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

by

Mark Drela

GTL Report #172 May 1983

GAS TURBINE & PLASMA DYNAMICS LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS

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

by

Mark Drela

GTL Report #172 May 1983

This research carried out in the Gas Turbine and Plasma Dynamics Laboratory, M.I.T. was supported by the NASA Lewis 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.

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 the 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 veloctiy to be carried past separation.

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 pre-

scribed 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, $a_o = \sqrt{\gamma RT_o}$, 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

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

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 result in systems with 5x5 blocks. In contrast, DBS has only 3x3 blocks and was at first an attempt to reduce CPU times. Although for a given number of grid points it does run faster, it also has higher truncation errors. Further investigation revealed that for a given level of accuracy, SBS and DBS require roughly the same CPU time. Since SBS is simpler and requires less coding it is preferred over DBS.

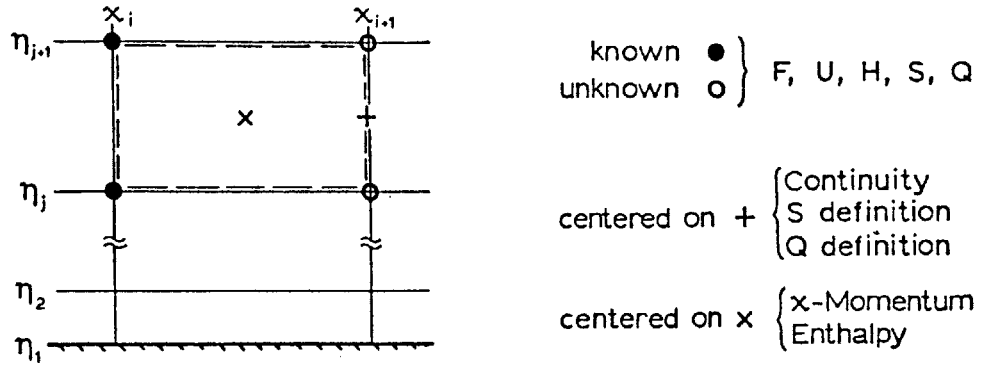

Figure 1. Keller's Box Scheme

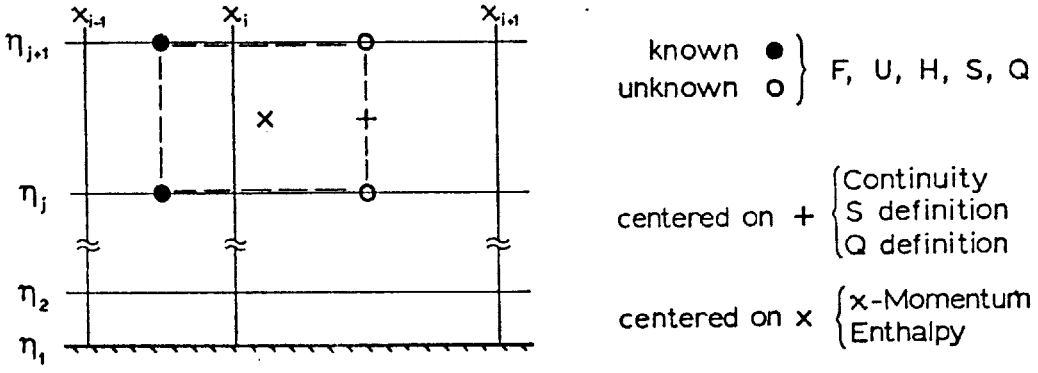

Figure 2. Shifted Box Scheme

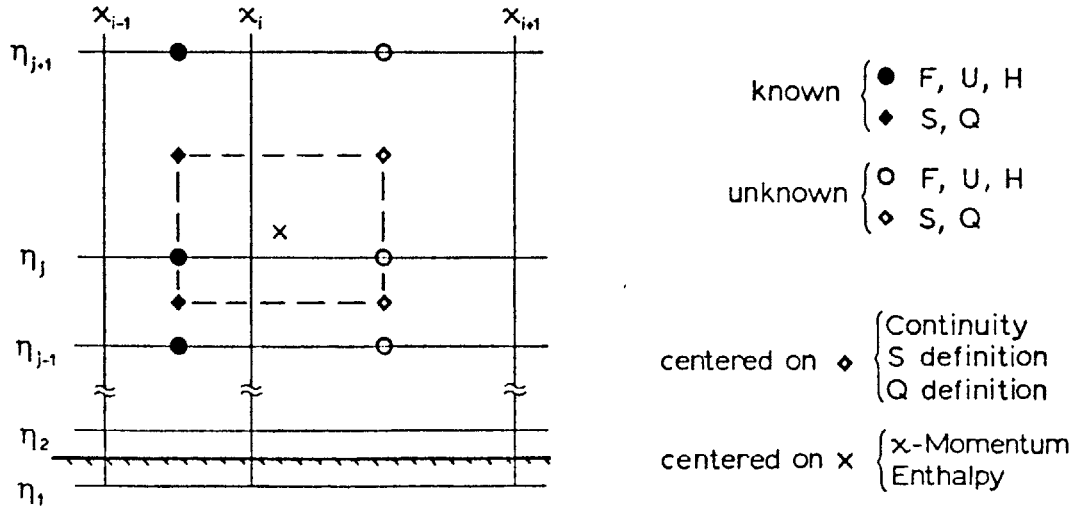

Figure 3. Double-Shifted Box Scheme

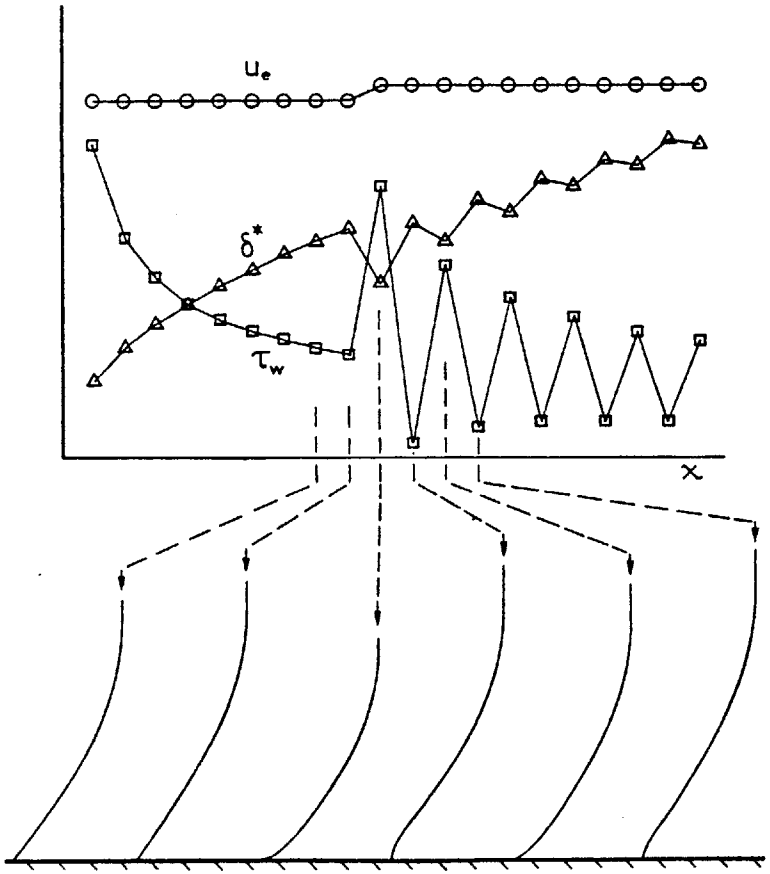

Figure 4. Response of Keller's Box Scheme to 5% edge velocity jump.

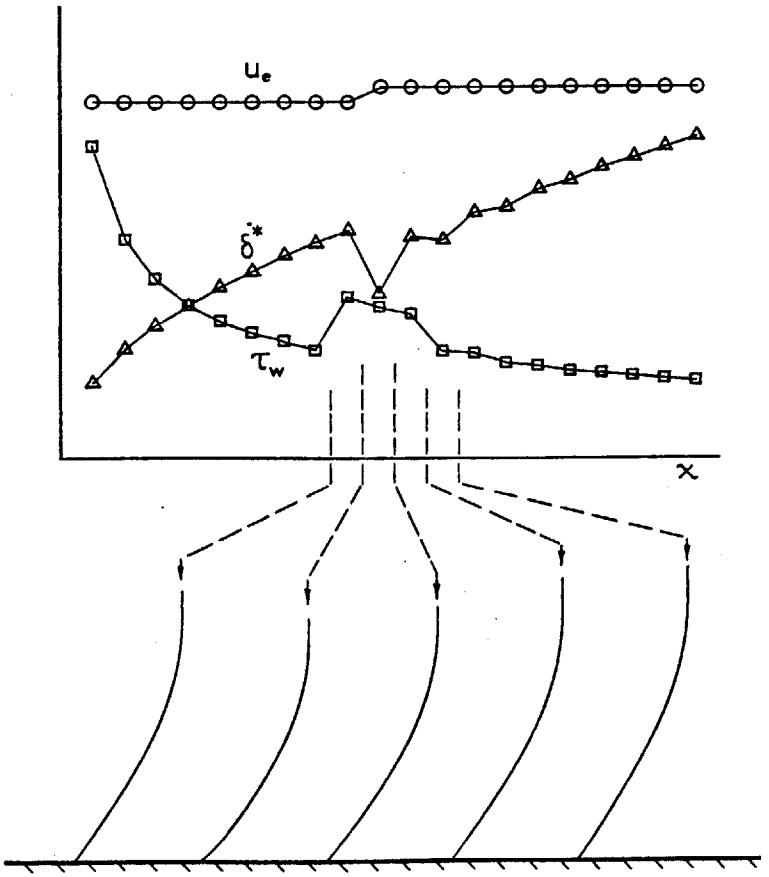

Figure 5. Response of Shifted Box Scheme to 5% edge velocity jump.

SOLUTION PROCEDURE

At each streamwise marching step, there are five unknowns for each η station at streamwise station $x_{i+\frac{1}{2}}$: 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 $u_{ei+\frac{1}{2}}$ and $n_{i+\frac{1}{2}}$, 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 $i+\frac{1}{2}$. The discretized gradient parameters are given by:

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

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.

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

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 δ 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.

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

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

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

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

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

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

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:

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

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 $U\,\partial U/\partial x$ and $U\,\partial H/\partial x$ 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 $U\,\partial U/\partial x$ 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.

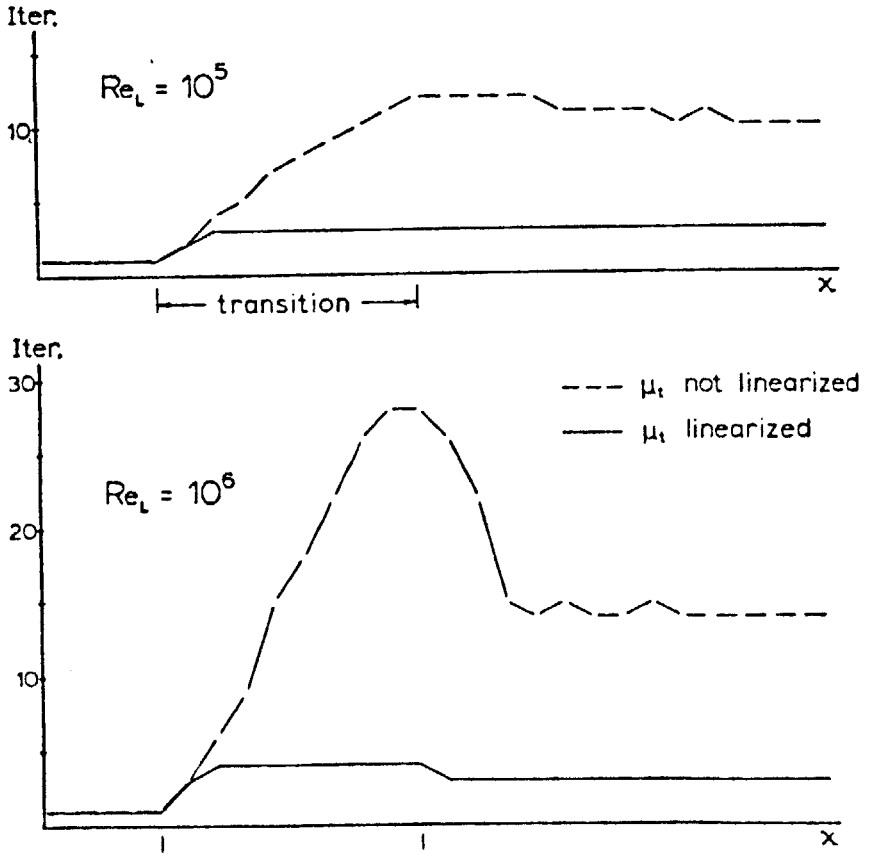

Figure 6. Effect of linearizing eddy viscosity on the number of iterations per streamwise station. Convergence criterion: $\delta U_{MAX} < 10^{-5}$

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.

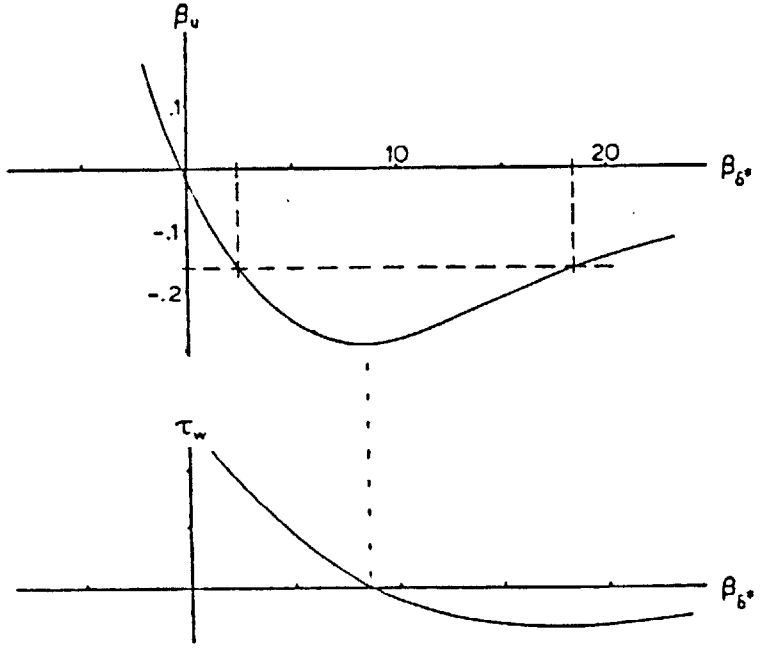

Figure 7a. Gradient parameter and wall shear relations far from separation.

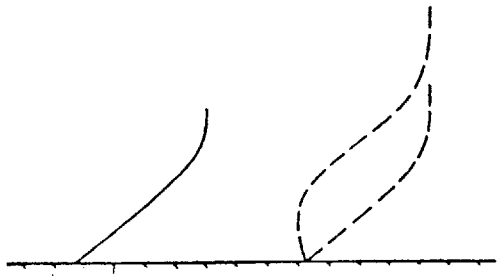

Figure 7b. Two profiles (dashed) corresponding to the same edge velocity. Upstream profile is far from separation.

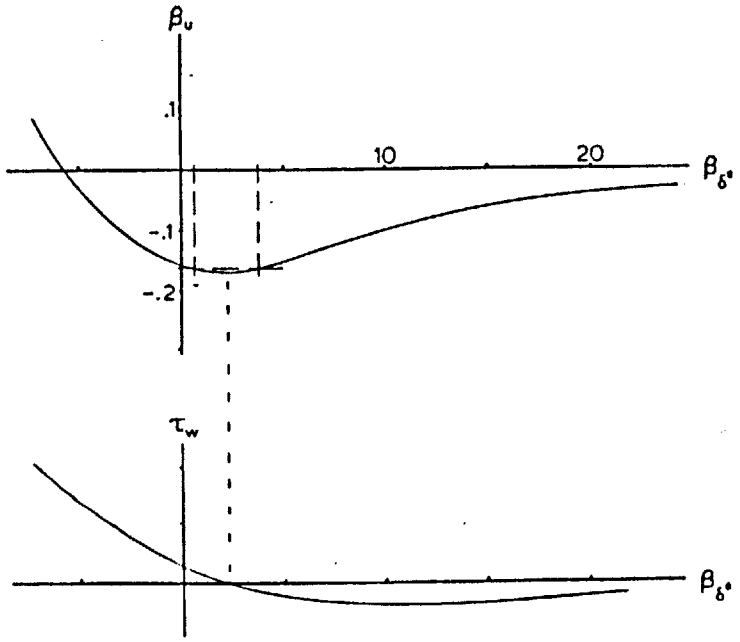

Figure 8a. Gradient parameter and wall shear relations close to separation.

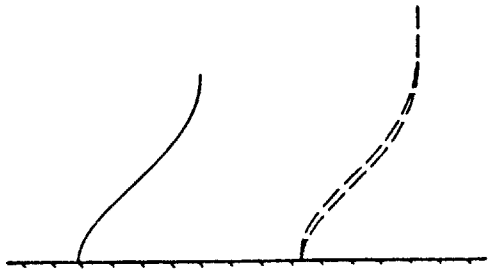

Figure 8b. Two profiles (dashed) corresponding to the same edge velocity. Upstream profile is close to separation.

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 SBS

The following shorthand definitions are used:

1) An overline ($\bar{}$) implies ($)_{i-\frac{1}{2}}$, and lack of one ( ) implies ($)_{i+\frac{1}{2}}$.
2) A tilde ($\tilde{}$) implies ($)_{j+\frac{1}{2}}$.

Example 1: x-Momentum, Equation (16)

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

$$L = \frac{S_{j+1} - S_j}{\eta_{j+1} - \eta_j} + \beta_n \frac{F_{j+1} + F_j}{2} \frac{U_{j+1} - U_j}{\eta_{j+1} - \eta_j} + \beta_u \left(1 - \frac{U_{j+1} + U_j}{2} \frac{F_{j+1} - F_j}{\eta_{j+1} - \eta_j} \right) \tag{A1}$$

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:

$$RHS = \frac{x + \bar{x}}{2} \left(\frac{F_{j+1} + \bar{F}_{j+1} - F_j - \bar{F}_j}{2(\eta_{j+1} - \eta_j)} \frac{U_{j+1} + U_j - \bar{U}_{j+1} - \bar{U}_j}{2(x - \bar{x})} - \frac{F_{j+1} + F_j - \bar{F}_{j+1} - \bar{F}_j}{2(x - \bar{x})} \frac{U_{j+1} + \bar{U}_{j+1} - U_j - \bar{U}_j}{2(\eta_{j+1} - \eta_j)} \right) \tag{A2}$$

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

$$\frac{1}{2}(L + \bar{L}) = RHS \tag{A3}$$

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

$$\delta L - 2\,\delta RHS = 2\,RHS - L - \bar{L} \tag{A4}$$

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

Before equation (A4) 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, δS, and global iterates δu_e, and δn. This is accomplished by straightforward differentiation:

$$\delta L = \delta F_{j+1}\left(\frac{\partial L}{\partial F_{j+1}}\right) + \delta F_j\left(\frac{\partial L}{\partial F_j}\right) + \delta U_{j+1}\left(\frac{\partial L}{\partial U_{j+1}}\right) + \delta U_j\left(\frac{\partial L}{\partial U_j}\right)$$
$$+ \delta S_{j+1}\left(\frac{\partial L}{\partial S_{j+1}}\right) + \delta S_j\left(\frac{\partial L}{\partial S_j}\right) + \delta \beta_u\left(\frac{\partial L}{\partial \beta_u}\right) + \delta \beta_n\left(\frac{\partial L}{\partial \beta_n}\right) \qquad \text{(A5)}$$

The iterate RHS is similarly broken down.

The $\delta\beta$ iterates in equation (A5) 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 β as described in the main text.

Example 2: Shear Definition, Equation (18)

The shear definition is discretized as:

$$\frac{S_{j+1} + S_j}{2} = \frac{\rho_e u_e x}{n^2}\left(\tilde{\mu} + \tilde{\mu}_t\right)\frac{U_{j+1} - U_j}{\eta_{j+1} - \eta_j} \qquad \text{(A6)}$$

Again, iterates δU, δS, $\delta\mu$, and the global iterates δu_e, and δn are introduced. The $\delta\mu$ iterates must further be reduced by differentiation of the formulas for μ and μ_t with respect to the profile and global variables similar to the way δL was reduced above. The formulas for μ and μ_t are given in Appendix B.

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:

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

$\bar{T}_{ref}$ 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

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

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

Inner formula

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 SBS

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

$$\delta F_j = r_{1j} - \delta u_e\, a_{1j} - \delta n\, b_{1j} - \delta U_\tau\, c_{1j} \tag{C1}$$

$$\delta U_j = r_{2j} - \delta u_e\, a_{2j} - \delta n\, b_{2j} - \delta U_\tau\, c_{2j} \tag{C2}$$

$$\delta H_j = r_{3j} - \delta u_e\, a_{3j} - \delta n\, b_{3j} - \delta U_\tau\, c_{3j} \tag{C3}$$

$$\delta S_j = r_{4j} - \delta u_e\, a_{4j} - \delta n\, b_{4j} - \delta U_\tau\, c_{4j} \tag{C4}$$

$$\delta Q_j = r_{5j} - \delta u_e\, a_{5j} - \delta n\, b_{5j} - \delta U_\tau\, c_{5j} \tag{C5}$$

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

$$\text{Relation 1:} \quad \delta u_e\, A_1 + \delta n\, B_1 + \delta U_\tau\, C_1 = D_1 \tag{C6}$$

$$\text{Relation 2:} \quad \delta u_e\, A_2 + \delta n\, B_2 + \delta U_\tau\, C_2 = D_2 \tag{C7}$$

$$\text{Relation 3:} \quad \delta u_e\, A_3 + \delta n\, B_3 + \delta U_\tau\, C_3 = D_3 \tag{C8}$$

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

Relation 1

Equation (24b) restated:

$$1 = \int_0^{\eta_e} U(1 - U)\, d\eta \tag{C9a}$$

Or, in discretized form:

$$1 = \sum_{j=1}^{J-1} \left(\frac{U_{j+1} + U_j}{2}\right)\left(1 - \frac{U_{j+1} + U_j}{2}\right)(\eta_{j+1} - \eta_j) \tag{C9b}$$

Using the shorthand from Appendix A, and introducing iterates:

$$1 = \sum_{j=1}^{J-1} U(1 - \tilde{U})(\eta_{j+1} - \eta_j) + \sum_{j=1}^{J-1} (\delta U_{j+1} + \delta U_j)\left(\frac{1}{2} - \tilde{U}\right)(\eta_{j+1} - \eta_j) \tag{C10}$$

By using equation (C2) to eliminate δU_j and δU_{j+1}, equation (C10) is readily put into the form of equation (C6). The coefficients are then given by:

$$A_1 = \sum_{j=1}^{J-1} (a_{2\,j+1} + a_{2\,j}) \left(\frac{1}{2} - \tilde{U}\right) (\eta_{j+1} - \eta_j) \qquad \text{(C11a)}$$

$$B_1 = \sum_{j=1}^{J-1} (b_{2\,j+1} + b_{2\,j}) \left(\frac{1}{2} - \tilde{U}\right) (\eta_{j+1} - \eta_j) \qquad \text{(C11b)}$$

$$C_1 = \sum_{j=1}^{J-1} (c_{2\,j+1} + c_{2\,j}) \left(\frac{1}{2} - \tilde{U}\right) (\eta_{j+1} - \eta_j) \qquad \text{(C11c)}$$

$$D_1 = \sum_{j=1}^{J-1} (d_{2\,j+1} + d_{2\,j}) \left(\frac{1}{2} - \tilde{U}\right) (\eta_{j+1} - \eta_j)$$

$$- 1 + \sum_{j=1}^{J-1} \tilde{U} (1 - \tilde{U}) (\eta_{j+1} - \eta_j) \qquad \text{(C11d)}$$

Relation 2:

U_τ definition: $\qquad U_\tau = \left(\frac{S_1}{R_1}\right)^{\frac{1}{2}} \qquad$ or $\qquad R_1 U_\tau^2 = S_1 \qquad$ (C12a-b)

Using the fact that U = 0 at the wall, R_1 is given by:

$$R_1 = \frac{1 - u_e^2/2h_e}{H_1} \qquad \text{(C13)}$$

Introducing iterates into equation (C12b) and linearizing (C13):

$$2 R_1 U_\tau \delta U_\tau + U_\tau^2 \delta R_1 - \delta S_1 = S_1 - R_1 U_\tau^2 \qquad \text{(C14)}$$

$$\delta R_1 = -\frac{R_1}{H_1} \delta H_1 - \frac{u_e}{h_e H_1} \delta u_e \qquad \text{(C15)}$$

Using equations (C4) and (C15), equation (C14) can be put in the form of equation (C7). The coefficients are given by:

$$A_2 = a_{4\,1} - U_\tau^2 \left(\frac{R_1}{H_1} a_{3\,1} + \frac{u_e}{h_e H_1} \right) \tag{C16a}$$

$$B_2 = b_{4\,1} - U_\tau^2 \left(\frac{R_1}{H_1} b_{3\,1} \right) \tag{C16b}$$

$$C_2 = c_{4\,1} - U_\tau^2 \left(\frac{R_1}{H_1} c_{3\,1} \right) + 2 R_1 U_\tau \tag{C16c}$$

$$D_2 = r_{4\,1} - U_\tau^2 \left(\frac{R_1}{H_1} r_{3\,1} \right) + S_1 - R_1 U_\tau^2 \tag{C16d}$$

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.

$$u_e + \delta u_e = u_{e_{sp}} \tag{C17}$$

This can be put immediately in the form of equation (C8), with the coefficients given by:

$$A_3 = 1 \qquad B_3 = 0 \qquad C_3 = 0 \qquad D_3 = u_{e_{sp}} - u_e \tag{C18a-d}$$

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

$$\rho_e u_e \delta^* + \delta(\rho_e u_e \delta^*) = m_{sp} \tag{C19}$$

The displacement thickness δ^* is expressed as:

$$\delta^* = \Delta \int_0^{\eta_e} (1 - RU)\, d\eta = \Delta \int_0^{\eta_e} \left(1 - \frac{\partial F}{\partial \eta}\right) d\eta = \Delta\, (\eta_e - F_J) \tag{C20}$$

Using (C20), equation (C19) becomes:

$$\delta n\, (\eta_e - F_J) - n\, \delta F_J = m_{sp} - n\, (\eta_e - F_J) \tag{C21}$$

Using equation (C1) to eliminate the δF iterate, equation (C21) is readily put into the form of equation (C8). The coefficients are:

$$A_3 = n\, a_{1_J} \tag{C22a}$$

$$B_3 = n\, b_{1_J} + \eta_e - F_J \tag{C22b}$$

$$C_3 = n\, c_{1_J} \tag{C22c}$$

$$D_3 = n\, r_{1_J} + m_{sp} - n\,(\eta_e - F_J) \tag{C22d}$$

Example 3: Displacement thickness δ^* specified.

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

From equation (C20)

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

Or, in linearized form:

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

The iterate $\delta(\rho u)_e$ in equation (C24b) 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):

$$\rho_e = \rho_{st}\left(\frac{T_e}{(\gamma-1)\,h_e}\right)^{\frac{1}{\gamma-1}} = \rho_{st}\left(1 - \frac{u_e^2}{2h_e}\right)^{\frac{1}{\gamma-1}} \tag{C25a}$$

$$\delta(\rho u)_e = \rho_e \delta u_e + u_e \delta\rho_e = \rho_e + u_e \frac{\partial \rho_e}{\partial u_e}\,\delta u_e = \rho_e(1 - M_e^2)\,\delta u_e \tag{C25b}$$

Substituting for $\delta(\delta^*)$, and eliminating δF_J, (C23) is put into the form of equation (C8). The coefficients are:

$$A_3 = \Delta\, a_{1_J} - \frac{\Delta}{u_e}(\eta_e - F_J)(1 - M_e^2) \tag{C26a}$$

$$B_3 = \Delta\, b_{1_J} + \frac{\eta_e - F_J}{\rho_e u_e} \tag{C26b}$$

$$C_3 = \Delta\, c_{1_J} \tag{C26c}$$

$$D_3 = \Delta\, r_{1_J} + \delta^*_{sp} - \Delta\,(\eta_e - F_J) \tag{C26d}$$

Example 4: Wall shear τ_w specified.

$$\tau_w + \delta\tau_w = \tau_{w_{sp}} \tag{C27}$$

From equations (13f) and (18): $$\tau_w = \frac{n\, u_e}{x} S_1 \tag{C28}$$

Therefore,

$$\frac{n}{x} S_1\, \delta u_e + \frac{u_e}{x} S_1\, \delta n + \frac{n\, u_e}{x}\, \delta S_1 = \tau_{w_{sp}} - \tau_w \tag{C29}$$

As in previous examples, equation (C27) can be put in the form of equation (C8), with the coefficients given by:

$$A_3 = -\frac{n\, u_e}{x} a_{41} + \frac{n}{x} S_1 \tag{C30a}$$

$$B_3 = -\frac{n\, u_e}{x} b_{41} + \frac{u_e}{x} S_1 \tag{C30b}$$

$$C_3 = -\frac{n\, u_e}{x} c_{41} \tag{C30c}$$

$$D_3 = -\frac{n\, u_e}{x} r_{41} + \tau_{w_{sp}} - \tau_w \tag{C30d}$$

APPENDIX D
PROGRAM LISTING

```
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 /C01/ A(5,5,31),B(5,3,31),C(2,5,31),R(5,4,31)
      COMMON /C05/ F(31), U(31), H(31), S(31), Q(31),
     &             FB(31),UB(31),HB(31),SB(31),QB(31),
     &             MU(31),MUT(31),DETA(31),ETAE,GEO,JJ
      COMMON /C06/ BH,BCON
      COMMON /C07/ XTR1,XTR2,TURB,UTAU,DUNORM
      COMMON /C08/ EPS,ITER,ITMAX,
     &             REO,SRE,PR,PRT,GAM,GM1,TVIS,TVCON
      COMMON /C09/ 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     (delta)
C      MS = mass scale         (n)      = Rhoe*Ue*Sc
C      DS = disp. thickness    (d*)
C      TH = mom. thickness
C      MD = mass defect        (m)      = Rhoe*Ue*Dstar
C      SR = wall shear        (tau)
C
      COMMON /C10/ 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(100),SPEC(100),RSTAG(100),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(31),VHP(31),VUO(31),VHO(31),
     &             TUP(31),THP(31),TUO(31),THO(31),
     &             VUE(31),TUE(31),TMS(31),TUT(31)
      COMMON /C16/ A1,A2,A3,B1,B2,B3,C1,C2,C3,D1,D2,D3
```

```
      PROGRAM BLAKE
      INCLUDE 'BLAKE.INC'
C
C     ************************************************
C     *                                              *
C     *    2-D, Compressible Boundary-Layer Program  *
C     *    Version 5.1                               *
C     *                                              *
C     *    Turbulence Model:                         *
C     *       Cebeci-Smith Two Layer Eddy Viscosity  *
C     *                                              *
C     *    Solution Scheme:                          *
C     *       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    August 1983                 *
C     *    MIT Gas Turbine and Plasma Dynamics Lab   *
C     *                                              *
C     ************************************************
C
      CALL INPUT
C
      IF(NSTR.GT.0) OPEN(UNIT=LSTR,NAME='STREAM.DAT',TYPE='NEW')
      IF(NPFL.GT.0) 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 = 0
      DO 1000 I=2, IEND-1
C
C------ calculate profiles at X(I+1/2)
        CALL INIT
```

```
        CALL PROFL
C
C------ output edge and integral quantities 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
 1000 CONTINUE
C
C---- output edge and integral quantities at last X station
      I = IEND
      CALL STROUT
C
      WRITE(LTTI,*) '[ BLAKE ]: Normal Termination'
C
      CALL STOPIT
C
C     The
      END



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



      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: 20
C      output flags: 0 = 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       RE0         : 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       : 110 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,*) RE0,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(RE0)
      GM1 = GAM - 1.0
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
C
    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 ! INPUT


      SUBROUTINE GRID
      INCLUDE 'BLAKE.INC'
C     ***************************************************
C      This routine calculates the DETA's for a geometric-
C      progression-type normal grid which are then scaled
C      to obtain the specified ETAE.
C     ***************************************************
C
C---- calculate normal grid spacing DETA(J) ... ETA(J+1) = ETA(J) + DETA(J)
      DETA(1) = 1.0
      TEST = 1.0
      DO 3 J=2, JJ-1
        DETA(J) = GEO*DETA(J-1)
        TEST = TEST + DETA(J)
    3 CONTINUE
C
C---- scale DETA(J) to get specified ETAE
      FUDGE = ETAE/TEST
      DO 5 J=1, JJ-1
        DETA(J) = FUDGE*DETA(J)
    5 CONTINUE
      RETURN
      END ! GRID


      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 = 0.5*(1.0 + BETU)      ! mass scale       "          "
      BETH = 0.                    ! total enthalpy   "          "
C
C---- these relationships must hold if there is similarity
      BETM = 0.5*(1.0 + BETU)      ! mass defect      "          "
      BETD = 0.5*(1.0 - BETU)      ! disp. thickness  "          "
      BETS = 0.5*(3.0*BETU - 1.0) ! wall shear       "          "
C
C---- there is no upstream station for similarity, so...
      BETUB = 0.
      BETNB = 0.
      BETHB = 0.
C
      TURB = 0.  ! no turbulence
C
      XF = 0.5*(X(1) + X(2))  ! similarity x position
C
      TST = TSTAG                        ! similarity
      RST = 0.5*(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 = 1.0 - BETU
      DFS = 0.64791 + BM1*(.200 + BM1*(.22973 + .6431*BM1**3))
      TFS = 0.29234 + BM1*(.125 + BM1*(.06660 + .1802*BM1**3))
      SFS = 1.23259 - BM1*(.560 + BM1*(.18213 + .1584*BM1**3))
      SHPF = DFS/TFS   ! shape parameter
C
C---- Similarity solutions with BETU=0 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.0.0 .AND. UGUESS.EQ.0.0) GO TO 500
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.0.0) UE = UGUESS
      IF(KODE.EQ.2 .AND. BETU.GT.0.0) UE = (MD/DFS)**2/XF
      IF(KODE.EQ.3 .AND. BETU.GT.0.0) UE = (DFS/DS)**2*XF
      IF(KODE.EQ.4 .AND. BETU.GT.0.0) UE = (XF*(SR/SFS)**2)**(1./3.)
C
C---- initialize MS for iteration
      EE = 0.5*GM1*UE**2/TST        ! edge kinetic energy/total enthalpy ratio
      EEC = 1.0 - EE
      RHOE = RST * EEC**(1.0/GM1)   ! edge density
      MS = TFS*SQRT(RHOE*UE*XF)     ! first guess for mass scale
C
C---- set initial profiles ... simple polynomials are used
      RNU = RHOE*UE*XF/MS**2
      Z = 0.
      DO 10 J=1, JJ       ! march up from the wall
        FB(J) = 0.
        UB(J) = 0.
        HB(J) = 0.
        SB(J) = 0.
        QB(J) = 0.
C
        H(J) = 1.0
        U(J) = Z*(2.0-Z)
        IF(Z.GT.1.0) U(J) = 1.0
C
        R2 = EEC/(H(J) - EE*U(J)**2)    ! density at ETAj
        IF(J.EQ.1) F(J) = 0.
        IF(J.GT.1) F(J) = F(J-1) + 0.5*DETA(J)*(R2*U(J) + R1*U(J-1))
C
        S(J) = RNU*2.0*(1.0 - Z)/7.5
        IF(Z.GT.1) S(J) = 0.0
        Q(J) = 0.0
C
        Z = Z + DETA(J)/7.5
        R1 = R2
   10 CONTINUE
C
C---- initialize everything else for iteration
      CALL ECALC  ! edge quantities
      CALL DCALC  ! Dstar, Dmom, and other thicknesses
```

```
      CALL VISC    ! viscosity
C
      DO 50 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 = 0.0
        DO 55 J=1, JJ
          F(J) = F(J) + R(1,1,J)
          U(J) = U(J) + R(2,1,J)
          H(J) = H(J) + R(3,1,J)
          S(J) = S(J) + R(4,1,J)
          Q(J) = Q(J) + R(5,1,J)
          DUMAX = AMAX1(DUMAX,ABS(R(2,1,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.0.0) GO TO 600
        IF(MS.LE.0.0) GO TO 700
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------ 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
      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
C
  500 WRITE(LTTI,*) '[ SIMIL ]: UGUESS must be given for inverse'
      WRITE(LTTI,*) '           flat plate similarity solution.'
      CALL STOPIT
C
  600 WRITE(LTTI,*) '[ SIMIL ]: Negative edge velocity was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged.'
      CALL STOPIT
C
  700 WRITE(LTTI,*) '[ SIMIL ]: Negative mass scale was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged.'
      CALL STOPIT
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 ! SIMIL



      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 = 0.
        DO 52 J=1, JJ
          F(J) = F(J) + R(1,1,J)
          U(J) = U(J) + R(2,1,J)
          H(J) = H(J) + R(3,1,J)
          S(J) = S(J) + R(4,1,J)
          Q(J) = Q(J) + R(5,1,J)
          DUMAX = AMAX1(DUMAX,ABS(R(2,1,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.0.0) GO TO 10
        IF(MS.LE.0.0) 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------ check for convergence or lack thereof
        DGLBL = ABS(DMS)/MS + ABS(DUE)/UE
        IF(DUMAX.LE.EPS .AND. DGLBL.LE.EPS) GO TO 20
C
    5 CONTINUE    ! end of Newton iteration loop
C
C
      WRITE(LTTI,*) '[ PROFL ]: CONVERGENCE FAILED at station ',I,'.5'
      WRITE(LTTI,*) '           Max U velocity residual:         ',DUMAX
      WRITE(LTTI,*) '           Uedge + Mass residuals :         ',DGLBL
      CALL STOPIT
C
   10 WRITE(LTTI,*) '[ PROFL ]: Negative edge velocity was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged.'
      CALL STOPIT
C
   11 WRITE(LTTI,*) '[ PROFL ]: Negative mass scale was calculated.'
      WRITE(LTTI,*) '           Solution probably diverged.'
      CALL STOPIT
C
   20 WRITE(LTTI,*) '[ PROFL ]: Station ',I,'.5 ...',ITER,' Iterations'
      RETURN
C
      END ! PROFL



      SUBROUTINE ECALC
      INCLUDE 'BLAKE.INC'
C     ****************************
C      This routine calculates edge
C      quantities at X(I+1/2).
C     ****************************
C
      EE = 0.5*GM1*UE**2/TST     ! edge Kinetic Energy to enthalpy ratio...
      EEC = 1.0 - EE             ! ...its complement
      TE = TST*EEC               ! edge static temperature
      RHOE = RST*EEC**(1.0/GM1)  ! edge static density
      PE = RHOE*TE/GAM           ! edge static pressure
```

```
      ME2 = UE*UE/TE              ! edge Mach Number squared...
      ME2C = 1.0 - ME2            ! ...its complement
      RNU = XF*RHOE*UE/MS**2      ! group in front of S and Q definitions
C
      RETURN
      END ! ECALC



      SUBROUTINE DCALC
      INCLUDE 'BLAKE.INC'
C     ****************************************
C      This routine calculates the profile
C      parameters DS, TH etc. at I+1/2
C     ****************************************
C
      DUNORM = 0.    ! normalized velocity thickness for outer eddy viscosity
      THNORM = 0.    ! normalized momentum thickness
      DO 10 J=1, JJ-1
        THNORM = THNORM + (F(J+1)-F(J)) * (1.0 - 0.5*(U(J+1)+U(J)))
        DUNORM = DUNORM + (1.0 - 0.5*(U(J+1)+U(J)))*DETA(J)
   10 CONTINUE
C
      DSNORM = ETAE - F(JJ)       ! 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
      SR = MS*UE/XF*S(1) ! wall shear
C
      RETURN
      END ! DCALC



      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**** 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**** 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 = 0.5*(RSTAG(I) + RSTAG(I+1))
      TST = TSTAG
C
C**** finally, set or initialize stuff at I+1/2 for iteration ***
C
      XF = 0.5*(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 = 0.5*(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 = 0.       ! since edge flow is adiabatic
C
C---- set turbulence weighting coefficient with cubic transition zone
      XT = 0.5
      IF(XTR1.NE.XTR2) XT = (2.0*XF - (XTR2+XTR1))/(XTR2-XTR1)
      TURB = 0.5 + 0.25*(3.0*XT - XT**3)
      IF(XF.LT.XTR1) TURB = 0.
      IF(XF.GE.XTR2) TURB = 1.0
C
C---- calculate edge quantities and viscosity
      CALL ECALC
      CALL DCALC
      CALL VISC
C
      RETURN
      END ! INIT



      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  ! power curve extrapolation
      MDIP = MDI*(X(I+1)/X(I))**BETM  ! for all these quantities
      DSIP = DSI*(X(I+1)/X(I))**BETD  ! .................................
      SRIP = 2.0*SR - SRI             ! linear extrapolation for wall shear
C
      RETURN
      END ! IPSET



      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      Viscosities are defined to be halfway between grid nodes:
C         MU(J) is at (ETA(J+1)+ETA(J))/2,   ditto for MUT(J)
C     ****************************************************************
C
C---- empirical turbulence constants
      DATA     VKAP,   DAMPC,   ALPHA,   PPC
     &      /  0.40,   26.0,    0.0168,  11.8  /
C
C---- zero out influence coefficients
      DO 10 J=1, JJ
        VUP(J) = 0.
        VHP(J) = 0.
        VUO(J) = 0.
        VHO(J) = 0.
        TUP(J) = 0.
        THP(J) = 0.
        TUO(J) = 0.
        THO(J) = 0.
        VUE(J) = 0.
        TUE(J) = 0.
        TMS(J) = 0.
        TUT(J) = 0.
   10 CONTINUE
C
C---- set wall shear velocity UTAU
      T = 0.5*TST*(H(1) + H(2))
      V1 = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
      SWALL = RNU*V1*(U(2)-U(1))/DETA(1)
      RWALL = 2.0*EEC/(H(1) + H(2))
      UTAU = SQRT(ABS(SWALL)/RWALL)
      IF(UTAU.LT.1.E-04) UTAU = 1.E-04     ! zero UTAU is a no-no
C
C---- assorted shorthand
      ECONST = SQRT(SRE*MS**3/(RHOE*UE*XF))
      BCONST = ECONST*UTAU/DAMPC
      DBDU = 0.
      IF(NSIM.EQ.0) DBDU = 1.0/(UE*FLOG)          ! dBETU/due
C
C---- set pressure gradient correction factor PN
      PTEMP = MU(1)/(ECONST*UTAU**3*RWALL**2)
      PPLUS = BETU*PTEMP
      PN2 = 1.0 - PPC*PPLUS
      IF(TURB.GT.0.0 .AND. PN2.LE.0.0) GO TO 800   ! test if correction factor
C                                                  ! is imaginary (!)
      PN = SQRT(ABS(PN2))
```

```
C
      TR1 = H(1) - EE*U(1)**2
C
      RU1 = 2.0*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 = 0.5*DETA(1)
C
CCC-- inner eddy viscosity loop
      DO 20 J=1, JJ-1
        JP = J+1
        TR2 = H(JP) - EE*U(JP)**2
C
        RU2 = 2.0*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 = 0.5*(T1 + T2)    ! temperature at J+1/2
        RHO = 0.5*(R1 + R2)    ! density     at J+1/2
C
C------ test if temperature is negative
        IF(T.LT.0.0) GO TO 700
C
C------ set molecular viscosity with Sutherland's formula
        MU(J) = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
        MUT(J) = 0.
C
CCC---- set coefficients for molecular viscosity iterates (dmu)j
C
C       dmu = (dmu/dU)dU + (dmu/dH)dH + ... etc
C
        DMUDT = 0.5*MU(J)*(1.5/T - 1.0/(T+TVCON))    ! dmu/dT
C
C------ Uj and Uj+1 influence coefficients
        VUP(J) = -DMUDT*2.0*TST*EE*U(JP)              ! dmu/dUj+1
        VUO(J) = -DMUDT*2.0*TST*EE*U(J)               ! dmu/dUj
C
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 = 0
        IF(TURB.EQ.0.0) GO TO 205
```

```
C
        US = ABS(U(JP) - U(J))
        SGN = 1.0
        IF(U(JP).LT.U(J)) SGN = -1.0
        BK = BCONST*PN*RHO/MU(J)
        EK = 0.
        IF(ETA*BK .LT. 30.0) EK = EXP(-ETA*BK)
        YL = VKAP*ETA*(1.0 - EK)
        VTP = RHO*YL*YL*MS*SRE/DETA(J)*TURB
C
C------ set inner eddy viscosity
        MUT(J) = VTP*US
C
C------ calculate outer eddy viscosity
        MUTOUT = ALPHA*RHO*MS*DUNORM*SRE*TURB
C
C------ go to outer viscosity loop if inner-outer match point has been reached
        IF(MUT(J).GT.MUTOUT) GO TO 30
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 = 0.
        IF(J.GT.1) DK = 2.0*CK*BK/YL
C
C------ Uj and Uj+1 influence coefficients
        TUP(J) =  SGN*VTP + 0.5*MUT(J)*RU2/RHO + MUT(J)*DK          ! dmut/dUj+1
     &   *(-VUP(J)/MU(J) + 0.5*RU2/RHO
     &     - 0.5*PPC*PPLUS/PN2*(VUP(J)/MU(1) - 0.5*RU2/RWALL))
        TUO(J) = -SGN*VTP + 0.5*MUT(J)*RU2/RHO + MUT(J)*DK          ! dmut/dUj+1
     &   *(-VUO(J)/MU(J) + 0.5*RU1/RHO
     &     - 0.5*PPC*PPLUS/PN2*(VUO(J)/MU(1) - 0.5*RU1/RWALL))
C
C------ Hj and Hj+1 influence coefficients
        THP(J) = 0.5*MUT(J)*RH2/RHO + MUT(J)*DK                     ! dmut/dHj+1
     &   *(-VHP(J)/MU(J) + 0.5*RH2/RHO
     &     - 0.5*PPC*PPLUS/PN2*(VHP(J)/MU(1) - 0.5*RH2/RWALL))
        THO(J) = 0.5*MUT(J)*RH1/RHO + MUT(J)*DK                     ! dmut/dHj
     &   *(-VHO(J)/MU(J) + 0.5*RH1/RHO
     &     - 0.5*PPC*PPLUS/PN2*(VHO(J)/MU(1) - 0.5*RH1/RWALL))
C
C------ Ue influence coefficient
        TUE(J) = MUT(J)*0.5*(RUE2+RUE1)/RHO + MUT(J)*DK             ! dmut/due
     &    *(-VUE(J)/MU(J) + 0.5*(RUE2+RUE1)/RHO - 0.5*EEC/UE
     &      - 0.5*PPC/PN2
     &       *(PTEMP*DBDU + 0.5*PPLUS*EEC/UE + PPLUS*VUE(1)/MU(1)
     &         + PPLUS/RWALL*2.0*GM1*UE/(TST*(H(1)+H(2))) ))
C
C------ Ms influence coefficient
```

```
        TMS(J) = MUT(J)/MS + MUT(J)*DK*(1.5 + 0.75*PPC*PPLUS/PN2)/MS ! dmut/dms
C
C------ Utau influence coefficient
        TUT(J) = MUT(J)*DK*(1.0 - 1.5*PPC*PPLUS/PN2)/UTAU       ! dmut/dUtau
C
  205   TR1 = TR2
        RU1 = RU2
        RH1 = RH2
        RUE1 = RUE2
        T1 = T2
        R1 = R2
        ETA = ETA + 0.5*(DETA(J)+DETA(JP))
   20 CONTINUE
      IF(TURB.EQ.0.0) RETURN
C
      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
   30 JSTART = J
      DO 40 J=JSTART, JJ-1
        JP = J+1
        TR2 = H(JP) - EE*U(JP)**2
        T2 = TST*TR2
        R2 = EEC/TR2
        RHO = 0.5*(R1 + R2) ! density     at J+1/2
        T = 0.5*(T1 + T2)   ! temperature at J+1/2
C
        IF(T.LT.0.0) GO TO 700
C
C------ set molecular and outer eddy viscosity
        MU(J) = SQRT((T/TVIS)**3)*(TVIS+TVCON)/(T+TVCON)
        MUT(J) = ALPHA*RHO*MS*DUNORM*SRE*TURB
C
CCC---- set coefficients for molecular viscosity iterates
        DMUDT = 0.5*MU(J)*(1.5/T - 1.0/(T+TVCON))
C
C------ Uj and Uj+1 influence coefficients
        VUP(J) = -DMUDT*2.0*TST*EE*U(JP)
        VUO(J) = -DMUDT*2.0*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) = MUT(J)/RHO*EE*EEC*U(JP)/TR2**2
        TUO(J) = MUT(J)/RHO*EE*EEC*U(J)/TR1**2
C
C------ Hj and Hj+1 influence coefficients
        THP(J) = -0.5*MUT(J)/RHO*EE/TR2**2
        THO(J) = -0.5*MUT(J)/RHO*EE/TR1**2
C
C------ Ue influence coefficient
        TUE(J) = -0.5*MUT(J)/RHO*GM1*UE/TST
     &      *(H(JP)-EE*U(JP)**2 + H(J)-EE*U(J)**2)
C
C------ Ms influence coefficient
        TMS(J) = MUT(J)/MS
C
C------ Utau influence coefficient
        TUT(J) = 0.    ! no wall shear effect on outer eddy viscosity
C
  401   TR1 = TR2
        T1 = T2
        R1 = R2
   40 CONTINUE
C
      RETURN
C
  700 WRITE(LTTI,*) '[ VISC ]: Negative temperature calculated.'
      WRITE(LTTI,*) '          Solution probably diverged.'
      CALL STOPIT
C
  800 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 ! VISC



      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=0).
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 = 0.                    ! XBAR multiplies the
      IF(NSIM.EQ.0) XBAR = 0.5*(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 = 0.                              ! for similarity, dBETU
      DBDN = 0.                              ! and dBETN are zero
      IF(NSIM.EQ.0) DBDU = 1.0/(UE*FLOG)
      IF(NSIM.EQ.0) DBDN = 1.0/(MS*FLOG)
C
      DO 2 J=1, JJ
        DO 21 N=1, 5
          DO 211 L=1, 5
            A(L,N,J) = 0.
            IF(N.LE.3) B(L,N,J) = 0.
            IF(L.LE.2) C(L,N,J) = 0.
            IF(N.LE.4) R(L,N,J) = 0.
  211     CONTINUE
   21   CONTINUE
    2 CONTINUE
C
CCC-- set first A and C blocks and righthand sides
C
C---- first line:  F = 0  wall boundary condition
      A(1,1,1) = 1.0
      R(1,1,1) = -F(1)
C
C---- second line: U = 0  boundary condition
      A(2,2,1) = 1.0
      R(2,1,1) = -U(1)
C
C---- third line:  bh H + (1-bh) Q = bcon   boundary condition
      A(3,3,1) = BH
      A(3,5,1) = 1.0 - BH
      R(3,1,1) = BCON + (BH-1.0)*Q(1) - BH*H(1)
C
      DO 1000 J=1, JJ-1
C
C---- set shorthand definitions
C
      JP = J+1
C
      FS = F(JP) + F(J)
      US = U(JP) + U(J)
      HS = H(JP) + H(J)
C
      FD = F(JP) - F(J)
      UD = U(JP) - U(J)
      HD = H(JP) - H(J)
```

```
C
      FY = FD + FB(JP) - FB(J)
      FX = FS - FB(JP) - FB(J)
C
      UY = UD + UB(JP) - UB(J)
      UX = US - UB(JP) - UB(J)
C
      HY = HD + HB(JP) - HB(J)
      HX = HS - HB(JP) - HB(J)
C
C
      SEP = 1.0                ! separation trigger
      IF(US .LE. 0.) SEP = 0. ! for Reyhner-Flugge-Lotz approximation
C
C---- x-momentum
      A(4,1,J) = 0.5*( BETU*US + BETN*UD + XBAR*(UX*SEP + UY))
      A(4,2,J) = 0.5*(-BETU*FD - BETN*FS - XBAR*(FY*SEP + FX))
      A(4,4,J) = -1.0
C
      C(1,1,J) = 0.5*(-BETU*US + BETN*UD + XBAR*(-UX*SEP + UY))
      C(1,2,J) = 0.5*(-BETU*FD + BETN*FS + XBAR*(-FY*SEP + FX))
      C(1,4,J) = 1.0
C
      R(4,1,J) = 0.5*XBAR*(FY*UX*SEP - FX*UY)
     &  - (SB(JP)-SB(J)
     &     + BETUB*(DETA(J) - 0.5*(UB(JP)+UB(J))*(FB(JP)-FB(J)))
     &     + BETNB*0.5*(FB(JP)+FB(J))*(UB(JP)-UB(J)))
     &  - (S(JP)-S(J)
     &     + BETU*(DETA(J) - 0.5*US*FD)
     &     + BETN*0.5*FS*UD)
C
      R(4,2,J) = (DETA(J) - 0.5*US*FD)*DBDU
      R(4,3,J) = 0.5*FS*UD*DBDN
C
C---- energy
      A(5,1,J) = 0.5*( BETH*HS + BETN*HD + XBAR*(HX*SEP + HY))
      A(5,3,J) = 0.5*(-BETH*FD - BETN*FS - XBAR*(FY*SEP + FX))
      A(5,5,J) = -1.0
C
      C(2,1,J) = 0.5*(-BETH*HS + BETN*HD + XBAR*(-HX*SEP + HY))
      C(2,3,J) = 0.5*(-BETH*FD + BETN*FS + XBAR*(-FY*SEP + FX))
      C(2,5,J) = 1.0
C
      R(5,1,J) = 0.5*XBAR*(FY*HX*SEP - FX*HY)
     &  - (QB(JP)-QB(J)
     &     - BETHB*0.5*(HB(JP)+HB(J))*(FB(JP)-FB(J))
     &     + BETNB*0.5*(FB(JP)+FB(J))*(HB(JP)-HB(J)))
     &  - (Q(JP)-Q(J)
     &     - BETH*0.5*HS*FD
     &     + BETN*0.5*FS*HD)
      R(5,3,J) = 0.5*FS*HD*DBDN
```

```
C
CCC-- continuity ---
C
      TR1 = H(J)  - EE*U(J)**2
      TR2 = H(JP) - EE*U(JP)**2
      R1 = EEC/TR1
      R2 = EEC/TR2
C
      B(1,1,JP) = -1.0
      B(1,2,JP) = -0.5*DETA(J)*R1 - DETA(J)*EE*EEC*(U(J)/TR1)**2
      B(1,3,JP) =  0.5*DETA(J)*U(J)/TR1**2*EEC
C
      A(1,1,JP) = 1.0
      A(1,2,JP) = -0.5*DETA(J)*R2 - DETA(J)*EE*EEC*(U(JP)/TR2)**2
      A(1,3,JP) =  0.5*DETA(J)*U(JP)/TR2**2*EEC
C
      R(1,1,JP) = -F(JP) + F(J) + 0.5*DETA(J)*(R2*U(JP) + R1*U(J))
      R(1,2,JP) = 0.5*DETA(J)*GM1*UE/TST
     & *(U(J)*(H(J)-U(J)**2)/TR1**2 + U(JP)*(H(JP)-U(JP)**2)/TR2**2)
C
C---- shear
      ST = (MU(J)+MUT(J))*UD/DETA(J)
      DSDV = 2.0*RNU*UD/DETA(J)
C
      B(2,1,JP) = A(4,1,J)
      B(2,2,JP) = -DSDV*(VUO(J)+TUO(J)) + 2.*RNU*(MU(J)+MUT(J))/DETA(J)
     &          + A(4,2,J)
      B(2,3,JP) = -DSDV*(VHO(J)+THO(J))
C
      A(2,1,JP) = C(1,1,J)
      A(2,2,JP) = -DSDV*(VUP(J)+TUP(J)) - 2.*RNU*(MU(J)+MUT(J))/DETA(J)
     &          + C(1,2,J)
      A(2,3,JP) = -DSDV*(VHP(J)*THP(J))
      A(2,4,JP) = 2.0
C
      R(2,1,JP) = 2.0*RNU*ST - (S(JP) + S(J)) + R(4,1,J)
      R(2,2,JP) = -DSDV*(VUE(J)+TUE(J)) - 2.0*ST*ME2C*RNU/UE + R(4,2,J)
      R(2,3,JP) = -DSDV*TMS(J) + 4.0*ST*RNU/MS + R(4,3,J)
      R(2,4,JP) = -DSDV*TUT(J) + R(4,4,J)
C
C---- heat flow
      TPR = MU(J)/PR + MUT(J)/PRT
      VPR = MU(J)*(1.0 - 1.0/PR)
      QT = (TPR*HD + VPR*EE*US*UD)/DETA(J)
      DQV = 2.0*RNU*(HD/PR + (1.0-1.0/PR)*EE*US*UD)/DETA(J)
      DQT = 2.0*RNU*HD/PRT/DETA(J)
C
      B(3,1,JP) = A(5,1,J)
      B(3,2,JP) =-DQV*VUO(J) - DQT*TUO(J) + 4.*RNU*VPR*EE*U(J)/DETA(J)
      B(3,3,JP) =-DQV*VHO(J) - DQT*THO(J) + 2.*RNU*TPR/DETA(J)
     &          + A(5,3,J)
```

```
C
      A(3,1,JP) = C(2,1,J)
      A(3,2,JP) =-DQV*VUP(J) - DQT*TUP(J) - 4.*RNU*VPR*EE*U(JP)/DETA(J)
      A(3,3,JP) =-DQV*VHP(J) - DQT*THP(J) - 2.*RNU*TPR/DETA(J)
     &          + C(2,3,J)
      A(3,5,JP) = 2.0
C
      R(3,1,JP) = 2.0*RNU*QT - (Q(JP) + Q(J)) + R(5,1,J)
      R(3,2,JP) = -DQV*VUE(J) - DQT*TUE(J)
     &          - 2.0*QT*ME2C*RNU/UE - 4.0*RNU*VPR*GM1*UE/TST*US*UD
     &          + R(5,2,J)
      R(3,3,JP) = -DQT*TMS(J) + 4.0*QT*RNU/MS + R(5,3,J)
      R(3,4,JP) = -DQT*TUT(J) + R(5,4,J)
C
 1000 CONTINUE   ! end of J loop
C
CCC-- fill last A and B blocks and righthand side vectors
C
C---- fourth line:  U = 1  edge boundary condition
      A(4,2,JJ) = 1.0
      R(4,1,JJ) = 1.0 - U(JJ)
C
C---- fifth line:  H = 1  edge boundary condition
      A(5,3,JJ) = 1.0
      R(5,1,JJ) = 1.0 - H(JJ)
C
      RETURN
      END ! SETUP



      SUBROUTINE SOLVE
      INCLUDE 'BLAKE.INC'
C     ***************************************************************************
C      This routine solves the block-tridiagonal system for the Newton-Raphson
C      Deltas and global iterate influence coefficients.  These are returned
C      in the Rj vectors. A small amount of pre-processing was done in SETUP
C      to obtain zero rows and columns in the C and B blocks:
C
C      |A C       ||d|   |R..R|          |0 0 0 0 0|          |* * * 0 0|
C      |B A C     ||d|   |R..R|          |0 0 0 0 0|          |* * * 0 0|
C      |  B . .   ||.| = |R..R|      C = |0 0 0 0 0|      B = |* * * 0 0|
C      |    . . C ||.|   |R..R|          |* * * * *|          |* * * 0 0|
C      |      B A ||d|   |R..R|          |* * * * *|          |* * * 0 0|
C
C     ***************************************************************************
C
CCC** Backward sweep: Elimination of upper block diagonal (C's).
      DO 1 J=JJ, 1, -1
        IF(J.EQ.JJ) GO TO 13
```

```
C
CCC---- calculate diagonal Aj block and modified righthand side (Rj's)
C       resulting from elimination of Cj block.
C
        JP = J+1
        DO 11 K=1, 2
          DO 111 L=1, 3
            A(K+3,L,J) = A(K+3,L,J) - (C(K,1,J)*B(1,L,JP)
     &       + C(K,2,J)*B(2,L,JP) + C(K,3,J)*B(3,L,JP)
     &       + C(K,4,J)*B(4,L,JP) + C(K,5,J)*B(5,L,JP))
  111     CONTINUE
          DO 112 L=1, 4
            R(K+3,L,J) = R(K+3,L,J) - (C(K,1,J)*R(1,L,JP)
     &       + C(K,2,J)*R(2,L,JP) + C(K,3,J)*R(3,L,JP)
     &       + C(K,4,J)*R(4,L,JP) + C(K,5,J)*R(5,L,JP))
  112     CONTINUE
   11   CONTINUE
C                                                                    -1
CCC---- multiply Bj block and righthand side Rj vectors by (Aj)
C       using Gaussian elimination.
C
   13   DO 14 NP=1, 4
          NP1 = NP+1
C
C-------- find max pivot index NX
          NX = NP
          DO 131 N=NP1, 5
            IF(ABS(A(N,NP,J))-ABS(A(NX,NP,J))) 131,131,1311
 1311        NX = N
  131     CONTINUE
C
          PIVOT = 1.0/A(NX,NP,J)
C
C-------- switch pivots
          A(NX,NP,J) = A(NP,NP,J)
C
C-------- switch rows & normalize pivot row
          DO 132 L=NP1, 5
            TEMP = A(NX,L,J)*PIVOT
            A(NX,L,J) = A(NP,L,J)
            A(NP,L,J) = TEMP
  132     CONTINUE
C
          DO 133 L=1, 3
            TEMP = B(NX,L,J)*PIVOT
            B(NX,L,J) = B(NP,L,J)
            B(NP,L,J) = TEMP
  133     CONTINUE
C
          DO 134 L=1, 4
            TEMP = R(NX,L,J)*PIVOT
```

```
            R(NX,L,J) = R(NP,L,J)
            R(NP,L,J) = TEMP
  134     CONTINUE
C
C-------- forward eliminate everything
          DO 135 K=NP1, 5
            DO 1351 L=NP1, 5
              A(K,L,J) = A(K,L,J) - A(K,NP,J)*A(NP,L,J)
 1351       CONTINUE
            B(K,1,J) = B(K,1,J) - A(K,NP,J)*B(NP,1,J)
            B(K,2,J) = B(K,2,J) - A(K,NP,J)*B(NP,2,J)
            B(K,3,J) = B(K,3,J) - A(K,NP,J)*B(NP,3,J)
            R(K,1,J) = R(K,1,J) - A(K,NP,J)*R(NP,1,J)
            R(K,2,J) = R(K,2,J) - A(K,NP,J)*R(NP,2,J)
            R(K,3,J) = R(K,3,J) - A(K,NP,J)*R(NP,3,J)
            R(K,4,J) = R(K,4,J) - A(K,NP,J)*R(NP,4,J)
  135     CONTINUE
C
   14   CONTINUE
C
C------ solve for last row
        PIVOT = 1.0/A(5,5,J)
        B(5,1,J) = B(5,1,J)*PIVOT
        B(5,2,J) = B(5,2,J)*PIVOT
        B(5,3,J) = B(5,3,J)*PIVOT
        R(5,1,J) = R(5,1,J)*PIVOT
        R(5,2,J) = R(5,2,J)*PIVOT
        R(5,3,J) = R(5,3,J)*PIVOT
        R(5,4,J) = R(5,4,J)*PIVOT
C
C------ back substitute everything
        DO 15 NP=4, 1, -1
          NP1 = NP+1
          DO 141 L=NP1, 5
            B(NP,1,J) = B(NP,1,J) - A(NP,L,J)*B(L,1,J)
            B(NP,2,J) = B(NP,2,J) - A(NP,L,J)*B(L,2,J)
            B(NP,3,J) = B(NP,3,J) - A(NP,L,J)*B(L,3,J)
            R(NP,1,J) = R(NP,1,J) - A(NP,L,J)*R(L,1,J)
            R(NP,2,J) = R(NP,2,J) - A(NP,L,J)*R(L,2,J)
            R(NP,3,J) = R(NP,3,J) - A(NP,L,J)*R(L,3,J)
            R(NP,4,J) = R(NP,4,J) - A(NP,L,J)*R(L,4,J)
  141     CONTINUE
   15   CONTINUE
    1 CONTINUE
C
CCC** Forward sweep: Back substitution using lower block diagonal (Bj's).
      DO 2 J=2, JJ
        JM = J-1
        DO 21 L=1, 4
          DO 211 N=1, 5
            R(N,L,J) = R(N,L,J)
```

```
     &  - (R(1,L,JM)*B(N,1,J) + R(2,L,JM)*B(N,2,J) + R(3,L,JM)*B(N,3,J))
  211     CONTINUE
   21   CONTINUE
    2 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 = 0.
      DNRES = 0.
      DNDUE = 0.              ! dNorm/due
      DNDMS = 0.              ! dNorm/dn
      DNDUT = 0.              ! dNorm/dUtau
      DO 100 J=1, JJ-1
        JP = J+1
        UMID = 0.5*(U(JP) + U(J))
        RNORM = RNORM + UMID*(1.0 - UMID)*DETA(J)
        DNRES = DNRES + DETA(J)*(0.5 - UMID)*(R(2,1,JP) + R(2,1,J))
        DNDUE = DNDUE + DETA(J)*(0.5 - UMID)*(R(2,2,JP) + R(2,2,J))
        DNDMS = DNDMS + DETA(J)*(0.5 - UMID)*(R(2,3,JP) + R(2,3,J))
        DNDUT = DNDUT + DETA(J)*(0.5 - UMID)*(R(2,4,JP) + R(2,4,J))
  100 CONTINUE
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 - 1.0 + DNRES
C
C---- second line ... drive current Utau to UTAU
      RWALL = EEC/H(1)                    ! density at wall
      RHW = -EEC/H(1)**2                  ! dRw/dHw
      RUE = -GM1*UE/TST/H(1)              ! dRw/due
C
      A2 = R(4,2,1) - UTAU**2 *(RHW*R(3,2,1) - RUE)
      B2 = R(4,3,1) - UTAU**2 * RHW*R(3,3,1)
```

```
      C2 = R(4,4,1) - UTAU**2 * RHW*R(3,4,1) + 2.0*RWALL*UTAU
      D2 = R(4,1,1) - UTAU**2 * RHW*R(3,1,1) + 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
   10 CALL GSOLVE
C
CCC-- correct profile iterates
      DO 12 J=1, JJ
        R(1,1,J) = R(1,1,J) - DUE*R(1,2,J) - DMS*R(1,3,J) - DUT*R(1,4,J)
        R(2,1,J) = R(2,1,J) - DUE*R(2,2,J) - DMS*R(2,3,J) - DUT*R(2,4,J)
        R(3,1,J) = R(3,1,J) - DUE*R(3,2,J) - DMS*R(3,3,J) - DUT*R(3,4,J)
        R(4,1,J) = R(4,1,J) - DUE*R(4,2,J) - DMS*R(4,3,J) - DUT*R(4,4,J)
        R(5,1,J) = R(5,1,J) - DUE*R(5,2,J) - DMS*R(5,3,J) - DUT*R(5,4,J)
   12 CONTINUE
C
      RETURN
      END ! DELTAS



      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 ! GSOLVE



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     ***************************************************************************
C
      SUBROUTINE KODE1
      INCLUDE 'BLAKE.INC'
C
C---- Ue specified
C
      A3 = 1.0
      B3 = 0.
      C3 = 0.
      D3 = SPECF - UE
C
      RETURN
      END ! KODE1



      SUBROUTINE KODE2
      INCLUDE 'BLAKE.INC'
C
C---- Rhoe*Ue*Dstar  (= mass defect) specified
C
      A3 = MS*R(1,2,JJ)
      B3 = MS*R(1,3,JJ) + ETAE - F(JJ)
      C3 = MS*R(1,4,JJ)
      D3 = MS*R(1,1,JJ) + SPECF - MD
C
      RETURN
      END ! KODE2



      SUBROUTINE KODE3
      INCLUDE 'BLAKE.INC'
C
C---- Dstar specified
C
      A3 = SC*R(1,2,JJ) - SC/UE*(ETAE - F(JJ))*ME2C
```

```
      B3 = SC*R(1,3,JJ) + (ETAE - F(JJ))/(RHOE*UE)
      C3 = SC*R(1,4,JJ)
      D3 = SC*R(1,1,JJ) + SPECF - DS
C
      RETURN
      END ! KODE3



      SUBROUTINE KODE4
      INCLUDE 'BLAKE.INC'
C
C---- Wall Shear specified
C
      A3 = MS*UE/XF*R(4,2,1) - MS/XF*S(1)
      B3 = MS*UE/XF*R(4,3,1) - UE/XF*S(1)
      C3 = MS*UE/XF*R(4,4,1)
      D3 = MS*UE/XF*R(4,1,1) + SR - SPECF
C
      RETURN
      END ! KODE4



      SUBROUTINE HEADER
      INCLUDE 'BLAKE.INC'
C
      IF(NSTR.EQ.0) RETURN
C
      IF(KODE.EQ.1) WRITE(LSTR,1001)
      IF(KODE.EQ.2) WRITE(LSTR,1002)
      IF(KODE.EQ.3) WRITE(LSTR,1003)
      IF(KODE.EQ.4) WRITE(LSTR,1004)
      WRITE(LSTR,1050) REO
      WRITE(LSTR,2000)
      RETURN
C
 1001 FORMAT('1 CODE 1:  Ue prescribed')
 1002 FORMAT('1 CODE 2:  Mass defect prescribed')
 1003 FORMAT('1 CODE 3:  Dstar prescribed')
 1004 FORMAT('1 CODE 4:  Wall shear prescribed')
 1050 FORMAT('0 RE =',E12.4)
 2000 FORMAT('0 Sta',6X,'x',7X,'Ue',6X,'Mach',
     &  6X,'Pe',8X,'m',6X,'Shear',4X,'Dstar',
     &  4X,'Dmom',6X,'H',7X,'Te',6X,'Twall',2X,'Heat flux'/
     &  1X,115('-'))
      END ! HEADER
```

```
      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.0) RETURN
      IF(MOD(I,NSTR).NE.0) RETURN
C
C---- set weights for interpolation ... similarity case
      WF = 1.0    ! I+1/2 weight
      WB = 0.     ! 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 = 0.5*GM1*UEI**2/TSTI
      TEI = TSTI*(1.0 - EEI)
      REI = RSTI*(1.0 - EEI)**(1.0/GM1)
      PEI = REI*TEI/GAM
      MACH = UEI/SQRT(TEI)
C
      SHPI = SHPF*WF + SHPB*WB
      THI = DSI/SHPI
C
      SX = WF*S(1) + WB*SB(1)
      QX = WF*Q(1) + WB*QB(1)
      HX = WF*H(1) + WB*HB(1)
      TWALL = TSTI*HX
C
      SHEAR = 0.
```

```
      HFLUX = 0.
      IF(X(I).GT.0.0) SHEAR = MSI*UEI/X(I)*SX
      IF(X(I).GT.0.0) HFLUX = -MSI*TSTI/(GM1*X(I))*QX
C
      WRITE(LSTR,1000) I,X(I),UEI,MACH,PEI,MDI,SHEAR,
     &  DSI,THI,SHPI,TEI,TWALL,HFLUX
      RETURN
C
 1000 FORMAT(1X,I4,8F9.4,F8.3,3F9.4)
      END ! STROUT



      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.0) RETURN
      IF(MOD(I,NPFL).NE.0) RETURN
C
      MACH = UE/SQRT(TE)
C
      WRITE(LPFL,1000)
      WRITE(LPFL,1010) I,XF,DS,TH,UE,MACH
      WRITE(LPFL,1011) SC,RHOE,BETU,SHPF
      WRITE(LPFL,1020)
C
C---- extrapolate wall viscosity
      VJ  = MU(1) - DETA(1)*(MU(2)-MU(1))/(DETA(2)+DETA(1))
      VTJ = 0.
      ETA = 0.
      DO 10 J=1, JJ
        RHO = EEC/(H(J) - EE*U(J)**2)
        IF(J.EQ.1 .OR. J.EQ.JJ) GO TO 101
C------- interpolate viscosities to grid nodes
         VJ  = (DETA(J)*MU(J-1) + DETA(J-1)*MU(J))/(DETA(J)+DETA(J-1))
         VTJ = (DETA(J)*MUT(J-1) + DETA(J-1)*MUT(J))/(DETA(J)+DETA(J-1))
  101   WRITE(LPFL,1050) J,ETA,F(J),U(J),S(J),RHO,H(J),Q(J),VJ,VTJ
        ETA = ETA + DETA(J)
   10 CONTINUE
      WRITE(LPFL,1070)
      RETURN
C
 1000 FORMAT('1',94('='))
 1010 FORMAT('0I =',I3,'.5   X =',F9.5,'  Dstar =',F8.4,
     & '  Dmom =',F8.4,'  Ue =',F8.4,'  Mach =',F8.4)
 1011 FORMAT('0Y scale =',F10.6,'   Rhoe =',F8.4,
```

```
     & '    BETAu =',F8.4,'    shape parameter =',F6.3)
1020 FORMAT('0',94('-')/'0 J',6X,'Eta',8X,'F',9X,'U',9X,'S',
     &  9X,'R',9X,'H',9X,'Q',10X,'Mu',7X,'Mut'/
     &  1X,3('-'),3X,7('-'),3X,7('-'),3X,7('-'),3X,7('-'),
     &  3X,7('-'),3X,7('-'),3X,8('-'),3X,7('-'),3X,7('-'))
1050   FORMAT(1X,I3,6F10.5,F11.6,2F10.5)
1070 FORMAT('0',94('-'))
     END ! PFLOUT
```