# High Performance Computing for Compressible Turbulent Flow

*Ekachai Juntasaro[1], Putchong Uthayopas[2],*
*Boonlue Sawatmongkhon[1] and Khongthep Boonmee[2]*

*[1]Computational Fluid Dynamics Laboratory (CFD Lab),*
*School of Mechanical Engineering, Institute of Engineering,*
*Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand.*
*Email: junta@ccs.sut.ac.th, Phone: (66 44) 224410 – 2, Fax: (66 44) 224411*

*[2]Parallel Research Group (PRG),*
*Department of Computer Engineering, Faculty of Engineering,*
*Kasetsart University, Bangkok 10900, Thailand.*
*Email: pu@ku.ac.th, Phone: (66 2) 9428555 ext 1416, Fax: (66 2) 5796245*

**ABSTRACT -- The aim of the present research and development work is to develop the computer program to simulate the steady two-dimensional compressible turbulent flow. The finite volume method is used to numerically solve the flow governing equations. The Navier-Stokes equations are solved for the velocity field and the SIMPLE algorithm is used to adjust the velocity field to satisfy the conservation law of mass. Since all the variables are stored at the center of each control volume, the Rhie-Chow interpolation is used to avoid the decoupling between the velocity and the pressure. The corrected velocity field is used to solve the $k-$ and $\varepsilon-$ equations. The eddy viscosity, that represents the influence of turbulence on the mean flow field, can then be calculated from those values of $k$ and $\varepsilon$ obtained. The energy equation is solved for the temperature field. The effects of temperature and pressure on the fluid density are taken into account via the equation of state. The boundary layer on a flat plate is employed as a test case because it is one of the standard benchmark problems for the validation of CFD software. The sequential-computing solver is first used to obtain the computed results. It is found that the computed results are in good agreement with the experimental data at subsonic speed. The parallel-computing solver is also implemented here and tested against the sequential-computing one. It is found that the parallel program can run faster than the sequential one up to 2.55 times for the best case. Furthermore, the governing equations are solved on the structured and body-fitted coordinates so that this computer program can be developed further for the simulation of flow over or inside any object of complex geometry in the future.**

**KEY WORDS** Compressible Turbulence Flow, High Performance Computing, Parallel Solver

## 1. Introduction

Fluid flow involves many advanced applications in science, engineering and technology. Understanding of the flow behavior is therefore important for the design and development of scientific and engineering innovations. In fluid dynamics, the flow behavior is governed by the continuity equation, the Navier-Stokes equations, the energy equation and the equation of state. For compressible flow, where the free-stream Mach number is higher than 0.3, the effects of temperature variation on fluid properties are so large that the fluid properties must be treated as variable.

To study turbulent flow, the continuity, Navier-Stokes, energy and state equations can be solved directly by any numerical method called Direct Numerical Simulation (DNS). However, the simulation requires the large number of grid points, volumes, elements or other form of sub-domains to capture the characteristics of turbulent flow. Therefore, the computation requires a supercomputer that have a large storage to store all essential data and good computing power to run the program as fast as possible. At present, the supercomputer can only provide the solution for the turbulent flow at low Reynolds number with simple geometries. In other words, the turbulent flow in engineering applications cannot practically be predicted and studied by this approach. In general, the turbulent flow is predicted and studied on the basis of mean quantities. By this way, the continuity, Navier-Stokes, energy and state equations are essentially time-averaged using the density-weighted technique. This technique gives rise to some extra unknown terms which need to be properly modeled. Turbulence models have been developed and widely used with success over a wide range of engineering applications.

The present work is aimed to develop the computer program for the simulation of steady two-dimensional compressible turbulent flow using the two-equation turbulence model. However, the recent emergence of Beowulf cluster computing technology, which is the use of commodity PC

and high speed network to build a cost effective supercomputer, has created a lot of interest around the world. The potential speed increases by parallelizing the CFD program to run on this plat form using portable standard programming such as MPI can be immense. Hence, there is a need to explore such technique to reduce the computation time and to increase productivity gained. Part of this work is performed according to that goal.

## 2. Governing Equations

Compressible flow is governed by the continuity, Navier-Stokes, energy and state equations where all the fluid properties are variable. For turbulent compressible flow, these governing equations are essentially time-averaged using the density-weighting technique and the resulting solution is the mean quantities. This technique gives rise to the extra unknown terms which cause a closure problem. This problem can be solved using an appropriate turbulence model. For steady two-dimensional mean flow, the governing equations with the turbulence model can be expressed in terms of tensor notation as follows:

### 2.1 Continuity Equation

$$\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j) = 0$$

where $\bar{\rho}$ is the fluid density and $\tilde{u}_j$ is the flow velocity.

### 2.2 Navier-Stokes Equations

$$\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j\tilde{u}_i) = \frac{\partial}{\partial x_j}(\bar{t}_{ij} + \tau_{ij}) - \frac{\partial \bar{P}}{\partial x_i}$$

where $\bar{P}$ is the pressure, and $\bar{t}_{ij}$ and $\tau_{ij}$ are the laminar- and turbulent-flow stresses respectively with the following definitions:

$$\bar{t}_{ij} = \mu\left[\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{u}_k}{\partial x_k}\right]$$

where $\mu$ is the fluid viscosity and $\delta_{ij}$ is the Kronecker delta: $\delta_{ij} = 0$ for $i \neq j$ and $\delta_{ij} = 1$ for $i = j$, and

$$\tau_{ij} = \mu_t\left[\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{u}_k}{\partial x_k}\right] - \frac{2}{3}\delta_{ij}\bar{\rho}k$$

where $\mu_t$ is the eddy viscosity and $k$ is the kinetic energy of turbulence.

### 2.3 Two-Equation Turbulence Model of Launder and Sharma (1974)

$$\mu_t = \bar{\rho}c_\mu f_\mu \frac{k^2}{\varepsilon}$$

where $c_\mu$ is the model constant, $f_\mu$ is the damping function, and $\varepsilon$ is the dissipation rate of $k$. The transport equations for $k$ and $\varepsilon$ are modelled as follows:

$$\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j k) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + \tau_{ij}\frac{\partial \tilde{u}_i}{\partial x_j} - \bar{\rho}\varepsilon + \bar{\rho}D$$

where $\sigma_k$ is the model constant and $D$ is the extra term. For compressible turbulent flow, $\varepsilon$ is split into two parts: $\varepsilon_s$ (Solenoidal dissipation rate of $k$) and $\varepsilon_d$ (Dilatation dissipation rate of $k$). Thus,

$$\varepsilon = \varepsilon_s + \varepsilon_d$$

Sarkar, Erlebacher, Hussaini & Kreiss (1991) have proposed that

$$\varepsilon_d = M_t^2\varepsilon_s$$

where $M_t$ is the turbulent Mach number defined as

$$M_t^2 = 2\frac{k}{a^2}$$

with $a$ is the speed of sound. $\varepsilon_s$ is calculated from the following equation:

$$\frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_j\varepsilon_s) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon_s}{\partial x_j}\right] + c_{\varepsilon 1}f_{\varepsilon 1}\frac{\varepsilon_s}{k}\tau_{ij}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) - \bar{\rho}c_{\varepsilon 2}f_{\varepsilon 2}\frac{\varepsilon_s^2}{k} - \frac{4}{3}\bar{\rho}\varepsilon_s\frac{\partial \tilde{u}_k}{\partial x_k} + \bar{\rho}E$$

where ( $\sigma_\varepsilon$ , $c_{\varepsilon 1}$ , $c_{\varepsilon 2}$ ) are the model constants, ( $f_{\varepsilon 1}$ , $f_{\varepsilon 2}$ ) are the damping functions, and $E$ is the extra term.

For the $k-\varepsilon$ turbulence model of Launder and Sharma (1974), the model constants, damping functions and extra terms are provided as follows:

$$c_\mu = 0.09\,, \quad \sigma_k = 1.0\,, \quad \sigma_\varepsilon = 1.3\,, \quad c_{\varepsilon 1} = 1.44\,, \quad c_{\varepsilon 2} = 1.92\,,$$

$$f_\mu = \exp\left[\frac{-3.4}{\left(1+\frac{R_t}{50}\right)^2}\right], f_{\varepsilon 1} = 1.0\,,$$

$$f_{\varepsilon 2} = 1 - 0.3\exp\left(-R_t^2\right),$$

$$D = -2\frac{\mu}{\overline{\rho}}\left(\frac{\partial\sqrt{k}}{\partial x_i}\right)^2, \quad \text{and} \quad E = 2\frac{\mu}{\overline{\rho}}\frac{\mu_t}{\overline{\rho}}\left(\frac{\partial^2\tilde{u}_i}{\partial x_j \partial x_k}\right)^2$$

where $R_t = \dfrac{\overline{\rho}k^2}{\mu\varepsilon}$.

### 2.4 Energy Equation

$$\frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{e}_T) = \frac{\partial}{\partial x_j}\left[\left(\frac{k_T}{c_v}+\frac{\mu_t}{Pr_t}\right)\frac{\partial\tilde{e}_T}{\partial x_j}\right] + \frac{\partial}{\partial x_j}\left[\tilde{u}_i\left(\bar{t}_{ij}+\tau_{ij}\right)\right] - \frac{\partial}{\partial x_j}\left(\tilde{u}_j\overline{P}\right) + \frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] - \frac{\partial}{\partial x_j}\left[\left(\frac{k_T}{c_v}+\frac{\mu_t}{Pr_t}\right)\frac{\partial}{\partial x_j}(K+k)\right]$$

where $k_T$ is the thermal conductivity, $c_v$ is the specific heat at constant volume, $Pr_t$ is the turbulent Prandtl number taken as 0.91, and $\tilde{e}_t$ is the total energy which is defined as $\tilde{e}_t = \tilde{e} + K + k$

where $\tilde{e}$ is the internal energy ( $\tilde{e} = c_v\tilde{T}$ where $\tilde{T}$ is the temperature) and $K$ is the kinetic energy of the mean flow, i.e. $K = 0.5(\tilde{u}^2 + \tilde{v}^2)$ .

### 2.5 Equation of State

$$\overline{P} = (\gamma-1)\overline{\rho}\left(\tilde{e}_t - K - k\right)$$

where $\gamma$ is the specific heat ratio.

The fluid properties, $\mu$ and $k_T$, of compressible flow can be influenced by the variation of the temperature so that they must be defined in terms of temperature as the following relations:

### 2.6 Sutherland's Law

$$\mu = \mu_\infty\left(\frac{\tilde{T}}{T_\infty}\right)^{3/2}\frac{T_\infty + 110}{\tilde{T}+110}$$

where the subscript $\infty$ denotes the value at free-stream.

### 2.7 Prandtl Number

$$Pr = \frac{\mu c_p}{k_T}$$

where $c_p$ is the specific heat at constant pressure.

## 3. Numerical Method

The finite volume method is used to numerically solve the governing equations which can be written in a general form as follows:

$$\frac{\partial}{\partial x_i}\left(\rho\bar{u}_i\phi\right) = \frac{\partial}{\partial x_i}\left(\Gamma\frac{\partial\phi}{\partial x_i}\right) + S^\phi$$

where $\phi$ is the general dependent variable, $\Gamma$ is the effective diffusion coefficient, and $S^\phi$ is the source/sink term of $\phi$. To be able to simulate the internal flow with variable cross-sectional area and the external flow past an object of complex shape, the general form of the governing equations is essentially transformed from the physical

domain $(x, y)$ into the computational domain $(\xi, \eta)$ as the following equation:

$$\frac{\partial}{\partial\xi}(\rho U\phi)+\frac{\partial}{\partial\eta}(\rho V\phi)=\frac{\partial}{\partial\xi}\left[\frac{\Gamma}{J}\left(\alpha\frac{\partial\phi}{\partial\xi}-\beta\frac{\partial\phi}{\partial\eta}\right)\right]+\frac{\partial}{\partial\eta}\left[\frac{\Gamma}{J}\left(\gamma\frac{\partial\phi}{\partial\eta}-\beta\frac{\partial\phi}{\partial\xi}\right)\right]+JS^{\phi}$$

where

$$U=\bar{u}\frac{\partial y}{\partial\eta}-\bar{v}\frac{\partial x}{\partial\eta},\ V=\bar{v}\frac{\partial x}{\partial\xi}-\bar{u}\frac{\partial y}{\partial\xi},$$

$$\alpha=\left(\frac{\partial x}{\partial\eta}\right)^2+\left(\frac{\partial y}{\partial\eta}\right)^2,\beta=\frac{\partial x}{\partial\xi}\frac{\partial x}{\partial\eta}+\frac{\partial y}{\partial\xi}\frac{\partial y}{\partial\eta},$$

$$\gamma=\left(\frac{\partial x}{\partial\xi}\right)^2+\left(\frac{\partial y}{\partial\xi}\right)^2,\text{ and } J=\frac{\partial x}{\partial\xi}\frac{\partial y}{\partial\eta}-\frac{\partial y}{\partial\xi}\frac{\partial x}{\partial\eta}.$$

Using the finite volume method, the computational domain is divided into a number of control volumes. The transformed equations can be integrated as follows:

$$[(\rho U\Delta\eta)\phi]_w^e+[(\rho V\Delta\xi)\phi]_s^n=\left[\frac{\Gamma\Delta\eta}{J}\left(\alpha\frac{\partial\phi}{\partial\xi}-\beta\frac{\partial\phi}{\partial\eta}\right)\right]_w^e+\left[\frac{\Gamma\Delta\xi}{J}\left(\gamma\frac{\partial\phi}{\partial\eta}-\beta\frac{\partial\phi}{\partial\xi}\right)\right]_s^n+(J\Delta\xi\Delta\eta)\bar{S}_P^{\phi}$$

where $\bar{S}_P^{\phi}$ is the mean value of $S^{\phi}$ at the center P of each control volume, and ($e$, $w$, $n$, $s$) are the east, west, north and south faces of each control volume. The convection terms are approximated by the first-order upwind differencing scheme and the diffusion terms are estimated by the second-order central differencing scheme. Therefore, the standard form of the finite volume equation can be obtained as

$$A_P^{\phi}\phi_P=A_E^{\phi}\phi_E+A_W^{\phi}\phi_W+A_N^{\phi}\phi_N+A_S^{\phi}\phi_S+b^{\phi}$$

where

$$A_E^{\phi}=\left(\frac{\Gamma}{J}\alpha\frac{\Delta\eta}{\Delta\xi}\right)_e+\max\left[0,-(\rho U\Delta\eta)_e\right],$$

$$A_W^{\phi}=\left(\frac{\Gamma}{J}\alpha\frac{\Delta\eta}{\Delta\xi}\right)_w+\max\left[0,(\rho U\Delta\eta)_w\right],$$

$$A_N^{\phi}=\left(\frac{\Gamma}{J}\gamma\frac{\Delta\xi}{\Delta\eta}\right)_n+\max\left[0,-(\rho V\Delta\xi)_n\right],$$

$$A_S^{\phi}=\left(\frac{\Gamma}{J}\gamma\frac{\Delta\xi}{\Delta\eta}\right)_s+\max\left[0,(\rho V\Delta\xi)_s\right],$$

$A_P^{\phi}=A_E^{\phi}+A_W^{\phi}+A_N^{\phi}+A_S^{\phi}$, and

$$b^{\phi}=(J\Delta\xi\Delta\eta)\bar{S}_P^{\phi}-\left[\frac{\Gamma\Delta\eta}{J}\left(\beta\frac{\partial\phi}{\partial\eta}\right)\right]_w^e-\left[\frac{\Gamma\Delta\xi}{J}\left(\beta\frac{\partial\phi}{\partial\xi}\right)\right]_s^n.$$

The standard SIMPLE algorithm is employed here to satisfy the conservation law of mass. The continuity equation is not solved directly with other governing equations. The $p'$-equation is solved instead to obtain the pressure correction $p'$ and its value is used to correct the values of pressure and velocities to satisfy the conservation law of mass. The $p'$-equation can be written in a standard form as follows:

$$A_P^p p'_P=A_E^p p'_E+A_W^p p'_W+A_N^p p'_N+A_S^p p'_S+m_P$$

where

$$A_E^p=\left(\rho B\frac{\Delta\eta}{\Delta\xi}\right)_e\ A_W^p=\left(\rho B\frac{\Delta\eta}{\Delta\xi}\right)_w,$$

$$A_N^p=\left(\rho C\frac{\Delta\xi}{\Delta\eta}\right)_n,\ A_S^p=\left(\rho C\frac{\Delta\xi}{\Delta\eta}\right)_s,$$

$A_P^p=A_E^p+A_W^p+A_N^p+A_S^p$, and

$$m_P=\left(\rho U^*\Delta\eta\right)_e-\left(\rho U^*\Delta\eta\right)_w+\left(\rho V^*\Delta\xi\right)_n-\left(\rho V^*\Delta\xi\right)_s.$$

$U^*$, $V^*$ are calculated from the resulting velocities of the Navier-Stokes equations, whereas

$$B=B^u\frac{\partial y}{\partial\eta}-B^v\frac{\partial x}{\partial\eta},\text{ and }C=C^v\frac{\partial x}{\partial\xi}-C^u\frac{\partial y}{\partial\xi}$$

where

$$B^u = -\frac{\Delta\xi\Delta\eta}{A_P^u}\frac{\partial y}{\partial \eta}, B^v = \frac{\Delta\xi\Delta\eta}{A_P^v}\frac{\partial x}{\partial \eta},$$

$$C^u = \frac{\Delta\xi\Delta\eta}{A_P^u}\frac{\partial y}{\partial \xi}, \text{ and } C^v = -\frac{\Delta\xi\Delta\eta}{A_P^v}\frac{\partial x}{\partial \xi}.$$

In general, the standard SIMPLE algorithm is implemented on the staggered grid system to prevent the decoupling between the velocity and the pressure. However, the staggered grid system is technically rather complicated for programming and requires a large amount of computer storage. This drawback becomes obvious when the computer program is developed further for real-world applications. The collocated grid system is employed in this work so that all the variables are stored at the center of each control volume. The problem of velocity-pressure decoupling is solved by the Rhie-Chow interpolation where ($U_e^*$, $U_w^*$, $V_n^*$, $V_s^*$) are calculated from the appropriate pressure gradient.

In the current work, the boundary layer on a flat plate is chosen as a test case and the implementation of the SIMPLE algorithm and the Rhie-Chow interpolation must be slightly changed to suit the flow problem. Physically, the pressure field of this flow is constant and presumably known throughout the flow domain. Therefore, the pressure correction $p'$ obtained is used to correct the velocities only, not to correct the pressure, because the pressure itself is already known and constant. Moreover, the Rhie-Chow interpolation is simplified to the linear interpolation of ($U_e^*$, $U_w^*$, $V_n^*$, $V_s^*$) between grid nodes without any effect of pressure gradient.

The algorithm for the simulation of turbulent compressible flow can be summarized as follows:

(1) Start the computation with an initial guess of velocities, pressure correction, turbulence kinetic energy, dissipation rate of turbulence kinetic energy, temperature, density and viscosity
(2) Calculate the Navier-Stokes equations for the velocities
(3) Calculate the $p'$-equation for the pressure correction
(4) Correct the velocities by the pressure correction
(5) Calculate the $k$-equation for the turbulence kinetic energy
(6) Calculate the $\varepsilon$-equation for the dissipation rate of turbulence kinetic energy
(7) Calculate the energy equation for the total energy, and hence temperature
(8) Calculate the density from the equation of state, then the viscosity from Sutherland's law, and the thermal conductivity from the definition of Prandtl number
(9) Repeat from step (2) until the solution converges

# 4. Development of Parallel Computer Program

Parallel computing is a technique of partitioning long computation tasks into many sub-tasks that execute concurrently on multiple computers. Many useful techniques and algorithms can be found in parallel computing text such as [14][15]. In general, to partition a sequential program to run on cluster system, this program has to be analyzed using the profiling program called "gprof" to discover the most compute intensive part. Below is some example captured from the result of gprof profiling.

```
%  cumulative  self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 30.81    678.05   678.05 2224080000     0.30     0.30  NavierStokes::Diff(int,
int, double **, char, char)

 28.87   1313.32   635.27     1500 423513.33 661914.16
NavierStokes::EnergyEquation(void)

 11.15   1558.71   245.39     1500 163593.33 359517.83
NavierStokes::Momentum(double **, double **)

  5.63   1682.62   123.91      800 154887.50 326566.29
NavierStokes::epslEquation(void)

  4.75   1787.12   104.50     1500 69666.67 456301.14
NavierStokes::SIMPLE(void)
```

The obtained results show that "Diff" method are the most using and calling in this program. However, Diff is a common subroutine that is called from other method. It turns out that "EnergyEquation" method call "Diff" method more than other method. Another long computation method is "Momentum" method, but "Momentum" method is called from "SIMPLE". Thus the decision are made to parallelize the "EnergyEquation" ,"Momentum" and "SIMPLE" method. But some methods have relation with three methods. Thus, we must to parallelize them. The main code consists of two levels of iteration. For the first iteration, program computes the one equation until variable DeltaOverallResidual less than $10^{-12}$. Within this loop has eight methods as SIMPLE, UpdateUV, OneEquation, UpdateMu_tOneEq, EnergyEquation, EquationOfState, SutherlandLaw and CalThermalConductivity. Every 50 iterations will compute DeltaOverallResidual for using check condition in iteration. When program finish first iteration will prepare initialized data for second iteration. Second iteration, program will compute by use two-equation method until DeltaOverallResidual less than $10^{-14}$. Within loop have five methods as SIMPLE, UpdateUV, kEquation, epslEquation, UpdateMu_tTwoEq, EnergyEquation, EquationOfState, SutherlandLaw and CalThermalConductivity. The pseudo codes of two main loops are as follows.

```
do{
        ++iteration;
        Turbulent.SIMPLE();
        Turbulent.UpdateUV();
        Turbulent.OneEquation();
        Turbulent.UpdateMu_tOneEq();
        Turbulent.EnergyEquation();
        Turbulent.EquationOfState();
        Turbulent.SutherlandLaw();
        Turbulent.CalThermalConductivity();
        if( iteration > 0  &&  iteration%50 == 0 ){
                Turbulent.CalOverallResidual();
                OverallResidualNew =
Turbulent.GetOverallResidual();
                DeltaOverallResidual = fabs(
                OverallResidualNew - OverallResidualOld
                );
                OverallResidualOld = OverallResidualNew;
                printf("\n%6d  OE %.2e ",iteration,
DeltaOverallResidual);
                Turbulent.Display();
        }
}while( DeltaOverallResidual > pow(10.0, -12.0) );
printf("\n\tSwitch to Two-Equations");
do{
        ++iteration;
        Turbulent.SIMPLE();
        Turbulent.UpdateUV();
        Turbulent.kEquation();
        Turbulent.epslEquation();
        Turbulent.UpdateMu_tTwoEq();
        Turbulent.EnergyEquation();
        Turbulent.EquationOfState();
        Turbulent.SutherlandLaw();
        Turbulent.CalThermalConductivity();
        if( iteration > 0  &&  iteration%50 == 0 ){
                Turbulent.CalOverallResidual();
                OverallResidualNew =
Turbulent.GetOverallResidual();
                DeltaOverallResidual = fabs(
                OverallResidualNew - OverallResidualOld
                );
                OverallResidualOld = OverallResidualNew;
                printf("\n%6d  OE %.2e ",iteration,
DeltaOverallResidual);
                Turbulent.Display();
        `
```

In order to parallelize this code, we divide the data into several parts using the column base partitioning approach. The important issue involved in data partitioning is to distribute the data to all nodes in a balanced fashion. A simple way of partitioning data can be used effectively here. For this program, the number of column is divided by number of computing node. If there is any column left, number of column in each node will be increases by one column. Thus, each node will have the same amount of data number to be processes.

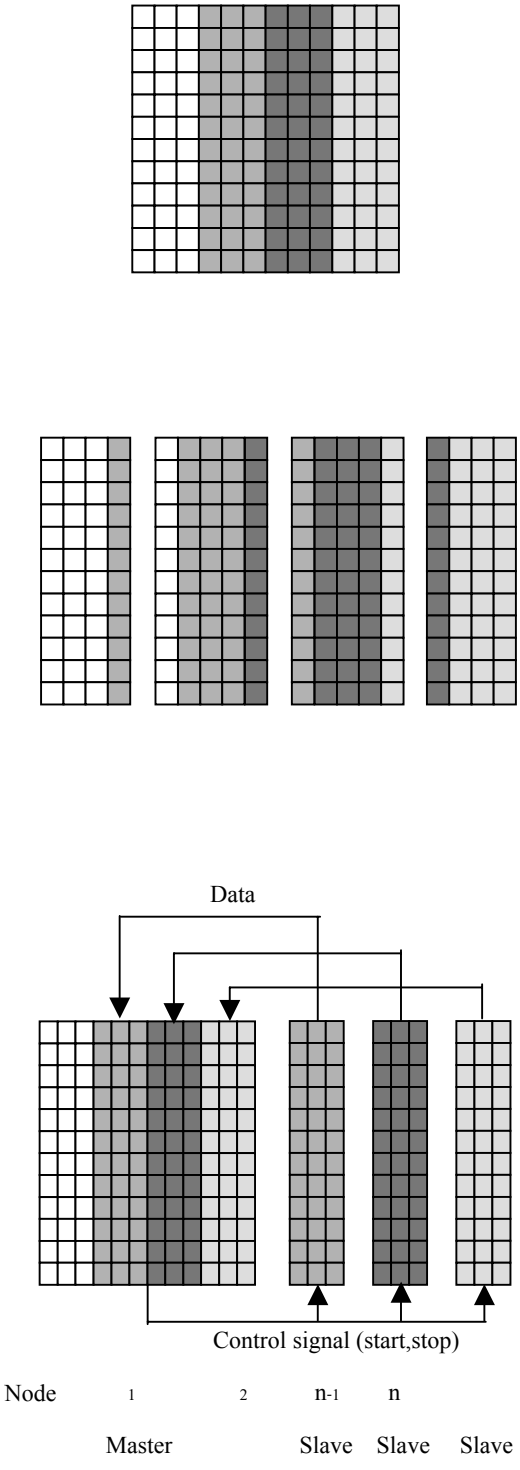



*Figure 3. Communication Model*

### 4.1 Communication Model

For our approach, we divide the computing node into two types: master node and slave node. The master node controls the processing step of parallel task. It send signal to slave node to start the computing process. Every 50 iterations, master node will receive the error data that will be used to compute the overall error. When overall error is lower than the minimum value required, master node will send signal to slave node to stop the computing process. Master node also computes the result using data that it has. In contrast, the main function of slave nodes is to compute the data that belong to them. When they receive the start signal from master node, slave node will start the computing process. When they receive stop signal from master node, they will stop the computing process and send result to master node.

### 4.2 Exchanging Data

The parallelization of this code has been done using MPI standard and MPICH [16][17] implementation from Argonne National Laboratory. For this program, there are many exchange of boundary data between nodes. This is done using MPI::COMM_WORLD.Send and MPI::COMM_WORLD.Receive primitive in MPI. The example statements excerpt from the code are as shown below.

```
void SendRight(int var_id,NavierStokes *Turbulent,double
*buffer,int rank)
{
        long number;
        Turbulent->GetArr(var_id,buffer,&number,Turbulent-
        >endxCV-6,Turbulent-> endxCV);
        MPI::COMM_WORLD.Send(&number,1,MPI::LONG,ran
        k+1,NUMTAG+100+rank+1);
        MPI::COMM_WORLD.Send(buffer,number,
        MPI::DOUBLE,rank+1,NUMTAG+100+rank+1);
}

void ReceiveLeft(int var_id,NavierStokes *Turbulent,double
*buffer,int rank)
{
        MPI::Status status;
        long number;
        MPI::COMM_WORLD.Recv(&number,1,MPI::LONG,
        rank-1,NUMTAG+100+rank,status);
        MPI::COMM_WORLD.Recv(buffer,number,
        MPI::DOUBLE,rank-1, NUMTAG+100+rank ,status);
        Turbulent->SetArr(var_id,buffer,Turbulent->startxCV-
```

In this computation, we assume that each computing node is ordered from left to right. The node that has rank less than other will be located on the left side. The first node locates on the left side of second node and so on. Next step is to identify the communication pattern by locating the variables that need to be updated. In the first loop consist of 8 method, every method modify the value of variables. Example of this is the SIMPLE method that computes the value of u, v, pCrtn, uCrtn and vCrtn. Thus, each node must exchange value of u, v, pCrtn, uCrtn and vCrtn. In the second loop, it contains 9 methods that update the value of variables. Thus, each node must exchange this data properly.

## 5. Results and Discussion

Computations are conducted for laminar and turbulent compressible flows, and input data are summarized in Table 1.

*Table 1. Input Data*

| Parameter | Compressible Laminar Flow | Compressible Turbulent Flow |
|---|---|---|
| $\xi_{max}$ | 151 | 151 |
| $\eta_{max}$ | 151 | 151 |
| $Re_L$ | 2,000,000 | 19,500,000 |
| $M_\infty$ | 0.4, 0.6, 0.8 | 0.824 |
| $T_\infty$ (K) | 300 | 300 |
| $P_\infty$ (Pa) | 101,325.0 | 110,995.5 |
| $T_W$ (K) | 300 | Adiabatic Recovery Temperature |
| Relaxation Factor | 0.5 | 0.5 |

where $\xi_{max}$ and $\eta_{max}$ are the numbers of grid lines used on the computational domain, $Re_L$ is the Reynolds number based on the length of the flat plate and the free-stream velocity, $M_\infty$ is the free-stream Mach number, $T_\infty$ is the free-stream temperature, $P_\infty$ is the free-stream pressure, $T_W$ is the wall temperature and the relaxation factor is used to stabilize the numerical scheme used.

### 5.1 Laminar Compressible Flow

Figure 4 shows the velocity distributions in which the numerical solutions are compared with the analytical solutions at three free-stream Mach numbers. The definitions of the normalized cross-stream distance and stream wise velocity are $y / \sqrt{x\mu_\infty / \rho_\infty u_\infty}$ and $u / u_\infty$ respectively. It is found that the numerical solutions are in very good agreement with the analytical solutions at all free-stream Mach numbers considered. For subsonic flow where the free-stream Mach number is as high as 0.8, the velocity distribution is not influenced by the Mach number. Physically, the compressibility effect is so little that its effect does not appear on the velocity distribution of the flow.

Figure 5 illustrates the temperature distributions where the numerical solutions are compared with the analytical solutions at three free-stream Mach numbers. The definition of the normalized temperarure is $T / T_\infty$ whereas the normalized cross-stream distance has the same definition as in Figure 4. The numerical solutions compared well with the analytical solutions at all three free-stream Mach numbers. The difference between the numerical solution and the analytical solution is larger as the Mach number is higher. The maximum temperature is higher as the Mach number

increases, that is, from about 0.5% at Mach 0.4 to around 2.5% at Mach 0.8.

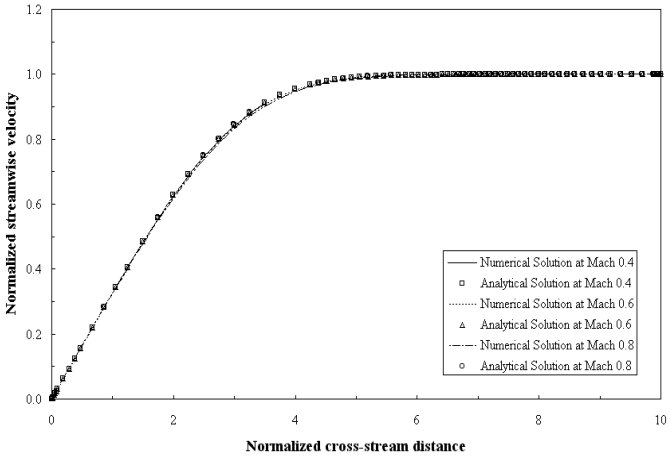



*Figure 4. Velocity distributions of the laminar boundary layer on a flat plate*

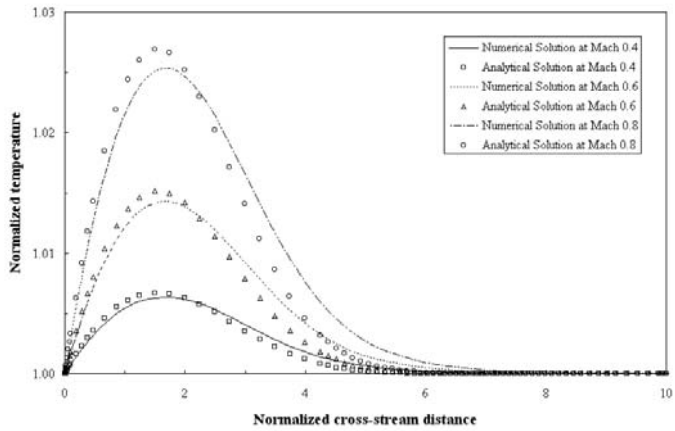



*Figure 5. Temperature distributions of the laminar boundary layer on a flat plate*

## 5.2 Turbulent Compressible Flow

Figures 6(a) and 6(b) show the velocity distributions of the turbulent boundary layer on a flat plate at Mach 0.824 where the numerical solution is compared with the law of the wall in Figure 6(a) while the experimental data of Motallebi (1994) is compared with the law of the wall in Figure 6(b). It is found that both the numerical solution and the Motallebi data are in good agreement with the law of the wall in a log-linear region where $5 < \ln(y\rho_w u_\tau / \mu_w) < 8$. In both figures, $u^*$ is the transformed velocity, which is defined by the van Driest transformation as follows:

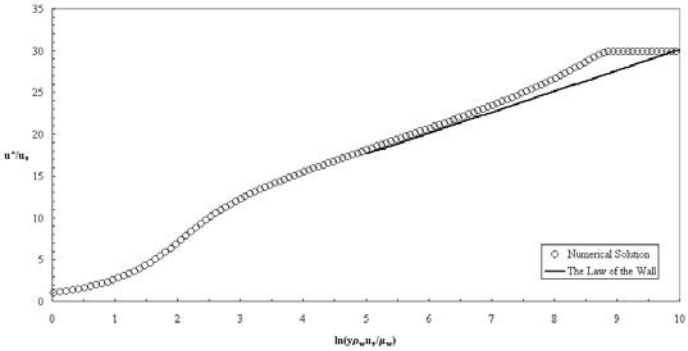
$$u^* = \frac{\tilde{u}_\delta}{b}\sin^{-1}\left[\frac{2b^2\frac{\tilde{u}}{\tilde{u}_\delta} - a}{\sqrt{a^2 + 4b^2}}\right]$$

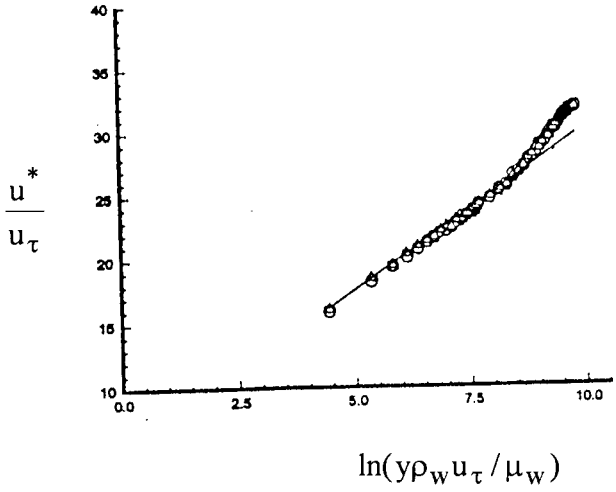
where

$$a = \frac{\tilde{T}_\delta}{\tilde{T}_w}\left[1 + r\frac{\gamma - 1}{2}M_\delta^2\right] - 1, \text{ and}$$

$$b^2 = r\frac{\gamma - 1}{2}M_\delta^2\frac{\tilde{T}_\delta}{\tilde{T}_w}$$

with $r$ is the recovery factor ($r = 0.89$ for turbulent flow) and the subscript $\delta$ denotes the edge of the boundary layer.

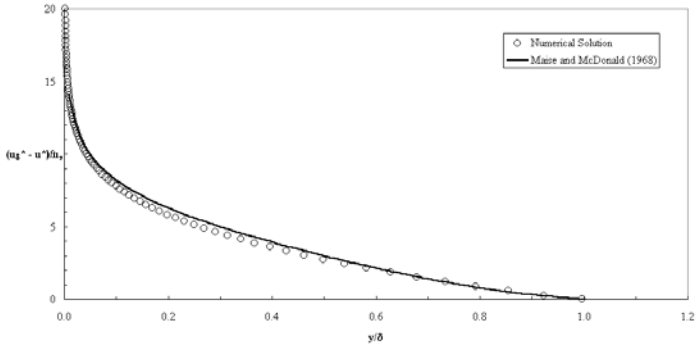


Figures 7(a) and 7(b) show the comparisons of the numerical solution and the experimental data of Motallebi (1994) with the following Maise and McDonald correlation respectively:

$$\frac{u_\delta^* - u^*}{u_\tau} = -2.5 \ln \frac{y}{\delta} + 1.25\left[1 + \cos\left(\pi \frac{y}{\delta}\right)\right]$$

where $\delta$ is the boundary layer thickness and $u_\tau$ is the friction velocity, i.e. $u_\tau = \sqrt{\tau_w / \overline{\rho}_w}$. It is found that both the numerical solution and the Motallebi data compare very well with this correlation.



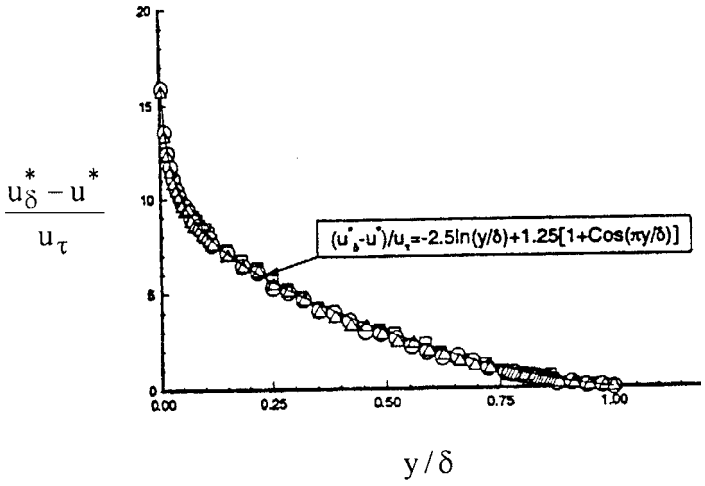




*Figure 7(b) Velocity distribution of the turbulent boundary layer on a flat plate;*
*Symbol for the experimental data of Motallebi (1994);*
*Line for the Maise and McDonald correlation*

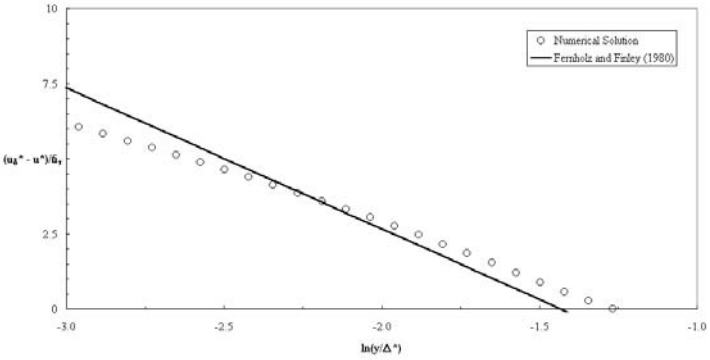
Figures 8(a) and 8(b) show the comparisons of the numerical solution and the experimental data of Motallebi (1994) with the following Fernholz and Finley correlation respectively:

$$\frac{u_\delta^* - u^*}{u_\tau} = -4.7 \ln \frac{y}{\Delta^*} - 6.74$$

where

$$\Delta^* = \delta \int_0^1 \left( \frac{u_\delta^* - u^*}{u_\tau} \right) d\left(\frac{y}{\delta}\right)$$

It is found that the numerical solution and the Motallebi data are reasonably well compared with the Fernholz and Finley correlation.



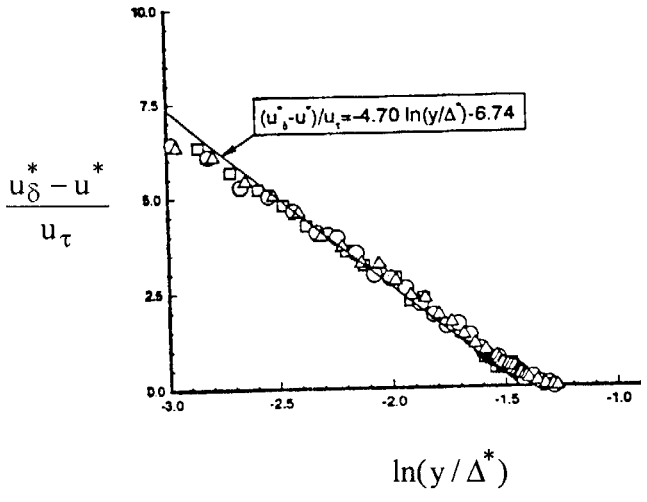




*Figure 8(b) Velocity distribution of the turbulent boundary layer on a flat plate;*
*Line for the Fernholz and Finley correlation*

### 5.3 Performance of Parallel Computer Program

To evaluate the performance of the system, parallel program has been tested on AMATA Beowulf system. This system consists of:

- 4 Athlon 950 MHz, 256 Mbytes RAM and 20 Gbytes Hard disk
- 4 Athlon 1GHz , 256 Mbytes RAM and 20 Gbytes Hard disk
- Fast Ethernet Switch Interconnection between nodes

First, the test has been conduct by running sequential program to measure the runtime. Then, parallel program has been run on 2, 4, and 8 nodes consequently. The runtime of parallel code has also been measured. The test has been repeated several times for several problem sizes. The results obtained are as depicted in Table 2. Also, the speedup curve has been plotted and illustrated in Figure 9.

*Table 2. Runtime results of the experiment*

| Test No. | Number of Grids | Sequential Runtime (seconds) | Parallel Runtime | | | Speedup | | |
|---|---|---|---|---|---|---|---|---|
| | | | 2 nodes | 4 nodes | 8 nodes | 2 nodes | 4 nodes | 8 nodes |
| 1 | 151*151 | 1065 | 1061.32 | 1206.02 | 1021.90 | 1.00 | 0.88 | 1.04 |
| 2 | 201*251 | 1856.6 | 1192.39 | 967.71 | 1236.52 | 1.56 | 1.92 | 1.50 |
| 3 | 251*151 | 3088 | 2142.85 | 1401.94 | 1373.27 | 1.44 | 2.20 | 2.25 |
| 4 | 301*151 | 5403 | 3230.65 | 2558.63 | 2506.05 | 1.67 | 2.11 | 2.16 |
| 5 | 351*151 | 7208 | 4645.10 | 2825.21 | 3161.77 | 1.55 | 2.55 | 2.28 |

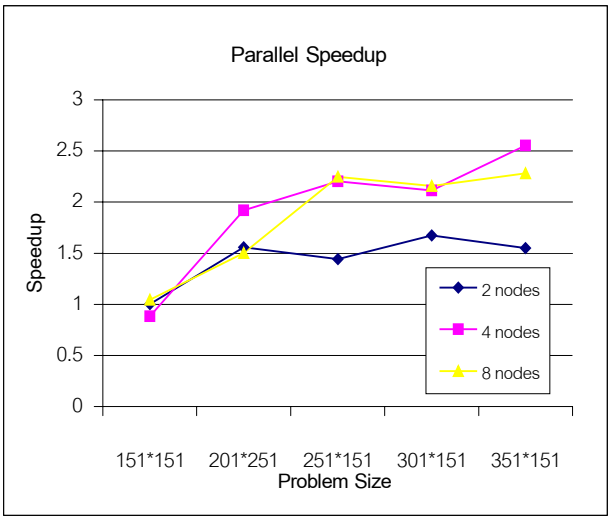



*Figure 9. Plot of speed up results*

From Figure 9, the parallel speed up shows this parallel algorithm receives the maximum speed when it run on 4 nodes. When number of computing node increase, speed up doesn't increase too. Because speed up of 8 computing nodes almost equal with speed up of 4 computing node. The maximum speed up obtained is as high as 2.55 times of the sequential execution speed.

However, as problem size increases, speed up will increase with it. But speed up doesn't increase follow to number of computing nodes. Thus this algorithm will improve computation/communication ratio for good performance.

## 6. Conclusions

Both laminar and turbulent compressible flows are simulated in the present work. The flow is considered at subsonic speed where the free-stream Mach number is as high as approximately 0.8. The numerical scheme of the computer program is validated using the laminar compressible boundary layer on a flat plate as a test case. It is found that the computer program is capable of simulating the laminar compressible boundary layers accurately at three free-stream Mach number considered. The turbulent compressible boundary layer on a flat plate at Mach 0.824 is used as a test case to validate the performance of the two-equation turbulence model of Launder and Sharma. It is found that the computer program can accurately simulate the turbulent compressible boundary layer at subsonic speed. The parallel implementation exhibits some moderate speedup. So, one of the most important future work is to analyze the performance of this parallel code and find out how to increases the speedup of this application.

## 7. Acknowledgements

Part of the present work of the first and third authors are financially supported from SUT Research Fund 55/2542. This support is greatly appreciated. We also would like to thank Dr.Varangrat Juntasaro for sharing her knowledge on the physics of compressible turbulent flow, and the modeling and numerical techniques. Some of the equipment used for performance assessment, especially the Athlon based cluster system, are supported by AMD Far East Inc. Part of the support is also from KURDI SRU Grant.

## References

[1] Fernholz, H.H., and Finley, P.J. (1980) "A Critical Commentary on Mean Flow Data for Two-Dimensional Compressible Turbulent Boundary Layers," AGARDograph 253.

[2] Gibson, M.M., Jones, W.P., and Whitelaw, J.H. (1992) "Turbulence Models for Computational Fluid Dynamics," Course Lecture Notes, 18-20 November 1992, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine.

[3] Gosman, A.D., and Issa, R.I. (1992) "Computational Fluid Dynamics," Post Experience Course, 16-18 November 1992, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine.

[4] Gropp, W., Lusk, E., and Thakur, R. (1999) "Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd edition", MIT Press.

[5] Gropp, W., Lusk, E., and Thakur, R., (1999) "Using MPI-2: Advanced Features of the Message-Passing Interface", MIT Press.

[6] Hinze, J.O. (1975) "Turbulence," 2nd edition, McGraw-Hill.

[7] Hutchings, B., and Iannuzzelli, R. (1987) "Benchmark Problems for Fluid Dynamics Codes," Mechanical Engineering, June, pp. 54-58.

[8] Juntasaro, E., and Sawatmongkhon, B. (1999) "Compressible Laminar Flow towards a Numerical Wind Tunnel," Proceedings of the 13th National Mechanical Engineering Conference, 2-3 December 1999, Royal Cliff Beach Resort Hotel, South Pattaya, Cholburi, Thailand, Vol. 1, pp. 132-137.

[9] Juntasaro, E., Uthayopas, P., Sawatmongkhon, B., and Boonmee, K. (2001) "High Performance Computing for Steady Two-Dimensional Turbulent Flow," Proceeding of the 5th Annual National Symposium on Computational Science and Engineering, 18-20 June 2001, Sofitel Central Plaza, Bangkok, Thailand, pp. 126-140.

[10] Karki, K.C., and Patankar, S.V. (1989) "Pressure Based Calculation Procedure for Viscous Flows at All Speeds in Arbitrary Configurations," AIAA Journal, Vol. 27, No. 9, pp. 1167-1174.

[11] Kumar, V., Grama, A., Gupta, A., and Karypis, G. (1994) "Introduction Parallel Computing Design and Analysis of Algorithms", Benjamin/Cummings.

[12] Lang, N.J., and Shih, T.H. (1991) "A Critical Comparison of Two-Equation Turbulence Models," NASA Technical Memorandum 105237.

[13] Launder, B.E., and Sharma, B.I. (1974) "Application of the Energy-Dissipation Model of Turbulence to the Calculation of a Flow near a Spinning Disk," Letters in Heat and Mass Transfer, Vol. 1, pp. 131-138.

[14] Maise, G., and McDonald, H. (1968) "Mixing Length and Kinematic Eddy Viscosity in a Compressible Boundary Layer," AIAA Journal, Vol. 6, No. 1, pp. 73-80.

[15] Motallebi, F. (1994) "Mean Flow Study of Two-Dimensional Subsonic Turbulent Boundary Layers," AIAA Journal, Vol. 32, No. 11, pp. 2153-2161.

[16] Motallebi, F. (1996) "Reynolds Number Effects on the Prediction of Mean Flow Data for Adiabatic 2-D Compressible Boundary Layers," Aeronautical Journal, Vol. 100, No. 992, pp. 53-59.

[17] Patankar, S.V. (1980) "Numerical Heat Transfer and Fluid Flow," Hemisphere.

[18] Patel, V.C., Rodi, W., and Scheuerer, G. (1985) "Turbulence Models for Near-Wall and Low Reynolds Number Flows: A Review," AIAA Journal, Vol. 23, No. 9, pp. 1308-1319.

[19] Rhie, C.M., and Chow, W.L. (1983) "Numerical Study of the Turbulent Flow past an Airfoil with Trailing Edge Separation," AIAA Journal, Vol. 21, No. 11, pp. 1525-1532.

[20] Sarkar, S., Erlebacher, G., Hussaini, M.Y., and Kreiss, H.O. (1991) "The Analysis and Modelling of Dilatational Terms in Compressible Turbulence," Journal of Fluid Mechanics, Vol. 227, pp. 473-493.

[21] Schlichting, H. (1979) "Boundary-Layer Theory," 7th edition, McGraw-Hill.

[22] Varangrat, S. (1999) "Computational Study of Compressible Flow in an S-Shaped Duct," Ph.D. Thesis, Department of Mechanical Engineering, Imperial College, University of London, U.K.

[23] Versteeg, H.K., and Malalasekera, W. (1995) "An Introduction to Computational Fluid Dynamics: The Finite Volume Method," Longman Scientific & Technical.

[24] White, F.M. (1991) "Viscous Fluid Flow," 2nd edition, McGraw-Hill.

[25] Wilcox, D.C. (1993) "Turbulence Modeling," DCW Industries.

[26] Wilkinson, B., and Allen, M. (1999) "Parallel Programming", Prentice-Hall.