FINITE-DIFFERENCE SOLUTION OF STEADY TWO-DIMENSIONAL BOUNDARY-LAYER EQUATIONS WITH HEAT TRANSFER

Thesis by

M. Mullainathan

In Partial Fulfillment of the Requirements

For the Degree of

Aeronautical Engineer

California Institute of Technology

Pasadena, California

1980

Submitted: October 2, 1979

ACKNOWLEDGEMENTS

The author likes to thank the Caltech community who in some form or other were helpful during his residence at Caltech. In particular, his sincere thanks are to Dr. Toshi Kubota for his co-operation as research advisor, Dr. C. D. Babcock for his moral support as option representative of the department and Karen Valente for all the help as secretary and especially for typing this thesis elegantly.

ABSTRACT

The incompressible boundary layer equations in two dimensions, with heat transfer have been solved numerically using three different methods and the results are compared. All three methods solve these equations when the pressure distribution is prescribed on the boundary, suction or blowing at the wall and the temperature distribution at the wall. The first method is the second-order Keller's box scheme and the second method is the fourth-order scheme using the Euler-Maclurin formula to replace an integral. The proposed third scheme is also a fourth-order scheme which uses a four point formula to replace an integral. All these schemes use a variable mesh in both co-ordinates. When the truncation error is specified the first scheme chooses an optimum spacing in the direction normal to the wall.

TABLE OF CONTENTS

LIST OF SYMBOLS

A, B, C	elements of $\mathcal{R}$ matrix
f	non-dimensional stream function
g	non-dimensional temperature
h	step size in η-direction
k	step size in x-direction
L	lower triangular matrix
N	number of grid points in the η-direction
Pr	Prandtle number
q	derivative of g with respect to η
R	Reynolds number
$\mathcal{R}$	tridiagnal matrix
r	right-hand side vector in the equation $\mathcal{R}\delta = r$
T	temperature
u	x component of velocity in the boundary layer
U	upper triangular matrix
U_e	velocity of the main stream at the edge of the boundary layer
U_e'	derivative of U_e with respect to x
v	y component of velocity
w	vector arrived at by multiplying the matrix U and the vector δ
x	distance along the surface measured from stagnation point
y	distance normal to the wall
a_1, a_2, θ	coefficients as defined in pages 31, 32, and 34

$\beta_2, \beta_3, \beta_4$	functions of a_1, a_2 and θ as defined in pages 32 and 34
Γ	matrix elements of the lower triangular matrix L
γ	elements of Γ matrix
Δ	matrix elements of the upper triangular matrix U
a_{iJ}	elements of Δ matrix
δ	vector whose elements are δf, δu, $\delta \tau$, etc.

Subscripts

J	count of steps in η-direction
w	quantities at the wall

Superscripts

n	count of steps in x-direction
i	iteration number

1. INTRODUCTION

While developing this computer code to solve the two-dimensional incompressible laminar boundary layer equations with heat transfer it is attempted to make this as general as possible. This is achieved to some extent by prescribing the boundary conditions in the most general form. They are:

a) arbitrary wall suction (or blowing),

b) arbitrary pressure distribution along the wall,

c) arbitrary temperature distribution along the wall.

The governing equations are parabolic partial differential equations with boundary conditions prescribed at the wall and at far away from the wall. A detailed discussion of the methods used to solve these equations can be found in reference 5. A common practice in numerical solution of the two-dimensional boundary-layer equation is to replace the streamwise derivative by a finite-difference approximation and reduce the initial/boundary-value problem for partial differential equations to the boundary-value problem for the ordinary differential equation in the direction transverse to the flow. The reduced problem is solved either by a shooting method or a finite difference method. The shooting method utilizes initial-value solvers for ordinary differential equations, and adjusts the initial values that are not specified in the original boundary-value problem until the solution meets the specified condition at the other end. In the finite-difference method, the ordinary differential equations are replaced by a set of coupled algebraic equations for a large number of unknowns by substituting finite-difference approximations for derivatives. When the original differential equations

are nonlinear, these algebraic equations also are nonlinear and some iterative schemes are used to obtain the solutions. When the memory capacity of a computer is limited, the shooting method is the only choice, but the method is relatively inefficient compared to finite-difference method and fails to converge when the initial value solution becomes very sensitive to the assumed initial values as, for example, in the boundary layer with strong blowing at the wall.

The memory requirement of the two-dimensional boundary-layer solution is no obstacle to the computers of today, and in the present investigation, finite-difference methods based on Keller's box scheme (ref. 1,2) and its modification for higher-order accuracy are formulated and coded in FORTRAN IV to be run on the IBM 370/3032 at the Caltech computing center. Numerical solutions are obtained for a laminar boundary layer subjected to unfavorable pressure gradient and the results from different schemes are compared.

Transformation of boundary layer equations, formulation of finite difference methods, Newton's iteration methods for solution of nonlinear equations, numerical results and discussion are presented in the following sections.

2. GOVERNING EQUATIONS

The continuity, momentum and energy equations for the two dimensional, incompressible, laminar boundary layer are given by (with notation as described in symbols):

$$(1)\quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

$$(2)\quad u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = U_e\frac{dU_e}{dx} + \frac{1}{R}\frac{\partial}{\partial y}\left(\nu\frac{\partial u}{\partial y}\right)$$

$$(3)\quad u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{1}{R\,Pr}\frac{\partial^2 T}{\partial y^2}$$

with boundary conditions

$$(4)\quad u(x,0) = 0 \qquad u(x,\infty) = u_e(x)$$

$$v(x,0) = v_w(x)$$

$$T(x,0) = T_w(x) \qquad T(x,\infty) = 1$$

In addition to the usual assumptions made in the boundary layer theory the following assumptions are made:

a) The density and thermal diffusivity do not change with temperature.

b) Kinematic viscosity is a function of temperature.

In order to avoid the difficulties in numerical integration caused by the boundary layer growth with distance x, the Falkner-Skan transformation is introduced:

$$\eta = y\left(\frac{U_e R}{2x}\right)^{\frac{1}{2}}$$

for y and

$$\varphi = \sqrt{2U_e x/R}\quad f(x,\eta)$$

for the streamfunction and

$$T = 1 + T_w(x)\, g(x, \eta)$$

for temperature.

In the new variables the equations (1-3) and boundary conditions (4) transform into

$$(5)\quad (\nu f_{\eta\eta})\,\eta + \left(1 + \frac{x}{U_e} U_e'\right) f f_{\eta\eta} + 2x \frac{U_e'}{U_e} (1 - f_\eta^2) = 2x(f_\eta f_{x\eta} - f_x f_{\eta\eta})$$

$$(6)\quad \frac{1}{Pr}\, g_{\eta\eta} + \left(1 + x \frac{U_e'}{U_e}\right) f g_\eta - 2x \frac{T_w'}{T_w}\, f_\eta g = 2x(f_\eta g_x - f_x g_\eta)$$

with boundary conditions

$$(7)\quad f(x, 0) = f_w(x)$$

$$f_\eta(x, 0) = 0 \qquad f_\eta(x, \infty) = 1.0$$

$$g(x, 0) = \frac{1}{T_w(x)} [T_w(x) - 1] \qquad g(x, \infty) = 0 \quad .$$

These equations are solved numerically and the method is given in the following sections.

3. FINITE DIFFERENCE APPROXIMATION

The partial differential equations (5-6) are approximated by a set of finite-difference equations using Keller's box scheme (ref. 1).

In order to implement Keller's box scheme for numerical solution, the equations (5-6) are written as a system of first-order partial differential equations:

$$(8)\quad f_\eta = u$$

$$(9)\quad u_\eta = \tau$$

$$(10)\quad (\nu\tau)_\eta + \left(1 + \frac{xU_e'}{U_e}\right) f\tau + 2x\,\frac{U_e'}{U_e}\,(1 - u^2) = 2x(u u_x - f_x \tau)$$

$$(11)\quad g_\eta = q$$

$$(12)\quad \frac{1}{Pr}\, q_\eta + \left(1 + \frac{xU_e'}{U_e}\right) fq - 2x\,\frac{T_w'}{T_w}\, ug = 2x(u g_x - q f_x)$$

with boundary conditions

$$(13)\quad f(x,0) = f_w(x)$$

$$u(x,0) = 0 \qquad u(x,\infty) = 1.0$$

$$g(x,0) = \frac{1}{T_w(x)}\,[T_w(x) - 1] \qquad g(x,\infty) = 0\ .$$

The computational domain ($0 \le x \le x_{max}$, $0 \le \eta \le \eta_{max}$) is divided into rectangular subdomains by grid work

$$x_1, x_2, x_3, \ldots\ldots\ldots\ldots\ldots\ x_n$$

$$\eta_1, \eta_2, \eta_3, \ldots\ldots\ldots\ldots\ldots\ \eta_J$$

The central difference scheme is used to approximate a derivative:

$$\left.\frac{\partial f}{\partial \eta}\right]_{J-\frac{1}{2}} = \frac{1}{h_J}\,(f_J^n - f_{J-1}^n) + O(h^2)\ .$$

and

$$f^n_{J-\frac{1}{2}} = \tfrac{1}{2}(f^n_J + f^n_{J-1}) \ .$$

Similarly

$$\left.\frac{\partial f}{\partial x}\right]^{n-\frac{1}{2}}_J = \frac{1}{k_n}\ (f^n_J - f^{n-1}_J) + 0(k^2)$$

and

$$f^{n-\frac{1}{2}}_J = \tfrac{1}{2}(f^n_J + f^{n-1}_J)$$

In these equations, we use the notation

$$f^n_J = f(x_n, \eta_J)$$

Similar expressions are used for the other derivatives. By substitution of these approximations for derivatives, the nonlinear partial differential equations are approximated by a system of nonlinear finite-difference equations. These nonlinear equations are solved iteratively by Newton's method, at every x.

Let i+1 denote the current iteration. Then

$$f^{i+1} = f^i + \delta f$$

Similar expressions hold good for the other variables. Assuming δf to be small, nonlinear terms are linearized for small corrections. For example, the product $f\tau$ can be written as

$$f^i + \delta f)\ (\tau^i + \delta\tau) = f^i\tau^i + \delta f\,\tau^i + f^i\,\delta\tau$$

by neglecting higher order terms. Then the equations reduce to the following set of linear equations for correction.

$$(14)\quad \delta f_J - \delta f_{J-1} - \frac{h_J}{2}\ (\delta u_J + \delta u_{J-1}) = r_{1,J}$$

$$(15)\quad \delta u_J - \delta u_{J-1} - \frac{h_J}{2}\ (\delta\tau_J + \delta\tau_{J-1}) = r_{2,J-1}$$

$$(16)\quad S_{1,J}\,\delta g_J + S_{2,J}\,\delta f_J + S_{3,J}\,\delta u_J + S_{4,J}\,\delta\tau_J$$

$$S_{11,J-1}\,\delta g_{J-1} + S_{2,J-1}\,\delta f_{J-1} + S_{3,J-1}\,\delta u_{J-1} + S_{5,J-1}\,\delta\tau_J$$

$$= r_{3,J}$$

$$(17)\quad \delta g_J - \delta g_{J-1} - \frac{h_J}{2}\,(\delta q_J + \delta q_{J-1}) = r_{4,J-1}$$

$$(18)\quad S_{6,J}\,\delta q_J + S_{7,J}\,\delta f_J + S_{8,J}\,\delta u_J + S_{9,J}\,\delta g_J$$

$$+ S_{10,J-1}\,\delta q_{J-1} + S_{7,J-1}\,\delta f_{J-1} + S_{8,J-1}\,\delta u_{J-1} + S_{9,J-1}\,\delta g_{J-1}$$

$$= r_{5,J}$$

$$2 \leq J \leq N$$

With these equations we have five boundary conditions

$$(19)\quad \delta f_1 = 0, \quad \delta u_1 = 0, \quad \delta g_1 = 0$$

$$(20)\quad \delta u_N = 0, \quad \delta g_N = 0$$

Complete derivations of these equations are given in Appendix 1.

We have 5(N-1) equations and 5 boundary conditions for 5N unknowns. Therefore these equations can be solved. Now these equations and the boundary conditions are arranged in such a way that if these equations are written in a matrix form the coefficient matrix of the unknowns will be a tridiagnal matrix.

In order to use matrix form the following vectors are introduced

$$\overline{\delta}_J = \begin{pmatrix} \delta f_J \\ \delta u_J \\ \delta\tau_J \\ \delta g_J \\ \delta q_J \end{pmatrix}$$

and

$$\bar{r}_J = \begin{pmatrix} r_{1,J} \\ r_{3,J} \\ r_{5,J} \\ r_{2,J} \\ r_{4,J} \end{pmatrix}$$

for $J=1,2,\ldots,N$.

To the system of equations (14) through (18) for $J=2,\ldots,N$, we add three equations from (19) at the top and two equations from (20) at the bottom. The first five equations involving $\bar{\delta}_1$ and $\bar{\delta}_2$ are,

$$\delta f_1 = 0$$

$$\delta u_1 = 0$$

$$\delta g_1 = 0$$

$$\delta u_2 - \frac{h_2}{2}\,\delta \tau_2 - \frac{h_2}{2}\,\delta \tau_1 = r_{2,1}$$

$$\delta g_2 - \frac{h_2}{2}\,\delta q_2 - \frac{h_2}{2}\,\delta q_1 = r_{4,1}$$

These are written in the following matrix form:

$$[A_1]\,\bar{\delta}_1 + [C_1]\,\bar{\delta}_2 = \bar{r}_1$$

where

$$[A_1] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{h_2}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{h_2}{2} \end{bmatrix}$$

and

$$[C_1] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{h_2}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{h_2}{2} \end{bmatrix}$$

At any station J=m similar matrix equation can be written:

$$B_m \bar{\delta}_{m-1} + A_m \bar{\delta}_m + C_m \bar{\delta}_{m+1} = \bar{r}_m$$

where

$$B_m = \begin{bmatrix} -1 & -\frac{1}{2}h_m & 0 & 0 & 0 \\ S_{2,m-1} & S_{3,m-1} & S_{5,m-1} & S_{11,m-1} & 0 \\ S_{7,m-1} & S_{8,m-1} & 0 & S_{9,m-1} & S_{10,m-1} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$2 \leq m \leq N$$

$$A_m = \begin{bmatrix} 1 & -\frac{1}{2}h_m & 0 & 0 & 0 \\ S_{2,m} & S_{3,m} & S_{4,m} & S_{1,m} & 0 \\ S_{7,m} & S_{8,m} & 0 & S_{9,m} & S_{6,m} \\ 0 & -1 & -\frac{1}{2}h_{m+1} & 0 & 0 \\ 0 & 0 & 0 & -1 & -\frac{1}{2}h_{m+1} \end{bmatrix}$$

$$2 \leq m < N$$

$$A_N = \begin{bmatrix} 1 & -\frac{1}{2}h_N & 0 & 0 & 0 \\ S_{2,N} & S_{3,N} & S_{4,N} & S_{1,N} & 0 \\ S_{7,N} & S_{8,N} & 0 & S_{9,N} & S_{6,N} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$C_m = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{1}{2}h_{m+1} & 0 & 0 \\ 0 & 0 & 0 & 1 & -\frac{1}{2}h_{m+1} \end{bmatrix}$$

and

$$C_N = 0$$

Thus, the whole system becomes a system of linear vector equations:

$$\begin{aligned} A_1\bar{\delta}_1 + C_1\bar{\delta}_2 &= \bar{r}_1 \\ B_2\bar{\delta}_1 + A_2\bar{\delta}_2 + C_2\bar{\delta}_3 &= \bar{r}_2 \\ \cdots\cdots + \cdots\cdots + \cdots\cdots &= \cdots \\ \cdots\cdots + \cdots\cdots + \cdots\cdots &= \cdots \\ B_N\bar{\delta}_{N-1} + A_N\bar{\delta}_N &= \bar{r}_N \end{aligned}$$

These equations collectively can be written in matrix form by denoting

$$\bar{r}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_{2,1} \\ r_{4,1} \end{bmatrix}$$

$$\bar{r}_m = \begin{bmatrix} r_{1,m} \\ r_{2,m} \\ r_{3,m} \\ r_{4,m} \\ r_{5,m} \end{bmatrix} \qquad 2 \leq m < N$$

$$\bar{r}_N = \begin{bmatrix} r_{1,N} \\ r_{2,N} \\ r_{3,N} \\ 0 \\ 0 \end{bmatrix}$$

and

$$\delta_m = \begin{bmatrix} \delta f_m \\ \delta u_m \\ \delta \tau_m \\ \delta g_m \\ \delta q_m \end{bmatrix} \qquad 1 \leq M \leq N$$

we obtain the block tridiagnal form:

$$\begin{bmatrix} A_1 & C_1 & & & & \\ B_2 & A_2 & C_2 & & & \\ & B_3 & A_3 & C_3 & & \\ & & & & B_N & A_N \end{bmatrix} \begin{bmatrix} \bar{\delta}_1 \\ \bar{\delta}_2 \\ \bar{\delta}_3 \\ \\ \bar{\delta}_N \end{bmatrix} = \begin{bmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \bar{r}_3 \\ \\ \bar{r}_N \end{bmatrix}$$

or

$$[\mathcal{R}] \quad [\delta] \quad = \quad [r]$$

This is a matrix equation at any i+1 iteration at any x_n. $\mathcal{R}$ is a function of f, u, τ, etc., of the ith iteration and [r] is a function of f, u, ... etc., of ith iteration of station x_n, which are known. When this equation is solved for δ,

$$f^{i+1} = f^{i} + \delta f \text{ , etc.}$$

are computed and iteration is carried out until convergence criteria are met.

In order to solve the linear equations

$$[\mathcal{R}] \quad [\delta] \quad = \quad [\bar{r}]$$

the L. U decomposition method is used. Where L is a lower triangular matrix and U is an upper triangular matrix. Let L be of the form

$$L = \begin{bmatrix} 1 & & & & \\ \Gamma_2 & 1 & & & \\ & \Gamma_3 & 1 & & \\ & & \Gamma_4 & 1 & \\ & & & \Gamma_N & 1 \end{bmatrix}$$

and U be of the form

$$U = \begin{bmatrix} \Delta_1 & C_1 & & & \\ & \Delta_2 & C_2 & & \\ & & \Delta_3 & C_3 & \\ & & & \Delta_{N-1} & C_{N-1} \\ & & & & \Delta_N \end{bmatrix}$$

then

$$[\mathcal{R}] \quad = \quad [L] \quad [U]$$

From this equation the following relations between Δ, Γ, A and B are derived

$$\Delta_1 = A_1$$

$$\Gamma_J \Delta_{J-1} = B_J \qquad J=2,3,4,\ldots N$$

$$\Delta_J = A_J - \Gamma_J C_{J-1} \qquad J=2,3,\ldots N$$

These equations can be solved sequentially. Hence $\Gamma_2 \ldots \Gamma_N$ and $\Delta_1 \ldots \Delta_N$ can be assumed to be known. The details are given in Appendix 2.

Thus the original equation is written as

$$[L]\ [U]\ [\delta] = [r]$$

If we let

$$[U]\ [\delta] = [w]$$

then we have

$$[L]\ [w] = [r]$$

from which we get

$$\overline{w}_1 = \overline{r}_1$$

$$\overline{w}_m = \overline{r}_m - \Gamma_m \overline{w}_{m-1} \qquad 2 \leq M \leq N$$

where

$$\overline{w}_m = \begin{bmatrix} w_{1,m} \\ w_{2,m} \\ w_{3,m} \\ w_{4,m} \\ w_{5,m} \end{bmatrix}$$

this equation can be solved without any difficulty since only matrix multiplication is involved. Hence w can be assumed to be known for further work. The next equation to be solved is:

$$[U]\ [\delta] = [w]$$

Since U and w are known this can be solved in the following way:

$$\begin{bmatrix} \Delta_1 & C_1 & & & & \\ & \Delta_2 & C_2 & & & \\ & & \Delta_3 & C_3 & & \\ & & & & & \\ & & & & \Delta_{N-1} & C_{N-1} \\ & & & & & \Delta_N \end{bmatrix} \begin{bmatrix} \overline{\delta}_1 \\ \overline{\delta}_2 \\ \\ \\ \overline{\delta}_{N-1} \\ \overline{\delta}_N \end{bmatrix} = \begin{bmatrix} \overline{w}_1 \\ \overline{w}_2 \\ \\ \\ \overline{w}_{N-1} \\ \overline{w}_N \end{bmatrix}$$

From which we get

$$\Delta_N \overline{\delta}_N = \overline{w}_N; \quad \Delta_m \delta_m = w_m - C_m \delta_{m+1}, \quad m = N-1, \ldots$$

Using Gaussian elimination at each step the above equations are solved sequentially.

After δ_M's are known

$$f^{i+1} = f^i + \delta f \quad \text{etc.}$$

are computed. Then we go back to the original equation $[\mathcal{R}][\delta] = [r]$, recompute $[\mathcal{R}]$ and $[r]$, and repeat the procedure until convergence criteria is met.

4. STEP SIZE CONTROL

In order to get an optimum spacing, a step size control subroutine is incorporated in the program. In the central difference scheme used in this computation, the truncation error is given by

$$\text{TRUNCATION ERROR} = 0(h^2 y'')$$

for an equation of the type

$$y' = f(x)$$

Step size is chosen such that the truncation error in equations (10 and 12) are below the prescribed limit. This is achieved in the following way. The equations are solved starting with a uniform grid and the mesh is improved according to the following criteria, by checking the truncation error of τ solution.

(1) For each J, let $\tilde{h}_J = \max(h_J, h_{J-1})$ and calculate the truncation error

$$\epsilon = \frac{\tilde{h}_J^2}{(h_J, h_{J-1})} \left| \left(\frac{\tau_{J+1} - \tau_J}{h_J} - \frac{\tau_J - \tau_{J-1}}{h_{J-1}} \right) \right|$$

if $\epsilon \leq$ maximum error allowed, nothing is done. If $\epsilon >$ maximum error allowed, then two new points are added such that

$$\tilde{\eta}_J = \tfrac{1}{2}(\eta_{J+1} + \eta_J) \quad \text{and} \quad \tilde{\eta}_{J-1} = \tfrac{1}{2}(\eta_J + \eta_{J-1})$$

(2) The new mesh is inspected and new points are introduced such that

$$\text{HLOW} \leq \frac{h_J}{h_{J-1}} \leq \text{HHIGH}$$

Typical values are 0.3 for HLOW and 3 for HHIGH.

(3) Solve the equations with the new mesh points.

(4) Examine the truncation error of the new solution.

(5) Repeat the process until the error is less than the prescribed limit everywhere.

Interpolation:

When the step size in the η direction is changed at a x_n station, the variables at the x_{n-1} station as well as the previous iterated values at the current station need to be interpolated for the new mesh points.

Suppose f(x) is a discrete function given at points

$$x = x_\nu \qquad \nu = 1, 2, \ldots N$$

Then a cubic polynominal P(x) is chosen for $x_{\nu-1} \leq x \leq x_\nu$ such that at the given points the following conditions are satisfied

$$P(x_\nu) = f(x_\nu) , \quad P(x_{\nu-1}) = f(x_{\nu-1})$$

$$P'(x_\nu) = f'(x_\nu) , \quad P'(x_{\nu-1}) = f'(x_{\nu-1})$$

In the program the subroutine CALCCF calculates the coefficients of the cubic polynominal and function subroutine PCUBIC calculates the value of the function at any desired point. (Ref. 3.)

5. A HIGHER ORDER METHOD TO SOLVE BOUNDARY LAYER EQUATIONS

The box scheme as described in Sections 4 and 5 is a first-order method in both x and η spacings. This calls for a fine mesh to achieve the desired accuracy compared to higher order schemes, which may require more computing time. Therefore, it is desirable to employ a higher-order scheme which may be more efficient for the same accuracy and which is easy to program. Here we describe one such method and compare it with another higher order method existing already. Reference 6 gives a brief review of other existing higher order schemes.

We consider the transformed equations (5, 6) with boundary conditions (7). They are written as a system of first-order equations (8, 9, 10, 11, 12) and boundary conditions (13). Only equations 10 and 12 involve the derivations with respect to x. Using central difference schemes we write

$$\left.\frac{\partial u}{\partial x}\right]^{n-\frac{1}{2}} = \frac{1}{k^n}\,(u^n - u^{n-1})$$

and

$$u^{n-\frac{1}{2}} = \tfrac{1}{2}(u^n + u^{n-1})$$

Appendix 3 gives the details of the equations and here we explain only the procedure. Using this we can write these equations at any $n-\frac{1}{2}$ station along x and reduce it to the form

$$Z = f(\eta)$$

integrating this with respect to η from η_{J-1} to η_J we get

$$Z_J - Z_{J-1} = \int_{\eta_{J-1}}^{\eta_J} f(\eta, Z)\, d\eta$$

where η_{J-1} and η_J are any consecutive points in the mesh. To evaluate this integral we use a NEWTON'S BACKWARD interpolating polynominal for $f(\eta)$ and then integrate. Using four points η_{J-2}, η_{J-1}, η_J, η_{J+1}, we get

$$Z_J - Z_{J-1} = C_1 f_{J-2} + C_2 f_{J-1} + C_3 f_J + C_4 f_{J+1}$$

where C_1, C_2, C_3 and C_4 are functions of h_{n-1}, h_n and h_{n+1}. Linearizing this for small corrections as done earlier we get

$$\delta Z_J - \delta Z_{J-1} - C_1 \delta f_{J-2} - C_2 \delta f_{J-1} - C_3 \delta f_J - C_4 \delta f_{J+1}$$

$$= Z_{J-1} - Z_J + C_1 f_{J-2} + C_2 f_{J-1} + C_3 f_J + C_4 f_{J+1}$$

At the boundaries one-sided four-point formulas are used and given in Appendix 3. This system of equations can be written as before in the matrix form

$$\mathcal{R}\,\delta = r \ .$$

Solving this equation is more involved compared to the first-order method since this equation contains unknowns at four points in every equation compared to two in the earlier case. In order to simplify this, the following approximation is used. Using Taylor's series expansions, we write the equation for δZ's as

$$\delta Z_J - \delta Z_{J-1} - (C_2 + C_1)\,\delta f_{J-1} - (C_3 + C_4)\delta f_J + 0(h^2)$$

$$= Z_{J-1} - Z_J + C_1 f_{J-2} + C_2 f_{J-1} + C_3 f_J + C_4 f_{J+1}$$

If we neglect $0(h^2)$ terms in the left-hand side the equation reduces to a form of the earlier case which can be solved easily.

In order to examine the consequences of this approximation,

let $\mathcal{R} = \mathcal{R}_1 + 0(h^2)$. Then $\mathcal{R}\delta = r$ becomes $\mathcal{R}_1\delta = r + 0(h^2)\delta$. Namely, the approximation introduces the error $0(h^2)\delta$ at each iteration, but since the condition of convergence is $\delta \to 0$ this error does not affect the final results. It affects only the rate of convergence. This simplification significantly reduces the complexity in solving this equation.

The equation $\mathcal{R}_1\delta = r$ contains new elements only in $\mathcal{R}_1$ and r. Hence only the subroutines RHS1 and RLHS need to be changed since the structure of the matrix is unchanged.

After making these changes the test case mentioned earlier has been solved and the results are given in Table 1. To compare this proposed method with existing methods another higher order method that uses the Euler-Maclurin formula to replace the integral has been programed and the results of this method also are given in Table 1. Reference 7 describes this method in detail and a brief description is given in Appendix 3.

6. DESCRIPTION OF THE PROGRAM

The above algorithm is written in Fortran IV for use in IBM 370-165 computer.

MAIN

The main program performs data input and calculation of parameters like α_1, α_2, β_2 ... etc., and calls various subprograms.

SUBROUTINES

RHS1 and RLHS calculates the right side vector r_1, r_2, ... etc., in the equation $R\delta = r$. Also calculates the elements of R matrix

GUESS gives the starting profile at x=0. For further calculations the profile at $x=x_{n-1}$ is used as initial guess at $x=x_n$. This helps to reduce the number of iterations at consecutive points.

DECOMP and SOLVE solve the equation $[R]\delta = r$ and gives new profile.

STEPZ computes the optimum step size when the maximum truncation error is specified.

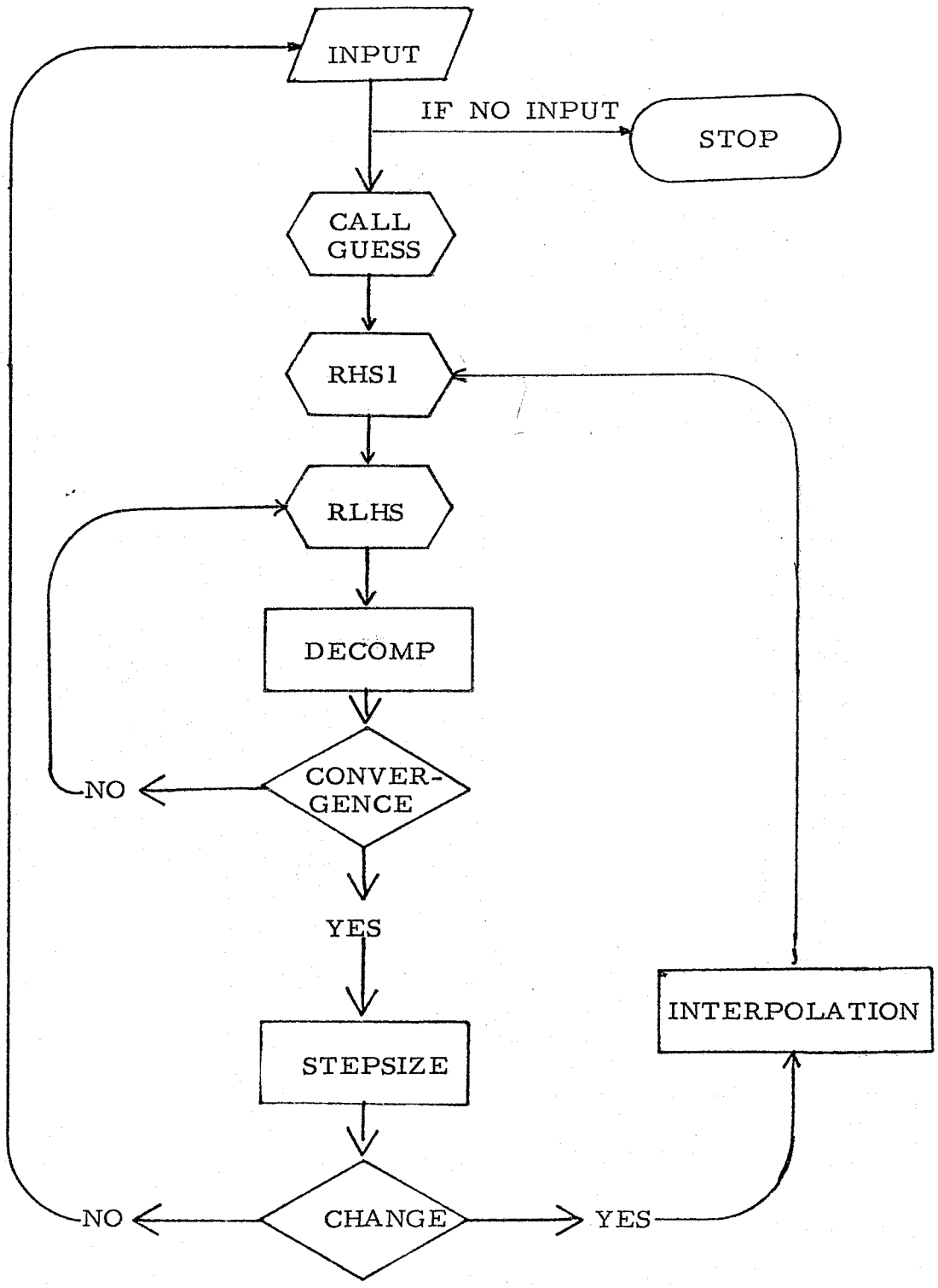
FMIDPT, CALCCF and PCUBIC are part of interpolating routine which gives the values for the new mesh points.

FUNCTION VISC and DVDT supplies the functional relation between ν and temperature.

A flow chart is given below and the listing of the program is presented in Appendix 4.

In the case of the four-point scheme a new subroutine COEF is added to calculate the coefficients of the four-point formula.

FLOW CHART


INPUT
IF NO INPUT
STOP
CALL GUESS
RHS1
RLHS
DECOMP
CONVER-GENCE
NO
YES
STEPSIZE
CHANGE
NO
YES
INTERPOLATION

7. RESULTS AND DISCUSSION

To compare these schemes the following test case has been used:

$$U_e = x - \beta x^5$$

where β is chosen such that the separation occurs at x=1. The Tifford series is used to determine β, which gives

$$\beta = 0.3411409$$

All three schemes are run with a uniform mesh in both x and η spacings. In the x spacing k was taken to be 0.025 and in the η spacing h was taken to be 0.1. The results are presented in Table 1 along with the results of Tifford's series for comparison. The computed separation points differ among all methods. The probable reason is that, near the separation point, the derivatives of flow properties with respect to x is large and at the separation point it is infinite. Hence all these methods fail near separation.

The results from both four- and two-point fourth-order methods agree very well with each other and also with Tifford's series except near the separation point. Tifford's series solutions are not valid in that region. Therefore, for the given β the separation could occur at a point different from x=1. For these reasons the failure to predict separation at x=1 need not be construed as a defect in these methods.

Goldstein (ref. 8) investigated the solution of laminar boundary layer equations near separation and concluded that the wall shear vanishes as the square root of the distance to the separation point. The separation point, x_s, was estimated by fitting a parabola to the

computed wall shear at $x = 0.9$ and 0.95. The fourth-order methods give $x_s = 0.9587$, while the second-order method gives $x_s = 0.9589$. Therefore, we conclude that all computed values beyond $x = 0.95$ are meaningless, and we also suspect that Tifford's series solution is unreliable beyond $x \simeq 0.8$.

8. CONCLUSION

Among the three methods considered, the second-order method has poor accuracy. This strongly emphasizes the need to adopt higher-order methods. The complexity in adopting such higher-order schemes prevented the wide use of such schemes in the past. Now it is clear that extension to higher-order is not necessarily a problem once certain approximations are made as demonstrated in the present study. Among the two higher-order methods each has its own advantages and disadvantages. The two-point formula which uses the Euler-Maclurin formula has a lower truncation error of

$$\frac{1}{720}\, h_J^5\, \frac{\partial^5 \tau}{\partial \eta^5}$$

compared to

$$\left[\frac{1}{30}\, h_J^5 + \frac{1}{12}\, h_J^4 (h_{J-1} + h_{J+1}) + \frac{1}{4}\, h_J^3\, h_{J-1}\, h_{J+1}\right] F[\eta_{J+2}, \eta_{J+1}, \eta_J, \eta_{J-1}, \eta_{J-2}]$$

where $F[\eta_{J+2}, \eta_{J+1}, \eta_J, \eta_{J-1}, \eta_{J-2}]$ is the fifth difference in the NEWTON's divided difference (ref. 3) the truncation error of the four-point formula at any station η_J. This reduces to

$$\frac{11}{720}\, h_J^5\, \frac{\partial^5 \tau}{\partial \eta^5}$$

for uniform mesh and the upper bound for non-uniform mesh occurs when h_{J-1} and $h_J \to 0$ and the error is

$$\frac{24}{720}\, h_J^5\, \frac{\partial^5 \tau}{\partial \eta^5} \quad .$$

These errors are so small in many cases that these errors will not have any effect up to at least six significant figures. Extension of these methods to higher-orders are possible. The four-point scheme is easy to extend since only the coefficients in the formula need to be redefined, whereas in the two-point scheme higher derivatives have to be evaluated. For the type of equations dealt with

in this study the evaluation of higher derivatives become cumbersome and time consuming. The author believes that the fourth-order methods are optimum. Between the two fourth-order schemes investigated the two-point scheme is found to be inefficient since the derivative evaluation is more involved as in the present case. By using the four-point scheme one can save up to 20% of computing time in the uniform mesh case and up to 17% in the case of non-uniform mesh without loss of accuracy. Nevertheless, an optimum method would be a combination of these two methods. That is to use two-point schemes where the derivatives come out of solution as in the case of the first three equations and use the four-point scheme as in the case of the last two equations for which the derivatives are not available. This will be more efficient than the two methods discussed earlier.

REFERENCES

1. T. Cebeci and P. Bradshaw "Momentum Transfer in Boundary Layers", 1977, McGraw-Hill Book Company.

2. T. Cebeci and A.M.O. Smith "Analysis of Turbulent Boundary Layers", 1974, Academic Press.

3. S.D. Conte and C. DeBoor "Elementary Numerical Analysis", 1972, McGraw-Hill Book Company.

4. H. Schlichting "Boundary Layer Theory", 1968, McGraw-Hill Book Company.

5. A.M.O. Smith and D.W. Clutter "Solution of the Incompressible Laminar Boundary Layer Equations", Douglas Aircraft Company Report No. ES 40446, 1961.

6. H.B. Keller "Numerical Methods in Boundary-Layer Theory", 1978 Annual Review of Fluid Mechanics, Vol. 10.

7. S.F. Wornom "A Critical Study of Higher-Order Numerical Methods for Solving the Boundary Layer Equations", 1977, AIAA Paper No. 77-637, June.

8. S. Goldstein "On Boundary-Layer Flow Near a Point of Separation", Quart. J. Mech. Appl. Math., Vol. 1, 1948, pp. 43-69.

TABLE 1

WALL SHEAR VS. x FOR DIFFERENT SCHEMES

($\Delta x = 0.025$, $\Delta\eta = 0.1$)

x	TIFFORD'S SERIES	FOURTH-ORDER METHODS		SECOND-ORDER ORDER
		FOUR-POINT	TWO-POINT	
0	1.232588	1.232588	1.232588	1.231682
0.1	1.23252	1.23252	1.23252	1.231615
0.2	1.23152	1.231514	1.231514	1.230613
0.3		1.227145	1.227145	1.22626
0.4	1.2153	1.215245	1.215245	1.214406
0.5		1.189458	1.189458	1.188718
0.6		1.139967	1.139968	1.139413
0.7		1.050123	1.050124	1.049892
0.8	0.8919	0.886646	0.886648	0.886966
0.9	0.5979	0.559499	0.559503	0.560802
0.95		0.215214	0.215222	0.217702
0.97	0.1815			
0.975		0.18715	-0.123194	-0.116905
1.0	0	0.026157	-0.178514	

APPENDIX 1. FINITE-DIFFERENCE FORMULATION

Finite difference approximation of equation

$$(\nu f_{\eta\eta})_\eta + \left(1 + \frac{x}{U_e} U_e'\right) f f_{\eta\eta} + 2x \frac{U_e'}{U_e} (1 - f_\eta^2)$$

$$= 2x(f_\eta f_{x\eta} - f_x f_{\eta\eta})$$

$$\frac{1}{Pr} g_{\eta\eta} + \left(1 + \frac{x}{U_e} U_e'\right) f g_\eta - \frac{2xT_w'}{T_w} f_\eta g$$

$$= 2x(f_\eta g_x - f_x g_\eta)$$

with boundary conditions

at

$\eta = 0$ $\quad f = f_w$

$f_\eta = 0$

$g = (T_{w-1})/T_w$

$\eta = \infty$ $\quad f_\eta = 1.0$

$g = 0$.

Before integrating numerically, these equations are written as a system of first-order partial differential equations.

$$f_\eta = u$$

$$u_\eta = \tau$$

$$(\nu\tau)_\eta + \left[1 + x \frac{U_e'}{U_e}\right] f\tau + 2x \frac{U_e'}{U_e} (1 - u^2) = 2x(uu_x - f_x\tau)$$

$$g_\eta = q$$

$$\frac{1}{Pr}\, q_\eta + \left(1 + x\,\frac{U_e'}{U_e}\right) fq - 2x\,\frac{T_w'}{T_w}\, ug = 2x(ug_x - qf_x)$$

These five first-order equations are written as finite difference equations in the following way.

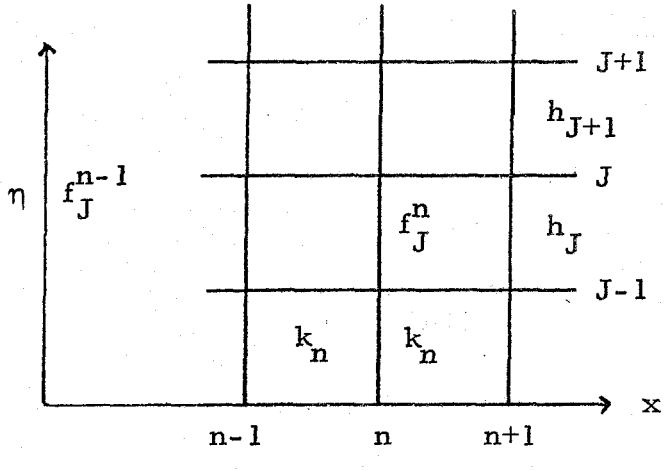
Description of Grid System

The computation domain ($0 \leq x \leq x_{max}$, $0 \leq \eta \leq \eta_{max}$) is covered by the network ($x = x_1, x_2, \ldots x_n$; $\eta = \eta_1, \eta_2 \ldots \eta_N$). The step size in the x-direction is

$$k_n = x_n - x_{n-1}$$

and in the η-direction

$$h_J = \eta_J - \eta_{J-1}$$



Here superscripts denote the x-index and subscripts denote the η-index.

Using the central differences scheme the derivatives are written as

$$\left.\frac{\partial f}{\partial \eta}\right|^{n}_{J-\frac{1}{2}} = \frac{f_J^n - f_{J-1}^n}{h_J}$$

and

$$\left.\frac{\partial f}{\partial x}\right|_J^{n-\frac{1}{2}} = \frac{f^n - f^{n-1}}{k_n}$$

A linear interpolation is used to get midpoint values. For example

$$f^n_{J-\frac{1}{2}} = \frac{f^n_J + f^n_{J-1}}{2}$$

and

$$f^{n-\frac{1}{2}}_J = \frac{f^n_J + f^{n-1}_J}{2}$$

(1) $\quad f_\eta = u$

This equation can be written as

$$\frac{f^n_J - f^n_{J-1}}{h_J} = u^n_{J-\frac{1}{2}} = \frac{u^n_J + u^n_{J-1}}{2}$$

$$f^n_J - \frac{h_J}{2}\, u^n_J - f^n_{J-1} - \frac{h_J}{2}\, u^n_{J-1} = 0$$

Linearization:

Set

$$f^{n,i+1} = f^{n,i} + \delta f\,, \quad u^{n,i+1} = u^{n,i} + \delta u$$

where the second superscript i is the iteration number. Substituting these into eq. (1) and retaining only the first powers of δf and δu, we obtain (dropping n)

$$\delta f_J - \frac{h_J}{2}\, \delta u_J - \delta f_{J-1} - \frac{h_J}{2}\, \delta u_{J-1} = r_{1,J}$$

where

$$r_{1,J} = - f^i_J + f^i_{J-1} + \frac{h_J}{2}\, u^i_J + \frac{h_J}{2}\, u^i_J \,.$$

Similarly the equations (2) and (4) can be written as

$$\delta u_J - \frac{h_J}{2}\, \delta \tau_J - \delta u_{J-1} - \frac{h_J}{2}\, \delta \tau_{J-1} = r_{2,J-1}$$

where

$$r_{2,J-1} = - u_J^i + u_{J-1}^i + \frac{h_J}{2} \tau_J^i + \frac{h_J}{2} \tau_{J-1}^i$$

$$\delta g_J - \delta g_{J-1} - \frac{h_J}{2} \delta q_J - \frac{h_J}{2} \delta q_{J-1} = r_{4,J-1}$$

where

$$r_{4,J-1} = g_{J-1}^i - g_J^i + \frac{h_J}{2} q_J^i + \frac{h_J}{2} q_{J-1}^i$$

Now equation (3) is written as follows

$$L = 2x(u u_x - f_x \tau)$$

where

$$L \equiv (\nu\tau)_\eta + \left(1 + x \frac{U_e'}{U_e}\right) f\tau + 2x \frac{U_e'}{U_e} (1 - u^2)$$

Finite differencing in the x-direction, we obtain at $x = x_{n-\frac{1}{2}}$, $\eta = \eta_J$

$$\tfrac{1}{2}(L_J^n + L_J^{n-1}) = \frac{x_{n-\frac{1}{2}}}{k_n} (u_J^n + u^{n-1})(u_J^n - u_J^{n-1})$$

$$- (f_J^n - f_J^{n-1})(\tau_J^n + \tau_J^{n-1})$$

By letting

$$\frac{2x_{n-\frac{1}{2}}}{k_n} = a_2^n$$

the above equation is written as

$$L_J^n - a_2^n (u_J^n)^2 + a_2^n f_J^n \tau_J^n + a_2^n f_J^n \tau_J^{n-1} - a_2^n f_J^{n-1} \tau_J^n = RL3_J$$

where

$$RL3_J = - L_J^{n-1} - a_2^n (u_J^{n-1})^2 + a_2^n f_J^{n-1} \tau_J^{n-1}$$

At any nth station the solutions at n-1 station are supposed to be known, and hence the right hand side of the above equation is known. Finite differencing in the η-direction, then, results in the equation (dropping n)

$$\frac{1}{h_J}\left[(\nu\tau)_J - (\nu\tau)_{J-1}\right] + \frac{\beta_2}{2} f_J \tau_J + \frac{\beta_2}{2} f_{J-1}\tau_{J-1}$$

$$+ 2a_1 - \frac{\beta_3}{2} u_J^2 - \frac{\beta_3}{2} u_{J-1}^2$$

$$+ \frac{a_2}{2} f_J \tau_J^{n-1} + \frac{a_2}{2} f_{J-1}\tau_{J-1}^{n-1}$$

$$- \frac{a_2}{2} f_J^{n-1}\tau_J - \frac{a_2}{2} f_{J-1}^{n-1}\tau_{J-1} = RL3_{J-\frac{1}{2}}$$

where

$$a_1 = x\, U_e'/U_e$$

$$\beta_2 = 1 + a_1 + a_2$$

$$\beta_3 = 2a_1 + a_2$$

Now introducing

$$\nu^{i+1} = \left[\nu^i + \frac{\partial\nu}{\partial g}\cdot \delta g\right] = \nu^i + (\nu')^i \delta g$$

$$\tau^{i+1} = \tau^i + \delta\tau^i \text{ , etc.}$$

and dropping higher order terms this equation becomes

$$S_{1,J}\delta g_J + S_{2,J}\delta f_J + S_{3,J}\delta u_J + S_{4,J}\delta\tau_J +$$

$$S_{11,J-1}\delta g_{J-1} + S_{2,J-1}\delta f_{J-1} + S_{3,J-1}\delta u_{J-1} + S_{5,J-1}\delta\tau_{J-1} = r_{3,J}$$

where

$$S_{1,J} = \frac{2}{h_J}\nu_J'\tau_J \text{ , with } \nu_J' = \left(\frac{\partial\nu}{\partial g}\right)_J$$

$$S_{2,J} = \beta_2\tau_J + a_2\tau_J^{n-1}$$

$$S_{3,J} = -2\beta_3 u_J$$

$$S_{4,J} = \frac{2}{h_J}\nu_J + \beta_2 f_J - a_2 f_J^{n-1}$$

$$S_{5,J-1} = -\frac{2}{h_J}\nu_{J-1} + \beta_2 f_{J-1} - a_2 f_{J-1}^{n-1}$$

$$S_{11,J-1} = -\frac{2}{h_J}\nu_{J-1}'\tau_{J-1}$$

and

$$r_{3,J} = 2\,RL3_{J-\frac{1}{2}} - 4a_1 - \beta_2 \tau_J f_J$$

$$- \beta_2 \tau_{J-1} f_{J-1} + \beta_3 u_J^2 + \beta_3 u_{J-1}^2$$

$$- a_2 \tau_J^{n-1} f_J - a_3 \tau_{J-1}^{n-1} f_{J-1}$$

$$+ a_2 \tau_J f_J^{n-1} + a_2 \tau_{J-1} f_{J-1}^{n-1}$$

$$- \frac{2}{h_J} \nu_J \tau_J + \frac{2}{h_J} \nu_{J-1} \tau_{J-1}$$

Following the same procedure equation (5) reduces to

$$S_{6,J}\,\delta q_J + S_{7,J}\,\delta f_J + S_{8,J}\,\delta u_J + S_{9,J}\,\delta g_J +$$

$$S_{10,J-1}\delta q_{J-1} + S_{7,J-1}\delta f_{J-1} + S_{8,J-1}\delta u_{J-1} + S_{9,J-1}\delta g_{J-1} = r_{5,J}$$

where

$$S_{6,J} = \frac{2}{Prh_J} + \beta_2 f_J - a_2 f_J^{n-1}$$

$$S_{7,J} = \beta_2 q_J + a_2 q_J^{n-1}$$

$$S_{8,J} = a_2 g_J^{n-1} - \beta_4 g_J$$

$$S_{9,J} = - \beta_4 u_J - a_2 u_J^{n-1}$$

$$S_{10,J-1} = \beta_2 f_{J-1} - a_2 f_{J-1}^{n-1} - \frac{2}{PrhJ}$$

$$r_{5,J} = 2\,RL5_{J-\frac{1}{2}} - \frac{2}{Prh_J} q_J + \frac{2}{Prh_J} q_{J-1}$$

$$- \beta_2 q_J f_J - \beta_2 q_{J-1} f_{J-1} + \beta_4 u_J g_J$$

$$+ \beta_4 u_{J-1} g_{J-1} + a_2 u_J^{n-1} g_J$$

$$+ a_2 u_{J-1}^{n-1} g_{J-1} - a_2 g_J^{n-1} u_J - a_2 g_{J-1}^{n-1} u_{J-1}$$

$$- a_2 q_J^{n-1} f_J - a_2 q_{J-1}^{n-1} f_{J-1} + a_2 q_J f_J^{n-1}$$

$$+ a_2 q_{J-1} f_{J-1}^{n-1}$$

and

$$\beta_4 = \theta + a_2 \quad \text{and} \quad \theta = 2x \frac{T_w'}{T_w}$$

APPENDIX 2. L.U. DECOMPOSITION OF $\mathcal{R}$ MATRIX

$\mathcal{R}$ is a tridiagnal band matrix of the form

$$\mathcal{R} = \begin{bmatrix} A_1 & C_1 & & & & 0 \\ B_2 & A_2 & C_2 & & & \\ & B_3 & A_3 & C_3 & & \\ & \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & & \\ 0 & & & B_{N-1} & A_{N-1} & C_{N-1} \\ & & & & B_N & A_N \end{bmatrix} \qquad \text{(A2-1)}$$

and to be decomposed into a form such that $\mathcal{R}$ = L.U., where L and U are of the form

$$L = \begin{bmatrix} 1 & & & & & 0 \\ \Gamma_2 & 1 & & & & \\ & \Gamma_3 & 1 & & & \\ & & \Gamma_4 & 1 & & \\ & & & & & \\ 0 & & & & \Gamma_N & 1 \end{bmatrix} \qquad \text{(A2-2)}$$

and

$$U = \begin{bmatrix} \Delta_1 & C_1 & & & & 0 \\ & \Delta_2 & C_2 & & & \\ & & \Delta_3 & C_3 & & \\ & & & & & \\ & & & & & \\ 0 & & & & \Delta_{N-1} & C_{N-1} \\ & & & & & \Delta_N \end{bmatrix} \qquad \text{(A2-3)}$$

By equating the matrix elements of $\mathcal{R}$ with the elements of L.U. after multiplication we get the following matrix equations:

$$\left.\begin{aligned} \Delta_1 &= A_1 \\ \Gamma_J \Delta_{J-1} &= \beta_J \qquad J=2,3,\ldots N \\ \Delta_J &= A_J - \Gamma_J C_{J-1} \qquad J=2,3,\ldots N \end{aligned}\right\} \qquad \text{(A2-4)}$$

It can be shown that Δ_J is of the form

$$\Delta_J = \begin{bmatrix} a_{11,J} & a_{12,J} & a_{13,J} & a_{14,J} & a_{15,J} \\ a_{21,J} & a_{22,J} & a_{23,J} & a_{24,J} & a_{25,J} \\ a_{31,J} & a_{32,J} & a_{33,J} & a_{34,J} & a_{35,J} \\ 0 & -1 & -\frac{1}{2}h_{J+1} & 0 & 0 \\ 0 & 0 & 0 & -1 & -\frac{1}{2}h_{J+1} \end{bmatrix}$$

and Γ_J is of the form

$$\Gamma_J = \begin{bmatrix} \gamma_{11,J} & \gamma_{12,J} & \gamma_{13,J} & \gamma_{14,J} & \gamma_{15,J} \\ \gamma_{21,J} & \gamma_{22,J} & \gamma_{23,J} & \gamma_{24,J} & \gamma_{25,J} \\ \gamma_{31,J} & \gamma_{32,J} & \gamma_{33,J} & \gamma_{34,J} & \gamma_{35,J} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since $\Delta_1 = A_1$ is known, the following equation can be solved.

$$\Gamma_2 \Delta_1 = \beta_2 \ .$$

This gives

$$\Gamma_2 = \begin{bmatrix} -1 & -\frac{h_2}{2} & 0 & 0 & 0 \\ 0 & -\frac{2}{h_2}S_{5,1} & 0 & -\frac{2}{h_2}S_{5,1} & 0 \\ 0 & 0 & -\frac{2}{h_2}S_{10,1} & 0 & -\frac{2}{h_2}S_{10,1} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After Γ_2 is known, Δ_2 is determined from

$$\Delta_2 = A_2 - \Gamma_2 C_1$$

This involves only matrix multiplication and addition. Γ_J and Δ_J for $J=3,4,\ldots,N$ may be determined by successive application of second and third equations of (A2-4). Determination of Δ_J after

Γ_J is known involves only matrix multiplication, but getting Γ_J from

$$\Gamma_J \Delta_{J-1} = \beta_J$$

involves solution of simultaneous equations.

Let us transpose this matrix equation

$$\Delta_{J-1}^T \Gamma_J^T = \beta_J^T$$

Since Γ_J contains only zeros in the last two rows, the above equation gives three equations for three row-vectors of Γ_J

$$[\Delta_{J-1}^T] \begin{bmatrix} r_{\ell 1, J} \\ r_{\ell 2, J} \\ r_{\ell 3, J} \\ r_{\ell 4, J} \\ r_{\ell 5, J} \end{bmatrix} = [D_\ell] \qquad \underline{\ell = 1, 2, 3}$$

where

$$D_1 = \begin{bmatrix} -1 \\ -\frac{h_J}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad D_2 = \begin{bmatrix} S_{2, J-1} \\ S_{3, J-1} \\ S_{5, J-1} \\ S_{11, J-1} \\ 0 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} S_{7, J-1} \\ S_{8, J-1} \\ 0 \\ S_{9, J-1} \\ S_{10, J-1} \end{bmatrix}$$

These three equations are solved using Gaussian elimination.

APPENDIX 3. HIGHER-ORDER FINITE DIFFERENCE

The boundary layer equations for constant density with heat-transfer case in the transformed variables as mentioned in Appendix 1 describing the second-order scheme.

As in Appendix 1, the central difference scheme is used to approximate a derivative in the x-direction and mid-grid value

$$u_x = \frac{1}{k_n} (u^n - u^{n-1})$$

$$u^{n-\frac{1}{2}} = \tfrac{1}{2} (u^n + u^{n+1})$$

Using these finite-difference approximation, we get the following equations at $x = x_{n+1}$ (refer to Appendix 1 for the grid system and other details). The superscript n+1 is omitted from the variables at $x = x_{n+1}$ for clarity.

$$f_\eta = u$$

$$u_\eta = \tau$$

$$(\nu\tau)_\eta = F$$

$$g_\eta = q$$

$$\frac{1}{Pr} q_\eta = G$$

where

$$F = \beta_3 u^2 - \beta_2 f\tau - 2a_1 + a_2 f^n\tau$$

$$- a_2 [\tau^n f + (u^n)^2 - f^n\tau^n] - L_3^n$$

$$G = ug(\theta+a_2) - (1+a_1+a_2)qf + a_2 (u^n g + g^n u - q^u f + f^n q)$$

$$- a_2 (u^n g^n - q^n f^n) - L_5^n$$

$$L_3^n = (\nu\tau)_\eta^n \quad (1+a_1^n)\, f^n \tau^n + 2a_1^n \,[1 - (u^n)^2]$$

$$L_5^n = \frac{1}{Pr}\, q_\eta^n + (1+a_1^n)\, f^n q^n - \theta^n u^n g^n$$

and a_1, a_2, β_2, β_3 and θ are defined in Appendix 1. Integrating these equations from η_{J-1} to η_J we get

$$f_J - f_{J-1} = \int_{\eta_{J-1}}^{\eta_J} u \, d\eta$$

$$u_J - u_{J-1} = \int_{\eta_{J-1}}^{\eta_J} \tau \, d\eta$$

$$(\nu\tau)_J - (\nu\tau)_{J-1} = \int_{\eta_{J-1}}^{\eta_J} F \, d\eta$$

$$g_J - g_{J-1} = \int_{\eta_{J-1}}^{\eta_J} q \, d\eta$$

$$\frac{1}{Pr}(q_J - q_{J-1}) = \int_{\eta_{J-1}}^{\eta_J} G \, d\eta$$

In the integral

$$\int_{\eta_{J-1}}^{\eta_J} F \, d\eta$$

F is replaced by a Newton's backward interpolating polynomial using the information available at η_{J+1}, η_J, η_{J-1} and η_{J-2}. Then we get

$$\int_{\eta_{J-1}}^{\eta_J} F\, d\eta = C_1(J) F_{J-2} + C_2(J) F_{J-1} + C_3(J) F_J + C_4(J) F_{J-1}$$

$$3 \leq J \leq N-1$$

For J=2, we use the values at η_1, η_2, η_3 and η_4 for the interpolation formula and get

$$\int_{\eta_1}^{\eta_2} F\, d\eta = b_1 F_1 + b_2 F_2 + b_3 F_3 + b_4 F_4$$

At the outer edge, the approximation

$$\int_{\eta_{N-1}}^{\eta_N} F\, d\eta = \tfrac{1}{2}(F_N + F_{N-1}) h_N$$

is used because all f, u, τ, etc., are linearly varying and hence this is exact. For different behavior of the function at the far end the form used at J=2 should be used.

After linearization as done earlier the equations are reduced to

$$\delta f_J - \delta f_{J-1} - (C_2 + C_1)\delta u_{J-1} - (C_3 + C_4)\delta u_J$$

$$= f_{J-1} - f_J + C_1 u_{J-2} + C_2 u_{J-1} + C_3 u_J + C_4 u_{J+1}$$

$$\delta u_J - \delta u_{J-1} - (C_2 + C_1)\delta \tau_{J-1} - (C_3 + C_4)\delta \tau_J$$

$$= u_{J-1} - u_J + C_1 \tau_{J-2} + C_2 \tau_{J-1} + C_3 \tau_J + C_4 \tau_{J+1}$$

$$\delta(\nu\tau)_J - \delta(\nu\tau)_{J-1} - a_1 \delta F_J - a_2 \delta F_{J-1}$$

$$= (\nu\tau)_{J-1} - (\nu\tau)_J + C_1 F_{J-2} + C_2 F_{J-1}$$

$$+ C_3 F_J + C_4 F_J$$

$$\delta g_J - \delta g_{J-1} - (C_1 + C_2)\delta q_{J-1} - (C_3 + C_4)\delta q_J$$

$$= g_{J-1} - g_J + C_1 q_{J-2} + C_2 q_{J-1} + C_3 q_J + C_4 q_{J+1}$$

$$\frac{1}{Pr}(\delta q_J - \delta q_{J-1}) - a_3 \delta G_J - a_4 \delta G_{J-1}$$

$$= \frac{1}{Pr}(q_{J-1} - q_J) + C_1 G_{J-2} + C_2 G_{J-1} + C_3 G_J + C_4 G_{J+1}$$

The equations have the same form as the first-order method except the elements of the left-hand side matrix and the r vector are different. This needs the modification in Routines RHS1 and RLHS in the program. After the modification a test case is solved and the results are presented in Table 1.

The coefficients b_1, b_2, b_3, etc., are found for two cases.

a) uniform mesh

$$b_1 = \frac{9}{24} h$$

$$b_2 = \frac{19}{24} h$$

$$b_3 = -\frac{5}{24} h$$

$$b_4 = \frac{1}{24} h$$

and

$$C_1(J) = -\frac{1}{24}\,h$$

$$C_2(J) = \frac{13}{24}\,h$$

$$C_3(J) = \frac{13}{24}\,h$$

$$C_4(J) = -\frac{1}{24}\,h$$

b) non-uniform mesh

for J=2

$$b_1 = \frac{h_2}{12H_1H_3}\left[(H_1+h_3)(H_3+H_3) + 2H_1H_3\right]$$

$$b_2 = \frac{h_2}{12H_2h_3}\left[(H_3+2H_2)(H_1+h_3) - H_2h_2\right]$$

$$b_3 = -\frac{h_2^3}{12H_1h_3h_4}\,(h_2+2H_2)$$

$$b_4 = \frac{h_2^3}{12H_2H_3h_4}\,(H_1+h_3)$$

where

$$H_1 = h_2 + h_3$$

$$H_2 = h_3 + h_4$$

$$H_3 = h_1 + h_2 + h_3$$

for $3 \leq J < N$

$$C_1(J) = -\frac{h_J^3(H_1+h_{j-1})}{12H_2h_{J-1}H_3}$$

$$C_2(J) = \frac{h_J}{12H_1h_{J-1}}\left[(H_1+h_{J+1})(H_2+h_{J-1}) + 2H_1h_{J-1}\right]$$

$$C_3(J) = \frac{h_J}{12H_2h_{J+1}}\left[(H_1+h_{J+1})(H_2+h_{J-1}) + 2H_2h_{J+1}\right]$$

$$C_4(J) = -\frac{h_J^3}{12H_1H_3h_{J+1}}\,(H_2+h_{J-1})$$

where

$$H_1(J) = h_{J+1} + h_J$$

$$H_2(J) = h_J + h_{J-1}$$

$$H_3(J) = h_{J+1} + h_J + h_{J-1}$$

Alternative Scheme Using Euler-Maclurin Formula

If we use the two-point Hermite interpolation instead of the four-point formula to approximate the integrand, we obtain the Euler-Maclurin formula

$$\int_{\eta_{J-1}}^{\eta_J} F\, d\eta = \frac{h_J}{2}(F_J + F_{J-1}) + \frac{h_J^2}{12}(F'_{J-1} - F'_J) + 0(h_J^5 g'')$$

where

$$F' = \frac{\partial F}{\partial \eta}$$

If we use this scheme, the equations reduce to

$$f_J - f_{J-1} = \frac{h_J}{2}(u_J + u_{J-1}) + \frac{h_J^2}{12}(\tau_{J-1} - \tau_J)$$

$$u_J - u_{J-1} = \frac{h_J}{2}(\tau_J + \tau_{J-1}) + \frac{h_J^2}{12}(\tau'_{J-1} - \tau'_J)$$

$$(\nu\tau)_J - (\nu\tau)_{J-1} = \frac{h_J}{2}(F_J + F_{J-1}) + \frac{h_J^2}{12}(F'_{J-1} - F'_J)$$

$$g_J - g_{J-1} = \frac{h_J}{2}(q_J + q_{J-1}) + \frac{h_J^2}{12}(G_{J-1} - G_J)$$

$$q_J - q_{J-1} = \frac{Prh_J}{2}(G_J + G_{J-1}) + \frac{h_J^2}{12}(G'_{J-1} - G'_J)$$

After linearization and omission of $0(h^2)\delta$ terms as mentioned earlier we get

$$\delta f_J - \delta f_{J-1} - \frac{h_J}{2}\,\delta u_J - h_J \delta u_{J-1}$$

$$= f_{J-1} - f_J + \frac{h_J}{2}\,(u_J + u_{J-1}) + \frac{h_J^2}{12}\,(\tau_{J-1} - \tau_J)$$

$$\delta u_J - \delta u_{J-1} - \frac{h_J}{2}\,\delta \tau_J - \frac{h_J^2}{12}\,\delta \tau_{J-1}$$

$$= u_{J-1} - u_J + \frac{h_J}{2}\,\tau_J + \frac{h_J}{2}\,\tau_{J-1} - \frac{h_J^2}{12}\,(\tau'_J - \tau'_{J-1})$$

$$\delta(\nu\tau)_J - \delta(\nu\tau)_{J-1} - \frac{h_J}{2}\,\delta F_J - \frac{h_J}{2}\,\delta F_{J-1}$$

$$= (\nu\tau)_{J-1} - (\nu\tau)_J + \frac{h_J}{2}\,(F_J + F_{J-1}) - \frac{h_J^2}{12}\,(F'_J - F'_{J-1})$$

τ' appearing in the second equation and implicitly in F' can be deduced from $(\nu\tau)_\eta = F$ and given by

$$\tau_\eta = (F - \nu' q \tau)/\nu$$

where ν is given by function VISC(T) and ν' is given by DVDT(T) in the program.

APPENDIX 4. FORTRAN LISTING OF COMPUTER CODE

MAIN

```
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION X(2),TW(2),RL3(100),RL5(100),Y(100),Y1(100)
      DIMENSION C(100,4)
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/CONSIP/AL1P,AL2P,BTA2P,BTA3P,BTA4P,THP
      COMMON/GUES/Y
      COMMON/TWPR/TW,PR
      COMMON/CALCF/Y1,C
      COMMON/RHS/RL3,RL5
      COMMON/CONS/AL1,AL2,BTA2,BTA3,BTA4,TH
      LOGICAL CONST
      COMMON/MESH/ CONST
  101 FORMAT (5X,'X =',F7.4,'  Y =',F7.4,'  F =',F11.6,'  U =',F11.6,
     . '  T =',F11.6,'  G =',F11.6,'  Q =',F11.6)
  102 FORMAT(/,5X,'STEP SIZE ITERATION=  ',I2,' DECOMP ITERATION=',I2,/)
      PR=1.
      KY=90
      KP=KY-1
      XK=.025
      X3=1.
      X(2)=0.
      B=0.3411409
      X(1)=0.0
      HJ=.1
      DO 3 J=1,KY
      H(J)=HJ
    3 CONTINUE
      CONST=.TRUE.
      IP=1
      TW(1)=1.
      TW(2)=1.
      TWP=0.
      UEP=0.
      FW=0.
      UE=1.
      CALL GUESS(KY,1)
      DO 4 J=1,KY
      Y1(J)=Y(J)
    4 CONTINUE
      IX=1
      AL1=1.
      AL2=0.
      IST=-1
      TH=0.
  333 CONTINUE
      ICT=0
```

```
      IF(IX.EQ.2) AL2=2.*X(2)/XK-1.
      AL2P=AL2
      TH=2.*X(IX)*IWP(IX)/IW(IX)
      BTA2=AL1+AL2
      BTA3=BTA2+AL1
      BTA2=1.+BTA2
      BTA4=TH+AL2
      G(1,IX)=(IW(IX)-1.)/IW(IX)
      F(1,IX)=FW
      CALL ELAPSE
      CALL DECOMP(KY,IX,ICT)
      CALL RLHS(KY,IX,IP)
      CALL ELAPSE
      WRITE (6,102) IST,ICT
      IF (IX .EQ. 1)  IP = 2
      DO 311 J=1,KY
      F(J,IP)=F(J,IX)
      U(J,IP)=U(J,IX)
      T(J,IP)=T(J,IX)
      G(J,IP)=G(J,IX)
      Q(J,IP)=Q(J,IX)
  311 CONTINUE
      IP=1
      DO 2 I=1,KY,KR
      WRITE(6,101) X(IX),Y1(I),F(I,1), U(I,1),T(I,1),G(I,1),Q(I,1)
    2 CONTINUE
      X(1)=X(2)
      IX=2
      AL1P=AL1
      BTA2P=BTA2
      BTA3P=BTA3
      BTA4P=BTA4
      THP=TH
      X(2)=X(1)+XK
      IF(X(1).GE.X3) STOP
      X1=X(2)**4*B
      UEP=1.-5.*X1
      UE=1.-X1
      AL1=UEP/UE
      GOTO 333
      END
```

```
      SUBROUTINE GUESS(KY,I)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION Y(100)
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/GUES/Y
      Y2=3.5*3.5
      Y1=3.5
C        BOUNDARY CONDITION
      F(1,I)=0.0
      U(1,I)=0.
      T(1,I)=0.5
      G(1,I)=1.
      Q(1,I)=0.5
      Y(1)=0.
      DO 300 J=2,KY
      Y(J)=Y(J-1)+H(J)
      YY=Y(J)
      IF(Y(J).GT.3.5) GOTO 190
      F(J,I)=YY**3/Y2*(1.-.5*YY/Y1)
      U(J,I)=YY**2/Y2*(3.-2.*YY/Y1)
      T(J,I)=6.*YY/Y2*(1.-YY/Y1)
      G(J,I)=1.-U(J,I)
      Q(J,I)=-T(J,I)
      GOTO 300
  190 CONTINUE
      F(J,I)=F(J-1,I)+H(J)
      U(J,I)=1.
      T(J,I)=0.
      G(J,I)=1.-U(J,I)
      Q(J,I)=-T(J,I)
  300 CONTINUE
C        INITIAL ESTIMATE COMPLETED
      RETURN
      END
```

```
      SUBROUTINE DECOMP(KY,IX,ICT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,8),H(100)
      DIMENSION R1(100),R2(100),R3(100),R4(100),R5(100)
      DIMENSION S1(100),S2(100),S3(100),S4(100),S5(100),S6(100),S7(100)
      DIMENSION S8(100),S9(100),S10(100),S11(100)
      DIMENSION A11(100),A12(100),A13(100),A14(100),A15(100)
      DIMENSION A21(100),A22(100),A23(100),A24(100),A25(100)
      DIMENSION A31(100),A32(100),A33(100),A34(100),A35(100)
      DIMENSION B11(100),B12(100),B13(100),B14(100),B15(100)
      DIMENSION B21(100),B22(100),B23(100),B24(100),B25(100)
      DIMENSION B31(100),B32(100),B33(100),B34(100),B35(100)
      DIMENSION W1(100),W2(100),W3(100),W4(100),W5(100)
      DIMENSION DF(100),DU(100),DT(100),DG(100),DQ(100)
      DIMENSION P(5),D(5)
      COMMON/CSOLVE/ A11,A13,A15,A21,A23,A25,A31,A33,A35,D,P
      COMMON/RHL/R1,R2,R3,R4,R5
      COMMON/COMP/S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11
      COMMON/ALL/F,U,T,G,Q,H
      LOGICAL SOLV
      COMMON/SOL/SOLV
      SOLV=.TRUE.
C        L-U DECOMPOSITION STARTS
      IP=1
      ICT=0
      CALL RHS1(KY,IX,IP)
  200 CONTINUE
      CALL RLHS(KY,IX,IP)
      K=KY+1
      H(K)=0.
      DU(K)=0.
      DT(K)=0.
      DG(K)=0.
      DQ(K)=0.
C        INITIAL VALUES FROM HAND COMPUTATION
      A11(1)=1.
      A22(1)=1.
      A34(1)=1.
      A12(1)=0.0
      A13(1)=0.0
      A14(1)=0.0
      A15(1)=0.0
      A21(1)=0.0
      A23(1)=0.0
      A24(1)=0.0
      A25(1)=0.0
```

```
      A31(1)=0.0
      A32(1)=0.0
      A33(1)=0.0
      A35(1)=0.0
      B11(2)=-1.
      B12(2)=-H(2)/2.
      B13(2)=0.
      B14(2)=0.
      B15(2)=0.
      B21(2)=0.
      B22(2)=-S5(1)*2./H(2)
      B23(2)=0.
      B24(2)=-S5(1)*2./H(2)
      B25(2)=0.
      B31(2)=0.
      B32(2)=0.
      B33(2)=      -S10(1)*2./H(2)
      B34(2)=0.
      B35(2)=-S10(1)*2./H(2)
C        INITIAL VALUES FROM HAND COMPUTATION OVER
      DO 306 J=2,KY
      HJ=-H(J)/2.
      JP=J-1
C     SINCE AT J=2 GAMMA MATRIX IS KNOWN FOLLWING STEP IS SKIPPED FOR 2
      IF(J.EQ.2) GOTO 210
C        GAUSSIAN ELIMINATION OF DELTA MATRIX STARTS
      B1=A13(JP)+HJ*A12(JP)
      B2=A23(JP)+HJ*A22(JP)
      B3=A33(JP)+HJ*A32(JP)
      C1=A15(JP)+HJ*A14(JP)
      C2=A25(JP)+HJ*A24(JP)
      C3=A35(JP)+HJ*A34(JP)
      D2=B2-A21(JP)*B1/A11(JP)
      D3=B3-A31(JP)*B1/A11(JP)
      E2=C2-A21(JP)*C1/A11(JP)
      E3=C3-A31(JP)*C1/A11(JP)  -D3*E2/D2
C        GAUSSIAN ELIMINATION OVER
C        USING THE SAME UTM THE EQNS ARE SOLVED FOR DIFFERANT ROWS
      D(1)=-1.
      D(2)=HJ
      D(3)=0.
      D(4)=0.
      D(5)=0.
      CALL SOLVE(B1,D2,D3,E2,E3,HJ,JP,C1)
      B11(J)=D(1)
      B12(J)=D(2)
      B15(J)=D(5)
      B13(J)=D(3)
      B14(J)=D(4)
      D(1)=S2(JP)
      D(2)=S3(JP)
      D(3)=S5(JP)
      D(4)=S11(JP)
      D(5)=0.
      CALL SOLVE(B1,D2,D3,E2,E3,HJ,JP,C1)
      B21(J)=D(1)
      B22(J)=D(2)
      B23(J)=D(3)
```

```
      B24(J)=D(4)
      B25(J)=D(5)
      D(1)=S7(JP)
      D(2)=S8(JP)
      D(3)=0.
      D(4)=S9(JP)
      D(5)=S10(JP)
      CALL SOLVE(B1,D2,D3,E2,E3,HJ,JP,C1)
      B31(J)=D(1)
      B32(J)=D(2)
      B33(J)=D(3)
      B34(J)=D(4)
      B35(J)=D(5)
  210 CONTINUE
C        GAMMA MATRIX AT NEXT STATION COMPUTATION OVER
C        DELTA MATRIX COMPUTATION STARTS
      A11(J)=1.
      A12(J)=HJ-B14(J)
      A13(J)=-B14(J)*HJ
      A14(J)=-B15(J)
      A15(J)=-B15(J)*HJ
      A21(J)=S2(J)
      A22(J)=S3(J)-B24(J)
C
      A23(J)=S4(J)-B24(J) *HJ
      A24(J)=S1(J)-B25(J)
      A25(J)=     -B25(J)*HJ
      A31(J)=S7(J)
      A32(J)=S8(J)-B34(J)
      A33(J)=     -B34(J)*HJ
      A34(J)=S9(J)-B35(J)
      A35(J)=-HJ*B35(J)+S6(J)
C        DELTA MATRIX COMPUTATION AT NEXT PT COMPLETED
C        RETURN TO GAMMA MATRIX COMPUTATION
  306 CONTINUE
C        INITIAL VALUES OF W MATIX
      W1(1)=R1(1)
      W2(1)=R3(1)
      W3(1)=R5(1)
      W4(1)=R2(1)
      W5(1)=R4(1)
C        COMPUTATION OF W MATRIXSTARTS
      DO 307 J=2,KY
      JP=J-1
      W1(J)=R1(J)-B11(J)*W1(JP)-B12(J)*W2(JP)-B13(J)*W3(JP)-B14(J)*W4(JP
     1)-B15(J)*W5(JP)
      W2(J)=R3(J)-B21(J)*W1(JP)-B22(J)*W2(JP)-B23(J)*W3(JP)-B24(J)*W4(JP
     1)-B25(J)*W5(JP)
      W3(J)=R5(J)-B31(J)*W1(JP)-B32(J)*W2(JP)-B33(J)*W3(JP)-B34(J)*W4(JP
     1)-B35(J)*W5(JP)
      W4(J)=R2(J)
      W5(J)=R4(J)
  307 CONTINUE
C        W MATRIX COMPUTATION OVER
C        SOLVING FOR DF,DU,.....   BEGINS
C        GAUSSIAN ELIMINATION
      DO 308 JJ=1,KY
      J=KY+1-JJ
```

```
      JP=J+1
      HJ=H(J+1)/2.
      B1=A21(J)/A11(J)
      B2=A22(J)-B1*A12(J)
      B3=A23(J)-B1*A13(J)
      B4=A24(J)-B1*A14(J)
      B5=A25(J)-B1*A15(J)
      C1=A31(J)/A11(J)
      C2=A32(J)-C1*A12(J)
      C3=A33(J)-C1*A13(J)
      C4=A34(J)-C1*A14(J)
      C5=A35(J)-C1*A15(J)
      D1=C2/B2
      D3=C3-D1*B3
      D4=C4-D1*B4
      D5=C5-D1*B5
      WB2=W2(J)-B1*W1(J)
      WC3=W3(J)-C1*W1(J)
      WD3=WC3   -D1*WB2
      W4(J)=W4(J)-(DU(JP)-HJ*DT(JP))
      W5(J)=W5(J)-(DG(JP)-HJ*DQ(JP))
      E1=B3-B2*HJ
      E2=B5-B4*HJ
C
      E3=D3
      E4=D5-D4*HJ
      V1=WB2+B4*W5(J)+B2*W4(J)
      V2=WD3+W5(J)*D4
C        CALCULATION OF DF,DU,.....
      DQ(J)=(V1*E3-V2*E1)/(E3*E2-E1*E4)
C
      DT(J)=(V1   -E2*DQ(J))/E1
      DU(J)=-W4(J)-HJ*DT(J)
      DG(J)=-W5(J)-HJ*DQ(J)
      DF(J)=(W1(J)-A12(J)*DU(J)-A15(J)*DQ(J)-A13(J)*DT(J)-A14(J)*DG(J))/
     1A11(J)
C        CALCULATION OF DF,DU,..... OVER AT J,J+1 STARTS
  308 CONTINUE
      DO 309 J=1,KY
      F(J,IX)=F(J,IX)+DF(J)
      U(J,IX)=U(J,IX)+DU(J)
      T(J,IX)=T(J,IX)+DT(J)
      G(J,IX)=G(J,IX)+DG(J)
      Q(J,IX)=Q(J,IX)+DQ(J)
  309 CONTINUE
      EPS=.00000000001
C        F,U,T,G,Q,....UPDATED     NEXT ITERATION STARTS
      ICT=ICT+1
      IF(ICT.GE.10) GOTO 220
      IF(DABS(DT(1)).GT.EPS.OR .DABS(DQ(1)).GT.EPS) GOTO 200
  220 CONTINUE
      IST=0
      SOLV=.FALSE.
      RETURN
      END
```

```
      SUBROUTINE SOLVE(B1,D2,D3,E2,E3,HJ,JP,C1)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A11(100),A13(100),A15(100),A21(100),A23(100),A25(100),
     1A31(100),A33(100),A35(100),D(5),P(5)
      COMMON/CSOLVE/ A11,A13,A15,A21,A23,A25,A31,A33,A35,D,P
      P(1)=D(1)
      P(3)=D(3)
      P(5)=D(5)
      P(2)=D(3)+HJ*D(2)-D(1)*B1/A11(JP)
      P(4)=D(5)+HJ*D(4)-D(1)*C1/A11(JP) -P(2)*E2/D2
      D(3)=P(4)/E3
      D(2)=(P(2)-D3*D(3))/D2
      D(1)=(P(1)-A21(JP)*D(2)-A31(JP)*D(3))/A11(JP)
      D(4)=(P(3)-A13(JP)*D(1)-A23(JP)*D(2)-A33(JP)*D(3))/HJ
      D(5)=(P(5)-A15(JP)*D(1)-A25(JP)*D(2)-A35(JP)*D(3))/HJ
      RETURN
      END
      DOUBLE PRECISION FUNCTION VISC(T)
      IMPLICIT REAL*8(A-H,O-Z)
      VISC=1.
      VISC=2.
      RETURN
      END
      DOUBLE PRECISION FUNCTION DVDT(T)
      IMPLICIT REAL*8(A-H,O-Z)
      DVDT=0.
      RETURN
      END
```

```
      SUBROUTINE STEPSZ(HRU,HRL,DET,DEQ,MP,IQ,IST,IX)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2) ,H(100)
      DIMENSION X(100),B(100)
      COMMON/GUES/X
      COMMON/ALL/F,U,T,G,Q,H
      N=KY
      NM=MP-1
      IP=1
      I=2
      IQ=0
      B(1)=X(1)
      DO 306 J=2,NM
      HJ=H(J)
      IF(HJ.LT.H(J+1)) HJ=H(J+1)
      HJ=HJ*HJ
      DEL=HJ*(DABS(((T(J+1,IX)-T(J,IX))/H(J+1)-(T(J,IX)-T(J-1,IX))/H(J))
     1/(H(J)+H(J+1))))
      IF(DEL.LE.DEI) GOTO 260
      IP=2
      I=I+IP
      IQ=IQ+1
      B(I-2)=(X(J)+X(J-1))/2.
      B(I-1)=X(J)
      B(I)=(X(J)+X(J+1))/2.
      GOTO 270
  260 CONTINUE
      I=I+IP
      IP=1
      B(I-1)=X(J)
  270 CONTINUE
  306 CONTINUE
      MP=I
      IF(IP.EQ.2) MP=I+1
      B(MP)=X(NM+1)
      DO 307 J=1,MP
      X(J)=B(J)
      IF(J.NE.1 ) H(J)=B(J)-B(J-1)
  307 CONTINUE
  400 CONTINUE
      NM=MP-1
      ICT=IC
      IC=0
      I=2
      B(2)=H(2)
      IG=0
      DO 308 J=3,MP
      I=I+1
      HR=H(J)/H(J-1)
```

```
      IF(HR.GT.HRU) GOTO 410
      IF(HR.GE.HRL) GOTO 420
      I=I+1
      IF(IG.EQ.1) GOTO420
      B(I-1)=H(J-1)/2.
      B(I-2)=H(J-1)/2.
      IC=IC+1
      GOTO 420
410   CONTINUE
      B(I)=H(J)/2.
      B(I+1)=H(J)/2.
      I=I+1
      IC=IC+1
      IG=1
      GOTO 430
420   CONTINUE
      B(I)=H(J)
      IG=0
430   CONTINUE
308   CONTINUE
      MP=I
       X(1)=0.
      DO 309 J=2,MP
      H(J)=B(J)
      X(J)=X(J-1)+H(J)
309   CONTINUE
      IF (IC.NE.0) GOTO 400
      IQ=IQ+IC+ICT
      IST=IST+1
      IF(IST.EQ.20) GOTO 440
      IF(IQ.NE.0) GOTO 200
440   CONTINUE
      IQ=0
200   CONTINUE
      RETURN
      END
```

```
      SUB ROUTINE FMIDPT(N,KY,IX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION Y1(100),Y (100),QD(100),TD(100),C(100,4)
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/CALCF/Y1,C
      COMMON/GUES/Y
C.....ON FIRST CALL N=KY OF PREVIOUS STATION X
C.....KY CURRENT VALUE
      NP=N-1
      DO 12 I=1,IX
      IM=I
      DO 1 J=2,NP
      HJ=1./(Y1(J+1)-Y1(J-1))
      TD(J)=(T(J+1,I)-T(J-1,I))/HJ
      QD(J)=(Q(J+1,I)-Q(J-1,I))/HJ
    1 CONTINUE
      H1=Y1(2)-Y1(1)
      TD(1)=(T(2,I)-T(1,I))/H1
      QD(1)=(Q(2,I)-Q(1,I))/H1
      HN=Y1(N)-Y1(NP)
      TD(N)=(T(N,I)-T(NP,I))/HN
      QD(N)=(Q(N,I)-Q(NP,I))/HN
      DO 2 J=1,N
      C(J,1)=F(J,I)
      C(J,2)=U(J,I)
    2 CONTINUE
      CALL CALCCF(NP,IM)
      DO 3 J=1,KY
      YB=Y(J)
      F(J,I)=PCUBIC(YB,NP,IM)
    3 CONTINUE
      DO 4 J=1,N
      C(J,1)=U(J,I)
      C(J,2)=T(J,I)
    4 CONTINUE
      CALL CALCCF(NP,IM)
      DO 5 J=1,KY
      YB=Y(J)
      U(J,I)=PCUBIC(YB,NP,IM)
    5 CONTINUE
      DO 6 J=1,N
      C(J,1)=T(J,I)
      C(J,2)=TD(J)
    6 CONTINUE
      CALL CALCCF(NP,IM)
      DO 7 J=1,KY
      YB=Y(J)
      T(J,I)=PCUBIC(YB,NP,IM)
    7 CONTINUE
      DO 8 J=1,N
```

```
      C(J,1)=G(J,I)
      C(J,2)=Q(J,I)
    8 CONTINUE
      CALL CALCCF(NP,IM)
      DO 9 J=1,KY
      YB=Y(J)
      G(J,I)=PCUBIC(YB,NP,IM)
    9 CONTINUE
      DO10 J=1,N
      C(J,1)=Q(J,I)
      C(J,2)=QD(J)
   10 CONTINUE
      CALL CALCCF(NP,IM)
      DO11 J=1,KY
      YB=Y(J)
      Q(J,I)=PCUBIC(YB,NP,IM)
   11 CONTINUE
   12 CONTINUE
      DO13 J=1,KY
      Y1(J)=Y(J)
   13 CONTINUE
      N=KY
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION PCUBIC(Y,N,IM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y1(100),C(100,4)
      COMMON/CALCF/Y1,C
      DATA I/1/
      H=Y-Y1(I)
      IF (H)              10,30,20
   10 IF(I.EQ.1)          GOTO 30
      I=I-1
      H=Y-Y1(I)
      IF(H)               10,30,30
   19 I=I+1
      H=HY
   20 IF(I.EQ.N)          GOTO 30
      HY=Y-Y1(I+1)
      IF(HY)              30,19,19
   30 PCUBIC=C(I,1)+H*(C(I,2)+H*(C(I,3)+H*C(I,4)))
      RETURN
      END
```

```
      SUBROUTINE CALCCF(N,IM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y1(100),C(100,4)
      COMMON/CALCF/Y1,C
      DO 10 I=1,N
      H=Y1(I+1)-Y1(I)
      DF1=(C(I+1,1)-C(I,1))/H
      DF3=C(I,2)+C(I+1,2)-2.*DF1
      C(I,3)=(DF1-C(I,2)-DF3)/H
   10 C(I,4)=DF3/H/H
      RETURN
      END
```

```
      SUBROUTINE RLHS(KY,IX,IP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION R1(100),R2(100),R3(100),R4(100),R5(100)
      DIMENSION S1(100),S2(100),S3(100),S4(100),S5(100),S6(100),S7(100)
      DIMENSION S8(100),S9(100),S10(100),S11(100)
      DIMENSION RL3(100),RL5(100),TW(2)
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/RHL/R1,R2,R3,R4,R5
      COMMON/COMP/S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11
      COMMON/RHS/RL3,RL5
      COMMON/TWPR/TW,PR
      COMMON/CONS/AL1,AL2,BTA2,BTA3,BTA4,TH
      TX=1.+(TW(IX)-1.)*G(J,IX)
      TP=1.+(TW(IX)-1.)*G(JP,IX)
      DO 305 J=2,KY
      JP=J-1
      HJ=H(J)/2.
      PRH=1./(PR*HJ)
C        VECTOR R UPDATED ON EVERY ITERATION
      R1(J)=F(JP,IX)-F(J,IX)+HJ*(U(J,IX)+U(JP,IX))
      R2(JP)=U(JP,IX)-U(J,IX)+HJ*(T(J,IX)+T(JP,IX))
      R31=BTA2*(T(J,IX)*F(J,IX)+T(JP,IX)*F(JP,IX))
      R32=BTA3*(U(J,IX)*U(J,IX)+U(JP,IX)*U(JP,IX))
      R33=AL2*(T(J,IP)*F(J,IX)+T(JP,IP)*F(JP,IX)-T(J,IX)*F(J,IP)-T(JP,IX
     1)*F(JP,IP))
      R34=(VISC(TX)*T(J,IX)-VISC(TP)*T(JP,IX))/HJ
      R3(J)=RL3(J)-R31+R32-R33-R34-4*AL1
      R4(JP)=G(JP,IX)-G(J,IX)+HJ*(Q(J,IX)+Q(JP,IX))
      R51=(Q(J,IX)-Q(JP,IX))*PRH
      R52=BTA2*(Q(J,IX)*F(J,IX)+Q(JP,IX)*F(JP,IX))
      R53=BTA4*(U(J,IX)*G(J,IX)+U(JP,IX)*G(JP,IX))
      R54=AL2*(U(J,IP)*G(J,IX)+U(JP,IP)*G(JP,IX)-G(J,IP)*U(J,IX)-G(JP,IP
     1)*U(JP,IX)-Q(J,IP)*F(J,IX)-Q(JP,IP)*F(JP,IX)+Q(J,IX)*F(J,IP)+Q(JP,
     2IX)*F(JP,IP))
      R5(J)=RL5(J)-R51-R52+R53+R54
C        VECTOR COMPUTATION OVER
C        COMPUTATION OF S VECTOR
      S1(J)=DVDT(TX)*T(J,IX)/HJ
      S2(J)=BTA2*T(J,IX)+AL2*T(J,IP)
      S3(J)=-2.*BTA3*U(J,IX)
      S4(J)=VISC(TX)/HJ+BTA2*F(J,IX)-AL2*F(J,IP)
      S5(JP)=-VISC(TP)/HJ+BTA2*F(JP,IX)-AL2*F(JP,IP)
      S6(J)=PRH+BTA2*F(J,IX)-AL2*F(J,IP)
      S7(J)=BTA2*Q(J,IX)+AL2*Q(J,IP)
      S8(J)=AL2*G(J,IP)-BTA4*G(J,IX)
      S9(J)=-BTA4*U(J,IX)-AL2*U(J,IP)
```

```
      S10(JP)=B1A2*F(JP,IX)-AL2*F(JP,IP)-PRH
      S11(JP)=-DVD↑(TP)*T(JP,IX)/HJ
  305 CONTINUE
C        VECTOR COMPUTATION OVER
C        R VECTOR SPECIFIED AT BOUNDARY
      R1(1)=0.
      R3(1)=0.
      R5(1)=0.
      R2(KY)=0.
      R4(KY)=0.
      RETURN
      END
```

```
      SUBROUTINE RHS1(KY,IX,IP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION RL3(100),RL5(100),TW(2)
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/TWPR/TW,PR
      COMMON/CONSIP/AL1,AL2,BTA2,BTA3,BTA4,TH
      COMMON/RHS/RL3,RL5
      IP=1.+(TW(IP)-1.)*G(JP,IP)
      TJ=1.+(TW(IP)-1.)*G(J,IP)
      DO 303 J=2,KY
C        COMPUTATION OF R VECTOR
      RL3(J)=0.
  303 RL5(J)=0.
      III=0
      IF(IX.EQ.1) GOTO 200
      DO 304 J=2,KY
      JP=J-1
      HJ=H(J)
      PRH=2./(PR*HJ)
      RL3(J)=2.*AL1     *(1.-(U(J,IP)**2)+1.-(U(JP,IP)**2))
      RL3(J)=RL3(J)+(1.+AL1     )*(F(J,IP)*T(J,IP)+F(JP,IP)*T(JP,IP))
      RL3(J)=RL3(J)+(VISC(TJ)*T(J,IP)-VISC(TP)*T(JP,IP))*2./HJ
      RL3(J)=AL2     *((T(J,IP)*F(J,IP)+T(JP,IP)*F(JP,IP))-(U(J,IP)*U(J,I
     1P)+U(JP,IP)*U(JP,IP)))-RL3(J)
      RL5(J)=(Q(J,IP)-Q(JP,IP))*PRH+(1+AL1     )*(F(J,IP)*Q(J,IP)+F(JP,IP
     1)*Q(JP,IP))- TH     *(U(J,IP)*G(J,IP)+U(JP,IP)*G(JP,IP))
      RL5(J)=AL2     *(Q(J,IP)*F(J,IP)+Q(JP,IP)*F(JP,IP)-U(J,IP)*G(J,IP)-
     2U(JP,IP)*G(JP,IP))-RL5(J)
  304 CONTINUE
C        ITERATION FOR EATA BEGINS
  200 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RLHS(KY,IX,IP)
C     THIS SUBROUTINE CALCULATES  THE MATRIX AND VECTOR
C     ELEMENTS FOR FOUR POINT SCHEME
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION S1(100),S2(100),S3(100),S4(100),S5(100),S6(100),S7(100)
      DIMENSION S8(100),S9(100),S10(100),S11(100)
      DIMENSION R1(100),R2(100),R3(100),R4(100),R5(100),TW(2)
      DIMENSION RFL(100),RGL(100),RFP(100),RGP(100),RF(100),RG(100)
      COMMON /RHS/ RFL,RFP,RGL,RGP,RF,RG
      DIMENSION C1(100),C2(100),C3(100),C4(100)
      COMMON/COEFF/C1,C2,C3,C4,B1,B2,B3,B4
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/RHL/R1,R2,R3,R4,R5
      COMMON/COMP/S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11
      COMMON/TWPR/TW,PR
      COMMON/CONS/AL1,AL2,BTA2,BTA3,BTA4,TH
      LOGICAL SOLV
      COMMON/SOL/ SOLV
      TX=1.+TW(IX)*G(J,IX)
      TP=1.+TW(IX)*G(JP,IX)
      DO10 J=1,KY
      RF(J)=BTA3*U(J,IX)**2-BTA2*F(J,IX)*T(J,IX)+AL2*(F(J,IP)*T(J,IX)-
     1T(J,IP)*F(J,IX))+RFL(J)-2.*AL1
      RG(J)=BTA4*U(J,IX)*G(J,IX)-BTA2*Q(J,IX)*F(J,IX)+AL2*(U(J,IP)*G(J,IX)
     1X)+G(J,IP)*U(J,IX)-Q(J,IP)*F(J,IX)+F(J,IP)*Q(J,IX))+RGL(J)
 10   CONTINUE
      IF(.NOT.SOLV) RETURN
      KP=KY-1
      DO 20 J=3,KP
      J2=J-2
      JP=J-1
      J1=J+1
      R1(J)=F(JP,IX)-F(J,IX)+C1(J)*U(J2,IX)+C2(J)*U(JP,IX)+C3(J)*U(J,IX)
     1+C4(J)*U(J1,IX)
      R2(JP)=U(JP,IX)-U(J,IX)+C1(J)*T(J2 ,IX)+C2(J)*T(JP,IX)+C3(J)*T(J,I
     1X)+C4(J)*T(J1,IX)
      R3(J)=VISC(TP)*T(JP,IX)-VISC(TX)*T(J,IX)+C1(J)*RF(J2)+C2(J)*RF(JP)
     1+C3(J)*RF(J)+C4(J)*RF(J1)
      R4(JP)=G(JP,IX)-G(J,IX)+C1(J)*Q(J2,IX)+C2(J )*Q(JP,IX)+C3(J)*Q(J,I
     1X)+C4(J)*Q(J1,IX)
      R5(J)=(Q(JP,IX)-Q(J,IX))/PR+C1(J)*RG(J2)+C2(J)*RG(JP)+C3(J)*RG(J)+
     1C4(J)*RG(J1)
 20   CONTINUE
      R1(2)=F(1,IX)-F(2,IX)+B1*U(1,IX)+B2*U(2,IX)+B3*U(3,IX)+B4*U(4,IX)
      R2(1)=U(1,IX)-U(2,IX)+B1*T(1,IX)+B2*T(2,IX)+B3*T(3,IX)+B4*T(4,IX)
      J=2
```

```
      JP=1
      R3(J)=VISC(TP)*T(JP,IX)-VISC(TX)*T(J,IX)+B1*RF(1)+B2*RF(2)+B3*RF(3
     1)+B4*RF(4)
      R4(1)=G(1,IX)-G(2,IX)+B1*Q(1,IX)+B2*Q(2,IX)+B3*Q(3,IX)+B4*Q(4,IX)
      R5(2)=(Q(1,IX)-Q(2,IX))/PR+B1*RG(1)+B2*RG(2)+B3*RG(3)+B4*RG(4)
      R1(KY)=F(KP,IX)-F(KY,IX)+(U(KP,IX)+U(KY,IX))*H(KY)/2.
      R2(KP)=U(KP,IX)-U(KY,IX)+(T(KP,IX)+T(KY,IX))*H(KY)/2.
      J=KY
      JP=KP
      R3(KY)=VISC(TP)*T(JP,IX)-VISC(TX)*T(J,IX)+(RF(KP)+RF(KY))*H(KY)/2.
      R4(KP)=G(KP,IX)-G(KY,IX)+(Q(KP,IX)+Q(KY,IX))*H(KY)/2.
      R5(KY)=(Q(KP,IX)-Q(KY,IX))/PR+(RG(KP)+RG(KY))*H(KY)/4.
      DO 30 J=1,KY
      HJ=H(J)/2.
      R3(J)=R3(J)/HJ
      R5(J)=R5(J)/HJ
   30 CONTINUE
C        COMPUTATION OF S VECTOR
      DO 40 J=2,KY
      HJ=H(J)/2.
      JP=J-1
      PRH=1./(PR*HJ)
      S1(J)=DVDT(TX)*T(J,IX)/HJ
      S2(J)=BTA2*T(J,IX)+AL2*T(J,IP)
      S3(J)=-2.*BTA3*U(J,IX)
      S4(J)=VISC(TX)/HJ+BTA2*F(J,IX)-AL2*F(J,IP)
      S5(JP)=-VISC(TP)/HJ+BTA2*F(JP,IX)-AL2*F(JP,IP)
      S6(J)=PRH+BTA2*F(J,IX)-AL2*F(J,IP)
      S7(J)=BTA2*Q(J,IX)+AL2*Q(J,IP)
      S8(J)=AL2*G(J,IP)-BTA4*G(J,IX)
      S9(J)=-BTA4*U(J,IX)-AL2*U(J,IP)
      S10(JP)=BTA2*F(JP,IX)-AL2*F(JP,IP)-PRH
      S11(JP)=-DVDT(TP)*T(JP,IX)/HJ
   40 CONTINUE
C        VECTOR COMPUTATION OVER
C        R VECTOR SPECIFIED AT BOUNDARY
      R1(1)=0.
      R3(1)=0.
      R5(1)=0.
      R2(KY)=0.
      R4(KY)=0.
      RETURN
      END
```

```
      SUBROUTINE RHS1(KY,IX,IP)
C     FOUR POINT SCHEME  PREVIOUS X POINT CALCULATION
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION RFL(100),RGL(100),RFP(100),RGP(100),RF(100),RG(100)
      DIMENSION C1(100),C2(100),C3(100),C4(100)
      LOGICAL CONST
      COMMON/MESH/ CONST
      COMMON/COEFF/C1,C2,C3,C4,B1,B2,B3,B4
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/TWPR/TW,PR
      COMMON /RHS/ RFL,RFP,RGL,RGP,RF,RG
      COMMON/CONSIP/AL1,AL2,BTA2,BTA3,BTA4,TH
      DIMENSION TW(2)
      TP=1.+TW(IP)*G(JP,IP)
      TJ=1.+TW(IP)*G(J,IP)
      IF(IX.EQ.1) GOTO 40
C        COMPUTATION OF R VECTOR
      DO 20 J=1,KY
      RFP(J)=RF(J)
      RGP(J)=RG(J)
      RFL(J)=(AL2-AL1-1.)*F(J,IP)*T(J,IP)+(2.*AL1-AL2)*U(J,IP)**2-2.*AL1
     1-RFP(J)
      RGL(J)=(TH-AL2)*U(J,IP)*G(J,IP)-(1.+AL1-AL2)*Q(J,IP)*F(J,IP)-RGP(J
     1)
   20 CONTINUE
      RETURN
   40 CONTINUE
      DO 10 J=2,KY
      RFL(J)=0.
      RGL(J)=0.
   10 CONTINUE
      IF(.NOT.CONST) RETURN
      KP=KY-1
      HJ=H(KP)/24.
      B1=HJ*9.
      B2=HJ*19.
      B3=-HJ*5.
      B4=HJ
      DO 30 J=2,KP
      C1(J)=-HJ
      C2(J)=HJ*13.
      C3(J)=C2(J)
      C4(J)=C1(J)
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COEF(H,KY)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION C1(100),C2(100),C3(100),C4(100),H(100)
      COMMON/COEFF/C1,C2,C3,C4,B1,B2,B3,B4
      LOGICAL CONST
      COMMON/MESH/ CONST
      IF(CONST) GO TO 20
      H1=H(2)+H(3)
      H2=H(3)+H(4)
      H3=H1+H(4)
      B1=H(2)*((H1+H(3))*(H2+H3)+2.*H1*H3)/(12.*H1*H3)
      B2=H(2)*((H3+2.*H2)*(H1+H(3))-H2*H(2))/(12.*H2*H(3))
      HC=H(2)**3/12.
      B3=-HC*(H(2)+2.*H2)/(H1*H(3)*H(4))
      B4=HC*(H1+H(3))/(H2*H3*H(4))
      KP=KY-1
      DO 10 J=3,KP
      JP=J-1
      J1=J+1
      J2=J-2
      H1=H(J1)+H(J)
      H2=H(J)+H(JP)
      H3=H1+H(JP)
      HC=H(J)**3/12.
      C1(J)=-HC*(H1+H(J1))/(H2*H(JP)*H3)
      C2(J)=H(J)*((H1+H(J1))*(H2+H(JP))+2.*H1*H(JP))/(H1*H(JP)*12.)
      C3(J)=H(J)*((H2+H(JP))*(H1+H(J1))+2.*H2*H(J1))/(12.*H2*H(J1))
      C4(J)=-HC*(H2+H(JP))/(H1*H3*H(J1))
   10 CONTINUE
      RETURN
   20 CONTINUE
      KP=KY-1
      HJ=H(KP)/24.
      B1=HJ*9.
      B2=HJ*19.
      B3=-HJ*5.
      B4=HJ
      DO 30 J=2,KP
      C1(J)=-HJ
      C2(J)=HJ*13.
      C3(J)=C2(J)
      C4(J)=C1(J)
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RLHS(KY,IX,IP)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION S1(100),S2(100),S3(100),S4(100),S5(100),S6(100),S7(100)
      DIMENSION S8(100),S9(100),S10(100),S11(100)
      DIMENSION R1(100),R2(100),R3(100),R4(100),R5(100),TW(2)
      DIMENSION RFL(100),DRFL(100),RFP(100),DRF (100),DT(100),DTP(100)
      DIMENSION RGL(100),DRGL(100),RGP(100),DRG (100 ),RF(100),RG(100)
      DIMENSION DRGP(100),DRFP(100)
      COMMON/DRFG/DRFP,DRGP
      COMMON/DERIV/RFL,DRFL,RFP,DRF ,DTP,DT,RGL,DRGL,RGP,DRG ,RF,RG
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/RHL/R1,R2,R3,R4,R5
      COMMON/COMP/S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11
      COMMON/TWPR/TW,PR
      COMMON/CONS/AL1,AL2,BTA2,BTA3,BTA4,TH
      LOGICAL SOLV
      COMMON/SOL/ SOLV
      TX=1.+(TW(IX)-1.)*G(J,IX)
      TP=1.+(TW(IX)-1.)*G(JP,IX)
      DO10 J=1,KY
      RF(J)=BTA3*U(J,IX)**2-BTA2*F(J,IX)*T(J,IX)+AL2*(F(J,IP)*T(J,IX)-
     1T(J,IP)*F(J,IX))-2.*AL1+RFL(J)
      DT(J)=(RF(J)-DVDT(TX)*Q(J,IX)*T(J,IX))/VISC(TX)
      DRF(J)=2.*BTA3*U(J,IX)*T(J,IX)-BTA2*(F(J,IX)*DT(J)+U(J,IX)*T(J,IX)
     1)+AL2*(F(J,IP)*DT(J)+T(J,IX)*U(J,IP)-T(J,IP)*U(J,IX)-F(J,IX)*DTP(J
     2))+DRFL(J)
      RG(J)=BTA4*U(J,IX)*G(J,IX)-BTA2*Q(J,IX)*F(J,IX)+AL2*(U(J,IP)*G(J,IX)
     1X)+G(J,IP)*U(J,IX)-Q(J,IP)*F(J,IX)+F(J,IP)*Q(J,IX))+RGL(J)
      DRG(J)=BTA4*(U(J,IX)*Q(J,IX)+G(J,IX)*T(J,IX))-BTA2*(Q(J,IX)*U(J,IX
     1)+F(J,IX)*PR*RG(J))+AL2*(U(J,IP)*Q(J,IX)+G(J,IX)*T(J,IP)+G(J,IP)*T
     2(J,IX)-F(J,IX)*PR*RGP(J)+F(J,IP)*PR*RG(J)+Q(J,IX)*U(J,IP))+DRGL(J)
  10  CONTINUE
      IF(.NOT.SOLV) RETURN
      DO 30 J=2,KY
      JP=J-1
      HJ=H(J)/2.
      HJP=H(J)**2/12.
      R1(J)=F(JP,IX)-F(J,IX)+HJ*(U(JP,IX)+U(J,IX))+HJP*(T(JP,IX)-T(J,IX)
     1)
      R2(JP)=U(JP,IX)-U(J,IX)+HJ*(T(JP,IX)+T(J,IX))+HJP*(DT(JP)-DT(J))
      R3(J)=VISC(TP)*T(JP,IX)-VISC(TX)*T(J,IX)+HJ*(RF(J)+RF(JP))+HJP*(DR
     1F(JP)-DRF(J))
      R4(JP)=G(JP,IX)-G(J,IX)+HJ*(Q(JP,IX)+Q(J,IX))+HJP*PR*(RG(JP)-RG(J))
     1)
      R5(J)=Q(JP,IX)-Q(J,IX)+HJ*(RG(JP)+RG(J))+HJP*(DRG(JP)-DRG(J))
      R3(J)=R3(J)/HJ
      R5(J)=R5(J)/HJ
  30  CONTINUE
```

```
C          COMPUTATION OF S VECTOR
      DO 40 J=2,KY
      HJ=H(J)/2.
      JP=J-1
      PRH=1./(PR*HJ)
      S1(J)=DVDT(TX)*T(J,IX)/HJ
      S2(J)=BTA2*T(J,IX)+AL2*T(J,IP)
      S3(J)=-2.*BTA3*U(J,IX)
      S4(J)=VISC(TX)/HJ+BTA2*F(J,IX)-AL2*F(J,IP)
      S5(JP)=-VISC(IP)/HJ+BTA2*F(JP,IX)-AL2*F(JP,IP)
      S6(J)=PRH+BTA2*F(J,IX)-AL2*F(J,IP)
      S7(J)=BTA2*Q(J,IX)+AL2*Q(J,IP)
      S8(J)=AL2*G(J,IP)-BTA4*G(J,IX)
      S9(J)=-BTA4*U(J,IX)-AL2*U(J,IP)
      S10(JP)=BTA2*F(JP,IX)-AL2*F(JP,IP)-PRH
      S11(JP)=-DVDT(IP)*T(JP,IX)/HJ
   40 CONTINUE
C          VECTOR COMPUTATION OVER
C          R VECTOR SPECIFIED AT BOUNDARY
      R1(1)=0.
      R3(1)=0.
      R5(1)=0.
      R2(KY)=0.
      R4(KY)=0.
      RETURN
      END
//
```

```
      SUBROUTINE RHS1(KY,IX,IP)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(100,2),U(100,2),T(100,2),G(100,2),Q(100,2),H(100)
      DIMENSION TW(2)
      DIMENSION RFL(100),DRFL(100),RFP(100),DRF (100),DT(100),DTP(100)
      DIMENSION RGL(100),DRGL(100),RGP(100),DRG (100 ),RF(100),RG(100)
      DIMENSION DRGP(100),DRFP(100)
      COMMON/DRFG/DRFP,DRGP
      COMMON/DERIV/RFL,DRFL,RFP,DRF ,DTP,DT,RGL,DRGL,RGP,DRG ,RF,RG
      COMMON/ALL/F,U,T,G,Q,H
      COMMON/TWPR/TW,PR
      COMMON/CONSIP/AL1,AL2,BTA2,BTA3,BTA4,TH
      TP=1.+(TW(IP)-1.)*G(JP,IP)
      TJ=1.+(TW(IP)-1.)*G(J,IP)
      DO 10 J=2,KY
      RFL(J)=0.
      DRFL(J)=0.
      RGL(J)=0.
      DRGL(J)=0.
   10 CONTINUE
      IF(IX.EQ.1) GOTO 30
C        COMPUTATION OF R VECTOR
      DO 20 J=1,KY
      DTP(J)=DT(J)
      RFP(J)=RF(J)
      DRFP(J)=DRF(J)
      RGP(J)=RG(J)
      DRGP(J)=DRG(J)
      RFL(J)=(AL2-AL1-1.)*F(J,IP)*T(J,IP)+(2.*AL1-AL2)*U(J,IP)**2-2.*AL1
     1-RFP(J)
      DRFL(J)=(2.*AL1-AL2)*2.*U(J,IP)*T(J,IP)-(1.+AL1-AL2)*(F(J,IP)*DTP(
     1J)+T(J,IP)*U(J,IP))-DRFP(J)
      RGL(J)=(TH-AL2)*U(J,IP)*G(J,IP)-(1.+AL1-AL2)*Q(J,IP)*F(J,IP)
     1-RGP(J)
      DRGL(J)=(TH-AL2)*(U(J,IP)*Q(J,IP)+G(J,IP)*T(J,IP))-(1.+AL1-AL2)*(Q
     1(J,IP)*U(J,IP)+F(J,IP)*PR*RGP(J))-DRGP(J)
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
```