The Proteus Navier-Stokes Code

Charles E. Towne
Trong T. Bui
Richard H. Cavicchi
Julianne M. Conley
Frank B. Molls
John R. Schwab

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

An effort is currently underway at NASA Lewis to develop two- and three-dimensional Navier-Stokes codes, called *Proteus*, for aerospace propulsion applications. The emphasis in the development of *Proteus* is not algorithm development or research on numerical methods, but rather the development of the code itself. The objective is to develop codes that are user-oriented, easily-modified, and well-documented. Well-proven, state-of-the-art solution algorithms are being used. Code readability, documentation (both internal and external), and validation are being emphasized. This paper is a status report on the *Proteus* development effort. The analysis and solution procedure are described briefly, and the various features in the code are summarized. The results from some of the validation cases that have been run are presented for both the two- and three-dimensional codes.

1. INTRODUCTION

Much of the effort in applied computational fluid dynamics consists of modifying an existing program for whatever geometries and flow regimes are of current interest to the researcher. Unfortunately, nearly all of the available nonproprietary programs were started as research projects with the emphasis on demonstrating the numerical algorithm rather than ease of use or ease of modification. The developers usually intend to clean up and formally document the program, but the immediate need to extend it to new geometries and flow regimes takes precedence.

The result is often a haphazard collection of poorly written code without any consistent structure. An extensively modified program may not even perform as expected under certain combinations of operating options. Each new user must invest considerable time and effort in attempting to understand the underlying structure of the program if intending to do anything more than run standard test cases with it. The user's subsequent modifications further obscure the program structure and therefore make it even more difficult for others to understand.

The *Proteus* two- and three-dimensional Navier-Stokes computer codes are intended to be user-oriented and easily-modifiable flow analysis programs, primarily for aerospace propulsion applications. Readability, modularity, and documentation have been the primary objectives. Every subroutine contains an extensive comment section describing the purpose, input variables, output variables, and calling sequence of the subroutine. With just three clearly-defined exceptions, the entire program is written in ANSI standard Fortran 77 to enhance portability. A master version of the program is maintained and periodically updated with corrections, as well as extensions of general interest, such as turbulence models.

The documentation is divided into three volumes. Volume 1 is the Analysis Description, and presents the equations and solution procedure used in *Proteus*. It describes in detail the governing equations, the turbulence models, the linearization of the equations and boundary conditions, the time and space differencing formulas, the ADI solution procedure, and the artificial viscosity models. Volume 2 is the User's Guide, and contains information needed to run the program. It describes the program's general features, the input and output, the procedure for setting up initial conditions, the computer resource requirements, the diagnostic messages that may be generated, the job control language used to run the program, and several test cases. Volume 3 is the Programmer's Reference, and contains detailed information useful when modifying the program. It describes the program structure, the Fortran variables stored in common blocks, and the details of each subprogram.

In this paper, the analysis and solution procedure are described briefly, and the various features in the code are summarized. The results from some of the validation cases that have been run are presented for both the two- and three-dimensional codes. The paper concludes with a brief status report on the *Proteus* development effort, including the work currently underway and our future plans.

2. ANALYSIS DESCRIPTION

In this section, the governing equations, the numerical solution method, and the turbulence models are described briefly. For a much more detailed description, see Volume 1 of the documentation (Towne, Schwab, Benson, and Suresh, 1990).

2.1 GOVERNING EQUATIONS

The basic governing equations are the compressible Navier-Stokes equations. In Cartesian coordinates, the two-dimensional planar equations can be written in strong conservation law form using vector notation as [1]

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{E}}{\partial x} + \frac{\partial \mathbf{F}}{\partial y} = \frac{\partial \mathbf{E}_V}{\partial x} + \frac{\partial \mathbf{F}_V}{\partial y} \tag{1}$$

where

$$\mathbf{Q} = \begin{bmatrix} \rho & \rho u & \rho v & E_T \end{bmatrix}^T \tag{2a}$$

$$\mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E_T + p)u \end{bmatrix} \tag{2b}$$

$$\mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E_T + p)v \end{bmatrix} \tag{2c}$$

$$\mathbf{E}_V = \frac{1}{Re_r} \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - \dfrac{1}{Pr_r} q_x \end{bmatrix} \tag{2d}$$

$$\mathbf{F}_V = \frac{1}{Re_r} \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - \dfrac{1}{Pr_r} q_y \end{bmatrix} \tag{2e}$$

The shear stresses and heat fluxes are given by

1. For brevity, in most instances this paper describes the two-dimensional *Proteus* code. The extension to three dimensions is relatively straightforward. Differences between the two-dimensional and three-dimensional codes are noted where relevant.

$$\tau_{xx} = 2\mu\frac{\partial u}{\partial x} + \lambda\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]$$

$$\tau_{yy} = 2\mu\frac{\partial v}{\partial y} + \lambda\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]$$

$$\tau_{xy} = \mu\left[\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right] \tag{3}$$

$$q_x = -k\frac{\partial T}{\partial x}$$

$$q_y = -k\frac{\partial T}{\partial y}$$

In these equations, t represents time; x and y represent the Cartesian coordinate directions; u and v are the velocities in the x and y directions; ρ, p, and T are the static density, pressure, and temperature; E_T is the total energy per unit volume; and μ, λ, and k are the coefficient of viscosity, the second coefficient of viscosity, and the coefficient of thermal conductivity.

In addition to the equations presented above, an equation of state is required to relate pressure to the dependent variables. The equation currently built into the *Proteus* code is the equation of state for thermally perfect gases, $p = \rho RT$, where R is the gas constant. For calorically perfect gases, this can be rewritten as

$$p = (\gamma - 1)\left[E_T - \frac{1}{2}\rho(u^2 + v^2)\right] \tag{4}$$

where γ is the ratio of specific heats, c_p / c_v. Additional equations are also used to define μ, λ, k, and c_p in terms of temperature for the fluid under consideration.

All of the equations have been nondimensionalized using appropriate normalizing conditions. Lengths have been nondimensionalized by L_r, velocities by u_r, density by ρ_r, temperature by T_r, viscosity by μ_r, thermal conductivity by k_r, pressure and total energy by $\rho_r u_r^2$, time by L_r / u_r, and gas constant and specific heat by u_r^2 / T_r. The reference Reynolds and Prandtl numbers are thus defined as $Re_r = \rho_r u_r L_r / \mu_r$ and $Pr_r = \mu_r u_r^2 / k_r T_r$.

Because the governing equations are written in Cartesian coordinates, they are not well suited for general geometric configurations. For most applications a body-fitted coordinate system is desired. This greatly simplifies the application of boundary conditions and the bookkeeping in the numerical method used to solve the equations. The equations are thus transformed from physical (x,y,t) coordinates to rectangular orthogonal computational (ξ,η,τ) coordinates. Equation (1) becomes

$$\frac{\partial \hat{\mathbf{Q}}}{\partial \tau} + \frac{\partial \hat{\mathbf{E}}}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}}{\partial \eta} = \frac{\partial \hat{\mathbf{E}}_V}{\partial \xi} + \frac{\partial \hat{\mathbf{F}}_V}{\partial \eta} \tag{5}$$

where

$$\hat{\mathbf{Q}} = \frac{\mathbf{Q}}{J}$$

$$\hat{\mathbf{E}} = \frac{1}{J}(\mathbf{E}\xi_x + \mathbf{F}\xi_y + \mathbf{Q}\xi_t)$$

$$\hat{\mathbf{F}} = \frac{1}{J}(\mathbf{E}\eta_x + \mathbf{F}\eta_y + \mathbf{Q}\eta_t)$$

$$\hat{\mathbf{E}}_V = \frac{1}{J}(\mathbf{E}_V\xi_x + \mathbf{F}_V\xi_y)$$

$$\hat{\mathbf{F}}_V = \frac{1}{J}(\mathbf{E}_V\eta_x + \mathbf{F}_V\eta_y)$$

In these equations the derivatives ξ_x, η_x, etc., are the metric scale coefficients for the generalized nonorthogonal grid transformation. J is the Jacobian of the transformation.

2.2 NUMERICAL METHOD

2.2.1 Time Differencing. The governing equations are solved by marching in time from some known set of initial conditions using a finite difference technique. The time differencing scheme currently used is the generalized method of Beam and Warming (1978). With this scheme, the time derivative term in equation (5) is written as

$$\frac{\partial\hat{\mathbf{Q}}}{\partial\tau} = \frac{\Delta\hat{\mathbf{Q}}^n}{\Delta\tau} = \frac{\theta_1}{1+\theta_2}\frac{\partial(\Delta\hat{\mathbf{Q}}^n)}{\partial\tau} + \frac{1}{1+\theta_2}\frac{\partial\hat{\mathbf{Q}}^n}{\partial\tau} + \frac{\theta_2}{1+\theta_2}\frac{\Delta\hat{\mathbf{Q}}^{n-1}}{\Delta\tau} + O\left[\left(\theta_1 - \frac{1}{2} - \theta_2\right)\Delta\tau, (\Delta\tau)^2\right] \tag{6}$$

where $\Delta\hat{\mathbf{Q}}^n = \hat{\mathbf{Q}}^{n+1} - \hat{\mathbf{Q}}^n$. The superscripts n and $n+1$ denote the known and unknown time levels, respectively. By choosing appropriate values for θ_1 and θ_2, the solution procedure can be either first- or second-order accurate in time.

Solving equation (5) for $\partial\hat{\mathbf{Q}}/\partial\tau$, substituting the result into equation (6) for $\partial(\Delta\hat{\mathbf{Q}}^n)/\partial\tau$ and $\partial\hat{\mathbf{Q}}^n/\partial\tau$, and multiplying by $\Delta\tau$ yields

$$\Delta\hat{\mathbf{Q}}^n = -\frac{\theta_1\Delta\tau}{1+\theta_2}\left[\frac{\partial(\Delta\hat{\mathbf{E}}^n)}{\partial\xi} + \frac{\partial(\Delta\hat{\mathbf{F}}^n)}{\partial\eta}\right] - \frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{\mathbf{E}}^n}{\partial\xi} + \frac{\partial\hat{\mathbf{F}}^n}{\partial\eta}\right] + \frac{\theta_1\Delta\tau}{1+\theta_2}\left[\frac{\partial(\Delta\hat{\mathbf{E}}_V^n)}{\partial\xi} + \frac{\partial(\Delta\hat{\mathbf{F}}_V^n)}{\partial\eta}\right] + \frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{\mathbf{E}}_V^n}{\partial\xi} + \frac{\partial\hat{\mathbf{F}}_V^n}{\partial\eta}\right]$$
$$+ \frac{\theta_2}{1+\theta_2}\Delta\hat{\mathbf{Q}}^{n-1} + O\left[\left(\theta_1 - \frac{1}{2} - \theta_2\right)(\Delta\tau)^2, (\Delta\tau)^3\right] \tag{7}$$

2.2.2 Linearization Procedure. Equation (7) is nonlinear, since, for example, $\Delta\hat{\mathbf{E}}^n = \hat{\mathbf{E}}^{n+1} - \hat{\mathbf{E}}^n$ and the unknown $\hat{\mathbf{E}}^{n+1}$ is a nonlinear function of the dependent variables and of the metric coefficients resulting from the generalized grid transformation. The equations must therefore be linearized to be solved by the finite difference procedure. For the inviscid terms, and for the non–cross-derivative viscous terms, this is done by expanding each nonlinear expression in a Taylor series in time about the known time level n. The cross-derivative viscous terms are simply lagged (i.e., evaluated at the known time level n and treated as source terms.)

The linearized form of equation (7) may be written as

$$\Delta\hat{Q}^n + \frac{\theta_1\Delta\tau}{1+\theta_2}\left\{\frac{\partial}{\partial\xi}\left[\left[\frac{\partial\hat{E}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right] + \frac{\partial}{\partial\eta}\left[\left[\frac{\partial\hat{F}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right]\right\} - \frac{\theta_1\Delta\tau}{1+\theta_2}\left\{\frac{\partial}{\partial\xi}\left[\left[\frac{\partial\hat{E}_{V_1}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right] + \frac{\partial}{\partial\eta}\left[\left[\frac{\partial\hat{F}_{V_1}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right]\right\} =$$

$$-\frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}}{\partial\xi} + \frac{\partial\hat{F}}{\partial\eta}\right]^n + \frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_1}}{\partial\xi} + \frac{\partial\hat{F}_{V_1}}{\partial\eta}\right]^n + \frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_2}}{\partial\xi} + \frac{\partial\hat{F}_{V_2}}{\partial\eta}\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_2}}{\partial\xi} + \frac{\partial\hat{F}_{V_2}}{\partial\eta}\right]^{n-1}$$

$$+\frac{\theta_2}{1+\theta_2}\Delta\hat{Q}^{n-1} + O\left[\left[\theta_1 - \frac{1}{2} - \theta_2\right](\Delta\tau)^2, (\theta_3-\theta_1)(\Delta\tau)^2, (\Delta\tau)^3\right] \tag{8}$$

where $\partial\hat{E}/\partial\hat{Q}$ and $\partial\hat{F}/\partial\hat{Q}$ are the Jacobian coefficient matrices resulting from the linearization of the convective terms, and $\partial\hat{E}_{V_1}/\partial\hat{Q}$ and $\partial\hat{F}_{V_1}/\partial\hat{Q}$ are the Jacobian coefficient matrices resulting from the linearization of the viscous terms.

The boundary conditions are treated implicitly, and may be viewed simply as additional equations to be solved by the ADI solution algorithm. In general, they also involve nonlinear functions of the dependent variables. They are therefore linearized using the same procedure as for the governing equations.

2.2.3 Solution Procedure. The governing equations, presented in linearized matrix form as equation (8), are solved by an alternating direction implicit (ADI) method. The form of the ADI splitting is the same as used by Briley and McDonald (1977), and by Beam and Warming (1978). Using approximate factorization, equation (8) can be split into the following two-sweep sequence.

Sweep 1 (ξ direction)

$$\Delta\hat{Q}^* + \frac{\theta_1\Delta\tau}{1+\theta_2}\frac{\partial}{\partial\xi}\left[\left[\frac{\partial\hat{E}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^*\right] - \frac{\theta_1\Delta\tau}{1+\theta_2}\frac{\partial}{\partial\xi}\left[\left[\frac{\partial\hat{E}_{V_1}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^*\right] = -\frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}}{\partial\xi} + \frac{\partial\hat{F}}{\partial\eta}\right]^n + \frac{\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_1}}{\partial\xi} + \frac{\partial\hat{F}_{V_1}}{\partial\eta}\right]^n$$

$$+\frac{(1+\theta_3)\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_2}}{\partial\xi} + \frac{\partial\hat{F}_{V_2}}{\partial\eta}\right]^n - \frac{\theta_3\Delta\tau}{1+\theta_2}\left[\frac{\partial\hat{E}_{V_2}}{\partial\xi} + \frac{\partial\hat{F}_{V_2}}{\partial\eta}\right]^{n-1} + \frac{\theta_2}{1+\theta_2}\Delta\hat{Q}^{n-1} \tag{9a}$$

Sweep 2 (η direction)

$$\Delta\hat{Q}^n + \frac{\theta_1\Delta\tau}{1+\theta_2}\frac{\partial}{\partial\eta}\left[\left[\frac{\partial\hat{F}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right] - \frac{\theta_1\Delta\tau}{1+\theta_2}\frac{\partial}{\partial\eta}\left[\left[\frac{\partial\hat{F}_{V_1}}{\partial\hat{Q}}\right]^n\Delta\hat{Q}^n\right] = \Delta\hat{Q}^* \tag{9b}$$

These equations represent the two-sweep alternating direction implicit (ADI) algorithm used to advance the solution from time level n to $n+1$. $\hat{Q}^*$ is the intermediate solution.

Spatial derivatives in equations (9a) and (9b) are approximated using second-order central difference formulas. The resulting set of algebraic equations can be written in matrix form with a block tri-diagonal coefficient matrix. They are solved using the block matrix version of the Thomas algorithm (e.g., see Anderson, Tannehill, and Pletcher, 1984).

2.2.4 Artificial Viscosity. With the numerical algorithm described above, high frequency nonlinear instabilities can appear as the solution develops. For example, in high Reynolds number flows oscillations can result from the odd-even decoupling inherent in the use of second-order central differencing for the inviscid terms. In addition, physical phenomena such as shock waves can cause instabilities when they are captured by the finite difference algorithm. Artificial viscosity, or smoothing, is normally added to the solution algorithm to suppress these high

frequency instabilities. Two artificial viscosity models are currently available in the *Proteus* computer code — a constant coefficient model used by Steger (1978), and the nonlinear coefficient model of Jameson, Schmidt, and Turkel (1981). The implementation of these models in generalized nonorthogonal coordinates is described by Pulliam (1986).

The constant coefficient model uses a combination of explicit and implicit artificial viscosity. The standard explicit smoothing uses fourth-order differences, and damps the high frequency nonlinear instabilities. Second-order explicit smoothing, while not used by Steger or Pulliam, is also available in *Proteus*. It provides more smoothing than the fourth-order smoothing, but introduces a larger error, and is therefore not used as often. The implicit smoothing is second order and is intended to extend the linear stability bound of the fourth-order explicit smoothing.

The explicit artificial viscosity is implemented in the numerical algorithm by adding the following terms to the right hand side of equation (9a) (i.e., the source term for the first ADI sweep.)

$$\frac{\varepsilon_E^{(2)}\Delta\tau}{J}(\nabla_\xi\Delta_\xi Q+\nabla_\eta\Delta_\eta Q)-\frac{\varepsilon_E^{(4)}\Delta\tau}{J}\left[(\nabla_\xi\Delta_\xi)^2Q+(\nabla_\eta\Delta_\eta)^2Q\right]$$

$\varepsilon_E^{(2)}$ and $\varepsilon_E^{(4)}$ are the second- and fourth-order explicit artificial viscosity coefficients. The symbols ∇ and Δ are backward and forward first difference operators.

The implicit artificial viscosity is implemented by adding the following terms to the left hand side of the equations specified.

$$-\frac{\varepsilon_I\Delta\tau}{J}\left[\nabla_\xi\Delta_\xi(J\Delta\hat{Q}^*)\right] \quad \text{to equation (9a)}$$

$$-\frac{\varepsilon_I\Delta\tau}{J}\left[\nabla_\eta\Delta_\eta(J\Delta\hat{Q}^n)\right] \quad \text{to equation (9b)}$$

The nonlinear coefficient artificial viscosity model is strictly explicit. Using the model as described by Pulliam (1986), but in the current notation, the following terms are added to the right hand side of equation (9a).

$$\nabla_\xi\left\{\left[\left[\frac{\psi}{J}\right]_{i+1}+\left[\frac{\psi}{J}\right]_i\right]\left(\varepsilon_\xi^{(2)}\Delta_\xi Q-\varepsilon_\xi^{(4)}\Delta_\xi\nabla_\xi\Delta_\xi Q\right)_i\right\}+\nabla_\eta\left\{\left[\left[\frac{\psi}{J}\right]_{j+1}+\left[\frac{\psi}{J}\right]_j\right]\left(\varepsilon_\eta^{(2)}\Delta_\eta Q-\varepsilon_\eta^{(4)}\Delta_\eta\nabla_\eta\Delta_\eta Q\right)_j\right\}$$

The subscripts i and j denote grid indices in the ξ and η directions. In the above expression, ψ is defined as

$$\psi=\psi_x+\psi_y$$

where ψ_x and ψ_y are spectral radii defined by

$$\psi_x=\frac{|U|+a\sqrt{\xi_x^2+\xi_y^2}}{\Delta\xi}$$

$$\psi_y=\frac{|V|+a\sqrt{\eta_x^2+\eta_y^2}}{\Delta\eta}$$

Here U and V are the contravariant velocities without metric normalization, defined by

$$U=\xi_t+\xi_x u+\xi_y v$$

$$V=\eta_t+\eta_x u+\eta_y v$$

and $a=\sqrt{\gamma RT}$, the speed of sound.

The parameters $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are the second- and fourth-order artificial viscosity coefficients. For the coefficients of the ξ direction differences,

$$\left(\varepsilon_\xi^{(2)}\right)_i = \kappa_2 \Delta\tau \max(\sigma_{i+1}, \sigma_i, \sigma_{i-1})$$

$$\left(\varepsilon_\xi^{(4)}\right)_i = \max\left[0, \kappa_4 \Delta\tau - \left(\varepsilon_\xi^{(2)}\right)_i\right]$$

where

$$\sigma_i = \left|\frac{p_{i+1} - 2p_i + p_{i-1}}{p_{i+1} + 2p_i + p_{i-1}}\right|$$

and κ_2 and κ_4 are constants. Similar formulas are used for the coefficients of the η direction differences. The parameter σ is a pressure gradient scaling parameter that increases the amount of second-order smoothing relative to fourth-order smoothing near shock waves. The logic used to compute $\varepsilon^{(4)}$ switches off the fourth-order smoothing when the second-order smoothing term is large.

2.3 TURBULENCE MODELS

Turbulence is modeled using either a generalized version of the Baldwin and Lomax (1978) algebraic eddy viscosity model, or the Chien (1982) low Reynolds number k-ε model.

2.3.1 Baldwin-Lomax Model. For wall-bounded flows, the Baldwin-Lomax turbulence model is a two-layer model, with

$$\mu_t = \begin{cases} (\mu_t)_{inner} & \text{for } y_n \le y_b \\ (\mu_t)_{outer} & \text{for } y_n > y_b \end{cases} \tag{10}$$

where y_n is the normal distance from the wall, and y_b is the smallest value of y_n at which the values of μ_t from the inner and outer region formulas are equal. For free turbulent flows, only the outer region value is used.

The outer region turbulent viscosity at a given ξ or η station is computed from

$$(\mu_t)_{outer} = KC_{cp}\rho F_{Kleb} F_{wake} Re_r \tag{11}$$

where K is the Clauser constant, taken as 0.0168, and C_{cp} is a constant taken as 1.6.

The parameter F_{wake} is computed from

$$F_{wake} = \begin{cases} y_{max} F_{max} & \text{for wall-bounded flows} \\ C_{wk} V_{diff}^2 \dfrac{y_{max}}{F_{max}} & \text{for free turbulent flows} \end{cases} \tag{12}$$

where C_{wk} is a constant taken as 0.25, and

$$V_{diff} = |\vec{V}|_{max} - |\vec{V}|_{min}$$

where $\vec{V}$ is the total velocity vector.

The parameter F_{max} in equation (12) is the maximum value of

$$F(y_n) = \begin{cases} y_n |\vec{\Omega}| \left[1 - e^{-y^+/A^+}\right] & \text{for wall-bounded flows} \\ y_n |\vec{\Omega}| & \text{for free turbulent flows} \end{cases} \tag{13}$$

and y_{max} is the value of y_n corresponding to F_{max}.

For wall-bounded flows, y_n is the normal distance from the wall. For free turbulent flows, two values of F_{max} and y_{max} are computed — one using the location of $|\vec{V}|_{max}$ as the origin for y_n, and one using the location of $|\vec{V}|_{min}$. The origin giving the smaller value of y_{max} is the one finally used for computing y_n, F_{max}, and y_{max}.

In equation (13), $|\vec{\Omega}|$ is the magnitude of the total vorticity, defined for two-dimensional planar flow as

$$|\vec{\Omega}| = \left|\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right| \tag{14}$$

The parameter A^+ is the Van Driest damping constant, taken as 26.0. The coordinate y^+ is defined as

$$y^+ = \frac{\rho_w u_\tau y_n}{\mu_w} Re_r = \frac{\sqrt{\tau_w \rho_w Re_r}}{\mu_w} y_n \tag{15}$$

where $u_\tau = \sqrt{\tau_w / \rho_w Re_r}$ is the friction velocity, τ is the shear stress, and the subscript w indicates a wall value. In *Proteus*, τ_w is set equal to $\mu_w |\vec{\Omega}|_w$.

The function F_{Kleb} in equation (11) is the Klebanoff intermittency factor. For free turbulent flows, $F_{Kleb} = 1$. For wall-bounded flows,

$$F_{Kleb} = \left[1 + B\left(\frac{C_{Kleb} y_n}{y_{max}}\right)^6\right]^{-1} \tag{16}$$

In equation (16), B and C_{Kleb} are constants taken as 5.5 and 0.3, respectively.

The inner region turbulent viscosity in the Baldwin-Lomax model is

$$(\mu_t)_{inner} = \rho l^2 |\vec{\Omega}| Re_r \tag{17}$$

where l is the mixing length, given by

$$l = \kappa y_n \left[1 - e^{-y^+/A^+}\right] \tag{18}$$

and κ is the Von Karman constant, taken as 0.4.

If both boundaries in a given coordinate direction are solid surfaces, the turbulence model is applied separately for each surface. An averaging procedure is used to combine the resulting two μ_t profiles into one.

The turbulent second coefficient of viscosity is simply defined as

$$\lambda_t = -\frac{2}{3}\mu_t$$

The turbulent thermal conductivity coefficient is defined using Reynolds analogy as

$$k_t = \frac{c_p \mu_t}{Pr_t} Pr_r$$

where c_p is the specific heat at constant pressure, and Pr_t is the turbulent Prandtl number.

2.3.2 **Chien k-ε Model.** The low Reynolds number k-ε formulation of Chien (1982) was chosen because of its reasonable approximation of the near wall region and because of its numerical stability. Here k and ε are the turbulent kinetic energy and the turbulent dissipation rate, respectively.

In Cartesian coordinates, the two-dimensional planar equations for the Chien k-ε model can be written using vector notation as

$$\frac{\partial \mathbf{W}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{S} + \mathbf{T} \tag{19}$$

where

$$\mathbf{W} = \begin{bmatrix} \rho k \\ \rho \varepsilon \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} \rho u k - \dfrac{1}{Re_r}\mu_k \dfrac{\partial k}{\partial x} \\ \rho u \varepsilon - \dfrac{1}{Re_r}\mu_\varepsilon \dfrac{\partial \varepsilon}{\partial x} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \rho v k - \dfrac{1}{Re_r}\mu_k \dfrac{\partial k}{\partial y} \\ \rho v \varepsilon - \dfrac{1}{Re_r}\mu_\varepsilon \dfrac{\partial \varepsilon}{\partial y} \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} P_k - Re_r \rho \varepsilon \\ C_1 P_k \dfrac{\varepsilon}{k} - Re_r C_2 \rho \dfrac{\varepsilon^2}{k} \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} -\dfrac{2}{Re_r}\dfrac{\mu}{y_n^2} k \\ -\dfrac{2}{Re_r}\dfrac{\mu e^{-y^+/2}}{y_n^2} \varepsilon \end{bmatrix}$$

and

$$\mu_k = \mu + \frac{\mu_t}{\sigma_k}$$

$$\mu_\varepsilon = \mu + \frac{\mu_t}{\sigma_\varepsilon}$$

$$C_2 = C_{2_c}\left[1 - \frac{2}{9}e^{-R_t^2/36}\right]$$

$$R_t = \frac{\rho k^2}{\mu \varepsilon}$$

$$P_k = \frac{\mu_t}{Re_r} P_1 - \frac{2}{3}\rho k P_2$$

$$P_1 = 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right] - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2$$

$$P_2 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

The turbulent viscosity is given by

$$\mu_t = C_\mu \rho \frac{k^2}{\varepsilon} \tag{20}$$

$$C_\mu = C_{\mu_\infty}\left(1 - e^{-C_3 y^+}\right)$$

In the above equations, C_1, C_2, C_3, σ_k, σ_ε, and C_{μ_∞} are constants equal to 1.35, 1.8, 0.0115, 1.0, 1.3, and 0.09, respectively. The parameter y_n is the minimum distance to the nearest solid surface, and y^+ is computed from y_n. In the above equations the mean flow properties have been nondimensionalized as described in Section 2.1. The turbulent kinetic energy k and the turbulent dissipation rate ε have been nondimensionalized by u_r^2 and $\rho_r u_r^4 / \mu_r$, respectively.

After transforming from physical to rectangular orthogonal computational coordinates, equation (19) becomes

$$\frac{\partial \hat{\mathbf{W}}}{\partial \tau} + \frac{\partial \hat{\mathbf{F}}}{\partial \xi} + \frac{\partial \hat{\mathbf{G}}}{\partial \eta} = \hat{\mathbf{S}} + \hat{\mathbf{T}} \tag{21}$$

where

$$\hat{\mathbf{W}} = \frac{1}{J}\begin{bmatrix}\rho k \\ \rho \varepsilon\end{bmatrix}$$

$$\hat{\mathbf{F}} = \hat{\mathbf{F}}_C - \hat{\mathbf{F}}_D - \hat{\mathbf{F}}_M$$

$$\hat{\mathbf{F}}_C = \frac{1}{J}\begin{bmatrix}\xi_x \rho u k + \xi_y \rho v k \\ \xi_x \rho u \varepsilon + \xi_y \rho v \varepsilon\end{bmatrix}$$

$$\hat{\mathbf{F}}_D = \frac{1}{J}\frac{1}{Re_r}\begin{bmatrix}\mu_k(\xi_x^2 + \xi_y^2)k_\xi \\ \mu_\varepsilon(\xi_x^2 + \xi_y^2)\varepsilon_\xi\end{bmatrix}$$

$$\hat{\mathbf{F}}_M = \frac{1}{J}\frac{1}{Re_r}\begin{bmatrix}\mu_k(\xi_x \eta_x + \xi_y \eta_y)k_\eta \\ \mu_\varepsilon(\xi_x \eta_x + \xi_y \eta_y)\varepsilon_\eta\end{bmatrix}$$

$$\hat{\mathbf{G}} = \hat{\mathbf{G}}_C - \hat{\mathbf{G}}_D - \hat{\mathbf{G}}_M$$

$$\hat{\mathbf{G}}_C = \frac{1}{J}\begin{bmatrix}\eta_x \rho u k + \eta_y \rho v k \\ \eta_x \rho u \varepsilon + \eta_y \rho v \varepsilon\end{bmatrix}$$

$$\hat{\mathbf{G}}_D = \frac{1}{J}\frac{1}{Re_r}\begin{bmatrix}\mu_k(\eta_x^2 + \eta_y^2)k_\eta \\ \mu_\varepsilon(\eta_x^2 + \eta_y^2)\varepsilon_\eta\end{bmatrix}$$

$$\hat{G}_M = \frac{1}{J}\frac{1}{Re_r}\begin{bmatrix}\mu_k(\xi_x\eta_x+\xi_y\eta_y)k_\xi\\ \mu_\varepsilon(\xi_x\eta_x+\xi_y\eta_y)\varepsilon_\xi\end{bmatrix}$$

$$\hat{S} = \frac{1}{J}\begin{bmatrix}P_k - Re_r\rho\varepsilon\\ C_1P_k\frac{\varepsilon}{k} - Re_rC_2\rho\frac{\varepsilon^2}{k}\end{bmatrix}$$

$$\hat{T} = \frac{1}{J}\begin{bmatrix}-\frac{2}{Re_r}\frac{\mu}{y_n^2}k\\ -\frac{2}{Re_r}\frac{\mu e^{-y^+/2}}{y_n^2}\varepsilon\end{bmatrix}$$

The time differencing scheme and linearization procedure described previously for the mean flow equations are also applied to equation (21). The mean flow variables are evaluated at the known time level n. This allows the k-ε equations to be uncoupled from the mean flow equations and solved separately. Spatial derivatives are approximated using first-order upwind differences for the convective terms, and second-order central differences for the viscous terms. In the two-dimensional *Proteus* code, the equations are solved by the same ADI procedure as the mean flow equations. In the three-dimensional code, they are solved by a two-sweep LU procedure, as described by Hoffmann (1989).

The turbulent second coefficient of viscosity λ_t and the turbulent thermal conductivity coefficient k_t are defined as described in the previous section.

3. CODE FEATURES

In this section the basic characteristics and capabilities of the two- and three-dimensional *Proteus* codes are summarized. For a much more detailed description, see Volumes 2 and 3 of the documentation (Towne, Schwab, Benson, and Suresh, 1990).

3.1 ANALYSIS

The *Proteus* codes solve the unsteady compressible Navier-Stokes equations in either two or three dimensions. The 2-D code can solve either the planar or axisymmetric form of the equations. Swirl is allowed in axisymmetric flow. The 2-D planar equations and the 3-D equations are solved in fully conservative form. As subsets of these equations, options are available to solve the Euler equations or the thin-layer Navier-Stokes equations. An option is also available to eliminate the energy equation by assuming constant total enthalpy.

The equations are solved by marching in time using the generalized time differencing of Beam and Warming (1978). The method may be either first- or second-order accurate in time, depending on the choice of time differencing parameters. Second-order central differencing is used for all spatial derivatives. Nonlinear terms are linearized using second-order Taylor series expansions in time. The resulting difference equations are solved using an alternating-direction implicit (ADI) technique, with Douglas-Gunn type splitting as written by Briley and McDonald (1977). The boundary conditions are also treated implicitly.

Artificial viscosity, or smoothing, is normally added to the solution algorithm to damp pre- and post-shock oscillations in supersonic flow, and to prevent odd-even decoupling due to the use of central differences in convection-dominated regions of the flow. Implicit smoothing and two types of explicit smoothing are available in *Proteus*. The implicit smoothing is second order with constant coefficients. For the explicit smoothing the user may choose a constant coefficient second- and/or fourth-order model (Steger, 1978), or a nonlinear coefficient mixed second- and fourth-order model (Jameson, Schmidt, and Turkel, 1981). The nonlinear coefficient model was designed specifically for flow with shock waves.

The equations are fully coupled, leading to a system of equations with a block tridiagonal coefficient matrix that can be solved using the block matrix version of the Thomas algorithm. Because this algorithm is recursive, the source code cannot be vectorized in the ADI sweep direction. However, it is vectorized in the non-sweep direction,

leading to an efficient implementation of the algorithm.

3.2 GEOMETRY AND GRID SYSTEM

The equations solved in *Proteus* were originally written in a Cartesian coordinate system, then transformed into a general nonorthogonal computational coordinate system. The code is therefore not limited to any particular type of geometry or coordinate system. The only requirement is that body-fitted coordinates must be used. In general, the computational coordinate system for a particular geometry must be created by a separate coordinate generation code and stored in an unformatted file that *Proteus* can read. However, simple Cartesian and polar coordinate systems are built in.

The equations are solved at grid points that form a computational mesh within this computational coordinate system. The number of grid points in each direction in the computational mesh is specified by the user. The location of these grid points can be varied by packing them at either or both boundaries in any coordinate direction. The transformation metrics and Jacobian are computed using finite differences in a manner consistent with the differencing of the governing equations.

3.3 FLOW AND REFERENCE CONDITIONS

As stated earlier, the equations solved by *Proteus* are for compressible flow. Incompressible conditions can be simulated by running at a Mach number of around 0.1. Lower Mach numbers may lead to numerical problems. The flow can be laminar or turbulent. The gas constant $\bar{R}$ is specified by the user, with the value for air as the default. The specific heats c_p and c_v, the molecular viscosity μ, and the thermal conductivity k can be treated as constants or as functions of temperature. The empirical formulas used to relate these properties to temperature are contained in a separate subroutine, and can easily be modified if necessary. The perfect gas equation of state is used to relate pressure, density, and temperature. This equation is also contained in a separate subroutine, which could be easily modified if necessary. All equations and variables in the program are nondimensionalized by normalizing values derived from reference conditions specified by the user, with values for sea level air as the default.

3.4 BOUNDARY CONDITIONS

The easiest way to specify boundary conditions in *Proteus* is by specifying the type of boundary (e.g., no-slip adiabatic wall, subsonic inflow, periodic, etc.). The program will then select an appropriate set of conditions for that boundary. For many applications this method should be sufficient. If necessary, however, the user may instead set the individual boundary conditions on any or all of the computational boundaries.

A variety of individual boundary conditions are built into the *Proteus* code, including: (1) specified values and/or gradients of Cartesian velocities u, v, and w, normal and tangential velocities V_n and V_t, pressure p, temperature T, and density ρ; (2) specified values of total pressure p_T, total temperature T_T, and flow angle; and (3) linear extrapolation. Another useful boundary condition is a "no change from initial condition" option for u, v, w, p, T, ρ, p_T, and/or T_T. Provision is also made for user-written boundary conditions. Specified gradient boundary conditions may be in the direction of the coordinate line intersecting the boundary or normal to the boundary, and may be computed using two-point or three-point difference formulas. For all of these conditions, the same type and value may be applied over the entire boundary surface, or a point-by-point distribution may be specified. Unsteady and time-periodic boundary conditions are allowed when applied over the entire boundary.

3.5 INITIAL CONDITIONS

Initial conditions are required throughout the flow field to start the time marching procedure. For unsteady flows they should represent a real flow field. A converged steady-state solution from a previous run would be a good choice. For steady flows, the ideal initial conditions would represent a real flow field that is close to the expected final solution.

The best choice for initial conditions, therefore, will vary from problem to problem. For this reason *Proteus* does not include a general-purpose routine for setting up initial conditions. The user must supply a subroutine, called INIT, that sets up the initial starting conditions for the time marching procedure. A version of INIT is, however, built into *Proteus* that specifies uniform flow with constant flow properties everywhere in the flow field. These conditions, of course, do represent a solution to the governing equations, and for many problems may help minimize starting transients in the time marching procedure. However, realistic initial conditions that are closer to the expected final solution should lead to quicker convergence.

3.6 TIME STEP SELECTION

Several different options are available for choosing the time step $\Delta\tau$, and for modifying it as the solution proceeds. $\Delta\tau$ may be specified directly, or through a value of the Courant-Friedrichs-Lewy (CFL) number. When specifying a CFL number, the time step may be either *global* (i.e., constant in space) based on the minimum CFL limit, or *local* (i.e., varying in space) based on the local CFL limit. For unsteady time-accurate flows global values should be used, but for steady flows using local values may lead to faster convergence. Options are available to increase or decrease $\Delta\tau$ as the solution proceeds based on the change in the dependent variables. An option is also available to cycle $\Delta\tau$ between two values in a logarithmic progression over a specified number of time steps.

3.7 CONVERGENCE

Five options are currently available for determining convergence. The user specifies a convergence criterion ε for each of the governing equations. Then, depending on the option chosen, convergence is based on: (1) the absolute value of the maximum change in the conservation variables ΔQ_{max} over a single time step; (2) the absolute value of the maximum change ΔQ_{max} averaged over a specified number of time steps; (3) the L_2 norm of the residual for each equation; (4) the average residual for each equation; or (5) the maximum residual for each equation.

It should be noted, however, that convergence is in the eye of the beholder. The amount of decrease in the residual necessary for convergence will vary from problem to problem. For some problems, it may be more appropriate to measure convergence by some flow-related parameter, such as the lift coefficient for an airfoil. Determining when a solution is sufficiently converged is, in some respects, a skill best acquired through experience.

3.8 INPUT/OUTPUT

Input to *Proteus* is through a series of namelists and, in general, an unformatted file containing the computational coordinate system. All of the input parameters have default values and only need to be specified by the user if a different value is desired. Reference conditions may be specified in either English or SI units. A restart option is also available, in which the computational mesh and the initial flow field are read from unformatted restart files created during an earlier run.

The standard printed output available in *Proteus* includes an echo of the input, boundary conditions, normalizing and reference conditions, the computed flow field, and convergence information. The user controls exactly which flow field parameters are printed, and at which time levels and grid points. Several debug options are also available for detailed printout in various parts of the program.

In addition to the printed output, several unformatted files can be written for various purposes. The first is an auxiliary file used for post-processing, usually called a plot file, that can be written at convergence or after the last time step if the solution does not converge. Plot files can be written for the NASA Lewis plotting program *CONTOUR* or the NASA Ames plotting program *PLOT3D*. If *PLOT3D* is to be used, two unformatted files are created, an *xyz* file containing the computational mesh and a *q* file containing the computed flow field. Another unformatted file written by *Proteus* contains detailed convergence information. This file is automatically incremented each time the solution is checked for convergence, and is used to generate the convergence history printout and with Lewis-developed post-processing plotting routines. And finally, two unformatted files may be written at the end of a calculation that may be used to restart the calculation in a later run. One of these contains the computational mesh, and the other the computed flow field.

3.9 TURBULENCE MODELS

For turbulent flow, *Proteus* solves the Reynolds time-averaged Navier-Stokes equations, with turbulence modeled using either the Baldwin and Lomax (1978) algebraic eddy-viscosity model or the Chien (1982) two-equation model.

3.9.1 Baldwin-Lomax Model. The Baldwin-Lomax model may be applied to either wall-bounded flows or to free turbulent flows. For wall-bounded flows, the model is a two-layer model. For flows in which more than one boundary is a solid surface, averaging procedures are used to determine a single μ_t profile. The turbulent thermal conductivity coefficient k_t is computed using Reynolds analogy.

3.9.2 Chien k-ε Model. With the Chien two-equation model, partial differential equations are solved for the turbulent kinetic energy k and the turbulent dissipation rate ε. These equations are lagged in time and solved separately from the mean flow equations. In the 2-D *Proteus* code, the equations are solved using the same solution

algorithm as for the mean flow equations, except that spatial derivatives for the convective terms are approximated using first-order upwind differencing. In the 3-D code, they are solved by a two-sweep LU procedure, as described by Hoffmann (1989).

Since the Chien two-equation model is a low Reynolds number formulation, the k-ε equations are solved in the near-wall region. No additional approximations are needed. Boundary conditions that may be used include: (1) no change from initial or restart conditions for k and ε; (2) specified values and/or gradients of k and ε; and (3) linear extrapolation. Specified gradient boundary conditions are in the direction of the coordinate line intersecting the boundary, and may be computed using two-point or three-point difference formulas. For all of these conditions, the same type and value may be applied over the entire boundary surface, or a point-by-point distribution may be specified. Spatially periodic boundary conditions for k and ε may also be used. Unsteady boundary conditions are not available for the k-ε equations. However, unsteady flows can still be computed with the Chien model using the unsteady boundary condition option for the mean flow quantities and appropriate boundary conditions for k and ε, such as specified gradients or linear extrapolation.

Initial conditions for k and ε are required throughout the flow field to start the time marching procedure. The best choice for initial conditions will vary from problem to problem, and the user may supply a subroutine, called KEINIT, that sets up the initial values of k and ε for the time marching procedure. A version of KEINIT is built into *Proteus* that computes the initial values from a mean initial or restart flow field based on the assumption of local equilibrium (i.e., production equals dissipation.) Variations of that scheme have been found to be useful in computing initial k and ε values for a variety of turbulent flows.

The time step used in the solution of the k-ε equations is normally the same as the time step used for the mean flow equations. However, the user can alter the time step, making it larger or smaller than the time step for the mean flow equations, by specifying a multiplication factor. The user can also specify the number of k-ε iterations per mean flow iteration.

4. VERIFICATION CASES

Throughout the *Proteus* development effort, verification of the code has been emphasized. A variety of cases have been run, and the computed results have been compared with both experimental data and exact solutions. Some cases are included in Volume 2 of the *Proteus* documentation (Towne, Schwab, Benson, and Suresh, 1990). Other cases have been reported by Conley and Zeman (1991), Saunders and Keith (1991), and Bui (1992).

Three cases are presented in this paper — flow past a circular cylinder, flow through a transonic diffuser, and flow through a square–cross-sectioned S-duct.

4.1 FLOW PAST A CIRCULAR CYLINDER

In this test case, steady flow past a two-dimensional circular cylinder was investigated. Both Euler and laminar viscous flow were computed.

4.1.1 Reference Conditions. In order to allow comparison of the *Proteus* results with incompressible experimental data and with potential flow results, this case was run with a low reference Mach number of 0.2. The cylinder radius was used as the reference length, and was set equal to 1 ft. Standard sea level conditions of 519 °R and 0.07645 lb_m/ft^3 were used for the reference temperature and density. The Reynolds number based on cylinder diameter was 40, matching the experimental value.

4.1.2 Computational Coordinates. For this problem a polar computational coordinate system was the obvious choice. The radial coordinate r varied from 1 at the cylinder surface to 30 at the outer boundary. Since the flow is symmetric, only the top half of the flow field was computed. The circumferential coordinate θ thus varied from 0° at the cylinder leading edge to 180° at the trailing edge. For the Euler flow case, a 21 (circumferential) × 51 (radial) mesh was used, with the radial grid packed moderately tightly near the cylinder surface. For the viscous flow case, a 51×51 mesh was used, with the radial grid packed more tightly near the cylinder surface.

4.1.3 Initial Conditions. Constant stagnation enthalpy was assumed, so only three initial conditions were required. For the Euler flow case, uniform flow with $u = 1$, $v = 0$, and $p = 1$ was used.

For the viscous flow case, the exact potential flow solution was used to set the initial conditions at all the non-wall points. Thus, with nondimensional free stream conditions of $\rho_\infty = u_\infty = T_\infty = p_\infty = 1$, the initial conditions

were[2]

$$u = 1 - \frac{1}{r^2}\cos(2\theta)$$

$$v = -\frac{1}{r^2}\sin(2\theta)$$

$$p = (p_T)_\infty - \frac{1}{2}\frac{\rho_\infty(u^2+v^2)}{R}$$

where

$$(p_T)_\infty = p_\infty + \frac{1}{2}\frac{\rho_\infty u_\infty^2}{R}$$

At the cylinder surface, the initial velocities u and v were set equal to zero, and the pressure p was set equal to the pressure at the grid point adjacent to the surface. Thus, with two-point one-sided differencing, $\partial p/\partial n = 0$ at the surface.

4.1.4 Boundary Conditions. Again, since we assumed constant stagnation enthalpy, only three boundary conditions were required at each computational boundary. For the Euler flow case, symmetry conditions were used along the symmetry line ahead of and behind the cylinder. At the cylinder surface, the radial velocity and the radial gradient of the circumferential velocity were set equal to zero. The radial gradient of pressure was computed from the polar coordinate form of the incompressible radial momentum equation written at the wall. The equation is (Hughes and Gaylord, 1964)

$$\rho v_r \frac{\partial v_r}{\partial r} + \frac{\rho v_\theta}{r}\frac{\partial v_r}{\partial \theta} - \rho\frac{v_\theta^2}{r} = -\frac{\partial p}{\partial r}$$

where v_r and v_θ are the radial and circumferential velocities, respectively. At the cylinder surface, $v_r = 0$. Thus,

$$\frac{\partial p}{\partial r} = \rho\frac{v_\theta^2}{r} = \rho\frac{u^2+v^2}{r}$$

And finally, at the outer boundary the free stream conditions were specified as boundary conditions.

For the viscous flow case, symmetry conditions were again used along the symmetry line ahead of and behind the cylinder. At the cylinder surface, no-slip conditions were used for the velocity, and the radial pressure gradient was set equal to zero. The outer boundary was split into an inlet region and wake region. The split was made, somewhat arbitrarily, at $\theta = 135°$. In the inlet region, the boundary values of u, v, and p were kept at their initial values, which were the potential flow values. In the wake region, the boundary values of p were kept at their initial values, and the radial gradients of u and v were set equal to zero.

4.1.5 Numerics. Both the Euler and viscous flow cases were run using a spatially varying time step, with a local CFL number of 10. The constant coefficient artificial viscosity model was used, with $\varepsilon_I = 2$ and $\varepsilon_E^{(4)} = 1$.

The Euler flow case converged in 210 time steps, and the viscous flow case converged in 360 time steps. The convergence criterion for both cases was that the L_2 norm of the residual for each equation drop below 0.001.

2. Note that the nondimensional gas constant R appears in these equations. This is because, in the *Proteus* input and output, the pressure is nondimensionalized by $\rho_r \bar{R} T_r$. Internal to the code, pressure is nondimensionalized by $\rho_r u_r^2$, as described in Section 2.1.

4.1.6 Computed Results. In Figure 1 the computed static pressure coefficient, defined as $(\bar{p}-p_r)/(\rho_r u_r^2/2g_c)$ is plotted as a function of θ for both the Euler and viscous flow cases. Also shown are the experimental data of Grove, Shair, Petersen, and Acrivos (1964), and the exact solution for potential flow. The *Proteus* results agree well with the data for the viscous flow case, and with the exact potential flow solution for the Euler flow case.

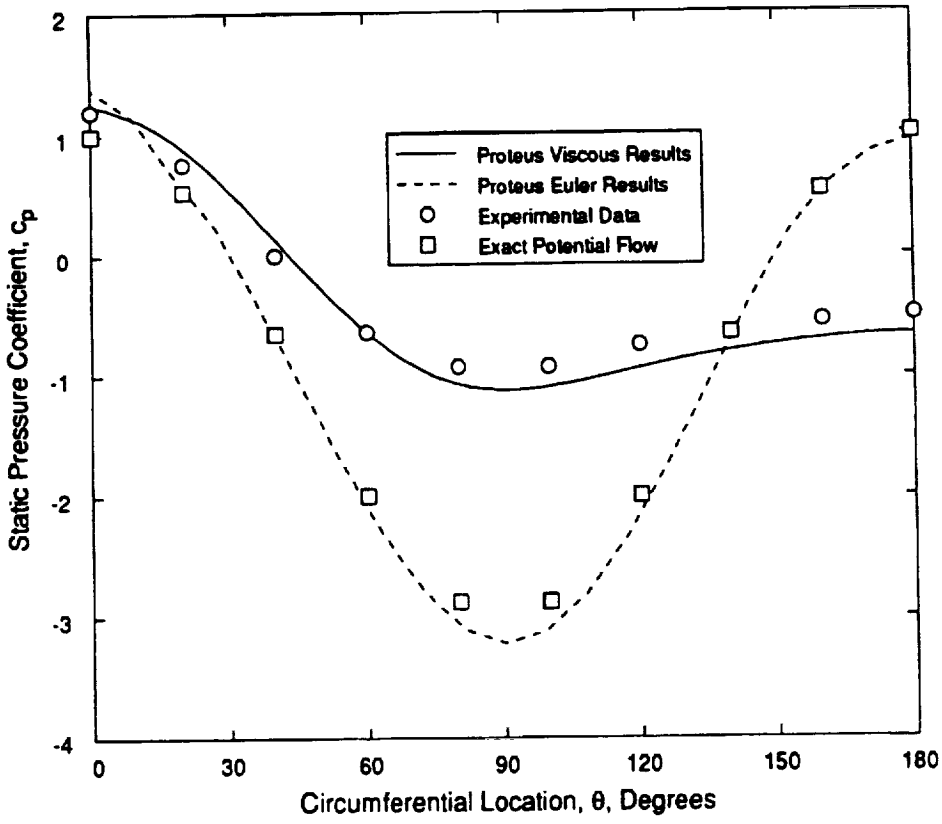

Figure 1. Pressure coefficient for flow past a circular cylinder.

4.2 TRANSONIC DIFFUSER FLOW

In this test case, two-dimensional transonic turbulent flow was computed in a converging-diverging duct. Turbulence was modeled using the Baldwin-Lomax model. The flow entered the duct subsonically, accelerated through the throat to supersonic speed, then decelerated through a normal shock and exited the duct subsonically. The computational domain is shown in Figure 2.

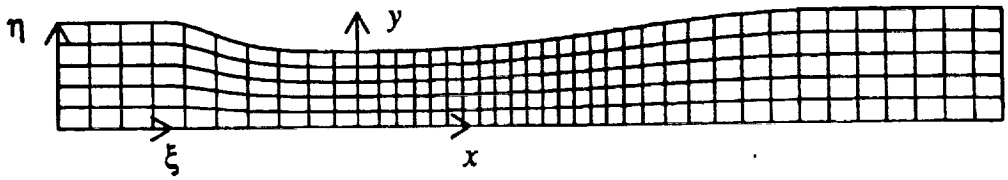

Figure 2. Computational domain for transonic diffuser flow.

4.2.1 Reference Conditions. The throat height of 0.14435 ft. was used as the reference length L_r. The reference velocity u_r was 100 ft/sec. The reference temperature and density were 525.602 °R and 0.1005 lb_m/ft^3, respectively. These values match the inlet total temperature and total pressure used in other numerical simulations of this

flow (Hsieh, Bogar, and Coakley, 1987).

4.2.2 Computational Coordinates. The x coordinate for this duct runs from –4.04 to +8.65. The Cartesian coordinates of the bottom wall are simply $y = 0$ for all x. For the top wall, the y coordinate is given by (Bogar, Sajben, and Kroutil, 1983)

$$y = \begin{cases} 1.4144 & \text{for } x \le -2.598 \\ \alpha \cosh\zeta / (\alpha - 1 + \cosh\zeta) & \text{for } -2.598 < x < 7.216 \\ 1.5 & \text{for } x \ge 7.216 \end{cases}$$

where the parameter ζ is defined as

$$\zeta = \frac{C_1 (x/x_l)[1 + C_2 x/x_l]^{C_3}}{(1 - x/x_l)^{C_4}}$$

The various constants used in the formula for the top wall height in the converging ($-2.598 \le x \le 0$) and diverging ($0 \le x \le 7.216$) parts of the duct are given in the following table.

Constant	Converging	Diverging
α	1.4114	1.5
x_l	–2.598	7.216
C_1	0.81	2.25
C_2	1.0	0.0
C_3	0.5	0.0
C_4	0.6	0.0

A body-fitted coordinate system was generated for the duct, with 81 points in the x direction and 51 points in the y direction. The coordinate system is shown in Figure 2. For clarity, the grid points are thinned by factors of 2 and 10 in the x and y directions, respectively. Note that for good resolution of the flow near the normal shock, the grid defining the computational coordinate system is denser in the x direction in the region just downstream of the throat. In the y direction, the actual computational mesh was tightly packed near both walls to resolve the turbulent boundary layers.

4.2.3 Initial Conditions. The initial conditions were simply zero velocity and constant pressure and temperature. Thus, $u = v = 0$ and $p = T = 1$ everywhere in the flow field.

4.2.4 Boundary Conditions. This calculation was performed in three separate runs. In the first run, the exit static pressure was gradually lowered to a value low enough to establish supersonic flow throughout the diverging portion of the duct. The pressure was lowered as follows:

$$p(t) = \begin{cases} 0.99 & \text{for } 1 < n \le 100 \\ -2.1405 \times 10^{-3} n + 1.20405 & \text{for } 101 \le n \le 500 \\ 0.1338 & \text{for } 501 \le n \le 3001 \end{cases}$$

where n is the time level. The equation for p for $101 \le n \le 500$ is simply a linear interpolation between $p = 0.99$ and $p = 0.1338$. In the second run, the exit pressure was gradually raised to a value consistent with the formation of a normal shock just downstream of the throat. Thus,

$$p(t) = \begin{cases} 3.4327 \times 10^{-4} n - 0.89636 & \text{for } 3001 < n \le 5000 \\ 0.82 & \text{for } 5001 \le n \le 6001 \end{cases}$$

Again, the equation for p for $3001 < n \le 5000$ is simply a linear interpolation between $p = 0.1338$ and $p = 0.82$. In the third run, the exit pressure was kept constant at 0.82 for $6001 < n \le 9000$.

The remaining boundary conditions were the same for all runs. At the inlet, the total pressure and total temperature were set equal to 1, and the y-velocity and the normal gradient of the x-velocity were both set equal to zero. At the exit, the normal gradients of temperature and both velocity components were set equal to zero. At both walls, no-slip adiabatic conditions were used, and the normal pressure gradient was set equal to zero.

4.2.5 Numerics. The case was run using a spatially varying time step. The local CFL number was 0.5 for the first two runs, and 5.0 for the third run. The nonlinear coefficient artificial viscosity model was used. For the first two runs, the coefficients $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ were 0.1 and 0.005, respectively. For the third run, $\varepsilon^{(4)}$ was lowered to 0.0004.

The convergence criterion was that the absolute value of the maximum change in the conservation variables ΔQ_{max} be less than 10^{-6}. At the end of the third run, the solution had not yet converged to this level. However, close examination of several parameters near the end of the calculation indicates that the solution is no longer changing appreciably with time, but oscillates slightly about some mean steady level. This type of result appears to be fairly common, especially for flows with shock waves. The reason is not entirely clear, but may be related to inadequate mesh resolution, discontinuities in metric information, etc. For this particular case, the cause may also be inherent unsteadiness in the flow. The experimental data for this duct show a self-sustained oscillation of the normal shock at Mach numbers greater than about 1.3 (Bogar, Sajben, and Kroutil, 1983).

4.2.6 Computed Results. The computed flow field is shown in Figure 3 in the form of constant Mach number contours.

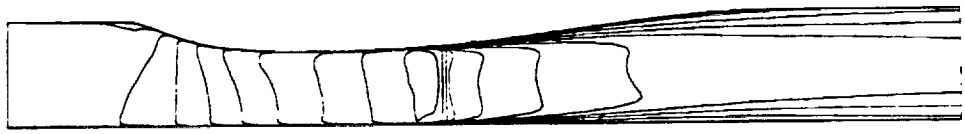

Figure 3. Computed Mach number contours for transonic diffuser flow.

The flow enters the duct at about $M = 0.46$, accelerates to just under $M = 1.3$ slightly downstream of the throat, shocks down to about $M = 0.78$, then decelerates and leaves the duct at about $M = 0.51$. The normal shock in the throat region and the growing boundary layers in the diverging section can be seen clearly. Because this is a shock capturing analysis, the normal shock is smeared in the streamwise direction.

The computed distribution of the static pressure ratio along the top and bottom walls is compared with experimental data (Hsieh, Wardlaw, Collins, and Coakley, 1987) in Figure 4. The static pressure ratio is here defined as $p/(p_T)_0$, where $(p_T)_0$ is the inlet core total pressure.

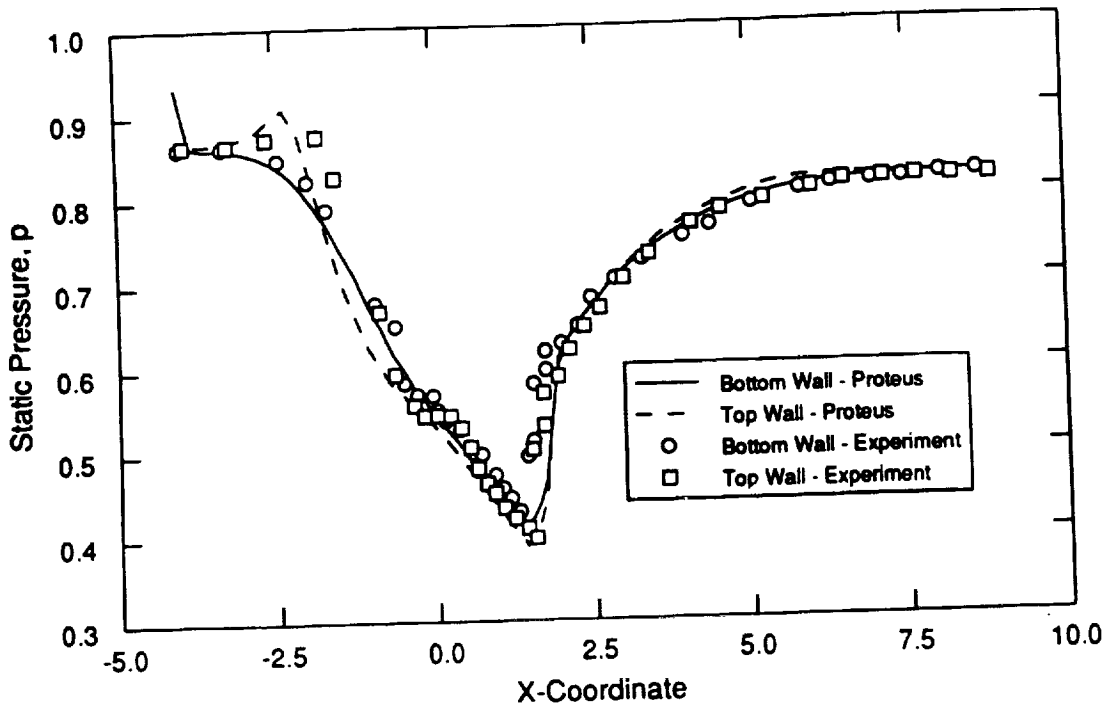

Figure 4. Computed and experimental static pressure distribution for transonic diffuser flow.

The computed results generally agree well with the experimental data, including the jump conditions across the normal shock. The predicted shock position, however, is slightly downstream of the experimentally measured position. The pressure change, of course, is also smeared over a finite distance. There is also some disagreement between analysis and experiment along the top wall near the inlet. This may be due to rapid changes in the wall contour in this region without sufficient mesh resolution.

4.3 TURBULENT S-DUCT FLOW

In this test case, three-dimensional turbulent flow in an S-duct was computed using first the Baldwin-Lomax algebraic turbulence model and then the Chien k-ε turbulence model. The S-duct consisted of two 22.5° bends with a constant area square cross section. The geometry and experimental data were obtained from a test conducted by Taylor, Whitelaw, and Yianneskis (1982).

4.3.1 Reference Conditions. The default standard sea level conditions for air of 519 °R and 0.07645 lb_m/ft^3 were used for the reference temperature and density. The specific heat ratio γ_r was set to 1.4. Since the experiment was incompressible, the reference Mach number M_r was set equal to 0.2 to minimize compressibility effects and, at the same time, achieve a reasonable convergence rate with the *Proteus* code. In the experiment, the Reynolds number based on the bulk velocity and the hydraulic diameter was 40,000. This value was therefore used as the reference Reynolds number Re_r in the calculation. The reference length L_r was set equal to 0.028658 ft. This value was computed from the definition of Re_r, where M_r and Sutherland's law were used to compute a_r and μ_r, respectively.

4.3.2 Computational Coordinates. Figure 5 illustrates the computational grid for the S-duct, created using the GRIDGEN codes (Steinbrenner, Chawner, and Fouts, 1991). For clarity, the grid is shown only on three of the computational boundaries, and the points have been thinned by a factor of two in each direction. The boundary grids were first created using the GRIDGEN 2D program. The 3-D volumetric grid was then generated from the boundary grids using GRIDGEN 3D.

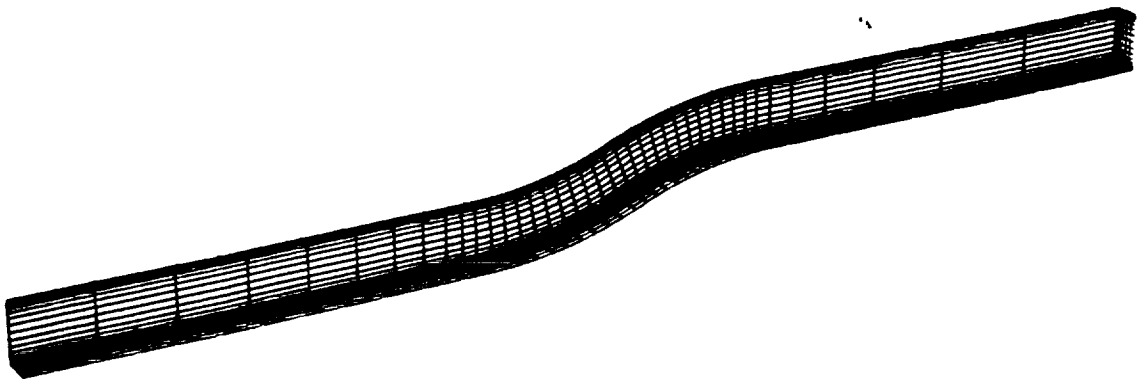

Figure 5. S-duct computational grid.

The computational grid extended from 7.5 hydraulic diameters upstream of the start of the first bend, to 7.5 hydraulic diameters downstream of the end of the second bend. The grid consisted of $81 \times 31 \times 61$ points in the ξ, η, and ζ directions, respectively. Since the S-duct is symmetric with respect to the $\eta = 1$ plane, only half of the duct was computed. To resolve the viscous layers, grid points were tightly packed near the solid walls using the default packing option in GRIDGEN 2D. At the grid point nearest the wall, the value of y^+ was about 0.5.

4.3.3 Initial Conditions. The computations were done in two separate major steps: a calculation using the Baldwin-Lomax turbulence model and a calculation using the Chien k-ε model. To start the Baldwin-Lomax calculations, the default initial profiles specified in subroutine INIT were used. Thus, the static pressure p was set equal to 1.0, and the velocity components u, v, and w were set equal to 0.0 everywhere in the duct. To start the Chien k-ε calculations, the initial values of u, v, w, p, and the turbulent viscosity μ_t were obtained from the Baldwin-Lomax solution. The initial values of k and ε were obtained using the default KEINIT subroutine in *Proteus*.

4.3.4 Boundary Conditions. For both calculations, constant stagnation enthalpy was assumed, eliminating the need for solving the energy equation. Therefore, only four boundary conditions were required for the mean flow at each computational boundary. In addition, for the Chien calculation, boundary conditions were required for k and ε at each computational boundary.

For the Baldwin-Lomax calculation, at the duct inlet the total pressure was specified as 1.02828, the gradient of u was set equal to zero, and the velocities v and w were set equal to zero. The inlet total pressure was calculated from the freestream static pressure and the reference Mach number using isentropic relations. At the duct exit, the static pressure was specified as 0.98416, and the gradients of u, v, and w were set equal to zero. The exit static pressure was found by trial and error in order to match the experimental mass flow rate. At the walls of the duct no-slip conditions were used for the velocities, and the normal pressure gradient was set to zero. Symmetry conditions were used in the symmetry plane.

For the Chien calculation, the boundary conditions for the mean flow were the same as for the Baldwin-Lomax calculation, with one exception. At the duct exit, the value of the static pressure was changed slightly, from 0.98416 to 0.98474, again in order to match the experimental mass flow rate. For the k-ε equations, at the upstream boundary the gradients of the turbulent kinetic energy k and the turbulent dissipation rate ε were set equal to zero for the first 20 time steps. After that time, the values of k and ε were kept constant. At the downstream boundary, the gradients of k and ε were set equal to zero. No-slip conditions were used at the solid boundaries, and symmetry conditions were used at the symmetry boundary.

4.3.5 Numerics. Both the Baldwin-Lomax and Chien calculations were run using a spatially varying time step. Since the flow field for the Baldwin-Lomax calculation was impulsively started from zero velocity everywhere, large CFL numbers specified at the very beginning of the calculation might result in an unphysical flow field and cause the calculation to blow up. Therefore, the calculations were run with a CFL number of 1 for the first 100 iterations, 5 for the next 200 iterations, and 10 for the remaining iterations. A total of 4,000 iterations was used for the Baldwin-Lomax calculation.

For the Chien case, a small CFL number was again used at the beginning of the calculation. The calculations were run with a CFL number of 1 for the first 120 iterations, 5 for the next 500 iterations, and 10 for the remaining iterations. A total of 2,520 iterations was used for the Chien calculation.

The constant coefficient artificial viscosity model was used for both cases, with $\varepsilon_I = 2$ and $\varepsilon_E^{(4)} = 1$.

The convergence criterion was that the average residual for each equation be less than 10^{-6}. However, both calculations were stopped before reaching this level of convergence when examination of several flow-related parameters indicated that the solution was no longer changing appreciably with time. The average residual at the end of the Baldwin-Lomax calculation ranged from 10^{-3} for the x-momentum equation to 3×10^{-5} for the continuity equation. For the Chien calculation the values were 3×10^{-4} for the x-momentum equation and 5×10^{-6} for the continuity equation. For both cases the residuals were continuing to drop when the calculations were stopped.

4.3.6 Computed Results. In Figure 6, the computed flow field from the Chien calculation is shown in the form of total pressure contours at five stations through the duct. (The upstream and downstream straight sections are not shown.) As the flow enters the first bend, the boundary layer at the bottom of the duct initially thickens due to the locally adverse pressure gradient in that region. In an S-duct, the high pressure at the outside (bottom) of the first bend drives the low energy boundary layer toward the inside (top) of the bend, while the core flow responds to centrifugal effects and moves toward the outside (bottom) of the bend. The result is a pair of counter-rotating secondary flow vortices in the upper half of the cross-section. These secondary flows cause a significant amount of flow distortion, as shown by the total pressure contours.

In the second bend, the direction of the cross-flow pressure gradients reverses, making the pressure higher in the upper half of the cross-section. However, the flow enters the second bend with a vortex pattern already established. The net effect is to tighten and concentrate the existing vortices near the top of the duct, in agreement with classical secondary flow theory. The resulting horseshoe-shaped distortion pattern at the exit of the second bend is typical of S-duct flows.

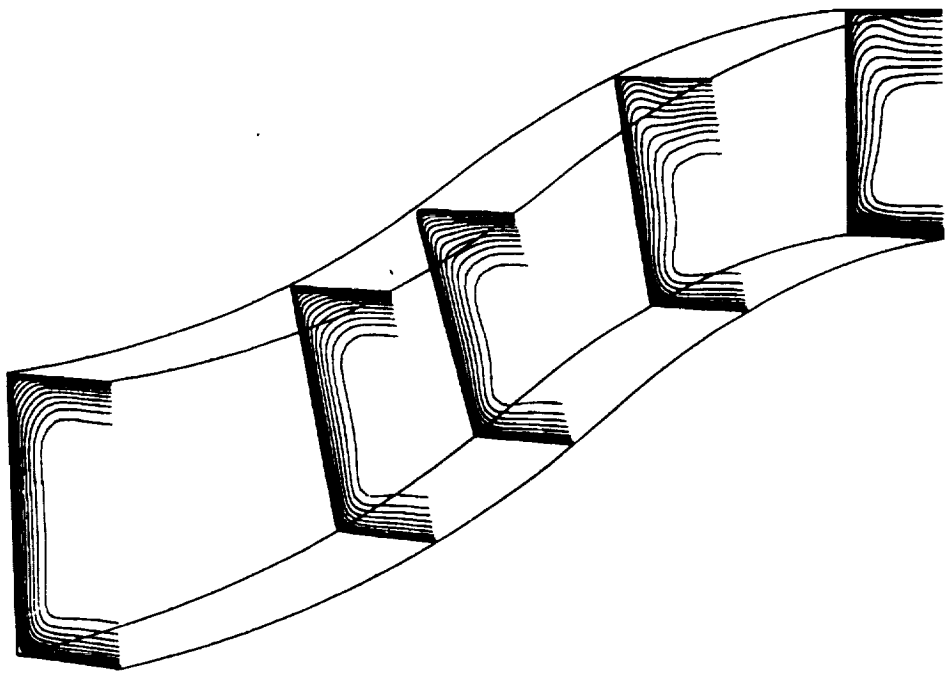

Figure 6. Computed total pressure contours for turbulent S-duct flow.

In Figure 7, the calculated wall pressure distribution is compared with the experimental data of Taylor, Whitelaw, and Yianneskis (1982). The agreement is very good. Both turbulence models correctly predicted the pressure trend and the pressure loss along the duct. The r and z coordinates noted in the legend are the same as those

defined by Taylor, Whitelaw, and Yianneskis.

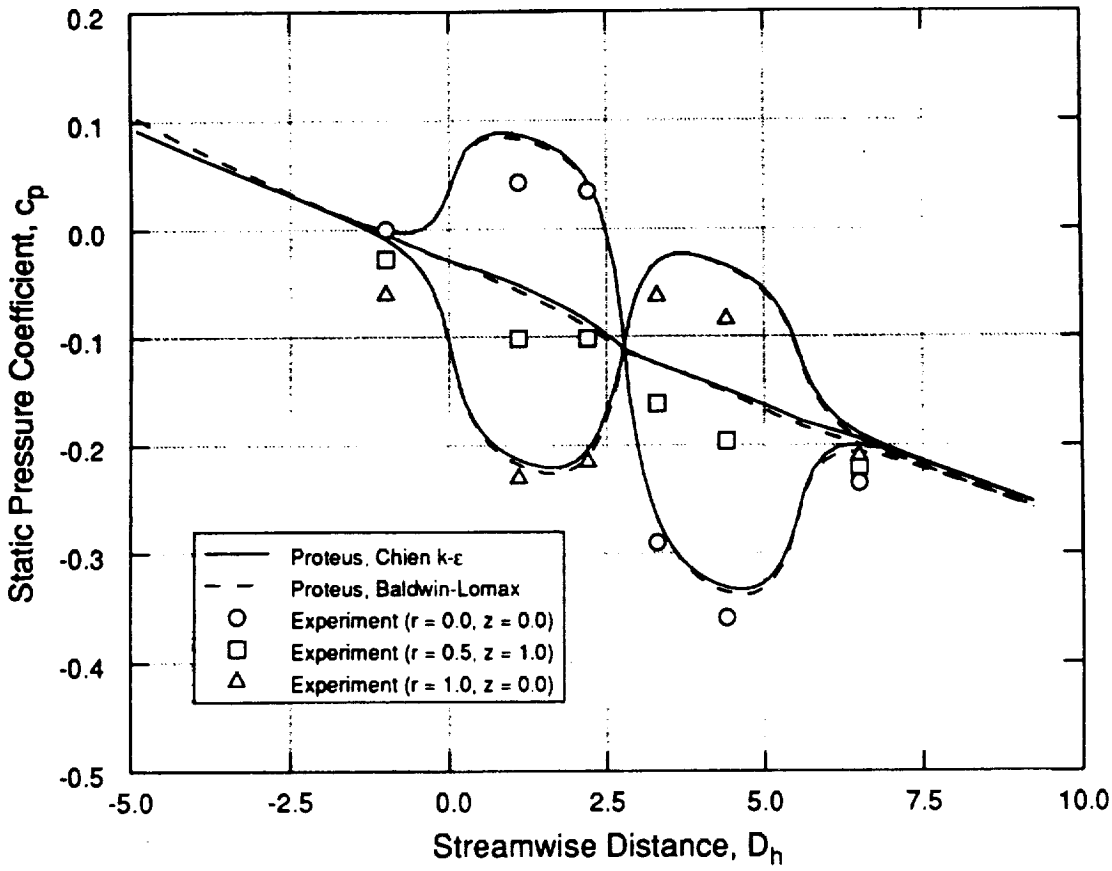

Figure 7. Computed surface static pressure distribution for turbulent S-duct flow.

In Figure 8, the experimental and computed velocity profiles in the symmetry plane are shown for the five streamwise stations that were surveyed in the experiment. These survey stations are at the same locations as the total pressure contours shown in Figure 6. The agreement between computation and experiment is excellent for both turbulence models. The asymmetry in the velocity profiles due to the pressure induced secondary motion is correctly predicted.

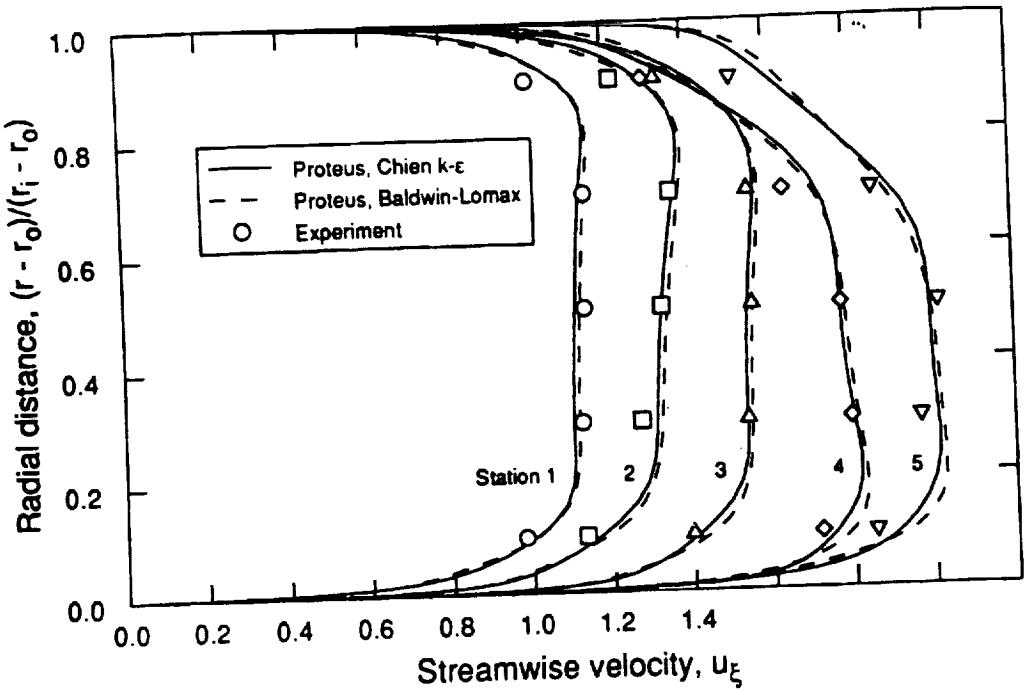

Figure 8. Computed streamwise velocity profiles for turbulent S-duct flow.

5. CONCLUDING REMARKS

The *Proteus* two- and three-dimensional Navier-Stokes codes recently developed at NASA Lewis have been described, and results have been presented from some of the validation cases. Version 1.0 of the two-dimensional code was released in late 1989 (Towne, Schwab, Benson, and Suresh, 1990), and version 2.0 was released in late 1991. Version 1.0 of the three-dimensional code was released in early 1992. Documentation for version 2.0 of the two-dimensional code and for version 1.0 of the three-dimensional code is available, but has not yet been formally published.

Current development work on the *Proteus* codes is being done to add a multiple-zone grid capability, a multi-grid convergence acceleration capability, and additional turbulence modeling options.

A wide variety of validation cases have been run, including: (1) several simplified flows for which exact Navier-Stokes solutions exist; (2) laminar and turbulent flat plate boundary layer flows; (3) two- and three-dimensional driven cavity flows; (4) flows with normal and oblique shock waves; (5) steady and unsteady flows past a cylinder; (6) developing laminar and turbulent flows in channels, pipes, and rectangular ducts; (7) steady and unsteady flows in a transonic diffuser; (8) flows in curved and S-shaped ducts; and (9) turbulent flow on a flat plate with a glancing shock wave. Current and future validation cases will emphasize three-dimensional duct flows and flows with heat transfer.

REFERENCES

Anderson, D. A., Tannehill, J. C., and Pletcher, R. H. (1984) *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, McGraw-Hill Book Company, New York.

Baldwin, B. S., and Lomax, H. (1978) "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper 78-257.

Beam, R. M., and Warming, R. F. (1978) "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," AIAA Journal, Vol. 16, No. 4, pp. 393–402.

Bogar, T. J., Sajben, M., and Kroutil, J. C. (1983) "Characteristic Frequencies of Transonic Diffuser Flow Oscillations," AIAA Journal, Vol. 21, No. 9, pp. 1232–1240.

Briley, W. R., and McDonald, H. (1977) "Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, pp. 373–397.

Bui, T. T. (1992) "Implementation/Validation of a Low Reynolds Number Two-Equation Turbulence Model in the Proteus Navier-Stokes Code — Two-Dimensional/Axisymmetric," NASA TM 105619.

Chien, K. Y. (1982) "Prediction of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model," AIAA Journal, Vol. 20, No. 1, pp. 33–38.

Conley, J. M., and Zeman, P. L. (1991) "Verification of the Proteus Two-Dimensional Navier-Stokes Code for Flat Plate and Pipe Flows," AIAA Paper 91-2013 (also NASA TM 105160).

Grove, A. S., Shair, F. H., Petersen, E. E., and Acrivos, A. (1964) "An Experimental Investigation of the Steady Separated Flow Past a Circular Cylinder," Journal of Fluid Mechanics, Vol. 19, pp. 60–80.

Hoffmann, K. A. (1989) *Computational Fluid Dynamics for Engineers*, Engineering Educational System, Austin, Texas.

Hsieh, T., Bogar, T. J., and Coakley, T. J. (1987) "Numerical Simulation and Comparison with Experiment for Self-Excited Oscillations in a Diffuser Flow," AIAA Journal, Vol. 25, No. 7, pp. 936–943.

Hsieh, T., Wardlaw, A. B., Jr., Collins, P., and Coakley, T. J. (1987) "Numerical Investigation of Unsteady Inlet Flow Fields," AIAA Journal, Vol. 25, No. 1, pp. 75–81.

Hughes, W. F., and Gaylord, E. W. (1964) *Basic Equations of Engineering Science*, Schaum's Outline Series, McGraw-Hill Book Company, New York.

Jameson, A., Schmidt, W., and Turkel, E. (1981) "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259.

Pulliam, T. H. (1986) "Artificial Dissipation Models for the Euler Equations," AIAA Journal, Vol. 24, No. 12, pp. 1931–1940.

Saunders, J. D., and Keith, Jr., T. G. (1991) "Results from Computational Analysis of a Mixed Compression Supersonic Inlet," AIAA Paper 91-2581.

Steger, J. L. (1978) "Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries," AIAA Journal, Vol. 16, No. 7, pp. 679–686.

Steinbrenner, J. P., Chawner, J. P., and Fouts, C. L. (1991) "The GRIDGEN 3D Multiple Block Grid Generation System, Volume II: User's Manual," WRDC-TR-90-3022, Vol. II.

Taylor, A. M. K. P., Whitelaw, J. H., and Yianneskis, M. (1982) "Developing Flow in S-Shaped Ducts," NASA CR 3550.

Towne, C. E., Schwab, J. R., Benson, T. J., and Suresh, A. (1990) "PROTEUS Two-Dimensional Navier-Stokes Computer Code — Version 1.0, Volumes 1–3," NASA TM's 102551–3.