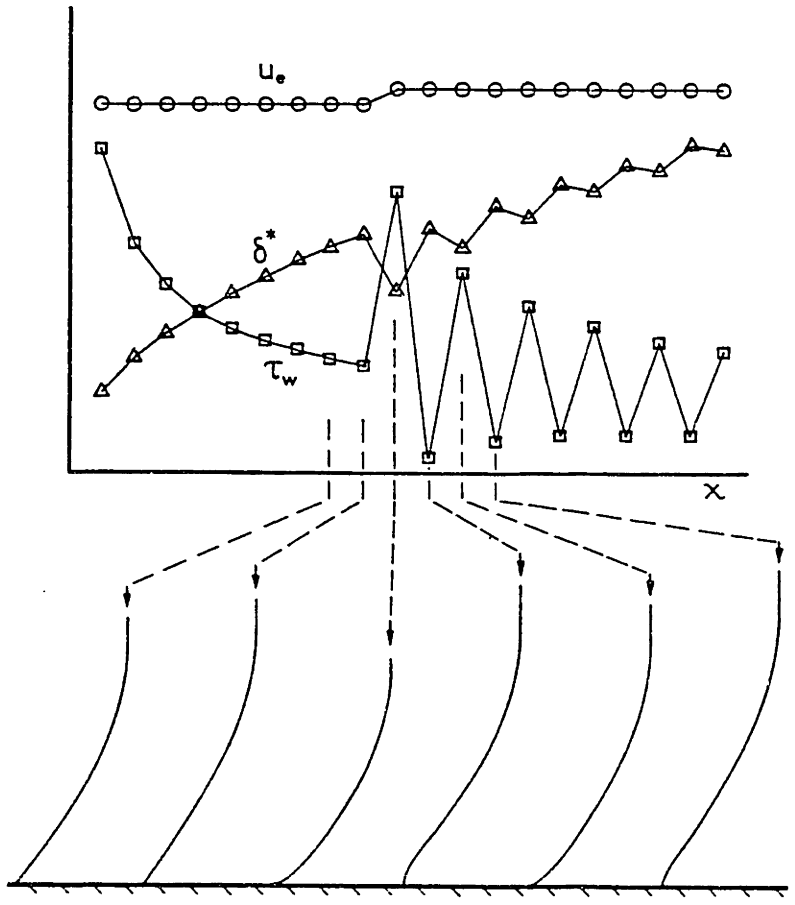

Figure 4. Response of Keller's Box Scheme to 5% edge velocity jump.

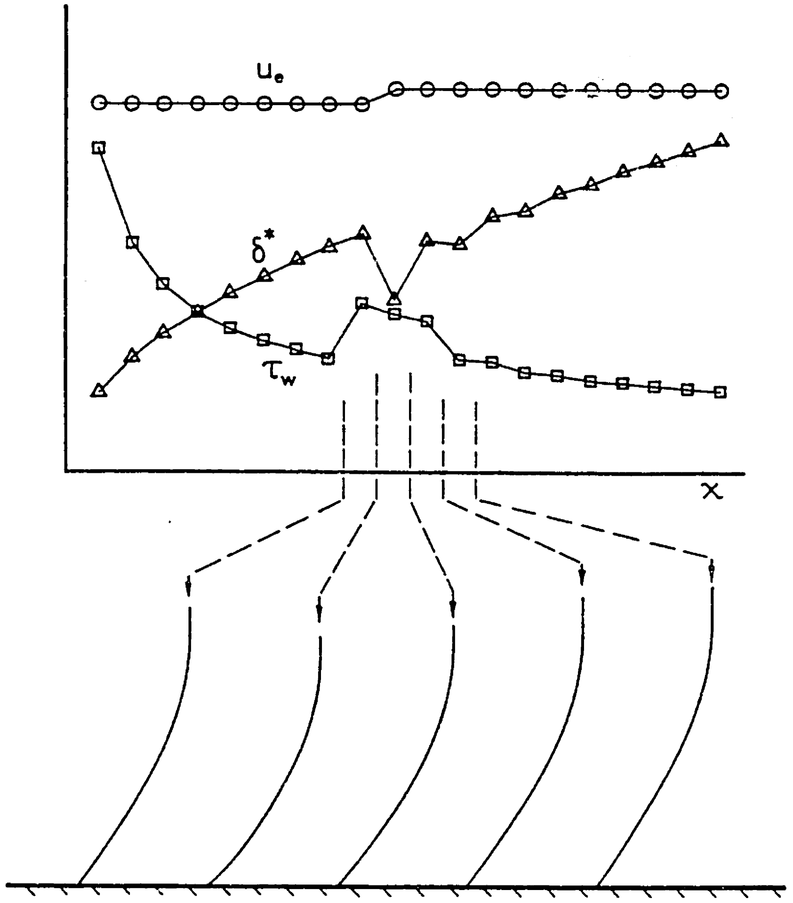

Figure 5. Response of Shifted Box Scheme to 5% edge velocity jump.

SOLUTION PROCEDURE

At each streamwise marching step, there are five unknowns for each η station at streamwise station $x_{i+1/2}$: F, U, H, S, and Q. In addition, there are two global (independent of η) unknowns at x_{i+1}: u_{ei+1}, and n_{i+1}. Although u_e is often prescribed for typical applications, it is convenient to always treat both u_e and n as unknown when the governing equations are discretized.

Since the discretized equations do not call for u_{ei+1} or n_{i+1}, but instead require the midpoint values $u_{ei+1/2}$ and $n_{i+1/2}$, the latter are temporarily taken as the global unknowns while the profiles are calculated. For convenience, the lack of a subscript will from now on imply i+1/2. The discretized gradient parameters are given by:

$$\beta_u = \frac{\ln (u_e/u_{ei})}{\ln (x/x_i)} \qquad \beta_n = \frac{\ln (n/n_i)}{\ln (x/x_i)} \qquad (28a\text{-}b)$$

In effect, u_e lies on a power curve in x between u_{ei} and u_{ei+1}, with β_u being the exponent of x (likewise for n and β_n). This interpolation scheme for u_e and n was chosen because it allows arbitrarily large streamwise steps in similar flows. Conventional linear interpolation of u_e and n does not have this property.

After u_e, n, β_u, β_n and the unknown profiles are calculated, u_{ei+1} and n_{i+1} are determined from the following relationships and stored for the next marching step.

$$u_{e_{i+1}} = u_{e_i}\left(\frac{x_{i+1}}{x_i}\right)^{\beta_u} \qquad n_{i+1} = n_i\left(\frac{x_{i+1}}{x_i}\right)^{\beta_n} \qquad (29a\text{-}b)$$

Because the discretized equations for each marching step are coupled and highly non-linear, the Newton-Raphson method is used to solve them iteratively. Following common practice, the iterates δF, δU, δH, δS, and δQ are introduced in the linearization and discretization process. For DBS, the iterates δS and δQ can be expressed as linear combinations of the other iterates and are thus eliminated. See Appendix A for discretization examples of equations (16) and (18).

The Cebeci-Smith two-layer eddy viscosity formulas given in Appendix B contain the wall shear velocity U_τ and the normalized velocity thickness Δ_u. Their respective iterates δU_τ and $\delta\Delta_u$ are therefore included in the linearized equations.

Together with δU_τ and $\delta\Delta_u$, the global iterates δu_e and δn are lumped on the righthand side to effectively produce five block tridiagonal systems with a common coefficient matrix of 5x5 (KBS and SBS) or 3x3 (DBS) blocks. The unknown column vector $\bar{\delta}$ contains the profile iterates δF, δU, δH (for DBS), and also δS, and δQ (for KBS and SBS):

$$\left[\,\bar{\bar{A}}\,\right] \times \left[\bar{\delta}\right] = \left[\bar{d}\right] - \delta u_e \left[\bar{e}\right] - \delta n \left[\bar{f}\right] - \delta U_\tau \left[\bar{g}\right] - \delta\Delta_u \left[\bar{h}\right] \tag{30}$$

All iterates (such as $\delta\mu$ and $\delta\mu_t$) which are not explicitly included in this system are expressed as linear combinations of the included iterates. Equations (31-33) are three examples of how these combinations are defined.

$$R = \frac{\rho}{\rho_e} = \frac{T_e}{T} = \frac{1 - u_e^2/2h_e}{H - U^2 u_e^2/2h_e} \tag{31a}$$

$$\delta R = \delta U \frac{\partial R}{\partial U} + \delta H \frac{\partial R}{\partial H} + \delta u_e \frac{\partial R}{\partial u_e} \tag{31b}$$

$$\beta_u = \frac{\ln\,(u_e/u_{ei})}{\ln\,(x/x_i)} \tag{32a}$$

$$\delta\beta_u = \delta u_e \frac{\partial\beta_u}{\partial u_e} = \delta u_e \frac{1}{u_e \ln\,(x/x_i)} \tag{32b}$$

$$\text{outer } \mu_t = 0.0168\ R\,\sqrt{Re_o}\ \Delta_u\ n\ \ \gamma_{tr} \tag{33a}$$

$$\delta\mu_t = \delta R \frac{\partial\mu_t}{\partial R} + \delta\Delta_u \frac{\partial\mu_t}{\partial\Delta_u} + \delta n \frac{\partial\mu_t}{\partial n} \tag{33b}$$

Since δR is not included in the block system, the δR in equation (33b) must still be eliminated by using equation (31b). Clearly, eliminating iterates not included in the system consists of repeated application of the chain rule of differentiation. Although very methodical, this process can and does get rather tedious, particularly with the inner eddy viscosity formula given in Appendix B. Nevertheless, the elimination is clearly worthwhile since it has a drastic effect on CPU time, as will be demostrated shortly.

In turbulent flow, the normalized velocity thickness Δ_u changes only slightly between Newton iterations. Its iterate can therefore be safely dropped from equation (30), simplifying the computational task somewhat. There is no noticable effect on the convergence rate.

After equation (30) is solved with a UL block factorization algorithm, each profile iterate is expressed as a residue r minus the global iterates times their respective influence coefficients a, b, and c:

$$\begin{bmatrix} \bar{\delta} \end{bmatrix} = \begin{bmatrix} \bar{r} \end{bmatrix} - \delta u_e \begin{bmatrix} \bar{a} \end{bmatrix} - \delta n \begin{bmatrix} \bar{b} \end{bmatrix} - \delta U_\tau \begin{bmatrix} \bar{c} \end{bmatrix} \qquad (34)$$

Since there are three unknowns left, namely δu_e, δn, and δU_τ, three more equations are necessary. One is obtained from the linearized definition of the scaling length Δ (equation (24b)). Another equation is obtained from the linearized definition of the wall shear velocity. The third equation results when some arbitrary point or integral quantity is prescribed. The derivations of these equations are given in Appendix C. Four different versions of the third equation are given, corresponding to specified u_e, $\rho_e u_e \delta^*$ (i.e. mass defect), δ^*, and τ_{wall}. These four versions are implemented in the program listed in Appendix D.

Once the three global iterates δu_e, δn, and δU_τ are calculated, the profile iterates δF, δU, δH (DBS), and also δS, and δQ (SBS and KBS) are easily determined from (34). The profile quantities are then updated and the process repeated to convergence.

Because all the governing equations are solved as a fully-coupled system (i.e. the variations of all quantities are taken into account by the chain rule elimination process), the entire system converges quadratically for both laminar and turbulent flow. Typically, two to four Newton iterations are needed per streamwise step. If the eddy viscosity formulas were not linearized, the calculation time would increase drastically for transitioning and turbulent flow, as shown in Figure 6. In this example, transition was achieved by artificially varying the turbulence intermittency factor in a continuous manner. Note that the higher the Reynolds Number, the stronger the effect of the turbulence on the momentum equation, and the higher the payoff of linearizing the eddy viscosity.

The Reyhner-Flugge-Lotz approximation, which is applied to regions of reverse flow, consists of setting the streamwise convective terms

U ∂U/∂x and U ∂H/∂x to zero. This is necessary to avoid growth of numerical errors and to prevent a zone of dependence violation. All the test cases run indicated that it is possible to retain the momentum convection term U ∂U/∂x in reverse flow simply by eliminating only its contribution to the variable iterates, thus avoiding artificial growth of numerical errors. This convection term is still retained in the residues (i.e. the righthand side of (30)). The fact that such a procedure results in stable calculations strongly suggests that upstream convection plays a very small role in limited separation regions. Of course, setting the variation of any term to zero adversely affects the quadratic convergence of the overall system. However, the contribution of the omitted terms is small, and as a result the number of iterations per streamwise step in separated flow rarely exceeds five. The separation behavior results which are presented in the next section were calculated using this modified Reyhner-Flugge-Lotz approximation.

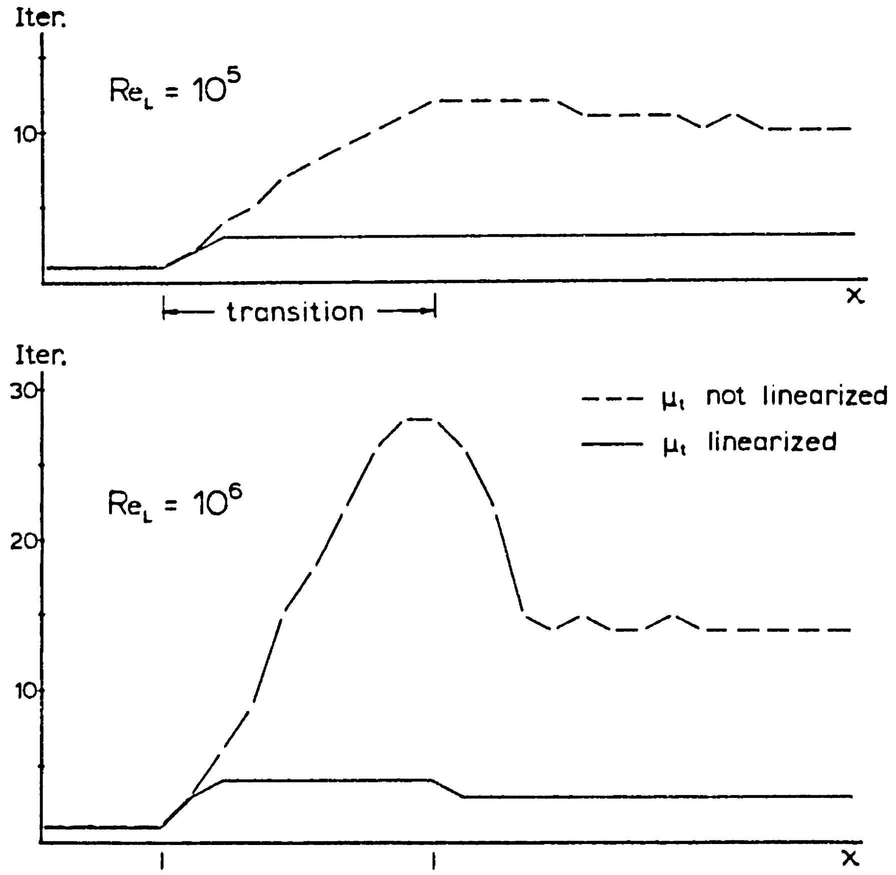

Figure 6. Effect of linearizing eddy viscosity on the number of iterations per streamwise station. Convergence criterion: $\delta U_{MAX} < 10^{-5}$

RESULTS AND DISCUSSION

Using the solution scheme presented here it is possible to investigate in detail the relationships between u_e, δ^* and wall shear at any given x station with relative ease, since the calculation mode (specified quantity) can be changed at any marching step. The separation behavior study given below was performed with SBS. DBS is a later development, but is expected to reproduce the results of SBS.

We first assume that all global quantities at the i-1th and ith stations, and the profiles midway between those two stations are known (see Figure 2). Now consider the problem of calculating the u_e and profiles at $x_{i+1/2}$ which correspond to a specified δ^*. If this specified δ^* is deliberately varied in some systematic manner, a relationship between u_e and δ^* (or, equivalently, between β_u and $\beta_{\delta^*} \equiv x/\delta^* \, d\delta^*/dx$) can be determined. Figure 7a shows such a relationship together with the corresponding wall shear at $x_{i+1/2}$. In this case the known upstream profile corresponds closely to the Blasius profile for zero pressure gradient. Several surprising features are apparent:

1) When β_u turns out to be negative, (i.e. u_e is less than u_{ei} and an adverse pressure gradient is present) there are two values of δ^* and corresponding β_{δ^*} which will produce this β_u. The numerical solution bifurcates whenever $\beta_u < 0$.

2) The smaller δ^* always gives a positive wall shear, the larger δ^* always gives a negative wall shear.

3) There is a minimum permissible β_u and hence a minimum permissible u_e. If u_e was specified to be less than this minimum, no solution to the finite difference equations would exist.

4) The minimum u_e occurs when the wall shear equals zero.

Assume now that a moderate adverse pressure gradient ($\beta_u = -0.16$) is specified at $x_{i+1/2}$. Figure 7a clearly shows that two distinct solutions are possible. However, the δ^* corresponding to attached flow produces a smooth continuation from the preceding stations, while the δ^* corresponding to separated flow is ridiculously large and has a radically different profile from the previous stations (see Figure 7b). Because the initial guesses for the profiles are obtained directly from the previous station, the iterative solution scheme in this case always converges on the "reasonable" leg of the bifurcating solution, since it is the one closest to the initial guess.

This situation changes significantly if the known upstream profile is close to separation. If the same pressure gradient parameter as in the previous case is specified (Figure 8a), the two possible values of δ^* are now quite close together. Furthermore, it is not clear which solution is reasonable and which is not since the two possible profiles are very nearly the same (see Figure 8b). Also note that β_u is locally quite insensitive to β_{δ^*} in contrast to the case in Figure 7a. This implies that specifying edge velocity poses a problem which is ill-conditioned near separation. Of course, it is also possible to specify a value u_e which is below the minimum and therefore has no solution. In either case, the iterative Newton-Raphson algorithm will fail spectacularly if convergence to a specified u_e is blindly attempted near this point. On the other hand, it is easy to see that convergence to a specified displacement thickness is well-conditioned at separation.

The relationships between β_u and β_{δ^*} shown in Figures 7 and 8 correspond to a freestream Mach Number of 0.0625, making the flow essentially incompressible. To determine what effect compressibility might have on solution behavior at separation, tests were also performed for Mach Numbers of 0.80 and 1.50. There was no qualitative change in the β_u-β_{δ^*} relationships shown in Figures 7 and 8.

It is highly unlikely that the bifurcation of the solution is due to the modified Reyhner-Flugge-Lotz approximation, although this is difficult to prove. It can only be stated here that at the separation point, where the occurence of solution bifurcation is most important, no upstream momentum convection exists.

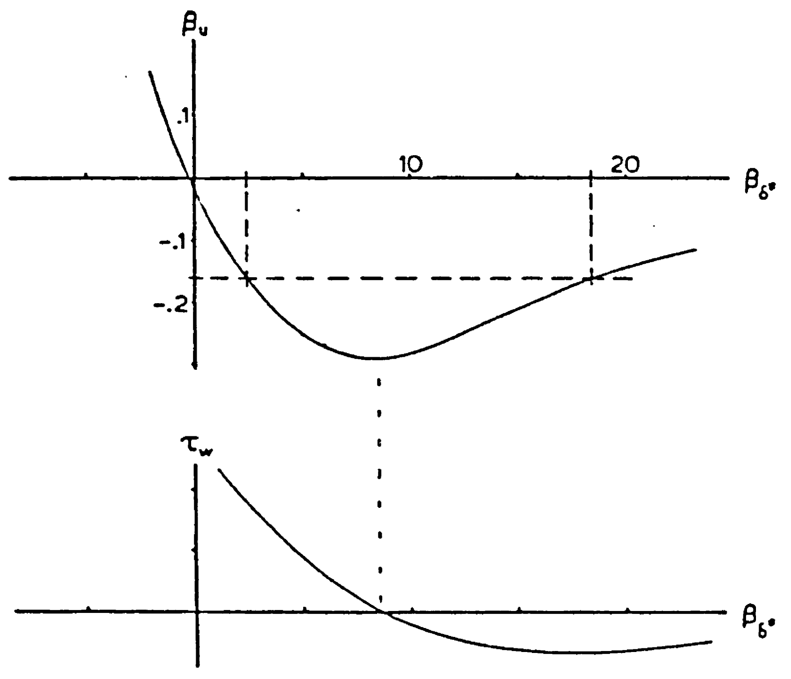

Figure 7a. Gradient parameter and wall shear relations far from separation.

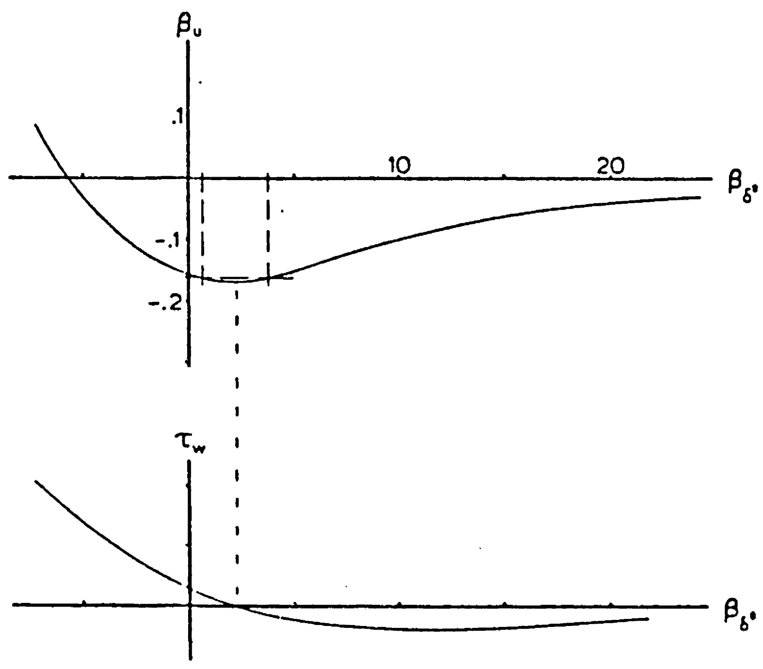

Figure 8a. Gradient parameter and wall shear relations close to separation.

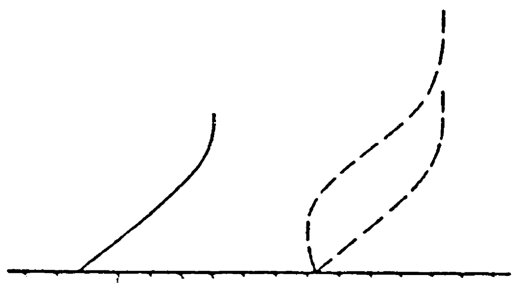

Figure 7b. Two profiles (dashed) corresponding to the same edge velocity. Upstream profile is far from separation.

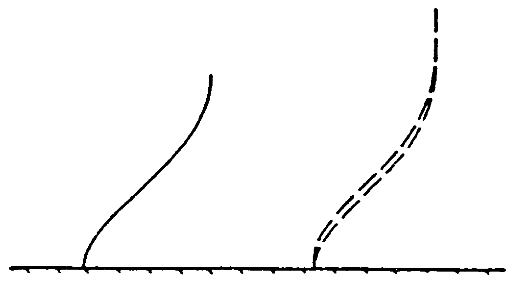

Figure 8b. Two profiles (dashed) corresponding to the same edge velocity. Upstream profile is close to separation.

REFERENCES

[1] J.E. Carter, "Inverse Solutions for Laminar Boundary-Layer Flows With Separation and Reattachment," NASA TR R-447, 1975.

[2] J.E. Carter, "Development of a Prediction Method for Transonic Shock Induced Separated Flow," UTRC Report R80-915213-4, 1980.

[3] T. Cebeci, H.B. Keller, and P.G. Williams, "Separating Boundary-Layer Flow Calculations," Academic Press, New York, 1979.

[4] T. Cebeci and P. Bradshaw, "Momentum Transfer in Boundary Layers," McGraw-Hill, New York, 1977.

[5] P. Bradshaw, T. Cebeci, and J.H. Whitelaw, "Engineering Calculation Methods for Turbulent Flow," Academic Press, New York, 1981.

[6] T. Cebeci and A.M.O. Smith, "Analysis of Turbulent Boundary Layers," Academic Press, New York, 1974.

[7] H. Schlichting, "Boundary Layer Theory," McGraw-Hill, New York, 1968.

[8] P.L. Ardonceau, T.A. de Roquefort, "Direct and Inverse Calculation of the Laminar Boundary Layer Solution," AIAA Journal, Nov 1980.

[9] M. Drela and W.T. Thompkins Jr., "A Study of Non-Unique Solutions of the Two-Dimensional Boundary Layer Equations at Laminar Separation and Reattachment Points," Proceedings of the Second Symposium on Numerical and Physical Aspects of Aerodynamic Flows, 1983.

APPENDIX A

DISCRETIZATION EXAMPLES FOR DBS

The following shorthand definitions are used:

1) An overline implies $i-\frac{1}{2}$, and lack of one implies $i+\frac{1}{2}$.

2) A "+" superscript implies $j+\frac{1}{2}$, a "-" superscript implies $j-\frac{1}{2}$.

Example 1: Shear Definition, Equation (18)

$$S^+ = \frac{\rho_e u_e x}{n^2}\left(\mu^+ + \mu_t^+\right)\frac{U_{j+1} - U_j}{\Delta\eta^+} \qquad \text{(A1a)}$$

$$S^- = \frac{\rho_e u_e x}{n^2}\left(\mu^- + \mu_t^-\right)\frac{U_j - U_{j-1}}{\Delta\eta^-} \qquad \text{(A1b)}$$

Example 2: x-Momentum, Equation (16)

Let L denote the discretized lefthand side of equation (16) at $i+\frac{1}{2}$:

$$L = 2\,\frac{S^+ - S^-}{\Delta\eta^+ + \Delta\eta^-} + \frac{\beta_n}{2}\left(\frac{F_{j+1} + F_j}{2}\,\frac{U_{j+1} - U_j}{\Delta\eta^+} + \frac{F_j + F_{j-1}}{2}\,\frac{U_j - U_{j-1}}{\Delta\eta^-}\right)$$
$$+ \beta_u\left(1 - \frac{1}{2}\left(\frac{U_{j+1} + U_j}{2}\,\frac{F_{j+1} - F_j}{\Delta\eta^+} + \frac{U_j + U_{j-1}}{2}\,\frac{F_j - F_{j-1}}{\Delta\eta^-}\right)\right) \qquad \text{(A2)}$$

Similarly, $\bar{L}$ denotes the entire lefthand side of equation (16) at $i-\frac{1}{2}$.

The discretized righthand side of equation (16) is defined as:

$$RHS = \frac{x + \bar{x}}{2}\left(\frac{F_{j+1} + \bar{F}_{j+1} - F_j - \bar{F}_j}{2\,\Delta\eta^+}\,\frac{U_{j+1} + U_j - \bar{U}_{j+1} - \bar{U}_j}{2\,\Delta x}\right.$$
$$- \frac{F_{j+1} + F_j - \bar{F}_{j+1} - \bar{F}_j}{2\,\Delta x}\,\frac{U_{j+1} + \bar{U}_{j+1} - U_j - \bar{U}_j}{2\,\Delta\eta^+}$$
$$+ \frac{F_j + \bar{F}_j - F_{j-1} - \bar{F}_{j-1}}{2\,\Delta\eta^-}\,\frac{U_j + U_{j-1} - \bar{U}_j - \bar{U}_{j-1}}{2\,\Delta x}$$
$$\left. - \frac{F_j + F_{j-1} - \bar{F}_j - \bar{F}_{j-1}}{2\,\Delta x}\,\frac{U_j + \bar{U}_j - U_{j-1} - \bar{U}_{j-1}}{2\,\Delta\eta^-}\right) \qquad \text{(A3)}$$

where Δx is the distance between the profiles: $\Delta x = x - \bar{x}$

The complete discretized form of equation (16) is therefore:

$$\frac{1}{2}\,(L + \bar{L}) \;=\; RHS \tag{A4}$$

Introducing iterates $L \rightarrow L + \delta L$ and $RHS \rightarrow RHS + \delta RHS$ gives:

$$\delta L \;-\; 2\,\delta RHS \;=\; 2\,RHS - L - \bar{L} \tag{A5}$$

Note that $\bar{L}$ contains only known quantities at $\bar{x}$ and therefore $\delta\bar{L} = 0$.

Before equation (A5) can be put into the block tridiagonal system (30), the iterates δL and δRHS must first be expressed in terms of the profile iterates δF, δU, δH, and global iterates δu_e, δn and δU_τ. This is accomplished by straightforward differentiation:

$$\begin{aligned}\delta L \;=\; & \delta F_{j+1}\left(\frac{\partial L}{\partial F_{j+1}}\right) + \delta F_j\left(\frac{\partial L}{\partial F_j}\right) + \delta F_{j-1}\left(\frac{\partial L}{\partial F_{j-1}}\right) \\ & + \delta U_{j+1}\left(\frac{\partial L}{\partial U_{j+1}}\right) + \delta U_j\left(\frac{\partial L}{\partial U_j}\right) + \delta U_{j-1}\left(\frac{\partial L}{\partial U_{j-1}}\right) \\ & + \delta S^+\left(\frac{\partial L}{\partial S^+}\right) + \delta S^-\left(\frac{\partial L}{\partial S^-}\right) + \delta\beta_u\left(\frac{\partial L}{\partial \beta_u}\right) + \delta\beta_n\left(\frac{\partial L}{\partial \beta_n}\right)\end{aligned} \tag{A6}$$

The iterate δRHS is similarly broken down.

The δS and $\delta\beta$ iterates in equation (A6) must still be expressed in terms of the profile and global iterates. Again, this is done by repeated differentiation of the finite difference expressions for S and β as described in the main text.

APPENDIX B
MOLECULAR AND EDDY VISCOSITY FORMULAS

As in the Analysis section, a bar denotes a dimensioned quantity and L, ρ_o, μ_o, T_o, $a_o = \sqrt{\gamma R T_o}$, $Re_o = \rho_o a_o L/\mu_o$ are dimensioned reference quantities.

Molecular Viscosity

Sutherland's Law as given by Schlichting [7] is:

$$\frac{\bar{\mu}}{\bar{\mu}_o} = \left(\frac{\bar{T}}{\bar{T}_{ref}}\right)^{\frac{3}{2}} \frac{\bar{T}_{ref} + \bar{T}_c}{\bar{T} + \bar{T}_c} \qquad \text{where} \qquad \bar{T}_c = 110 \text{ K}^0 \quad \text{for air} \tag{B1}$$

$\bar{T}_{ref}$ is the temperature at which $\bar{\mu} = \mu_o$. It is not necessary that $\bar{T}_{ref} = T_o$. Using T_o to non-dimensionalize all temperatures gives

$$\mu = \left(\frac{T}{T_{ref}}\right)^{\frac{3}{2}} \frac{T_{ref} + T_c}{T + T_c} \tag{B2}$$

In terms of the profile variables and u_e, the local temperature T is:

$$T = (\gamma - 1)\left(h_e H - \frac{1}{2} u_e^2 U^2\right) \tag{B3}$$

Eddy Viscosity

This is the two-layer Cebeci-Smith model as given in Cebeci and Smith [6]. Starting from the wall, the inner formula is used up to the point where $(\mu_t)_{inner} > (\mu_t)_{outer}$. The outer formula is used from there on.

Outer formula

$$\bar{\mu}_t = \alpha \bar{\rho} \int_0^{\bar{y}_e} (\bar{u}_e - \bar{u})\, d\bar{y} \;\; \gamma_{tr} \qquad \text{where} \qquad \alpha = 0.0168 \tag{B4}$$

γ_{tr} is the intermittency factor which varies from 0 to 1 in the transition zone. Although empirical formulas for γ_{tr} are available, for simplicity it is user-prescribed in the program listed in Appendix D.

In the transformed variables, (B4) becomes:

$$\mu_t = \alpha \; R \; n \; \Delta_u \sqrt{Re_o} \; \gamma_{tr} \tag{B5}$$

where
$$\Delta_u = \int_0^{\eta_e} (1 - U) \, d\eta \tag{B6}$$

<u>Inner formula</u>

For brevity, the inner eddy viscosity is given directly in terms of the transformed variables..

$$\mu_t = R \; n \; \lambda^2 \left| \frac{\partial U}{\partial \eta} \right| \sqrt{Re_o} \; \gamma_{tr} \tag{B7}$$

$$\lambda = \kappa \; \eta \left(1 - \exp\left(-\frac{\eta}{A} \right) \right) \qquad \text{where} \quad \kappa = 0.40 \tag{B8}$$

$$A = \frac{26}{N} \sqrt{\frac{\rho_e u_e x}{n^3}} \, \frac{\mu}{R} \, \frac{1}{U_\tau} Re_o^{-\frac{1}{4}} \tag{B9}$$

$$N = \left(1 - 11.8 \; p^+ \right)^{\frac{1}{2}} \tag{B10}$$

$$p^+ = \beta_u \sqrt{\frac{\rho_e u_e x}{n^3}} \, \frac{\mu_w}{R_w^2} \, \frac{1}{U_\tau^3} \; Re_o^{-\frac{1}{4}} \tag{B11}$$

When p^+ is linearized, the variations $\delta\mu_w$ and δR_w are approximated by the local variations $\delta\mu$ and δR. Since μ and R do not vary substantially across the inner layer or between Newton iterations, these are good approximations, and hence convergence rate is not noticably affected.

APPENDIX C

GLOBAL ITERATE SOLUTION FOR DBS

After solution of the block tridiagonal system (30), the profile iterates are in the following form (equation (34)):

$$\delta F_j = r_{1_j} - \delta u_e \, a_{1_j} - \delta n \, b_{1_j} - \delta U_\tau \, c_{1_j} \quad \text{(C1)}$$

$$\delta U_j = r_{2_j} - \delta u_e \, a_{2_j} - \delta n \, b_{2_j} - \delta U_\tau \, c_{2_j} \quad \text{(C2)}$$

$$\delta H_j = r_{3_j} - \delta u_e \, a_{3_j} - \delta n \, b_{3_j} - \delta U_\tau \, c_{3_j} \quad \text{(C3)}$$

The residues r and influence coefficients a, b, and c are known. To determine the profile iterates δF, δU, and δH, three more linearized relations are needed. These will produce a 3x3 system which is then readily solved for δu_e, δn, and δU_τ:

Relation 1: $\delta u_e \; A_1 + \delta n \; B_1 + \delta U_\tau \; C_1 = D_1$ (C4)

Relation 2: $\delta u_e \; A_2 + \delta n \; B_2 + \delta U_\tau \; C_2 = D_2$ (C5)

Relation 3: $\delta u_e \; A_3 + \delta n \; B_3 + \delta U_\tau \; C_3 = D_3$ (C6)

The coefficients A, B, C, and D are derived below for each relation.

Relation 1

Equation (24b) restated: $$1 = \int_0^{\eta_e} U(1 - U)\, d\eta \quad \text{(C7a)}$$

Or, in discretized form:

$$1 = \sum_{j=1}^{J-1} T \, \frac{U_{j+1} + U_j}{2} \left(1 - \frac{U_{j+1} + U_j}{2}\right) \Delta\eta_j \quad \text{(C7b)}$$

$$\text{where} \quad T = \begin{cases} \frac{1}{2} & \text{for } j = 1,\ j = J-1 \\ 1 & \text{for } 1 < j < J-1 \end{cases}$$

Letting $U_j^\dagger = (U_{j+1} + U_j)/2$, and introducing iterates $U_j \rightarrow U_j + \delta U_j$:

$$1 = \sum_{j=1}^{J-1} T \, U_j^\dagger \,(1 - U_j^\dagger)\, \Delta\eta_j + \sum_{j=1}^{J-1} T \,(\delta U_{j+1} + \delta U_j)(\tfrac{1}{2} - U_j^\dagger)\, \Delta\eta_j \quad \text{(C8)}$$

By using equation (C2) to eliminate δU_j and δU_{j+1}, equation (C8) is readily put into the form of equation (C4). The coefficients are then given by:

$$A_1 = \sum_{j=1}^{J-1} T\ (a_{2\,j+1} + a_{2\,j})\ (\tfrac{1}{2} - U_j^+)\ \Delta\eta_j \qquad \text{(C9a)}$$

$$B_1 = \sum_{j=1}^{J-1} T\ (b_{2\,j+1} + b_{2\,j})\ (\tfrac{1}{2} - U_j^+)\ \Delta\eta_j \qquad \text{(C9b)}$$

$$C_1 = \sum_{j=1}^{J-1} T\ (c_{2\,j+1} + c_{2\,j})\ (\tfrac{1}{2} - U_j^+)\ \Delta\eta_j \qquad \text{(C9c)}$$

$$D_1 = \sum_{j=1}^{J-1} T\ (d_{2\,j+1} + d_{2\,j})\ (\tfrac{1}{2} - U_j^+)\ \Delta\eta_j$$

$$- \ 1 \ + \sum_{j=1}^{J-1} T\ U_j^+\ (1 - U_j^+)\ \Delta\eta_j \qquad \text{(C9d)}$$

<u>Relation 2:</u>

U_τ definition: $$U_\tau = \sqrt{\frac{S_w}{R_w}} \qquad \text{or} \qquad R_w\, U_\tau^2 = S_w \qquad \text{(C10a-b)}$$

Using the fact that $\mu_t = 0$ and $U = 0$ at the wall, S_w and R_w are given by:

$$S_w = \frac{\rho_e u_e x}{n^2}\ \mu_1\ \frac{U_2 - U_1}{\Delta\eta_1} \qquad R_w = \frac{2 - u_e^2/h_e}{H_2 + H_1} \qquad \text{(C11a-b)}$$

Introducing iterates into equation (C10b) and linearizing (C11a-b):

$$2\ R_w U_\tau\ \delta U_\tau\ +\ U_\tau^2\ \delta R_w\ -\ \delta S_w\ =\ S_w - R_w U_\tau^2 \qquad \text{(C12)}$$

$$\delta S_w = \left(\frac{\partial S_w}{\partial(\rho u)_e}\right)\delta(\rho u)_e + \left(\frac{\partial S_w}{\partial n}\right)\delta n + \left(\frac{\partial S_w}{\partial U_2}\right)\delta U_2 + \left(\frac{\partial S_w}{\partial U_1}\right)\delta U_1 + \left(\frac{\partial S_w}{\partial \mu_1}\right)\delta\mu_1 \qquad \text{(C13a)}$$

$$\delta R_w = \left(\frac{\partial R_w}{\partial u_e}\right)\delta u_e + \left(\frac{\partial R_w}{\partial H_2}\right)\delta H_2 + \left(\frac{\partial R_w}{\partial H_1}\right)\delta H_1 \qquad \text{(C13b)}$$

The iterate $\delta(\rho u)_e$ in equation (C13a) can be expressed solely in terms of δu_e as follows (ρ_{st} denotes edge stagnation density and $M_e^2 = u_e^2/T_e$ is the edge Mach number squared):

$$\rho_e = \rho_{st}\left(\frac{T_e}{(\gamma-1)\,h_e}\right)^{\frac{1}{\gamma-1}} = \rho_{st}\left(1 - \frac{u_e^2}{2h_e}\right)^{\frac{1}{\gamma-1}} \qquad \text{(C14a)}$$

$$\delta(\rho u)_e = \rho_e\delta u_e + u_e\delta\rho_e = \left(\rho_e + u_e\frac{\partial\rho_e}{\partial u_e}\right)\delta u_e = \rho_e(1 - M_e^2)\,\delta u_e \qquad \text{(C14b)}$$

The iterate $\delta\mu_1$ is similarly expressed in terms of δu_e, δH_1, and δH_2 by straightforward differentiation of Sutherland's formula for viscosity listed in Appendix B.

Using equations (C13), (C14), and the expression for $\delta\mu_1$, equation (C12) can be put in the form of equation (C5). The coefficients are given by:

$$A_2 = \left(\frac{\partial S_w}{\partial U_2}\right)a_{22} + \left(\frac{\partial S_w}{\partial U_1}\right)a_{21} + \frac{\partial S_w}{\partial\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)a_{32} + \left(\frac{\partial\mu_1}{\partial H_1}\right)a_{31} - \frac{\partial\mu_1}{\partial u_e}\right) - \left(\frac{\partial S_w}{\partial(\rho u)_e}\right)\rho_e(1-M_e^2)$$
$$- U_\tau^2\left(\left(\frac{\partial R_w}{\partial H_2}\right)a_{32} + \left(\frac{\partial R_w}{\partial H_1}\right)a_{31} - \frac{\partial R_w}{\partial u_e}\right) \qquad \text{(C15a)}$$

$$B_2 = \left(\frac{\partial S_w}{\partial U_2}\right)b_{22} + \left(\frac{\partial S_w}{\partial U_1}\right)b_{21} + \frac{\partial S_w}{\partial\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)b_{32} + \left(\frac{\partial\mu_1}{\partial H_1}\right)b_{31}\right) - \frac{\partial S_w}{\partial n}$$
$$- U_\tau^2\left(\left(\frac{\partial R_w}{\partial H_2}\right)b_{32} + \left(\frac{\partial R_w}{\partial H_1}\right)b_{31}\right) \qquad \text{(C15b)}$$

$$C_2 = \left(\frac{\partial S_w}{\partial U_2}\right)c_{22} + \left(\frac{\partial S_w}{\partial U_1}\right)c_{21} + \frac{\partial S_w}{\partial\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)c_{32} + \left(\frac{\partial\mu_1}{\partial H_1}\right)c_{31}\right)$$
$$- U_\tau^2\left(\left(\frac{\partial R_w}{\partial H_2}\right)c_{32} + \left(\frac{\partial R_w}{\partial H_1}\right)c_{31}\right) + 2\,R_wU_\tau \qquad \text{(C15c)}$$

$$D_2 = \left(\frac{\partial S_w}{\partial U_2}\right)r_{22} + \left(\frac{\partial S_w}{\partial U_1}\right)r_{21} + \frac{\partial S_w}{\partial\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)r_{32} + \left(\frac{\partial\mu_1}{\partial H_1}\right)r_{31}\right)$$
$$- U_\tau^2\left(\left(\frac{\partial R_w}{\partial H_2}\right)r_{32} + \left(\frac{\partial R_w}{\partial H_1}\right)r_{31}\right) + S_w - R_wU_\tau^2 \qquad \text{(C15d)}$$

Relation 3

This relation is completely arbitrary. However, for stable calculations it must produce a well-posed problem. Four examples of this relation are given, corresponding to the four mode options implemented the program listed in Appendix D. The "sp" subscript denotes a specified quantity.

Example 1: Edge velocity u_e specified.

$$u_e + \delta u_e = u_{e_{sp}} \tag{C16}$$

This can be put immediately in the form of equation (C6), with the coefficients given by:

$$A_3 = 1 \qquad B_3 = 0 \qquad C_3 = 0 \qquad D_3 = u_{e_{sp}} - u_e \tag{C17a-d}$$

Example 2: Mass defect $m \equiv \rho_e u_e \delta^*$ specified.

$$\rho_e u_e \delta^* + \delta(\rho_e u_e \delta^*) = m_{sp} \tag{C18}$$

The displacement thickness δ^* is expressed as:

$$\delta^* = \Delta \int_0^{\eta_e} (1 - RU)\, d\eta = \Delta \int_0^{\eta_e} (1 - \frac{\partial F}{\partial \eta})\, d\eta = \Delta\, (\eta_e - F_e) \tag{C19}$$

where $F_e = \frac{1}{2}(F_{J-1} + F_J) \qquad \eta_e = \frac{1}{2}(\eta_{J-1} + \eta_J)$ (C20a-b)

Expanding $\delta(\rho_e u_e \delta^*)$ in equation (C18) and using (C19) gives:

$$\delta n\, (\eta_e - F_e) - \frac{n}{2}(\delta F_{J-1} + \delta F_J) = m_{sp} - n\, (\eta_e - F_e) \tag{C21}$$

Using equation (C1) to eliminate the δF iterates, equation (C21) is readily put into the form of equation (C6). The coefficients are:

$$A_3 = \frac{n}{2}(a_{1_{J-1}} + a_{1_J}) \tag{C22a}$$

$$B_3 = \frac{n}{2}(b_{1_{J-1}} + b_{1_J}) + \eta_e - F_e \tag{C22b}$$

$$C_3 = \frac{n}{2}(c_{1_{J-1}} + c_{1_J}) \tag{C22c}$$

$$D_3 = \frac{n}{2}(r_{1_{J-1}} + r_{1_J}) + m_{sp} - n\, (\eta_e - F_e) \tag{C22d}$$

Example 3: Displacement thickness δ^* specified.

$$\delta^* + \delta(\delta^*) = \delta^*_{sp} \tag{C23}$$

From equation (C19)

$$\delta^* = \Delta(\eta_e - F_e) = \frac{n}{\rho_e u_e}(\eta_e - F_e) \tag{C24a}$$

Or, in linearized form:

$$\delta(\delta^*) = \frac{\eta_e - F_e}{\rho_e u_e}\delta n - \frac{n}{\rho_e u_e}\delta F_e - \frac{n}{\rho_e^2 u_e^2}(\eta_e - F_e)\,\delta(\rho u)_e \tag{C24b}$$

Substituting for $\delta(\delta^*)$, multiplying through by $\rho_e u_e$, and eliminating δF_e with equation (C1), (C23) is put into the form of equation (C6). The coefficients are:

$$A_3 = \frac{n}{2}(a_{1_{J-1}} + a_{1_J}) - \frac{n}{u_e}(\eta_e - F_e)(1 - M_e^2) \tag{C25a}$$

$$B_3 = \frac{n}{2}(b_{1_{J-1}} + b_{1_J}) + \eta_e - F_e \tag{C25b}$$

$$C_3 = \frac{n}{2}(c_{1_{J-1}} + c_{1_J}) \tag{C25c}$$

$$D_3 = \frac{n}{2}(r_{1_{J-1}} + r_{1_J}) + \rho_e u_e \delta^*_{sp} - n(\eta_e - F_e) \tag{C25d}$$

Example 4: Wall shear τ_w specified.

$$\tau_w + \delta\tau_w = \tau_{w_{sp}} \tag{C26}$$

From equations (13f) and (18): $\tau_w = \frac{\rho_e u_e^2}{n}\,\mu_1\,\frac{U_2 - U_1}{\Delta\eta_1}$ (C27a)

Or, in linearized form:

$$\delta\tau_w = \frac{\tau_w}{u_e}(2 - M_e^2)\,\delta u_e - \frac{\tau_w}{n}\delta n + \frac{\tau_w}{U_2 - U_1}(\delta U_2 - \delta U_1) + \frac{\tau_w}{\mu_1}\delta\mu_1 \tag{C27b}$$

As in previous examples, equation (C26) can be put in the form of equation (C6), with the coefficients given by:

$$A_3 = \frac{\tau_w}{U_2 - U_1}\left(a_{2\,2} - a_{2\,1}\right) + \frac{\tau_w}{\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)a_{3\,2} + \left(\frac{\partial\mu_1}{\partial H_1}\right)a_{3\,1} - \frac{\partial\mu_1}{\partial u_e}\right) - \frac{\tau_w}{u_e}\,(2-M_e^2) \qquad (C28a)$$

$$B_3 = \frac{\tau_w}{U_2 - U_1}\left(b_{2\,2} - b_{2\,1}\right) + \frac{\tau_w}{\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)b_{3\,2} + \left(\frac{\partial\mu_1}{\partial H_1}\right)b_{3\,1}\right) + \frac{\tau_w}{n} \qquad (C28b)$$

$$C_3 = \frac{\tau_w}{U_2 - U_1}\left(c_{2\,2} - c_{2\,1}\right) + \frac{\tau_w}{\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)c_{3\,2} + \left(\frac{\partial\mu_1}{\partial H_1}\right)c_{3\,1}\right) \qquad (C28c)$$

$$D_3 = \frac{\tau_w}{U_2 - U_1}\left(r_{2\,2} - r_{2\,1}\right) + \frac{\tau_w}{\mu_1}\left(\left(\frac{\partial\mu_1}{\partial H_2}\right)r_{3\,2} + \left(\frac{\partial\mu_1}{\partial H_1}\right)r_{3\,1}\right) + \tau_w - \tau_{w_{sp}} \qquad (C28d)$$

APPENDIX D
PROGRAM LISTING

```
      PROGRAM BLAKE
      INCLUDE 'BLAKE.INC'
C
C     **************************************************
C     *                                                *
C     *    2-D, Compressible Boundary-Layer Program    *
C     *                                                *
C     *    Turbulence Model:                           *
C     *       Cebeci-Smith Two Layer Eddy Viscosity    *
C     *                                                *
C     *    Solution Scheme:                            *
C     *       Double-Shifted Box Scheme,               *
C     *       second order accurate for all grids.     *
C     *                                                *
C     *    Options currently implemented               *
C     *    (streamwise quantity prescribed:            *
C     *       1) Ue                                    *
C     *       2) Rhoe*Ue*Dstar  ( = mass defect)       *
C     *       3) Dstar                                 *
C     *       4) Wall Shear                            *
C     *                                                *
C     *    Mark Drela      May 1983                    *
C     *    MIT Gas Turbine and Plasma Dynamics Lab     *
C     *                                                *
C     **************************************************
C
      CALL INPUT
C
      IF(NSTR.GT.Ø) OPEN(UNIT=LSTR,NAME='STREAM.DAT',TYPE='NEW')
      IF(NPFL.GT.Ø) OPEN(UNIT=LPFL,NAME='PROFIL.DAT',TYPE='NEW')
C
C---- generate starting solution between first two X stations
      NSIM = 1
      CALL SIMIL
C
C---- output first station solution from similarity solution
      I = 1
      CALL HEADER
      CALL STROUT
C
C---- output profiles at X(1+1/2)
      CALL PFLOUT
C
C---- march downstream
      NSIM = Ø
      DO 1ØØØ I=2, IEND-1
C
C------ calculate profiles at X(I+1/2)
        CALL INIT
        CALL PROFL
C
C------ output solution at X(I)
        CALL STROUT
C
C------ output profiles at X(I+1/2)
        CALL PFLOUT
C
C------ set edge quantities at X(I+1)
        CALL IPSET
C
 1ØØØ CONTINUE
C
C---- output last station solution
      I = IEND
      CALL STROUT
C
      WRITE(LTTI,*) '[ BLAKE ]: Normal Termination'
C
      CALL STOPIT
C
C     The
      END
```

```
C      *********************************************
C
C       This is file BLAKE.INC which is INCLUDED
C       at compile time in each subroutine.
C
C      *********************************************
C
       IMPLICIT REAL (M)
C
       COMMON /CØ1/ A11(41),A12(41),A13(41),
     &              A21(41),A22(41),A23(41),
     &              A31(41),A32(41),A33(41)
       COMMON /CØ1/ B11(41),B12(41),B13(41),
     &              B21(41),B22(41),B23(41),
     &              B31(41),B32(41),B33(41)
       COMMON /CØ1/ C11(41),C12(41),C13(41),
     &              C21(41),C22(41),C23(41),
     &              C31(41),C32(41),C33(41)
       COMMON /CØ4/ R11(41),R12(41),R13(41),R14(41),
     &              R21(41),R22(41),R23(41),R24(41),
     &              R31(41),R32(41),R33(41),R34(41)
       COMMON /CØ5/ F(41), U(41), H(41), S(41), Q(41),
     &              FB(41),UB(41),HB(41),SB(41),QB(41),
     &              VIS(41),VTB(41),DY(41),ETAE,GEO,JJ
       COMMON /CØ6/ BH,BCON
       COMMON /CØ7/ XTR1,XTR2,TURB,UTAU,DUNORM
       COMMON /CØ8/ EPS,ITER,ITMAX,
     &              REØ,SRE,PR,PRT,GAM,GM1,TVIS,TVCON
       COMMON /CØ9/ DUE,DMS,DUT
C
C---- assorted quantities at X(I+1/2)
C
C       UTAU   = wall shear velocity          (for inner eddy viscosity)
C       DUNORM = normalized velocity thickness (for outer eddy viscosity)
C
C       RHOE = edge density
C       UE = edge velocity
C       SC = length scale
C       MS = mass scale               = Rhoe*Ue*Sc
C       DS = displacement thickness = Dstar
C       TH = momentum thickness      = Dmom
C       MD = mass defect             = Rhoe*Ue*Dstar
C       SR = physical wall shear
C
C              ye
C             /
C       Sc  = | U(1 - U) dy
C             /
C             o
C
       COMMON /C1Ø/ UE,  MS,  SC,  MD,  DS,  SR,  TH,
     &              UEI, MSI, SCI, MDI, DSI, SRI,
     &              UEIP,MSIP,SCIP,MDIP,DSIP,SRIP,
     &              TE,EE,EEC,ME2,ME2C,PE,RHOE,TST,RST
       COMMON /C11/ PPAR,UGUESS,RNU
       COMMON /C12/ I,IEND,X(1ØØ),SPEC(1ØØ),RSTAG(1ØØ),TSTAG,
     &              BETN, BETU, BETH, BETM, BETD, BETS,
     &              BETNB,BETUB,BETHB,
     &              XF,XB,XLOG,FLOG,SHPF,SHPB,SPECF
       COMMON /C13/ KODE,NSTR,NPFL,NSIM
       COMMON /C14/ LINP,LFLO,LTTI,LSTR,LPFL
       COMMON /C15/ VUP(41),VHP(41),VUO(41),VHO(41),
     &              TUP(41),THP(41),TUO(41),THO(41),
     &              VUE(41),TUE(41),TMS(41),TUT(41)
       COMMON /C16/ A1,A2,A3,B1,B2,B3,C1,C2,C3,D1,D2,D3
```

```
      SUBROUTINE INPUT
      INCLUDE 'BLAKE.INC'
C     ****************************************************************
C      This routine reads the input files INPUT.DAT and FLOW.DAT
C
C----  Description of INPUT.DAT  ----------------------------------
C      KODE       ! option number...see label in main progam
C      EPS        ! convergence epsilon...recommended: 1.e-5
C      ITMAX      ! maximum number of Newton iterations...recommended: 2Ø
C      output flags: Ø = no output
C                    1 = output every x station
C                    2 = output every 2nd x station, etc.
C      NSTR       ! STREAM.DAT output flag
C      NPFL       ! PROFIL.DAT output flag
C      REØ        ! reference Reynolds Number...mainly used in turbulence model
C      PR         ! Prandtl Number
C      PRT        ! turbulent Prandtl Number
C      GAM        ! Cp/Cv
C      TSTAG      ! freestream stagnation temperature
C      TVIS       ! temperature corresponding to reference viscosity
C      TVCON      ! 11Ø Kelvin normalized with reference temperature
C      XTR1,XTR2  ! x positions marking beginning and end of transition zone
C      BH,BCON    ! constants in wall BC:   bh*Hwall + (1-bh)*Qwall = bcon
C      PPAR       ! pressure gradient parameter  x/ue due/dx at leading edge
C      UGUESS     ! initial edge velocity guess for KODEs 2 & 3 (see SIMIL)
C      JJ         ! number of normal grid lines
C      GEO        ! geometric grid stretching constant  geo = dETAj+1/dETAj
C      ETAE       ! edge value of ETA...recommended: 14
C
C----  Description of FLOW.DAT  ----------------------------------
C      IEND       ! number of streamwise stations
C      X(I)       ! x value array
C      RSTAG(I)   ! stagnation density array
C      SPEC(I)    ! specified quantity array...interpreted according to KODE
C
C     ****************************************************************
C
C---- set logical unit numbers
      LINP = 1    ! global input file
      LFLO = 2    ! streamwise station input file
      LTTI = 5    ! terminal
      LSTR = 7    ! streamwise output file
      LPFL = 8    ! profile output file (caution! tends to get large real fast)
C
C---- read main input
      OPEN(UNIT=LINP,NAME='INPUT.DAT',TYPE='OLD')
      READ(LINP,*) KODE,EPS,ITMAX
      READ(LINP,*) NSTR,NPFL
      READ(LINP,*) REØ,PR,PRT,GAM
      READ(LINP,*) TSTAG,TVIS,TVCON
      READ(LINP,*) XTR1,XTR2
      READ(LINP,*) BH,BCON
      READ(LINP,*) PPAR,UGUESS
      READ(LINP,*) JJ,GEO,ETAE
      CLOSE(UNIT=LINP)
C
      SRE = SQRT(REØ)
      GM1 = GAM - 1.Ø
C
C---- generate normal grid
      CALL GRID
C
C---- read streamwise station input
      OPEN(UNIT=LFLO,NAME='FLOW.DAT',TYPE='OLD')
      READ(LFLO,*) IEND
      DO 4 I=1, IEND
        READ(LFLO,*,END=5) X(I),RSTAG(I),SPEC(I)
    4 CONTINUE
      CLOSE(UNIT=LFLO)
C
      RETURN
    5 IEND = I - 1
      WRITE(LTTI,*) '[ INPUT ]: Number of streamwise stations found
     & was less than expected.'
      WRITE(LTTI,*) '           IEND changed to ',IEND
      CLOSE( UNIT=LFLO)
C
      RETURN
      END


      SUBROUTINE GRID
      INCLUDE 'BLAKE.INC'
C     ****************************************************
C      This routine calculates the DY's for a geometric-
C      progression-type normal grid which are then scaled
C      to obtain the specified ETAE.
C     ****************************************************
C
C---- calculate nor.ıal grid spacing DY(J) ... ETA(J+1) = ETA(J) + DY(J)
      DY( 1) = 1.Ø
      YTEST = Ø.
      DO 3 J=2, JJ-1
        DY( J) = GEO*DY( J-1)
        YTEST = YTEST + Ø.5*( DY( J)+DY( J-1))
    3 CONTINUE
C
C---- scale DY(J) to get specified ETAE
      FUDGE = ETAE/YTEST
      DO 5 J=1, JJ-1
        DY( J) = FUDGE*DY( J)
    5 CONTINUE
      RETURN
      END
      SUBROUTINE SIMIL
      INCLUDE 'BLAKE.INC'
C     ****************************************************************
C      This routine calculates a similarity solution using the
C      same transformation as the main program.  The solution
C      is calculated midway between X(1) and X(2).
C      The specified edge quantity is assumed to be in SPEC(2).
C      Four types of similarity solutions are implemented
C      corresponding to the four modes of the main program,
C      although similarity with prescribed wall shear is
C      probably not very useful due to the singular nature of
C      the wall shear at a leading edge for certain cases.
C     ****************************************************************
C
C---- set prescribed gradient parameters
      BETU = PPAR                    ! edge velocity gradient parameter
      BETN = Ø.5*(1.Ø + BETU)        ! mass scale          "          "
      BETH = Ø.                      ! total enthalpy      "          "
C
C---- these relationships must hold if there is similarity
      BETM = Ø.5*(1.Ø + BETU)        ! mass defect         "          "
      BETD = Ø.5*(1.Ø - BETU)        ! disp. thickness     "          "
      BETS = Ø.5*( 3.Ø*BETU - 1.Ø)   ! wall shear          "          "
C
C---- there is no upstream station for similarity, so...
      BETUB = Ø.
      BETNB = Ø.
      BETHB = Ø.
C
      TURB = Ø.   ! no turbulence
C
      XF = Ø.5*(X( 1) + X( 2))   ! similarity x position
C
      TST = TSTAG                           ! similarity
      RST = Ø.5*(RSTAG(1) + RSTAG( 2))      ! stagnation
      PST = RST*TST/GAM                     ! quantities
C
C---- calculate Falkner-Skan Dstar, Theta, and Shear with empirical formulas...
C        ...necessary for initial estimates to start the Newton-Raphson procedure
      BM1 = 1.Ø - BETU
      DFS = Ø.64791 + BM1*( .2ØØ + BM1*( .22973 + .6431*BM1**3))
      TFS = Ø.29234 + BM1*( .125 + BM1*( .Ø666Ø + .18Ø2*BM1**3))
      SFS = 1.23259 - BM1*( .56Ø + BM1*( .18213 + .1584*BM1**3))
      SHPF = DFS/TFS    ! shape parameter
C
C---- Similarity solutions with BETU=Ø and specified Mass Defect or Dstar
C-    are non-unique if they exist at all.  There is a high and low Mach Number
C-    solution for each case.  UGUESS is the first guess for Ue which will put
C-    the Newton-Raphson solver on one of the two branches.
C---- But first we must see if UGUESS was given:
C
      IF(( KODE.EQ.2 .OR. KODE.EQ.3)
     &    .AND. BETU.EQ.Ø.Ø .AND. UGUESS.EQ.Ø.Ø) GO TO 5ØØ
C
C---- set SPECF at XF for whatever KODE it may be
      IF( KODE.EQ.1) SPECF = SPEC( 2)*( XF/X( 2))**BETU
      IF( KODE.EQ.2) SPECF = SPEC( 2)*( XF/X( 2))**BETM
      IF( KODE.EQ.3) SPECF = SPEC( 2)*( XF/X( 2))**BETD
      IF( KODE.EQ.4) SPECF = SPEC( 2)*( XF/X( 2))**BETS
C
C---- set specified quantity for some KODE
      UE = SPECF    ! assumes KODE=1
      MD = SPECF    ! assumes KODE=2
      DS = SPECF    ! assumes KODE=3
      SR = SPECF    ! assumes KODE=4
C
C---- initialize UE for iteration for KODEs other than 1
      IF( KODE.NE.1 .AND. BETU.EQ.Ø.Ø) UE = UGUESS
      IF( KODE.EQ.2 .AND. BETU.GT.Ø.Ø) UE = (MD/DFS)**2/XF
      IF( KODE.EQ.3 .AND. BETU.GT.Ø.Ø) UE = (DFS/DS)**2*XF
      IF( KODE.EQ.4 .AND. BETU.GT.Ø.Ø) UE = (XF*(SR/SFS)**2)**(1./3.)
```

```
C
C---- initialize MS for iteration
      EE = Ø.5*GM1*UE**2/TST        ! edge kinetic energy/total enthalpy ratio
      EEC = 1.Ø - EE
      RHOE = RST * EEC**(1.Ø/GM1)  ! edge density
      MS = TFS*SQRT(RHOE*UE*XF)     ! first guess for mass scale
C
C---- set initial profiles ... simple polynomials are used
      Z = -Ø.5*DY(1)/7.5
      DO 1Ø J=1, JJ      ! march up from the wall
        FB(J) = Ø.
        UB(J) = Ø.
        HB(J) = Ø.
        SB(J) = Ø.
        QB(J) = Ø.
C
        H(J) = 1.Ø
        U(J) = Z*(2.Ø-Z)
        IF(Z.GT.1.Ø) U(J) = 1.Ø
C
        R2 = EEC/(H(J) - EE*U(J)**2)    ! density at eta(J)
        IF(J.EQ.1) F(J) = Ø.
        IF(J.GT.1) F(J) = F(J-1) + Ø.5*DY(J-1)*(R2*U(J) + R1*U(J-1))
C
C------ Note: S(J) and Q(J) will be set by SQSET
C
        Z = Z + DY(J)/7.5
        R1 = R2
   1Ø CONTINUE
C
C---- initialize everything else for iteration
      CALL ECALC   ! edge quantities
      CALL DCALC   ! Dstar, Dmom, and other thicknesses
      CALL VISC    ! viscosity
      CALL SQSET   ! S and Q arrays
C
      DO 5Ø ITER=1, ITMAX    ! Newton iteration loop
C
C------ fill blocks of tridiagonal system
        CALL SETUP
C
C------ get base profile iterates and global iterate influence coefficients
        CALL SOLVE
C
C------ get global variable iterates and corrected profile iterates
        CALL DELTAS
C
C------ update profile variables
        DUMAX = Ø.Ø
        DO 55 J=1, JJ
          F(J) = F(J) + R11(J)
          U(J) = U(J) + R21(J)
          H(J) = H(J) + R31(J)
          DUMAX = AMAX1(DUMAX,ABS(R21(J)))
   55   CONTINUE
C
C------ update edge velocity UE and mass scale MS
        UE = UE + DUE
        MS = MS + DMS
C
C------ test for negative edge values (divergence)
        IF(UE.LE.Ø.Ø) GO TO 6ØØ
        IF(MS.LE.Ø.Ø) GO TO 7ØØ
C
C------ recalculate edge quantities
        CALL ECALC
C
C------ recalculate DS, TH, MD, SC, and shape parameter
        CALL DCALC
C
C------ recalculate viscosity
        CALL VISC
C
C------ recalculate S and Q arrays
        CALL SQSET
```

```
C
C------ test for convergence
        DGLBL = ABS(DMS)/MS + ABS(DUE)/UE
        IF(DUMAX.LE.EPS .AND. DGLBL.LE.EPS) GO TO 900
C
   50 CONTINUE    ! end of Newton iteration loop
C
C
C!!!! PANIC messages
C
      WRITE(LTTI,*) '[ SIMIL ]: Newton iteration did not converge.'
      WRITE(LTTI,*) '           Max U velocity iterate   : ',DUMAX
      WRITE(LTTI,*) '           Ue + mass scale iterates : ',DGLBL
      IF(KODE.EQ.2 .AND. BETU.EQ.0.0)
     & WRITE(LTTI,*) '           Specified Mass is possibly too small.'
      IF(KODE.EQ.3 .AND. BETU.EQ.0.0)
     & WRITE(LTTI,*) '           Specified Dstar is possibly too small.'
      CALL STOPIT   ! Crash softly
C
  500 WRITE(LTTI,*) '[ SIMIL ]: UGUESS must be given for inverse'
      WRITE(LTTI,*) '           flat plate similarity solution.'
      CALL STOPIT   ! Crash softly
C
  600 WRITE(LTTI,*) '[ SIMIL ]: Negative edge velocity was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged. Crashing...'
      CALL STOPIT   ! Crash softly
C
  700 WRITE(LTTI,*) '[ SIMIL ]: Negative mass scale was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged. Crashing...'
      CALL STOPIT   ! Crash softly
C
C
C---- The normal graceful exit
C
  900 WRITE(LTTI,*)'[ SIMIL ]: Similarity        ...',ITER,' Iterations'
C
C---- set edge quantities for X(2) station
      UEIP = UE*(X(2)/XF)**BETU
      MSIP = MS*(X(2)/XF)**BETN
      MDIP = MD*(X(2)/XF)**BETM
      DSIP = DS*(X(2)/XF)**BETD
      SRIP = SR*(X(2)/XF)**BETS
C
C---- set edge quantities for X(1) station...
C       ...assume first that streamwise gradients are zero
      UEI = UEIP
      MSI = MSIP
      MDI = MDIP
      DSI = DSIP
C
C---- and if they are not zero...
      IF(BETU.NE.0.0) UEI = UE*(X(1)/XF)**BETU
      IF(BETN.NE.0.0) MSI = MS*(X(1)/XF)**BETN
      IF(BETM.NE.0.0) MDI = MD*(X(1)/XF)**BETM
      IF(BETD.NE.0.0) DSI = DS*(X(1)/XF)**BETD
C
C---- treat shear carefully, it might be infinite at leading edge...
      SRI = 99.9999            ! ...or at least very large
      IF(BETS.EQ.0.0) SRI = SRIP
      IF(BETS.NE.0.0 .AND. X(1).GT.0.0) SRI = SR*(X(1)/XF)**BETS
C
C---- One last thing to take care of...
C     ... for BETU > 0, warn if incompressibility assumption is invalid
      MACH = SQRT(ME2)
      IF(BETU.EQ.0.0 .OR. MACH.LE.0.05) RETURN   ! the 0.05 is arbitrary
C
      WRITE(LTTI,*) '[ SIMIL ]: WARNING! Edge Mach number = ',MACH
      WRITE(LTTI,*) '           Heat production might upset similarity.'
      WRITE(LTTI,*) '           X(1) and/or X(2) should be smaller.'
      RETURN    ! keep going anyway
C
      END
      SUBROUTINE INIT
      INCLUDE 'BLAKE.INC'
C     ******************************************
C      This routine initializes everything for
C      solution of profiles and edge quantities
C      between the Ith and I+1th stations.
C     ******************************************
C
C
C**** first, set stuff for the previous profile station I-1/2 ***
C
C---- set profiles at I-1/2
      DO 2 J=1, JJ
        FB(J) = F(J)
        UB(J) = U(J)
        HB(J) = H(J)
        SB(J) = S(J)
        QB(J) = Q(J)
    2 CONTINUE
C
C---- set gradient parameters at I-1/2
      BETUB = BETU
      BETNB = BETN
      BETHB = BETH
C
C---- set X value at I-1/2
      XB = XF
C
C---- set shape parameter at I-1/2 for the output routines
      SHPB = SHPF
C
C
C**** next, set stuff for station I ***
C
C---- set UEI, MSI, etc.
      UEI = UEIP
      MSI = MSIP
      MDI = MDIP
      DSI = DSIP
      SRI = SRIP
C
C---- set known TST and PST
      RST = Ø.5*( RSTAG( I) + RSTAG( I+1))
      TST = TSTAG
C
C
C**** finally, set or initialize stuff at I+1/2 for iteration ***
C
      XF = Ø.5*( X( I+1) + X( I))
      XLOG = ALOG( X( I+1)/X( I))
      FLOG = ALOG( XF/X( I))
C
C---- the normal power-curve interpolation of SPECF is done here...
C       ...this is exact for similar flows
      IF( KODE.NE.4) BSPEC = ALOG( SPEC( I+1)/SPEC( I))/XLOG
      IF( KODE.NE.4) SPECF = SPEC( I)*( XF/X( I))**BSPEC
      IF( KODE.EQ.1) BETU = BSPEC
      IF( KODE.EQ.2) BETM = BSPEC
      IF( KODE.EQ.3) BETD = BSPEC
C
C---- linear interpolation is used for wall shear since it might be negative...
C       ...this is NOT exact for similar flows and requires smaller x steps
      IF( KODE.EQ.4) SPECF = Ø.5*( SPEC( I+1) + SPEC( I))
C
C---- set or initialize UE and MS
      UE = UEI*( XF/X( I))**BETU
      MS = MSI*( XF/X( I))**BETN
C
C---- set known total enthalpy gradient parameter
      BETH = Ø.
```

```
C---- set turbulence weighting coefficient with cubic transition zone
      XT = Ø.5
      IF(XTR1.NE.XTR2) XT = (2.Ø*XF - (XTR2+XTR1))/(XTR2-XTR1)
      TURB = Ø.5 + Ø.25*(3.Ø*XT - XT**3)
      IF(XF.LT.XTR1) TURB = Ø.
      IF(XF.GE.XTR2) TURB = 1.Ø
C
C---- calculate edge quantities, viscosity, S and Ω
      CALL ECALC
      CALL DCALC
      CALL VISC
      CALL SΩSET
C
      RETURN
      END



      SUBROUTINE IPSET
      INCLUDE 'BLAKE.INC'
C     ********************************************
C      This routine sets streamwise quantities at
C      I+1 after calculation of profiles at I+1/2
C     ********************************************
C
C---- calculate gradient parameters for power curve extrapolation
      BETM = ALOG(MD/MDI)/FLOG
      BETD = ALOG(DS/DSI)/FLOG
C
C---- set quantities for the I+1th station
      UEIP = UEI*(X(I+1)/X(I))**BETU
      MSIP = MSI*(X(I+1)/X(I))**BETN
      MDIP = MDI*(X(I+1)/X(I))**BETM
      DSIP = DSI*(X(I+1)/X(I))**BETD
      SRIP = 2.Ø*SR - SRI             ! linear extrapolation for wall shear
C
      RETURN
      END
      SUBROUTINE PROFL
      INCLUDE 'BLAKE.INC'
C     ***************************************************
C      This routine calculates the BL profiles between
C      the Ith and I+1th stations using Newton-Raphson.
C     ***************************************************
C
      DO 5 ITER=1, ITMAX    ! Newton iteration loop
C
C------ fill block tridiagonal system
        CALL SETUP
C
C------ get uncorrected profile iterates and global influence coefficients
        CALL SOLVE
C
C------ get global variable iterates and corrected profile iterates
        CALL DELTAS
C
C------ update profiles and get max U iterate
        DUMAX = Ø.
        DO 52 J=1, JJ
          F(J) = F(J) + R11(J)
          U(J) = U(J) + R21(J)
          H(J) = H(J) + R31(J)
          DUMAX = AMAX1(DUMAX,ABS(R21(J)))
   52   CONTINUE
C
C------ update UE and/or MS
        UE = UE + DUE
        MS = MS + DMS
C       UTAU will be updated from its definition in VISC
C
C------ check for divergence
        IF(UE.LE.Ø.Ø) GO TO 1Ø
        IF(MS.LE.Ø.Ø) GO TO 11
C
C------ recalculate edge quantities
        CALL ECALC
C
C------ recalculate DS, TH, and all that
        CALL DCALC
C
C------ recalculate gradient parameters
        BETU = ALOG(UE/UEI)/FLOG
        BETN = ALOG(MS/MSI)/FLOG
C
C------ recalculate UTAU, viscosity, and viscosity influence coefficients
        CALL VISC
C
C------ recalculate S and Q arrays
        CALL SQSET
C
C------ check for convergence or lack thereof
        DGLBL = ABS(DMS)/MS + ABS(DUE)/UE
        IF(DUMAX.LE.EPS .AND. DGLBL.LE.EPS) GO TO 2Ø
C
    5 CONTINUE   ! end of Newton iteration loop
C
C
C!!!! PANIC Messages.  We normally don't get to this point
C
      WRITE(LTTI,*) '[ PROFL ]: CONVERGENCE FAILED at station ',I,'.5'
      WRITE(LTTI,*) '             Max U velocity residual:          ',URES
      WRITE(LTTI,*) '             Uedge + Mass residuals :          ',DRES
      CALL STOPIT  ! Crash softly
C
   1Ø WRITE(LTTI,*) '[ PROFL ]: Negative edge velocity was calculated.'
      WRITE(LTTI,*) '             Solution probably diverged. Crashing...'
      CALL STOPIT  ! Crash softly
C
   11 WRITE(LTTI,*) '[ PROFL ]: Negative mass scale was calculated.'
      WRITE(LTTI,*) '             Solution probably diverged. Crashing...'
      CALL STOPIT  ! Crash softly
```

```
C---- We normally DO get to this point.
C
   2Ø WRITE(LTTI,*) '[ PROFL ]: Station ',I,'.5 ...',ITER,' Iterations'
      RETURN
C
      END



      SUBROUTINE ECALC
      INCLUDE 'BLAKE.INC'
C     ******************************
C      This routine calculates edge
C      quantities at X(I+1/2).
C     ******************************
C
      EE = Ø.5*GM1*UE**2/TST      ! edge Kinetic Energy to enthalpy ratio...
      EEC = 1.Ø - EE              ! ...its complement
      TE = TST*EEC                ! edge static temperature
      RHOE = RST*EEC**(1.Ø/GM1)   ! edge static density
      PE = RHOE*TE/GAM            ! edge static pressure
      ME2 = UE*UE/TE              ! edge Mach Number squared...
      ME2C = 1.Ø - ME2            ! ...its complement
      RNU = XF*RHOE*UE/MS**2      ! group in front of S and Q definitions
C
      RETURN
      END



      SUBROUTINE DCALC
      INCLUDE 'BLAKE.INC'
C     ****************************************
C      This routine calculates the profile
C      parameters DS, TH etc. at I+1/2
C     ****************************************
C
      DUNORM = Ø.    ! normalized velocity thickness for outer eddy viscosity
      THNORM = Ø.    ! normalized momentum thickness
      DO 1Ø J=1, JJ-1
        UPS = 1.Ø
        IF(J.EQ.1 .OR. J.EQ.JJ-1) UPS = Ø.5
        THNORM = THNORM + UPS*(F(J+1)-F(J)) * (1.Ø - Ø.5*(U(J+1)+U(J)))
        DUNORM = DUNORM + UPS*(1.Ø - Ø.5*(U(J+1)+U(J)))*DY(J)
   1Ø CONTINUE
C
      DSNORM = ETAE - Ø.5*(F(JJ) + F(JJ-1)) ! normalized displacement thickness
C
      SHPF = DSNORM/THNORM        ! shape parameter
C
      SC = MS/(RHOE*UE)  ! normal scaling length
      TH = THNORM*SC     ! momentum thickness
      DS = DSNORM*SC     ! displacement thickness
      MD = RHOE*UE*DS    ! mass defect
C
      RETURN
      END


      SUBROUTINE SQSET
      INCLUDE 'BLAKE.INC'
C
C---- set S and Q
      DO 1Ø J=1, JJ-1
        JP = J+1
        S(J) = RNU*(VIS(J)+VTB(J))*(U(JP)-U(J))/DY(J)
        Q(J) = RNU/DY(J)*
     &  ((VIS(J)/PR + VTB(J)/PRT)*(H(JP) - H(J))
     &    + VIS(J)*(1.- 1./PR)*EE*(U(JP)**2 - U(J)**2))
   1Ø CONTINUE
C
C---- set physical wall shear SR
      SR = S(1)*MS*UE/XF
C
      RETURN
      END
      SUBROUTINE SETUP
      INCLUDE 'BLAKE.INC'
C     ************************************************************
C      This routine sets up the block-tridiagonal system for either
C      the similarity (NSIM=1) or marching problem (NSIM=Ø).
C      Influence coefficients for variations of molecular and eddy
C      viscosities are received from subroutine VISC and incorporated
C      into the block matrix to obtain overall quadratic convergence.
C     ************************************************************
C
      IP = I+1
      IM = I-1
C
      IF(NSIM.EQ.1) XBAR = Ø.                      ! XBAR multiplies the
      IF(NSIM.EQ.Ø) XBAR = Ø.5*(XF+XB)/(XF-XB)  ! x-dependent terms
C
C---- set variational conversion factors for BETU and BETN...
C     ... dBETU = DBDU x dUE  ;  dBETN = DBDN x dMS
      DBDU = Ø.                               ! for similarity, dBETU
      DBDN = Ø.                               ! and dBETN are zero
      IF(NSIM.EQ.Ø) DBDU = 1.Ø/(UE*FLOG)
      IF(NSIM.EQ.Ø) DBDN = 1.Ø/(MS*FLOG)
C
CCC-- fill last A and B blocks and righthand side vectors
C
C---- first line: continuity (will be set in the first DO loop pass below)
C
C---- second line:  U = 1  edge boundary condition
      B21(JJ) = Ø.
      B22(JJ) = 1.Ø
      B23(JJ) = Ø.
C
      A21(JJ) = Ø.
      A22(JJ) = 1.Ø
      A23(JJ) = Ø.
C
      R21(JJ) = 1.Ø - Ø.5*(U(JJ) + U(JJ-1))
      R22(JJ) = Ø.
      R23(JJ) = Ø.
      R24(JJ) = Ø.
C
C---- third line:  H = 1  edge boundary condition
      B31(JJ) = Ø.
      B32(JJ) = Ø.
      B33(JJ) = 1.Ø
C
      A31(JJ) = Ø.
      A32(JJ) = Ø.
      A33(JJ) = 1.Ø
C
      R31(JJ) = 1.Ø - Ø.5*(H(JJ) + H(JJ-1))
      R32(JJ) = Ø.
      R33(JJ) = Ø.
      R34(JJ) = Ø.
C
      DO 1ØØØ JBACK=1, JJ-1    ! sweep from edge to wall
C
C---- set shorthand definitions
      J = JJ - JBACK + 1
      JM = J-1
      JP = J+1
C
      DYP = DY(J)
      DYM = DY(JM)
      DYO = Ø.5*(DYP+DYM)
C
      FSM = F(J) + F(JM)
      USM = U(J) + U(JM)
      HSM = H(J) + H(JM)
C
      FDM = F(J) - F(JM)
      UDM = U(J) - U(JM)
      HDM = H(J) - H(JM)
      SEP = 1.Ø                    ! separation trigger
      IF(USM .LE. Ø.) SEP = Ø. !
C
      FYM = (FDM + FB(J) - FB(JM))*SEP
      UYM =  UDM + UB(J) - UB(JM)
      HYM =  HDM + HB(J) - HB(JM)
C
      FXM =  FSM - FB(J) - FB(JM)
      UXM = (USM - UB(J) - UB(JM))*SEP
      HXM = (HSM - HB(J) - HB(JM))*SEP
C
CCC-----------------
CCC-- continuity ---
C
C---- set more shorthand
      TRM = H(JM) - EE*U(JM)**2
      TRO = H(J) - EE*U(J)**2
      RM = EEC/TRM
      RO = EEC/TRO
C
C---- fill first line of A, B, C  blocks
      B11(J) = 2.Ø
      B12(J) = DYM*(RM + 2.Ø*EE*EEC*(U(JM)/TRM)**2)
      B13(J) = -DYM*U(JM)*EEC/TRM**2
C
      A11(J) = -2.Ø
      A12(J) = DYM*(RO + 2.Ø*EE*EEC*(U(J)/TRO)**2)
      A13(J) = -DYM*U(J)*EEC/TRO**2
C
      C11(J) = Ø.
      C12(J) = Ø.
      C13(J) = Ø.
C
C---- fill first line of righthand side column vectors
      R11(J) = 2.Ø*(F(J)-F(JM)) - DYM*(RO*U(J)+RM*U(JM))
      R12(J) = -DYM*GM1*UE/TST
     & *(U(JM)*(H(JM)-U(JM)**2)/TRM**2 + U(J)*(H(J)-U(J)**2)/TRO**2)
      R13(J) = Ø.
      R14(J) = Ø.
C
      IF(J.EQ.JJ) GO TO 11Ø
C
CCC-----------------
CCC-- x-momentum ---
C
C---- set weights for shear influence coefficients
      AUM = RNU*(VIS(JM)+VTB(JM))/DYM   ! dS(J-1/2)/dU(J)
      AUP = RNU*(VIS(J)+VTB(J))/DYP     ! dS(J+1/2)/dU(JP)
      AVTM = RNU*UDM/DYM     ! dS(J-1/2)/dmu(J-1/2)
      AVTP = RNU*UDP/DYP     ! dS(J+1/2)/dmu(J+1/2)
C
C---- fill second line of A, B, C  blocks
      B21(J) = Ø.25*( BETU*USM + BETN*UDM + XBAR*(UYM + UXM))/DYM
      B22(J) = Ø.25*(-BETU*FDM - BETN*FSM - XBAR*(FXM + FYM))/DYM
     & - (-AUM + AVTM*(VUO(JM)+TUO(JM)))/DYO
      B23(J) = -AVTM*(VHO(JM)+THO(JM))/DYO
C
      A21(J) = Ø.25*( BETU*USP + BETN*UDP + XBAR*(UYP + UXP))/DYP
     &       + Ø.25*(-BETU*USM + BETN*UDM + XBAR*(UYM - UXM))/DYM
      A22(J) = Ø.25*(-BETU*FDP - BETN*FSP - XBAR*(FXP + FYP))/DYP
     &       + Ø.25*(-BETU*FDM - BETN*FSM + XBAR*(FXM - FYM))/DYM
     & + (-AUP + AVTP*(VUO(J)+TUO(J)))/DYO
     & - ( AUM + AVTM*(VUP(JM)+TUP(JM)))/DYO
      A23(J) = AVTP*(VHO(J)+THO(J))/DYO
     &       - AVTM*(VHP(JM)+THP(JM))/DYO
C
      C21(J) = Ø.25*(-BETU*USP + BETN*UDP + XBAR*(UYP - UXP))/DYP
      C22(J) = Ø.25*(-BETU*FDP + BETN*FSP + XBAR*(FXP - FYP))/DYP
     & + (AUP + AVTP*(VUP(J)+TUP(J)))/DYO
      C23(J) = AVTP*(VHP(J)+THP(J))/DYO
```

```
C---- still more shorthand
      UDFDY = Ø.25*(USP*FDP/DYP + USM*FDM/DYM)
      FDUDY = Ø.25*(FSP*UDP/DYP + FSM*UDM/DYM)
      RLHS = (SB(J)-SB(JM))/DYO + BETUB
     & - BETUB*((UB(JP)+UB(J))*(FB(JP)-FB(J))/DYP
     &          +(UB(J)+UB(JM))*(FB(J)-FB(JM))/DYM)*Ø.25
     & + BETNB*((FB(JP)+FB(J))*(UB(JP)-UB(J))/DYP
     &          +(FB(J)+FB(JM))*(UB(J)-UB(JM))/DYM)*Ø.25
C
C---- fill second line of righthand side column vectors
      R21(J) = Ø.25*XBAR*((FYP*UXP-FXP*UYP)/DYP + (FYM*UXM-FXM*UYM)/DYM)
     & - (S(J)-S(JM))/DYO - BETU + BETU*UDFDY - BETN*FDUDY - RLHS
      R22(J) = (1.Ø - UDFDY)*DBDU
     & + (AVTP*(VUE(J)+TUE(J)) - AVTM*(VUE(JM)+TUE(JM)))/DYO
     & + (S(J)-S(JM))/DYO*ME2C/UE
      R23(J) = FDUDY*DBDN
     & + (AVTP*TMS(J) - AVTM*TMS(JM))/DYO
     & - (S(J)-S(JM))/DYO*2.Ø/MS
      R24(J) = (AVTP*TUT(J) - AVTM*TUT(JM))/DYO
C
CCC-------------
CCC-- energy ---
C
C---- set weights for heat flux influence coefficients
      AUM = VIS(JM)*(1.-1./PR)*2.Ø*EE*RNU/DYM           ! dQ(J-1/2)/dU(J)
      AHM = (VIS(JM)/PR + VTB(JM)/PRT)*RNU/DYM          ! dQ(J-1/2)/dH(J)
      AVM = (HDM/PR + (1.-1./PR)*EE*USM*UDM)*RNU/DYM ! dQ(J-1/2)/dmu(J-1/2)
      ATM = HDM/PRT*RNU/DYM                             ! dQ(J-1/2)/dmut(J-1/2)
      AUEM = VIS(JM)*(1.-1./PR)*GM1*UE/TST*USM*UDM*RNU/DYM ! dQ(J-1/2)/due
C
      AUP = VIS(J)*(1.-1./PR)*2.Ø*EE*RNU/DYP            ! dQ(J+1/2)/dU(JP)
      AHP = (VIS(J)/PR + VTB(J)/PRT)*RNU/DYP            ! dQ(J+1/2)/dH(JP)
      AVP = (HDP/PR + (1.-1./PR)*EE*USP*UDP)*RNU/DYP ! dQ(J+1/2)/dmu(J+1/2)
      ATP = HDP/PRT*RNU/DYP                             ! dQ(J+1/2)/dmut(J+1/2)
      AUEP = VIS(J)*(1.-1./PR)*GM1*UE/TST*USP*UDP*RNU/DYP  ! dQ(J+1/2)/due
C
C---- fill second line of A, B, C  blocks
      B31(J) = Ø.25*( BETH*HSM + BETN*HDM + XBAR*(HYM + HXM))/DYM
      B32(J) = -(-AUM*U(JM) + AVM*VUO(JM) + ATM*TUO(JM))/DYO
      B33(J) = Ø.25*(-BETH*FDM - BETN*FSM - XBAR*(FXM + FYM))/DYM
     & - (-AHM + AVM*VHO(JM) + ATM*THO(JM))/DYO
C
      A31(J) = Ø.25*( BETH*HSP + BETN*HDP + XBAR*(HYP + HXP))/DYP
     &        + Ø.25*(-BETH*HSM + BETN*HDM + XBAR*(HYM - HXM))/DYM
      A32(J) = (-AUP*U(J) + AVP*VUO(J) + ATP*TUO(J))/DYO
     &        - ( AUM*U(J) + AVM*VUP(JM) + ATM*TUP(JM))/DYO
      A33(J) = Ø.25*(-BETH*FDP - BETN*FSP - XBAR*(FXP + FYP))/DYP
     &        + Ø.25*(-BETH*FDM + BETN*FSM + XBAR*(FXM - FYM))/DYM
     & + (-AHP + AVP*VHO(J) + ATP*THO(J))/DYO
     & - ( AHM + AVM*VHP(JM) + ATM*THP(JM))/DYO
C
      C31(J) = Ø.25*(-BETH*HSP + BETN*HDP + XBAR*(HYP - HXP))/DYP
      C32(J) = (AUP*U(JP) + AVP*VHP(J) + ATP*THP(J))/DYO
      C33(J) = Ø.25*(-BETH*FDP + BETN*FSP + XBAR*(FXP - FYP))/DYP
     & + (AHP + AVP*VHP(J) + ATP*THP(J))/DYO
C
      HDFDY = Ø.25*(HSP*FDP/DYP + HSM*FDM/DYM)
      FDHDY = Ø.25*(FSP*HDP/DYP + FSM*HDM/DYM)
      RLHS = (QB(J)-QB(JM))/DYO
     & - BETHB*((HB(JP)+HB(J))*(FB(JP)-FB(J))/DYP
     &          +(HB(J)+HB(JM))*(FB(J)-FB(JM))/DYM)*Ø.25
     & + BETNB*((FB(JP)+FB(J))*(HB(JP)-HB(J))/DYP
     &          +(FB(J)+FB(JM))*(HB(J)-HB(JM))/DYM)*Ø.25
C
C---- fill third line of righthand side column vectors
      R31(J) = Ø.25*XBAR*((FYP*HXP-FXP*HYP)/DYP + (FYM*HXM-FXM*HYM)/DYM)
     & - (Q(J)-Q(JM))/DYO + BETH*HDFDY - BETN*FDHDY - RLHS
      R32(J) = (AVP*VUE(J) + ATP*TUE(J) - AVM*VUE(JM) - ATM*TUE(JM))/DYO
     & + (Q(J)-Q(JM))/DYO*ME2C/UE
     & + (AUEP-AUEM)/DYO
      R33(J) = FDHDY*DBDN
     & + (ATP*TMS(J) - ATM*TMS(JM))/DYO
     & - (Q(J)-Q(JM))/DYO*2.Ø/MS
      R34(J) = (ATP*TUT(J) - ATM*TUT(JM))/DYO
```

```
C---- set shorthand definitions for next J loop sweep
C
  11Ø FSP = FSM
      USP = USM
      HSP = HSM
C
      FDP = FDM
      UDP = UDM
      HDP = HDM
C
      FYP = FYM
      UYP = UYM
      HYP = HYM
C
      FXP = FXM
      UXP = UXM
      HXP = HXM
C
 1ØØØ CONTINUE    ! end of J loop
C
CCC-- set first A and C blocks and righthand sides
C
C---- first line:  F = Ø  wall boundary condition
      A11(1) = 1.Ø
      A12(1) = Ø.
      A13(1) = Ø.
C
      C11(1) = 1.Ø
      C12(1) = Ø.
      C13(1) = Ø.
C
      R11(1) = -(F(1) + F(2))
      R12(1) = Ø.
      R13(1) = Ø.
      R14(1) = Ø.
C
C---- second line: U = Ø  boundary condition
      A21(1) = Ø.
      A22(1) = 1.Ø
      A23(1) = Ø.
C
      C21(1) = Ø.
      C22(1) = 1.Ø
      C23(1) = Ø.
C
      R21(1) = -(U(1) + U(2))
      R22(1) = Ø.
      R23(1) = Ø.
      R24(1) = Ø.
C
C---- third line:  bh H + (1-bh) dH/dy = bcon   boundary condition
      A31(1) = Ø.
      A32(1) = Ø.
      A33(1) = Ø.5*BH - (1.Ø-BH)/DY(1)
C
      C31(1) = Ø.
      C32(1) = Ø.
      C33(1) = Ø.5*BH + (1.Ø-BH)/DY(1)
C
      R31(1) = BCON + (BH-1.Ø)*(H(2)-H(1))/DY(1) - .5*BH*(H(1)+H(2))
      R32(1) = Ø.
      R33(1) = Ø.
      R34(1) = Ø.
C
      RETURN
      END
      SUBROUTINE SOLVE
      INCLUDE 'BLAKE.INC'
C     ****************************************************
C      This routine solves the block-tridiagonal system
C      (with four righthand sides) received from SETUP.
C      A UL decomposition method is used.  Diagonal
C      dominance of the diagonal blocks is assumed.
C     ****************************************************
C
      DO 100 JBACK=1, JJ  ! backward elimination loop
      J = JJ - JBACK + 1
      JP = J + 1
C
      IF(J.EQ.JJ) GO TO 110  ! last Aj block is unchanged
C
C---- eliminate Cj block (calculate modified Aj block and Rj vectors)
       A11(J) = A11(J) - (C11(J)*B11(JP)+C12(J)*B21(JP)+C13(J)*B31(JP))
       A12(J) = A12(J) - (C11(J)*B12(JP)+C12(J)*B22(JP)+C13(J)*B32(JP))
       A13(J) = A13(J) - (C11(J)*B13(JP)+C12(J)*B23(JP)+C13(J)*B33(JP))
C
       A21(J) = A21(J) - (C21(J)*B11(JP)+C22(J)*B21(JP)+C23(J)*B31(JP);
       A22(J) = A22(J) - (C21(J)*B12(JP)+C22(J)*B22(JP)+C23(J)*B32(JP))
       A23(J) = A23(J) - (C21(J)*B13(JP)+C22(J)*B23(JP)+C23(J)*B33(JP))
C
       A31(J) = A31(J) - (C31(J)*B11(JP)+C32(J)*B21(JP)+C33(J)*B31(JP))
       A32(J) = A32(J) - (C31(J)*B12(JP)+C32(J)*B22(JP)+C33(J)*B32(JP))
       A33(J) = A33(J) - (C31(J)*B13(JP)+C32(J)*B23(JP)+C33(J)*B33(JP))
C
       R11(J) = R11(J) - (C11(J)*R11(JP)+C12(J)*R21(JP)+C13(J)*R31(JP))
       R12(J) = R12(J) - (C11(J)*R12(JP)+C12(J)*R22(JP)+C13(J)*R32(JP))
       R13(J) = R13(J) - (C11(J)*R13(JP)+C12(J)*R23(JP)+C13(J)*R33(JP))
       R14(J) = R14(J) - (C11(J)*R14(JP)+C12(J)*R24(JP)+C13(J)*R34(JP))
C
       R21(J) = R21(J) - (C21(J)*R11(JP)+C22(J)*R21(JP)+C23(J)*R31(JP))
       R22(J) = R22(J) - (C21(J)*R12(JP)+C22(J)*R22(JP)+C23(J)*R32(JP))
       R23(J) = R23(J) - (C21(J)*R13(JP)+C22(J)*R23(JP)+C23(J)*R33(JP))
       R24(J) = R24(J) - (C21(J)*R14(JP)+C22(J)*R24(JP)+C23(J)*R34(JP))
C
       R31(J) = R31(J) - (C31(J)*R11(JP)+C32(J)*R21(JP)+C33(J)*R31(JP))
       R32(J) = R32(J) - (C31(J)*R12(JP)+C32(J)*R22(JP)+C33(J)*R32(JP))
       R33(J) = R33(J) - (C31(J)*R13(JP)+C32(J)*R23(JP)+C33(J)*R33(JP))
       R34(J) = R34(J) - (C31(J)*R14(JP)+C32(J)*R24(JP)+C33(J)*R34(JP))
C
CCC-- solve AjGj = Bj and AjWj = Rj systems by Gaussian elimination
C      Gj is stored in Bj space and Wj is stored in Rj space
C
C---- normalize 1st row
  110 A11INV = 1.0/A11(J)
       A12(J) = A12(J)*A11INV
       A13(J) = A13(J)*A11INV
C
       B11(J) = B11(J)*A11INV
       B12(J) = B12(J)*A11INV
       B13(J) = B13(J)*A11INV
C
       R11(J) = R11(J)*A11INV
       R12(J) = R12(J)*A11INV
       R13(J) = R13(J)*A11INV
       R14(J) = R14(J)*A11INV
C
C---- eliminate 2nd row
       A22(J) = A22(J) - A12(J)*A21(J)
       A23(J) = A23(J) - A13(J)*A21(J)
C
       B21(J) = B21(J) - B11(J)*A21(J)
       B22(J) = B22(J) - B12(J)*A21(J)
       B23(J) = B23(J) - B13(J)*A21(J)
C
       R21(J) = R21(J) - R11(J)*A21(J)
       R22(J) = R22(J) - R12(J)*A21(J)
       R23(J) = R23(J) - R13(J)*A21(J)
       R24(J) = R24(J) - R14(J)*A21(J)
```

```
C---- eliminate 3rd row
      A32(J) = A32(J) - A12(J)*A31(J)
      A33(J) = A33(J) - A13(J)*A31(J)
C
      B31(J) = B31(J) - B11(J)*A31(J)
      B32(J) = B32(J) - B12(J)*A31(J)
      B33(J) = B33(J) - B13(J)*A31(J)
C
      R31(J) = R31(J) - R11(J)*A31(J)
      R32(J) = R32(J) - R12(J)*A31(J)
      R33(J) = R33(J) - R13(J)*A31(J)
      R34(J) = R34(J) - R14(J)*A31(J)
C
C---- normalize 2nd row
      A22INV = 1.Ø/A22(J)
      A23(J) = A23(J)*A22INV
C
      B21(J) = B21(J)*A22INV
      B22(J) = B22(J)*A22INV
      B23(J) = B23(J)*A22INV
C
      R21(J) = R21(J)*A22INV
      R22(J) = R22(J)*A22INV
      R23(J) = R23(J)*A22INV
      R24(J) = R24(J)*A22INV
C
C---- eliminate 3rd row
      A33(J) = A33(J) - A23(J)*A32(J)
C
      B31(J) = B31(J) - B21(J)*A32(J)
      B32(J) = B32(J) - B22(J)*A32(J)
      B33(J) = B33(J) - B23(J)*A32(J)
C
      R31(J) = R31(J) - R21(J)*A32(J)
      R32(J) = R32(J) - R22(J)*A32(J)
      R33(J) = R33(J) - R23(J)*A32(J)
      R34(J) = R34(J) - R24(J)*A32(J)
C
C---- normalize 3rd row
      A33INV = 1.Ø/A33(J)
      B31(J) = B31(J)*A33INV
      B32(J) = B32(J)*A33INV
      B33(J) = B33(J)*A33INV
C
      R31(J) = R31(J)*A33INV
      R32(J) = R32(J)*A33INV
      R33(J) = R33(J)*A33INV
      R34(J) = R34(J)*A33INV
C
CCC-- back substitution
C
C---- 2nd row
      B21(J) = B21(J) - B31(J)*A23(J)
      B22(J) = B22(J) - B32(J)*A23(J)
      B23(J) = B23(J) - B33(J)*A23(J)
C
      R21(J) = R21(J) - R31(J)*A23(J)
      R22(J) = R22(J) - R32(J)*A23(J)
      R23(J) = R23(J) - R33(J)*A23(J)
      R24(J) = R24(J) - R34(J)*A23(J)
C
C---- 1st row
      B11(J) = B11(J) - B21(J)*A12(J) - B31(J)*A13(J)
      B12(J) = B12(J) - B22(J)*A12(J) - B32(J)*A13(J)
      B13(J) = B13(J) - B23(J)*A12(J) - B33(J)*A13(J)
C
      R11(J) = R11(J) - R21(J)*A12(J) - R31(J)*A13(J)
      R12(J) = R12(J) - R22(J)*A12(J) - R32(J)*A13(J)
      R13(J) = R13(J) - R23(J)*A12(J) - R33(J)*A13(J)
      R14(J) = R14(J) - R24(J)*A12(J) - R34(J)*A13(J)
C
  1ØØ CONTINUE
```

```
      DO 200 J=2, JJ   ! forward substitution loop
      JM = J-1
C
      R11(J) = R11(J) - (B11(J)*R11(JM)+B12(J)*R21(JM)+B13(J)*R31(JM))
      R12(J) = R12(J) - (B11(J)*R12(JM)+B12(J)*R22(JM)+B13(J)*R32(JM))
      R13(J) = R13(J) - (B11(J)*R13(JM)+B12(J)*R23(JM)+B13(J)*R33(JM))
      R14(J) = R14(J) - (B11(J)*R14(JM)+B12(J)*R24(JM)+B13(J)*R34(JM))
C
      R21(J) = R21(J) - (B21(J)*R11(JM)+B22(J)*R21(JM)+B23(J)*R31(JM))
      R22(J) = R22(J) - (B21(J)*R12(JM)+B22(J)*R22(JM)+B23(J)*R32(JM))
      R23(J) = R23(J) - (B21(J)*R13(JM)+B22(J)*R23(JM)+B23(J)*R33(JM))
      R24(J) = R24(J) - (B21(J)*R14(JM)+B22(J)*R24(JM)+B23(J)*R34(JM))
C
      R31(J) = R31(J) - (B31(J)*R11(JM)+B32(J)*R21(JM)+B33(J)*R31(JM))
      R32(J) = R32(J) - (B31(J)*R12(JM)+B32(J)*R22(JM)+B33(J)*R32(JM))
      R33(J) = R33(J) - (B31(J)*R13(JM)+B32(J)*R23(JM)+B33(J)*R33(JM))
      R34(J) = R34(J) - (B31(J)*R14(JM)+B32(J)*R24(JM)+B33(J)*R34(JM))
C
  200 CONTINUE
C
      RETURN
      END
      SUBROUTINE DELTAS
      INCLUDE 'BLAKE.INC'
C     ******************************************
C      This routine calculates the iterates of
C      global unknowns and uses them to correct
C      the profile iterates using the influence
C      coefficients calculated by SOLVE.
C     ******************************************
C
C---- calculate RNORM and its global iterate influence coefficients
      RNORM = Ø.
      DNRES = Ø.
      DNDUE = Ø.                 ! dNorm/due
      DNDMS = Ø.                 ! dNorm/dn
      DNDUT = Ø.                 ! dNorm/dUtau
      DO 1ØØ J=1, JJ-1
        JP = J+1
        UMID = Ø.5*(U(JP) + U(J))
        UPS = 1.Ø
        IF(J.EQ.1 .OR. JP.EQ.JJ) UPS = Ø.5
        RNORM = RNORM + UPS*UMID*(1.Ø - UMID)*DY(J)
        DNRES = DNRES + UPS*DY(J)*(Ø.5 - UMID)*(R21(JP) + R21(J))
        DNDUE = DNDUE + UPS*DY(J)*(Ø.5 - UMID)*(R22(JP) + R22(J))
        DNDMS = DNDMS + UPS*DY(J)*(Ø.5 - UMID)*(R23(JP) + R23(J))
        DNDUT = DNDUT + UPS*DY(J)*(Ø.5 - UMID)*(R24(JP) + R24(J))
  1ØØ CONTINUE
C
C---- calculate influence coefficients for U, H, Ue, Ms, and Utau iterates
      SMU = RNU*(U(2)-U(1))/DY(1)          ! dSw/dmu
      SU2 =  RNU*VIS(1)/DY(1)              ! dSw/dU2
      SU1 = -RNU*VIS(1)/DY(1)              ! dSw/dU1
      SUE = S(1)*ME2C/UE                   ! dSw/due
      SMS = -2.Ø*S(1)/MS                   ! dSw/dn
C
      RH2 = -2.Ø*EEC/(H(2) + H(1))**2      ! dRw/dH2
      RH1 = -2.Ø*EEC/(H(2) + H(1))**2      ! dRw/dH1
      RUE = -2.Ø*GM1*UE/TST/(H(2) + H(1)) ! dRw/due
C
C
C**** Set up system for DUE, DMS, and DUT
C
C---- first line ... drive RNORM to 1
      A1 = DNDUE
      B1 = DNDMS
      C1 = DNDUT
      D1 = RNORM - 1.Ø + DNRES
C
C---- second line ... drive current Utau to UTAU
      RWALL = 2.Ø*EEC/(H(2) + H(1))        ! density at wall
      A2 = SU2*R22(2) + SU1*R22(1)
     &   + SMU*(VHP(1)*R32(2) + VHO(1)*R32(1) - VUE(1)) - SUE
     &   - UTAU**2 * (RH2*R32(2) + RH1*R32(1) - RUE)
      B2 = SU2*R23(2) + SU1*R23(1)
     &   + SMU*(VHP(1)*R33(2) + VHO(1)*R33(1)) - SMS
     &   - UTAU**2 * (RH2*R33(2) + RH1*R33(1))
      C2 = SU2*R24(2) + SU1*R24(1)
     &   + SMU*(VHP(1)*R34(2) + VHO(1)*R34(1))
     &   - UTAU**2 * (RH2*R34(2) + RH1*R34(1)) + 2.Ø*RWALL*UTAU
      D2 = SU2*R21(2) + SU1*R21(1)
     &   + SMU*(VHP(1)*R31(2) + VHO(1)*R31(1))
     &   - UTAU**2 * (RH2*R31(2) + RH1*R31(1))
     &   + S(1) - RWALL*UTAU**2
C
C---- third line ... drive (whatever's specified) to specified value
      IF(KODE.EQ.1) CALL KODE1
      IF(KODE.EQ.2) CALL KODE2
      IF(KODE.EQ.3) CALL KODE3
      IF(KODE.EQ.4) CALL KODE4
```

```
C
C--
C--                    |A1   B1   C1| |DUE|     |D1|
C--                    |          | | |   |     |  |
C--   3X3 system:      |A2   B2   C2|X|DMS|  =  |D2|
C--                    |          | | |   |     |  |
C--                    |A3   B3   C3| |DUT|     |D3|
C--
C
C---- solve 3x3 system for global iterates
   1Ø CALL GSOLVE
C
CCC-- correct profile iterates
      DO 12 J=1, JJ
        R11(J) = R11(J) - DUE*R12(J) - DMS*R13(J) - DUT*R14(J)
        R21(J) = R21(J) - DUE*R22(J) - DMS*R23(J) - DUT*R24(J)
        R31(J) = R31(J) - DUE*R32(J) - DMS*R33(J) - DUT*R34(J)
   12 CONTINUE
C
      RETURN
      END


      SUBROUTINE GSOLVE
      INCLUDE 'BLAKE.INC'
C
C---- solve 3x3 system by using Cramer's rule
      DET =  A3*(B1*C2 - C1*B2)
     &      -B3*(A1*C2 - C1*A2)
     &      +C3*(A1*B2 - B1*A2)
      DUE = (D3*(B1*C2 - C1*B2)
     &      -B3*(D1*C2 - C1*D2)
     &      +C3*(D1*B2 - B1*D2))/DET
      DMS = (A3*(D1*C2 - C1*D2)
     &      -D3*(A1*C2 - C1*A2)
     &      +C3*(A1*D2 - D1*A2))/DET
      DUT = (A3*(B1*D2 - D1*B2)
     &      -B3*(A1*D2 - D1*A2)
     &      +D3*(A1*B2 - B1*A2))/DET
C
      RETURN
      END
```

```
C     ************************************************************************
C
C      Each KODEn routine sets up the third line of the 3x3 system for the
C      global iterates which is then solved in DELTAS. Quadratic convergence
C      to some specified quantity (stored in SPECF) is then achieved.
C
C     ************************************************************************

      SUBROUTINE KODE1
      INCLUDE 'BLAKE.INC'
C
C---- Ue specified
C
      A3 = 1.Ø
      B3 = Ø.
      C3 = Ø.
      D3 = SPECF - UE
C
      RETURN
      END


      SUBROUTINE KODE2
      INCLUDE 'BLAKE.INC'
C
C---- Rhoe*Ue*Dstar  (= mass defect) specified
C
      JM = JJ-1
      A3 = Ø.5*MS*(R12(JJ) + R12(JM))
      B3 = Ø.5*MS*(R13(JJ) + R13(JM)) + ETAE - Ø.5*(F(JJ) + F(JM))
      C3 = Ø.5*MS*(R14(JJ) + R14(JM))
      D3 = Ø.5*MS*(R11(JJ) + R11(JM)) + SPECF - MD
C
      RETURN
      END


      SUBROUTINE KODE3
      INCLUDE 'BLAKE.INC'
C
C---- Dstar specified
C
      JM = JJ-1
      A3 = Ø.5*MS*(R12(JJ) + R12(JM))
     &   - MS/UE*(ETAE - Ø.5*(F(JJ) + F(JM)))*ME2C
      B3 = Ø.5*MS*(R13(JJ) + R13(JM)) + ETAE - Ø.5*(F(JJ) + F(JM))
      C3 = Ø.5*MS*(R14(JJ) + R14(JM))
      D3 = Ø.5*MS*(R11(JJ) + R11(JM)) + RHOE*UE*SPECF - MD
C
      RETURN
      END


      SUBROUTINE KODE4
      INCLUDE 'BLAKE.INC'
C
C---- Wall Shear specified
C
C     SR =  current physical wall shear  (set in SQSET)
C
      SU2 = UE/SC*VIS(1)/DY(1)   ! dSr/d(U2-U1)
      SMU = SR/VIS(1)            ! dSr/dmu
C
      A3 = SU2*(R22(2) - R22(1))
     &   + SMU*(VHP(1)*R32(2) + VHO(1)*R32(1) - VUE(1))
     &   - SR/UE*(2.Ø - ME2)
      B3 = SU2*(R23(2) - R23(1))
     &   + SMU*(VHP(1)*R33(2) + VHO(1)*R33(1)) + SR/MS
      C3 = SU2*(R24(2) - R24(1))
     &   + SMU*(VHP(1)*R34(2) + VHO(1)*R34(1))
      D3 = SU2*(R21(2) - R21(1))
     &   + SMU*(VHP(1)*R31(2) + VHO(1)*R31(1)) + SR - SPECF
C
      RETURN
      END
      SUBROUTINE VISC
      INCLUDE 'BLAKE.INC'
C     ******************************************************************
C      This routine calculates molecular and eddy viscosities using
C      the current boundary layer profiles.  Sutherland's formula and
C      the Cebeci-Smith 2-layer turbulence model is used.  Influence
C      coefficients for viscosity variations are also calculated to
C      give overall quadratic convergence.
C     ******************************************************************
C
C---- empirical turbulence constants
      DATA      VKAP,   DAMPC,    ALPHA,   PPC
     &       /  Ø.4Ø,   26.Ø,     Ø.Ø168,  11.8  /
C
C---- set wall shear velocity UTAU
      T = Ø.5*TST*(H(1) + H(2))
      V1 = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
      SWALL = RNU*V1*(U(2)-U(1))/DY(1)
      RWALL = 2.Ø*EEC/(H(1) + H(2))
      UTAU = SQRT(ABS(SWALL)/RWALL)
      IF(UTAU.LT.1.E-Ø4) UTAU = 1.E-Ø4     ! zero UTAU is a no-no
C
C---- assorted shorthand
      ECONST = SQRT(SRE*MS**3/(RHOE*UE*XF))
      BCONST = ECONST*UTAU/DAMPC
      DBDU = Ø.
      IF(NSIM.EQ.Ø) DBDU = 1.Ø/(UE*FLOG)           ! dBETU/due
C
C---- set pressure gradient correction factor PN
      PTEMP = VIS(1)/(ECONST*UTAU**3*RWALL**2)
      PPLUS = BETU*PTEMP
      PN2 = 1.Ø - PPC*PPLUS
      IF(PN2.LE.Ø.Ø) GO TO 8ØØ    ! test if correction factor is imaginary (!)
      PN = SQRT(PN2)
C
      TR1 = H(1) - EE*U(1)**2
C
      RU1 = 2.Ø*EE*EEC*U(1)/TR1**2
      RH1 = -EEC/TR1**2
      RUE1 = -GM1*UE*(H(1) - U(1)**2)/(TST*TR1**2)
C
      T1 = TST*TR1
      R1 = EEC/TR1
      ETA = Ø.
C
CCC-- inner eddy viscosity loop
      DO 2Ø J=1, JJ-1
        JP = J+1
        TR2 = H(JP) - EE*U(JP)**2
C
        RU2 = 2.Ø*EE*EEC*U(JP)/TR1**2                       ! dRj+1/dUj+1
        RH2 = -EE/TR2**2                                    ! dRj+1/dHj+1
        RUE2 = -GM1*UE*(H(JP) - U(JP)**2)/(TST*TR2**2)      ! dRj+1/due
C
        T2 = TST*TR2
        R2 = EEC/TR2
        T = Ø.5*(T1 + T2)    ! temperature at J+1/2
        R = Ø.5*(R1 + R2)    ! density     at J+1/2
C
C------ test if temperature is negative
        IF(T.LT.Ø.Ø) GO TO 7ØØ
C
C------ set molecular viscosity with Sutherland's formula
        VIS(J) = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
        VTB(J) = Ø.
C
CCC---- set coefficients for molecular viscosity iterates (dmu)j
C
C       dmu = (dmu/dU)dU + (dmu/dH)dH + ... etc
C
        DMUDT = Ø.5*VIS(J)*(1.5/T - 1.Ø/(T+TVCON))    ! dmu/dT
C
C------ Uj and Uj+1 influence coefficients
        VUP(J) = -DMUDT*2.Ø*TST*EE*U(JP)              ! dmu/dUj+1
        VUO(J) = -DMUDT*2.Ø*TST*EE*U(J)               ! dmu/dUj
```

```
C------ Hj and Hj+1 influence coefficients
        VHP(J) = DMUDT*TST                               ! dmu/dHj+1
        VHO(J) = DMUDT*TST                               ! dmu/dHj
C
C------ Ue influence coefficient
        VUE(J) = -DMUDT*GM1*UE*(U(JP)**2 + U(J)**2)   ! dmu/due
C
C------ don't bother calculating eddy viscosity if TURB = Ø
        IF(TURB.EQ.Ø.Ø) GO TO 2Ø5
C
        US = ABS(U(JP) - U(J))
        SGN = 1.Ø
        IF(U(JP).LT.U(J)) SGN = -1.Ø
        BK = BCONST*PN*R/VIS(J)
        EK = Ø.
        IF(ETA*BK .LT. 3Ø.Ø) EK = EXP(-ETA*BK)
        YL = VKAP*ETA*(1.Ø - EK)
        VTP = R*YL*YL*MS*SRE/DY(J)*TURB
C
C------ set inner eddy viscosity
        VTB(J) = VTP*US
C
C------ calculate outer eddy viscosity
        VTBOUT = ALPHA*R*MS*DUNORM*SRE*TURB
C
C------ go to outer viscosity loop if inner-outer match point has been reached
        IF(VTB(J).GT.VTBOUT) GO TO 3Ø
C
CCC---- set coefficients for inner eddy viscosity iterates (dmut)j
C
C       dmut = (dmut/dU)dU + (dmut/dH)dH + ... etc
C
        CK = VKAP*ETA*ETA*EK
        DK = Ø.
        IF(J.GT.1) DK = 2.Ø*CK*BK/YL
C
C------ Uj and Uj+1 influence coefficients
        TUP(J) =  SGN*VTP + Ø.5*VTB(J)*RU2/R + VTB(J)*DK        ! dmut/dUj+1
     &   *(-VUP(J)/VIS(J) + Ø.5*RU2/R
     &    - Ø.5*PPC*PPLUS/PN2*(VUP(J)/VIS(1) - Ø.5*RU2/RWALL))
        TUO(J) = -SGN*VTP + Ø.5*VTB(J)*RU2/R + VTB(J)*DK        ! dmut/dUj+1
     &   *(-VUO(J)/VIS(J) + Ø.5*RU1/R
     &    - Ø.5*PPC*PPLUS/PN2*(VUO(J)/VIS(1) - Ø.5*RU1/RWALL))
C
C------ Hj and Hj+1 influence coefficients
        THP(J) = Ø.5*VTB(J)*RH2/R + VTB(J)*DK                   ! dmut/dHj+1
     &   *(-VHP(J)/VIS(J) + Ø.5*RH2/R
     &    - Ø.5*PPC*PPLUS/PN2*(VHP(J)/VIS(1) - Ø.5*RH2/RWALL))
        THO(J) = Ø.5*VTB(J)*RH1/R + VTB(J)*DK                   ! dmut/dHj
     &   *(-VHO(J)/VIS(J) + Ø.5*RH1/R
     &    - Ø.5*PPC*PPLUS/PN2*(VHO(J)/VIS(1) - Ø.5*RH1/RWALL))
C
C------ Ue influence coefficient
        TUE(J) = VTB(J)*Ø.5*(RUE2+RUE1)/R + VTB(J)*DK           ! dmut/due
     &    *(-VUE(J)/VIS(J) + Ø.5*(RUE2+RUE1)/R - Ø.5*EEC/UE
     &      - Ø.5*PPC/PN2
     &        *(PTEMP*DBDU + Ø.5*PPLUS*EEC/UE + PPLUS*VUE(1)/VIS(1)
     &          + PPLUS/RWALL*2.Ø*GM1*UE/(TST*(H(1)+H(2))) ))
C
C------ Ms influence coefficient
        TMS(J) = VTB(J)/MS + VTB(J)*DK*(1.5 + Ø.75*PPC*PPLUS/PN2)/MS ! dmut/dms
C
C------ Utau influence coefficient
        TUT(J) = VTB(J)*DK*(1.Ø - 1.5*PPC*PPLUS/PN2)/UTAU       ! dmut/dUtau
C
  2Ø5   TR1 = TR2
        RU1 = RU2
        RH1 = RH2
        RUE1 = RUE2
        T1 = T2
        R1 = R2
        ETA = ETA + Ø.5*(DY(J) + DY(JP))
   2Ø CONTINUE
      IF(TURB.EQ.Ø.Ø) RETURN
      WRITE(LTTI,*) '[ VISC ]: WARNING! Streamwise station ',I
      WRITE(LTTI,*) '          Inner turbulence model reached BL edge.'
      WRITE(LTTI,*) '          Local Reynolds Number is too low.'
      RETURN
C
CCC-- outer eddy viscosity loop
   3Ø JSTART = J
      DO 4Ø J=JSTART, JJ-1
        JP = J+1
        TR2 = H(JP) - EE*U(JP)**2
        T2 = TST*TR2
        R2 = EEC/TR2
        R = Ø.5*(R1 + R2)    ! density     at J+1/2
        T = Ø.5*(T1 + T2)    ! temperature at J+1/2
C
        IF(T.LT.Ø.Ø) GO TO 7ØØ
C
C------ set molecular and outer eddy viscosity
        VIS(J) = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
        VTB(J) = ALPHA*R*MS*DUNORM*SRE*TURB
C
CCC---- set coefficients for molecular viscosity iterates
        DMUDT = Ø.5*VIS(J)*(1.5/T - 1.Ø/(T+TVCON))
C
C------ Uj and Uj+1 influence coefficients
        VUP(J) = -DMUDT*2.Ø*TST*EE*U(JP)
        VUO(J) = -DMUDT*2.Ø*TST*EE*U(J)
C
C------ Hj and Hj+1 influence coefficients
        VHP(J) = DMUDT*TST
        VHO(J) = DMUDT*TST
C
C------ Ue influence coefficient
        VUE(J) = -DMUDT*GM1*UE*(U(JP)**2 + U(J)**2)
C
CCC---- set coefficients for outer eddy viscosity iterates
C
C------ Uj and Uj+1 influence coefficients
        TUP(J) = VTB(J)/R*EE*EEC*U(JP)/TR2**2
        TUO(J) = VTB(J)/R*EE*EEC*U(J)/TR1**2
C
C------ Hj and Hj+1 influence coefficients
        THP(J) = -Ø.5*VTB(J)/R*EE/TR2**2
        THO(J) = -Ø.5*VTB(J)/R*EE/TR1**2
C
C------ Ue influence coefficient
        TUE(J) = -Ø.5*VTB(J)/R*GM1*UE/TST
     &     *(H(JP)-EE*U(JP)**2 + H(J)-EE*U(J)**2)
C
C------ Ms influence coefficient
        TMS(J) = VTB(J)/MS
C
C------ Utau influence coefficient
        TUT(J) = Ø.    ! no wall shear effect on outer eddy viscosity
C
  4Ø1   TR1 = TR2
        T1 = T2
        R1 = R2
   4Ø CONTINUE
C
      RETURN
C
  7ØØ WRITE(LTTI,*) '[ VISC ]: Negative temperature calculated.'
      WRITE(LTTI,*) '          Solution probably diverged.'
      CALL STOPIT
C
  8ØØ WRITE(LTTI,*) '[ VISC ]: Negative dUe/dx correction factor.'
      WRITE(LTTI,*) '          Local Reynolds Number is too low or'
      WRITE(LTTI,*) '          dUe/dx is too high to be corrected for.'
      CALL STOPIT
C
      END
      SUBROUTINE HEADER
      INCLUDE 'BLAKE.INC'
C
      IF(NSTR.EQ.Ø) RETURN
C
      IF(KODE.EQ.1) WRITE(LSTR,1ØØ1)
      IF(KODE.EQ.2) WRITE(LSTR,1ØØ2)
      IF(KODE.EQ.3) WRITE(LSTR,1ØØ3)
      IF(KODE.EQ.4) WRITE(LSTR,1ØØ4)
C
 1ØØ1 FORMAT('1 CODE 1:  Ue prescribed')
 1ØØ2 FORMAT('1 CODE 2:  Mass defect prescribed')
 1ØØ3 FORMAT('1 CODE 3:  Dstar prescribed')
 1ØØ4 FORMAT('1 CODE 4:  Wall shear prescribed')
C
      WRITE(LSTR,1Ø5Ø) REØ
 1Ø5Ø FORMAT('Ø RE =',E12.4)
C
      WRITE(LSTR,2ØØØ)
 2ØØØ FORMAT('Ø Sta',6X,'x',7X,'Ue',6X,'Mach',
     &  6X,'Pe',8X,'m',6X,'Shear',4X,'Dstar',
     &  4X,'Dmom',6X,'H',7X,'Te',6X,'Twall',4X,'Qwall'/
     &  1X,115('-'))
      RETURN
      END


      SUBROUTINE STROUT
      INCLUDE 'BLAKE.INC'
C     ***************************************************
C      This routine outputs X(I) station quantities to
C      unit LSTR. If needed, profile values at X(I) are
C      interpolated from X(I-1/2) and X(I+1/2).
C     ***************************************************
C
      IF(NSTR.EQ.Ø) RETURN
      IF(MOD(I,NSTR).NE.Ø) RETURN
C
C---- set weights for interpolation ... similarity case
      WF = 1.Ø    ! I+1/2 weight
      WB = Ø.     ! I-1/2 weight
      IF(NSIM.EQ.1) GO TO 3
C
      IF(I.LT.IEND) GO TO 2
C----- set weights for extrapolation ... I = IEND case  (unless I < 3)
       IF(I.LT.3) GO TO 3
       WF = (X(I) - X(I-2)) / (X(I-1) - X(I-2))
       WB = (X(I-1) - X(I)) / (X(I-1) - X(I-2))
       UEI = UEIP
       MSI = MSIP
       MDI = MDIP
       DSI = DSIP
       SRI = SRIP
C
      IF(I.EQ.IEND) GO TO 3
C----- set weights for interpolation ... normal case
    2  WF = (X(I) - X(I-1)) / (X(I+1) - X(I-1))
       WB = (X(I+1) - X(I)) / (X(I+1) - X(I-1))
C
    3 TSTI = TSTAG
      RSTI = RSTAG(I)
C
      EEI = Ø.5*GM1*UEI**2/TSTI
      TEI = TSTI*(1.Ø - EEI)
      REI = RSTI*(1.Ø - EEI)**(1.Ø/GM1)
      PEI = REI*TEI/GAM
      MACH = UEI/SQRT(TEI)
C
      SHPI = SHPF*WF + SHPB*WB
      THI = DSI/SHPI
C
      QX = WF*Q(1) + WB*QB(1)
      HX = Ø.5*(WF*(H(1)+H(2)) + WB*(HB(1)+HB(2)))
      TWALL = TSTI*HX
```

```
C
      WRITE(LSTR,1ØØØ) I,X(I),UEI,MACH,PEI,MDI,SRI,
     &  DSI,THI,SHPI,TEI,TWALL,QX
 1ØØØ FORMAT(1X,I4,8F9.4,F8.3,3F9.4)
C
      RETURN
      END


      SUBROUTINE PFLOUT
      INCLUDE 'BLAKE.INC'
C     *********************************
C      This routine outputs profiles
C      at X(I+1/2) to unit LPFL.
C     *********************************
C
      IF(NPFL.EQ.Ø) RETURN
      IF(MOD(I,NPFL).NE.Ø) RETURN
C
      MACH = UE/SQRT(TE)
C
      WRITE(LPFL,1ØØØ)
 1ØØØ FORMAT('1',94('='))
C
      WRITE(LPFL,1Ø1Ø) I,XF,DS,TH,UE,MACH
 1Ø1Ø FORMAT('ØI =',I3,'.5   X =',F9.5,'  Dstar =',F8.4,
     & '  Dmom =',F8.4,'  Ue =',F8.4,'  Mach =',F8.4)
C
      WRITE(LPFL,1Ø11) SC,RHOE,BETU,SHPF
 1Ø11 FORMAT('ØY scale =',F1Ø.6,'   Rhoe =',F8.4,
     & '   BETAu =',F8.4,'   shape parameter =',F6.3)
C
      WRITE(LPFL,1Ø2Ø)
 1Ø2Ø FORMAT('Ø',94('-')/'Ø J',6X,'Eta',8X,'F',9X,'U',9X,'S',
     &  9X,'R',9X,'H',9X,'Q',1ØX,'Mu',7X,'Mut'/
     &  1X,3('-'),3X,7('-'),3X,7('-'),3X,7('-'),3X,7('-'),
     &  3X,7('-'),3X,7('-'),3X,8('-'),3X,7('-'),3X,7('-'))
C
      ETA = Ø.
      DO 1Ø J=1, JJ-1
        JP = J+1
C
C------ calculate values midway between eta grid lines
        FM = Ø.5*(F(JP) + F(J))
        UM = Ø.5*(U(JP) + U(J))
        HM = Ø.5*(H(JP) + H(J))
        RM = Ø.5*(EEC/(H(J) - EE*U(J)**2) + EEC/(H(JP) - EE*U(JP)**2))
C
        WRITE(LPFL,1Ø5Ø) J,ETA,FM,UM,S(J),RM,HM,Q(J),VIS(J),VTB(J)
 1Ø5Ø   FORMAT(1X,I3,6F1Ø.5,F11.6,2F1Ø.5)
C
        ETA = ETA + Ø.5*(DY(JP) + DY(J))
C
   1Ø CONTINUE
      WRITE(LPFL,1Ø7Ø)
 1Ø7Ø FORMAT('Ø',94('-'))
      RETURN
      END




      SUBROUTINE STOPIT
      INCLUDE 'BLAKE.INC'
      CLOSE(UNIT=LSTR)
      CLOSE(UNIT=LPFL)
      STOP
      END
```