A Formulation for True Unsteady Flows Using Iterative Convergence Techniques and the Finite Element Method

THESIS

Submitted in partial fulfillment of the requirements of

BITS F421T Thesis

By

Srihari Menon

2011A4TS209G

Under the supervision of

Dr. T. Jayachandran
Deputy Director, PRSO
VSSC, ISRO

BITS Pilani
K K Birla Goa Campus

December 2014

CERTIFICATE

This is to certify that the Thesis entitled A Formulation for True Unsteady Flows Using Iterative Convergence Techniques and the Finite Element Method is submitted by Srihari Menon ID 2011A4TS209G in partial fulfillments of the requirements of BITS F421T Thesis and embodies the work done by him under my supervision.

Signature of supervisor

Name

Date: Designation

ACKNOWLEDGEMENT

I first owe my thanks to Dr. Shibu clement, for having obliged me with this opportunity. Had it not been for his faith, kindness, guidance and support, this dissertation would’ve never seen the light of day. I am forever indebted to him for giving me the opportunity, guiding me all along, and continuously helping me understand the subject and the world of research. I call myself privileged and lucky to be his student. He will forever be one of the few individuals whom I shall always accredit for shaping my future.

I thank my supervisor, Dr. T. Jayachandran, Deputy Director, PRSO, VSSC, ISRO for being an extremely patient teacher. The fact that he holds an important office and would still be patient enough to teach me the basics of computational fluid dynamics spanning several hours speaks volumes of both his knowledge and character. I’ve not only come to learn a lot academically from him, but professionally as well. I shall forever be indebted to him and idolize him as the embodiment of knowledge, intellect and virtue.

I express my gratitude to Dr. S. Ramakrishnan, Former Director, VSSC, ISRO for giving me the opportunity of working in his prestigious and esteemed organization.

I also would like to thank Mr. M. Ajith, Engineer, FMTD for being my mentor. He helped me immensely by clearing a lot of doubts and fixing errors in my code. I thank him too, for his patience and assistance.

I would also like to mention Mr. Jaya Bharata Reddy, Mr. Praveen Kumar, Dr. Praveen Nair, Mr. CP Peeraiah, Mr. Samik Jash, Engineers, FMTD and Dr. Dileep K.N., Deputy Division Head, FMTD for having helped me one way or another and for making me feel welcomed. It was a pleasure and a learning experience working with them at the Fluid Mechanics and Thermal analysis Division (FMTD).

Thesis title: A Formulation for True Unsteady Flows Using Iterative Convergence Techniques and the Finite Element Method

Supervisor: Dr. T. Jayachandran

Semester: First 2014

Student: Srihari Menon 2011A4TS209G

THESIS ABSTRACT

This thesis work implements a Galerkin method for true unsteady problems to achieve greater precision and faster convergence while solving the continuity, Navier stokes and energy equations for a two dimensional compressible or incompressible flow. This is done by recalculating the equations iteratively for every time step in the following fashion. The flux variables and the right hand side of the equations are first evaluated at the beginning of every time step. Then the difference of the lumped mass and consistent mass matrix is multiplied to the change in the flux variables which is then added to the right hand side. This entire “corrected” RHS is then solved by simply dividing by the LHS. The process of correcting the RHS and then calculating the flux variables is carried out till the desired precision is achieved. At the end of each time step, the newly calculated flux variables are then broken down to obtain the primitive variables, which are the final result. The process is repeated for as many time steps as desired. The code works for unstructured linear triangular element grids, and is designed using the finite element method. This code is validated for flow past a cylinder, flow in a shock tube and supersonic flow past a blunt body.

NOMENCLATURE

ρ Density ,Kg/m^3

P Pressure, Pa

Cp Specific heat , J/KgK

K Conductivity, W/mK

T Temperature, K

t Time, sec

Ni, Nj Shape functions

Ω element/ element area, m^2

γ ratio of specific heats

Q Heat generated per unit volume, J/m3

Δt Time step, sec

ΔL Characteristic length of the element, m

T_0 Stagnation temperature, K

P_0 Stagnation pressure, Pa

1. INTRODUCTION

1.1 Overview

Several numerical methods and schemes exist under the domain of computational fluid dynamics that are applied in concordance with the kind of results desired so as to better understand and exploit fluid flow. The scheme being implemented here is done with the intent of gaining precision of flow variables. This is not only required for getting more believable results, but it also keeps the computations from divergence.

The code is able enough to handle both compressible and incompressible flows, which is also a desirable trait in a fluid flow solver. This is a challenge because in case the flow is compressible, the rate of change of density is not zero, thus the first term of the continuity equation is non-zero. Because of this the density is not constant, and thus doesn't get cancelled out of the continuity equation. This especially helps when the continuity is being solved along with the Navier stokes and energy equations. However, so is not the case for the incompressible case, for then the density gets cancelled out of the continuity equation ad the continuity equation gets "disconnected" from the other two sets of equations. Obviously, no fluid flow can be dubbed as being absolutely incompressible. However, Mach 0.3 is assumed to be a reasonable approximation to differentiate compressible from incompressible flow, flows with lesser speed being incompressible.

The code is also made to work on the finite element method for the discretization of the domain. The flow domain is divided, or discretized, into linear triangular elements. This method was chosen over the finite difference and finite volume methods for gaining greater spatial accuracy: the flow equations are integrated against the shape functions. This means that the value of variables doesn't just change from node to node, but can vary along the edge of an element as well. The drawback of such a scheme, however, is that the results are going to be relatively more diffused across the elements. This is crucial especially in case of shock capturing.

The code also incorporates artificial dissipation, a calculation incorporated to "smooth out" results across neighboring nodes and elements. This exercise is purely computational in nature, and has nothing to do with the physics of the problem at hand. An artificially dissipated set of calculations present a more visually comprehensible solution set. There are several methods of implementing artificial dissipation which shall be discussed in subsequent chapters.

1.2 Purpose

The primary purpose of this research effort is to develop an efficient, high-resolution, high order accurate, explicit scheme for simulating a wide range of transient, steady and viscous compressible and incompressible flows on two dimensional unstructured grids. The research uses the finite element method, and has its crux in a convergence scheme that reduces error by at least two orders every ten iterations.

The solver works as follows. The geometry is read and essential data for implementation of boundary conditions is deduced. The initial conditions are read and accordingly initialized throughout the domain. Then the advective flux vector is calculated for all nodes in order to prepare for solving the continuity, Navier-stokes and energy equations simultaneously. The equations are solved for values of the next time level using data of the preceding time level only. Thus the scheme is explicit. The calculation itself is simple: the right-hand side is divided by the left-hand side of the equations in order to find the unknown variables of the next time-step. The variables themselves are not primitive but flux variables. Once the time-step iterations begin, at every time-step, convergence iterations are run to gain precision of results. At the end of the convergence iterations, the primitive variables are then calculated from the flux variables. It is important to note that temperature itself is not explicitly solved for. It is deduced from the energy variable obtained after the aforesaid calculations. The calculations are done including artificial dissipation. The calculations continue till the specified number of time-steps are arrived upon.

The solver can be used to solve for incompressible and compressible cases. In certain supersonic cases, the velocity may grow much faster than the energy variable, thus causing fluctuations in temperature, and may even cause it to go negative. This could hinder the calculations, as a valid temperature is required for time-step calculation. It is thus advisable, if feasible, to run the program for a lower-than-desired Mach number for a certain number of iterations, then restart the calculations for a higher Mach number. This solver has been validated for flow past a cylinder to account for subsonic cases and flow past a blunt body for supersonic cases.

1.3 Governing hydrodynamic equations

The governing hydrodynamic equations under consideration for this research are the compressible Euler (inviscid) and Navier-Stokes (viscous) equations. The governing equations can be defined in two-dimensional, differential form as

$$\frac{\partial U}{\partial t}+\frac{\partial f}{\partial x}+\frac{\partial g}{\partial y}=\frac{\partial f_v}{\partial x}+\frac{\partial g_v}{\partial y}+Q \tag{1.1}$$

or in compact differential vector form,

$$\frac{\partial U}{\partial t}+\vec{\nabla}\bullet\vec{F}=\vec{\nabla}\bullet\vec{F}_v+Q\,. \tag{1.2}$$

In equation (1.2), U is the column vector of the conservative variables given by

$$U=\begin{Bmatrix}\rho\\ \rho\vec{u}\\ \rho e_t\end{Bmatrix}=\begin{Bmatrix}\rho\\ \rho u\\ \rho v\\ \rho e_t\end{Bmatrix},$$

where ρ is density, u and v are the x and y components of the velocity vector $\vec{u}$, and e_t is the specific total energy. In terms of conserved variables, ρu and ρv are the momentum components and ρe_t is the total energy. The advective flux vector, $\vec{F}$ in equation (1.2), is defined by its components f and g as

$$f=\begin{Bmatrix}\rho u\\ \rho u^2+P\\ \rho u v\\ \rho u H\end{Bmatrix}\quad g=\begin{Bmatrix}\rho v\\ \rho u v\\ \rho v^2+P\\ \rho v H\end{Bmatrix},$$

where H is the specific total enthalpy and P is the pressure. The viscous flux vector, $\vec{F}_v$ in equation (1.2), is defined by its components f_v and g_v as

$$f_v = \begin{Bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{Bmatrix} \quad g_v = \begin{Bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{Bmatrix},$$

where τ_{xx}, τ_{xy}, and τ_{yy} are the components of the viscous stress tensor and q_x and q_y are the components of the heat flux vector. For a Newtonian fluid with Stokes hypothesis, the viscous stress tensor components are defined as

$$\tau_{xx} = \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right),$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),$$

and

$$\tau_{yy} = \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right),$$

where μ is the dynamic viscosity coefficient. Following Fourier's law of heat conduction, the vector components of the heat flux vector are

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

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

where k is the coefficient of thermal conductivity and T is the absolute temperature. Q in equation (1.2) is the column vector of source terms. Many terms can be included in Q, such as a mass source, momentum drag terms, and internal energy generation.

The mathematical classification of the governing equations of compressible flow is connected to the concept of characteristics, which are defined as families of surfaces in unsteady flow, along which certain combinations of variables remain constant or certain derivatives can become discontinuous. A system of first-order partial differential equations are defined as hyperbolic if its homogeneous part allows wave-like solutions. The left-hand side of equation 1.2, commonly referred to as the Euler equations, are strictly hyperbolic in time and space. The behavior of the mathematical properties of this system of equations are dominated by wave-like solutions. This is of prime importance when addressing boundary conditions where the wave-like behavior dictates what specified information is necessary at the boundary in order to satisfy the governing equations. When the right hand side of equation 1.2 is included, the wave-like behavior may be damped and the equation system is parabolic in nature. If all waves are damped the equations are classified as elliptic. Therefore the system of time dependent Navier stokes equations is parabolic in time and space, although the conservation of mass equation is hyperbolic in nature. The time dependent Navier stokes equation are then classified as mixed parabolic-hyperbolic. Incorporating the same reasoning, the steady state form of the Navier stokes equations is then classified as mixed elliptic-hyperbolic.

Because of its wide range of applicability, the ideal gas equation of state will be use throughout this research. In most cases, a compressible gas can be considered as an ideal gas, even if viscous effects are taken into account. The ideal gas equation of state is

$$P = \rho R_c T \qquad \textbf{(1.3)}$$

where R_c is the gas constant per unit mass and is equal to the universal gas constant divided by the molecular mass of the fluid. R_c is related to the specific heat coefficient at constant pressure by

$$C_p = \frac{\gamma}{\gamma - 1} R_c$$

Where γ is the ratio of specific heats

$$\gamma = \frac{C_p}{C_v}$$

and c_v is the specific heat at constant volume. The internal energy *e* and the enthalpy *h* are only functions of temperature and have the following relations

$$e = C_v T = \frac{1}{\gamma - 1}\frac{P}{\rho}$$

and

$$h = c_p T = \frac{\gamma}{\gamma - 1}\frac{P}{\rho}. \tag{1.5}$$

The stagnation variables can be derived from the total enthalpy H, which can be defined as

$$H = \frac{\rho e_t + P}{\rho} = h + \frac{\vec{u}^2}{2} = c_p T_o, \tag{1.6}$$

where the total or stagnation temperature T_o is defined by

$$T_o = T + \frac{\vec{u}^2}{2c_p} = T\left(1 + \frac{\gamma - 1}{2}M^2\right). \tag{1.7}$$

In equations (1.7), the Mach number M is the ratio of the magnitude of velocity to the sound speed,

$$M = \frac{|\vec{u}|}{c}.$$

The sound speed of the fluid is the change in pressure with respect to density at constant entropy. For an ideal gas, this translates into

$$c^2 = \left(\frac{\partial P}{\partial \rho}\right)_s = \gamma R_c T = \frac{\gamma P}{\rho}, \tag{1.8}$$

where the subscript s indicates constant entropy. The stagnation pressure can then be represented in terms of the stagnation temperature and Mach number as

$$P_o = P\left(\frac{T_o}{T}\right)^{\frac{\gamma}{\gamma-1}} = P\left(1 + \frac{\gamma - 1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}. \tag{1.9}$$

The specific total energy can now be written in terms of the stagnation temperature or in terms of the sum of internal energy and kinetic energy,

$$e_t = c_v T_o = e + \frac{\vec{u}^2}{2}. \tag{1.10}$$

The static pressure and temperature can now be expressed in terms more useful for CFD applications. In terms of the solution variables, the pressure and temperature are

$$P = (\gamma - 1)\left(\rho e_t - \frac{\rho \vec{u} \bullet \vec{u}}{2}\right) \qquad \textbf{(1.11)}$$

and

$$T = \frac{1}{c_v}\left(e_t - \frac{\vec{u} \bullet \vec{u}}{2}\right), \qquad \textbf{(1.12)}$$

respectively.

2. FINITE ELEMENT METHOD

2.1. Overview

Every flow variable can be represented in the form of a mathematical function defined across a flow domain, in relation with other flow variables. In other words, a flow variable is a function of other flow variables and of location. An analytical solution or exact solution is a mathematical expression that gives the value of the desired unknown quantity at any location in the domain. However, analytical solutions are practically impossible to find for complex problems involving complicated geometric, material properties, boundary conditions, etc. In such cases, a viable alternative is to resort to the numerical methods, which give an approximate but acceptable solution.

Finite element, Finite volume and Finite difference are the most common among the numerical schemes used for the solution of heat transfer an fluid flow problems. Finite element technique has certain distinct advantages over other methods. Its first essential characteristic is that the continuum field, or domain, is subdivided into cells, called elements, which form a grid. The elements have a triangular or a quadrilateral form (in 2D) and can be rectilinear or curved. The grid itself need not be structured. With unstructured grids and curved cells, complex geometries can be handled with ease. This important advantage of the method is not shared by the finite difference method (FDM) which needs a structured grid, which, however, can be curved. The finite volume method (FVM), on the other hand, has the same geometric flexibility as the FEM.

The second essential characteristic of the FEM is that the solution of the discrete problem is assumed beforehand to have a prescribed form. The solution has to belong to a function space, which is built by varying function values in a given way, for instance linearly or quadratically, between values in nodal points. The nodal points, or nodes, are typical points of the elements such as vertices, mid-side points, mid-element points, etc. To put it simply, flow variables change only from node to node in FVM, but change across the nodes as well in FEM. Due to this choice, the representation of the solution is strongly linked to the geometric representation of the domain. This link is thus not as strong in the FVM.

The third essential characteristic is that a FEM does not look for the solution of the differential equations itself, but looks for a solution of an integral form of the differential equation. The most general integral form is obtained from a weighted residual formulation. By this formulation the method acquires the ability to naturally incorporate differential type boundary conditions and allows easily the construction of higher order accurate methods. The ease in obtaining higher order accuracy and the ease of implementation of boundary conditions form a second important advantage of the FEM. With respect to accuracy, the FEM is superior to the FVM, where higher order accurate formulations are quite complicated.

The method essentially consists of assuming the piecewise continuous function for the solution and obtaining the parameters of the functions in a manner that reduces the error in the solution. It works like this. Every variable that can be defined at the nodal level (such as velocity and pressure), is defined. Then, variables are calculated for each element. This is done by multiplying the nodal variables to shape functions, (dimensionless mathematical functions that represent elements with respect to the domain). This is what is meant by piecewise continuous functions: functions that are otherwise continuous throughout the domain but are discretized into pieces to fit the discretized domain. Once we have each variable for an element, they are substituted into the equations, and then the RHS an LHS are summed up for all the elements in the domain, as if it were being assembled. Now, it is here that we must recall that the FEM is just a numerical method for approximating a solution. Therefore analytically, the left hand side of the equation mustn't exactly be equal to the right hand side of the equation. There is thus an error, or what is called as a residue. All methods that exist in numerical computations are nothing but efforts to reduce this residue. The weighted residual formulation is one such attempt, where the residue is multiplied with a weighting function and is then integrated so as to bring the residue to zero. One could differentiate and equate the residue to zero so as to get a minima, but that it would only be so for a specific point in space. The residue is integrated to zero so as to bring down the error of the entire domain. When the weighting function is unity, one is solving using the FVM. When the weighting function is the shape functions themselves, one is solving using the FEM.

The finite elements procedure was first proposed by Zienkiewicz and Cheung for field problems numerous reports on the application of the Finite element method, steady, unsteady, linear and non-linear heat conduction problems involving multi-dimensional geometries and complex boundary conditions are available. Weighted residual method and Galerkin method are incorporated to solve the complex problems. Pepper and Heinrich describe in detail the application of Finite element method to one, two and three dimensional heat transfer problems.

Lary J. Segerlind makes an extensive coverage on the applications of finite element method for the multidimensional problems. The derivation of elemental matrices for a range of lower and higher order elements is also described. Huebner and Thornton gives detailed formulation and solutions for various problems of conduction, convection and radiation using finite element method. Backett and Chu discusses the application of finite element method to the solution of heat conduction problem involving non-linear radiation-boundary conditions. The application of finite element space discretization of unsteady heat conduction equations results in a system of ordinary differential equation, which are to be integrated in time to obtain the unsteady temperature response. Various implicit and explicit schemes have been suggested for this purpose. A comparison of different time marching schemes for transient heat conduction is presented by Wood and Lewis.

A good discussion on the stability and oscillation characteristics of finite element methods are given by Mayer. Purely implicit schemes are unconditionally stable and the calculated values do not oscillate about the correct values. But the disadvantage is that, a large system of simultaneous equations is to be solved and which require large computer memory. It may be also noticed that for non-linear problems, the implicit scheme may numerically oscillates for larger values of time. Explicit schemes are conditionally stable and are known to be unstable when the time step exceeds a certain value and they do not require matrix inversions.

The standard Galerkin procedures were found to be oscillating for “Pe” higher than 2 and most of the practical problems are in this unstable region. Upwind schemes have been proposed as a remedy and its extension to the multidimensional problems using finite element method are not straight forward. In 1984 Donea proposed a Taylor Galerkin algorithm for connective and transport problems. In 1992 Comini et al published the Taylor-Galerkin algorithm for convection type problems. Biji Philip Mathew solved 3-D convection-diffusion equation explicitly by lumping the capacitance matrix using Taylor-Galerkin algorithm and proved the applicability of Taylor-Galerkin algorithm for practical 2-D and 3-D problems.

The development of the finite element method in fluid dynamics is at present still far from ended. For the simplest problems such as potential flows, both compressible and incompressible, and incompressible Navier-Stokes flows at low Reynolds numbers, the finite element method is more or less full-grown, although new evolutions, certainly for Navier-Stokes problems, are still continuing. More complex problems like compressible flows governed by Euler- or Navier-Stokes equations or incompressible viscous flows at high Reynolds numbers still form an area of active research.

2.2. Application to hydrodynamic equations

The differential equations that define a flow are stated as the following:

for conservation of mass,

$$\frac{\partial \rho}{\partial t} = -\frac{\partial \rho u}{\partial x} - \frac{\partial \rho v}{\partial y}, \qquad \textbf{(2.1)}$$

for the balance of momentum in the x-direction,

$$\frac{\partial \rho u}{\partial t} = -\frac{\partial \rho u^2}{\partial x} - \frac{\partial \rho uv}{\partial y} - \frac{\partial P}{\partial x}, \qquad \textbf{(2.2a)}$$

for the balance of momentum in the y-direction,

$$\frac{\partial \rho v}{\partial t} = -\frac{\partial \rho uv}{\partial x} - \frac{\partial \rho v^2}{\partial y} - \frac{\partial P}{\partial y}, \qquad \textbf{(2.2b)}$$

and for conservation of total energy,

$$\frac{\partial \rho e_t}{\partial t} = -\frac{\partial}{\partial x}\left[\rho u\left(\frac{\rho e_t + P}{\rho}\right)\right] - \frac{\partial}{\partial y}\left[\rho v\left(\frac{\rho e_t + P}{\rho}\right)\right]. \qquad \textbf{(2.3)}$$

Where (2.1) is the continuity equation, (2.2a) and (2.2b) are the Navier Stokes equations in x and y directions and (2.3) is the energy equation. When being treated numerically, the formulae are discretized to the following form:

$$\frac{\rho^{n+1}-\rho^{n}}{\Delta t}=-\frac{\partial \rho u^{n+1}}{\partial x}-\frac{\partial \rho v^{n+1}}{\partial y}, \qquad \textbf{(2.4)}$$

$$\frac{\rho u^{n+1}-\rho u^{n}}{\Delta t}=-\frac{\partial \rho u^{2}}{\partial x}\bigg|^{n}-\frac{\partial \rho u v}{\partial y}\bigg|^{n}-\frac{\partial P^{n+1}}{\partial x}, \qquad \textbf{(2.5a)}$$

$$\frac{\rho v^{n+1}-\rho v^{n}}{\Delta t}=-\frac{\partial \rho u v}{\partial x}\bigg|^{n}-\frac{\partial \rho v^{2}}{\partial y}\bigg|^{n}-\frac{\partial P^{n+1}}{\partial y}, \qquad \textbf{(2.5b)}$$

and

$$\frac{\rho e_t^{n+1}-\rho e_t^{n}}{\Delta t}=-\frac{\partial}{\partial x}\left[\rho u^{n+1}\left(\frac{\rho e_t^{n}+P^{n+1}}{\rho^{n+1}}\right)\right]-\frac{\partial}{\partial y}\left[\rho v^{n+1}\left(\frac{\rho e_t^{n}+P^{n+1}}{\rho^{n+1}}\right)\right]. \qquad \textbf{(2.6)}$$

In equations (2.4)-(2.6), the superscripts n and $n+1$ denote values at the beginning and end of a time step, respectively, and Δt is the incremental time step size. The superscripts on the conserved variables ρu, ρv and ρe_t, denote the time level for the whole variable, not just the primitive variables u, v and e.

Let us refer to equn.(1.2). In the calculations of this thesis, there are no source terms. Therefore Q can be dropped from the same. Furthermore, $\vec{F}$ and $\overrightarrow{F_v}$ can be combined into a single $\vec{F}$, since from here onwards, the math takes over the physics and it is necessary to simplify. Then we would have,

$$U=\begin{Bmatrix}\rho\\ \rho u\\ \rho v\\ \rho e\end{Bmatrix} \qquad \textbf{(2.7)}$$

$$f=\begin{Bmatrix}\rho u\\ \rho u^{2}+P-\tau_{xx}\\ \rho u v-\tau_{xy}\\ (\rho e+P)u-k\frac{\partial T}{\partial x}-u\tau_{xx}-v\tau_{xy}\end{Bmatrix} \qquad \textbf{(2.8)}$$

$$g = \begin{Bmatrix} \rho v \\ \rho uv - \tau_{xy} \\ \rho v^2 + P - \tau_{yy} \\ (\rho e + P)v - k\frac{\partial T}{\partial y} - u\tau_{xy} - v\tau_{yy} \end{Bmatrix} \tag{2.9}$$

Where,

f and g are the components of $\vec{F}$,

$k\frac{\partial T}{\partial x}$ is the heat flux term and k is the conductivity of the fluid.

Thus on assembling these equations, we simply have

$$\frac{\partial U}{\partial t} + \nabla \cdot \vec{F} = 0 \tag{2.10}$$

Which can be rewritten as,

$$\frac{\partial U}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \tag{2.11}$$

We now introduce the idea of piecewise continuous functions. In the finite element domain, every variable can be represented as the summation of the product of that variable at discrete points of an element and the shape functions (or the derivatives of the shape functions) of that element. That is, a variable, say, P, can either be averaged over an element or can be written as,

P_e=N_1P_1+N_2P_2+N_3P_3

Where *e* stands for element, N_i represents the shape functions associated with that element and P_i are the pressures at the nodes of that element.

Similarly, certain variables can only be evaluated at the elemental level and not at the nodal level, such as,

$$\frac{\partial u}{\partial x} = \frac{\partial N_1}{\partial x}u_1 + \frac{\partial N_2}{\partial x}u_2 + \frac{\partial N_3}{\partial x}u_3 \tag{2.12}$$

$$\frac{\partial v}{\partial y} = \frac{\partial N_1}{\partial y}v_1 + \frac{\partial N_2}{\partial y}v_2 + \frac{\partial N_3}{\partial y}v_3 \tag{2.13}$$

And similarly for $\frac{\partial u}{\partial y}$ and $\frac{\partial v}{\partial x}$. These are then to be used for evaluating the shear stresses. It is important to understand that all three variables, U, f and g, must be evaluated at the elemental level.

The scheme used in this research is to solve the domain one element at a time and then to assemble the results *en masse.* This is computationally more efficient. Therefore, for every element, we can write the governing equations as,

$$\sum_{i=1}^{3} N_i \frac{\partial U_i}{\partial t} + \frac{\partial N_i}{\partial x} f_i + \frac{\partial N_i}{\partial y} g_i = 0 \qquad \textbf{(2.14)}$$

Here, an element is being assumed to be a linear triangular element, thus the summation of 3.

Furthermore, f_i and $g_i$for the nodes of an element are constants (after all, these are values at discrete points in space). Therefore, only the shape functions are to be differentiated spatially. Also, shape functions don't change with time (the domain is a fixed structure) therefore it shan't be temporally differentiated.

2.3. Galerkin method of weighted residuals

When the solution is approximated using an expression of the form (2.14), in general we cannot obtain the true solution to the differential equation. That means, when solving we shan't get an absolute zero but a non-zero residual in the RHS instead. Now, the residual could be reduced if the number of nodes and elements in the domain were increased, but it can never be brought to an absolute zero. The idea of the weighted residuals is that we can multiply the residual with a weighting function and force the integral of the product to vanish.

Therefore, the LHS of equn. (2.14) is to be integrated with the weighting function as follows

$$\sum_{e} \int W \; LHS \; d\Omega = 0$$

Where W is the weight function and the LHS referred to is that of the aforementioned equation, and all of which is integrated against the area of the element under question, which is the assembled across the domain. Choosing different weighting functions bears different results. In the Galerkin method, the shape function itself is taken to be the weighting function.

Therefore, on integrating equn.(2.14)with a shape weighting function against the element area variable, we have,

$$\sum_e \int N_i N_j \frac{\delta U}{\Delta t} + N_i \frac{\partial N_j}{\partial x} f_j + N_i \frac{\partial N_j}{\partial y} g_j \, d\Omega = 0$$

In this study, the method adopted to attain solutions is explicit in nature. Therefore, in the aforementioned equation the only unknown is δU. The values of $\vec{F}$ are calculated at *t*=*n*. Here,

$$\delta U = U^{n+1} - U^n$$

Where U^{n+1} are values at the next time step. This, specifically is the unknown.

Again, it is important to note that the flow variables themselves are constants in the equation above, therefore the only terms that are going to be integrated are the products of shape functions and that of shape functions and their derivatives.

On transferring the second and third terms to the right hand side, we have

$$\sum_e \int N_i N_j d\Omega \delta U \;= \Delta t \sum_e \int -N_i \frac{\partial N_j}{\partial x} d\Omega f_j - \Delta t \sum_e \int N_i \frac{\partial N_j}{\partial y} d\Omega g_j - \Delta t \sum_e \int N_j d\tau \, \vec{\eta} \cdot \vec{F} \qquad \textbf{(2.15)}$$

The third term is a boundary integral term. Here, τ defines the boundary of the domain and $\vec{\eta}$ is the unit vector normal to the boundary. The flux vector for boundaries are evaluated in the same way as is in equn.(2.8) and (2.9), except that it is evaluated for two nodes (the ones forming the boundary), and not three nodes.

The integral of the two shape functions on the right hand side of (2.15) is called the consistent mass matrix, and is denoted as $M_{c.}$

$$M_c = \sum_e \int N_i N_j d\Omega$$

The integrated matrices of all the aforementioned integrations can be found in appendix A2.

The equation (2.15) can be written in the form,

$$M_c \delta U = R \qquad \textbf{(2.16)}$$

Where R stands for the right hand side of equn. (2.15).

2.4. Iterative procedure for precision

Solving equn.(2.16) iteratively gives room for a lot of oscillations in the variable values. These oscillations can, and often do, lead to interruptions in calculation (such as yielding a negative temperature which would make calculating time steps impossible), accumulation of error and consequent divergence of solution. In order to accelerate the convergence of results, this study employs a certain iterative procedure developed by Donea that reduces the order of change of variables by at least two orders every ten iterations per time-step.

This is done by first finding out the lumped mass matrix of the consistent mass matrix.

$$Diag\{M_l\}_j = \sum_{i=1}^{n} \{M_c\}_{i,j}$$

Where *i* is the row and j is the column number respectively. Every other entry in the diagonal lumped matrix is zero. The consistent mass matrix is a symmetric matrix, and thus the diagonal entries in its diagonal matrix are all going to be the same value. In other words, the lumped mass matrix is going to be nothing more than a scalar multiple of the identity matrix.

Next, the following changes are brought to equn. (2.16):

$$M_l \delta U + M_c \delta U = R + M_l \delta U$$

This can be rearranged as,

$$M_l \delta U_{i+1} = R + (M_l - M_c)\delta U_i \qquad \textbf{(2.17)}$$

Where the subscripts *i* and *i+1* denote convergence sub-steps in a time-step.

The fact that the lumped mass matrix is a scalar multiple of the identity matrix simplifies the equation above into an equation where the unknown can be found by simply dividing the equation by a scalar quantity.

In the case of this study, since the equations and thus the variables are to be calculated element-wise, the lumped mass matrix is a three-by-three matrix, the scalar multiple being one-third the element area. Details of calculation are given in the flow solver chapter.

3. ARTIFICIAL DISSIPATION

3.1. Overview

Requirements of positivity and accuracy of compressible flow quantities cannot be mutually achieved with the use of traditional second-order spatially accurate discretized numerical schemes. Godunov (1959) showed that linear second-order or higher-order spatially accurate methods cannot guarantee monotonicity. First-order methods, such as the Lax-Friedrichs method are monotonic and guarantee positivity but are extremely diffusive, which tends to smear high gradients and discontinuities over many computational cells, thus rendering the solution inaccurate. Second-order methods, such as the Lax-Wendroff or MacCormack methods, are less diffusive but are susceptible to nonlinear instabilities and nonphysical oscillations occurring in regions where the solution contains high changes in gradient (curvature) and discontinuities. These oscillations can cause conserved variables, which physically (and computationally) require positivity, to become negative and possibly render the simulation unstable. Oran and Boris (1987) give an excellent description of how these problems occur with low and high-order methods. Some form of artificial dissipation is required in the numerical simulation of compressible flows in order to maintain a stable and physically relevant representation of the flow.

Introducing artificial diffusion into the algorithm is one traditional approach to dampen the oscillations created by high-order methods. However, this diffusion also spreads high gradients over more computational cells as the overshoots and undershoots are suppressed. Total damping of the oscillations requires as much numerical diffusion as first-order methods.

Because the addition of artificial dissipation is essentially introducing nonphysical behavior into the flow simulation, it must be carefully applied in such a manner that excessive error is not introduced into the solution and that conservation is strictly maintained. A variable dissipation method is needed that automatically senses where the artificial dissipation should be applied. The method should only apply dissipation in regions where the solution contains high changes in solution gradient, not in regions where the solution is already smooth. Two such artificial dissipation methods are presented here. The first was developed by Lapidus (1967) and later applied to the finite element method by Löhner et al., (1985b). It is a third-order diffusion term that is locally one-dimensional in the direction of the velocity vector. It uses the magnitude of the local velocity gradient as a sensor to locate a discontinuity. The second artificial dissipation method was developed by Peraire (Morgan and Peraire,1987, and Peraire et al., 1988) for finite element computations. This method incorporates a sensor that is based upon the change in the local pressure gradient. Both artificial dissipation methods are used as filters. They are applied after a high-order solution has been obtained to control the non-physical oscillations. Both methods have shown good results for flow simulations that do not contain strong discontinuities, such as normal shocks. In regions of strong discontinuities, monotonic behavior in the solution can generally not be maintained unless the diffusion coefficient is excessively large. This type of artificial dissipation may be preferred over high resolution methods for simulation of flows that do not contain strong discontinuities because they are far less costly from a computational point of view.

Nonlinear high-resolution monotone methods were invented to solve the problem of maintaining positivity and monotonicity with accuracy. The first nonlinear, monotone, positivity-preserving technique was the FCT algorithm developed by Boris and Book (1973- 1976). The FCT technique constructs the net transportive flux point by point, nonlinearly, as a weighted average of a flux computed by a low-order scheme and a flux computed by a high-order scheme. The weighting is done in a manner which insures that the high-order flux is used to the greatest extent possible without introducing overshoots and undershoots. This weighting procedure is referred to as "flux-correction" or "flux limiting" because, in effect, the weighting is such that the fluxes are limited so that no new extrema in the solution are introduced. While Boris and Book's FCT solver is one-dimensional in nature, Zalesak (1979) developed a generalized, alternative approach to FCT that is truly multidimensional without introducing the flow direction bias of operator splitting. Zalesak's FCT algorithm allows the use of unstructured grids which makes the algorithm attractive for implementation into the finite element method. Löhner et al., (1987 and 1988) adapted Zalesak's FCT algorithm for controlling the oscillations generated by the Two-Step Taylor-Galerkin FEM on unstructured meshes.

3.2. Jameson Schmidt Turkel artificial dissipation

This solver uses JST artificial dissipation technique. Higher order non-oscillatory schemes can be derived by introducing anti-diffusive terms in a controlled manner. An early attempt to produce a high resolution scheme by this approach is the Jameson-Schmidt-Turkel (JST) scheme.

This dissipation scheme provides an artificial viscosity in order to both prevent oscillations near shocks and damp high frequencies, enabling the iteration procedure to reach a steady state. In the Jameson, Schmidt, Turkel (JST) formulation these artificial viscosities are provided by second and fourth differences of the variables with a scalar coefficient included.

This scalar coefficient depends on the largest eigenvalue (in each direction) to scale the size of the viscosity. In addition, the coefficient depends on the second difference of the pressure to sense shocks. In the neighborhood of shocks the fourth difference is turned on while the second difference prevents overshoots. In smooth regions of the flow the second difference (which leads to first-order accuracy) is minimal while the fourth difference damps the high-frequency errors.

The artificial viscosity is calculated as is shown for every node

$$\vartheta = \frac{\sum_{i=1}^{n} |P_i - P|}{\sum_{i=1}^{n} |P_i + P|}$$

Where P is the pressure at that node, i is the index for a neighboring node and n are the total neighbor nodes of the node under question. The second and fourth differences are calculated as

$$\nabla^2\varphi = \sum_{i=1}^{n} C_2 * Max(\vartheta_i, \vartheta)(\vec{F_i} - \vec{F})/\Delta t$$

and, $$\nabla^4\varphi = \sum_{i=1}^{n}(C_4 - C_2 * Max(\vartheta_i, \vartheta))(\nabla^2\varphi_i - \nabla^2\varphi)$$

Where $\vec{F}$ is a flux variable, C_2 and C_4 are the scalar coefficients. Usually, they are assigned the values 0.25 and 0.00390625, respectively. The same are used in the solver here.

Finally, the dissipation term is evaluated as

$$D=\nabla^2\varphi - \nabla^4\varphi$$

It is important to note that the dissipation is calculated for every individual node and flux variable.

This dissipation term is then added to the right hand side of the equations when being solved for.

4. THE FLOW SOLVER

4.1. Overview

In this section, the process in which the solver works, and the method it employs to gain precision of results, is going to be discussed. First the geometrical properties of the given domain are calculated, such as deducing the neighbor nodes to every node, identifying the boundary nodes an elements, and identifying the closest non-boundary node to each boundary node. The methods of the same are mentioned in the appendices. Apart from these, the derivatives of the shape functions are calculated element-wise, and the left hand side of the principal equation to be solved, which consists of the "average element area per node" is calculated node-wise. The shortest edge of each element is also calculated in order to determine the minimum time-step for every iteration.

Next the domain is initialized. The initial conditions are read from an input file provided by the user, which is then used to initialize all variables in the domain. The content of initialization varies from problem to problem, but broadly the input given is that of P_{∞}, T_{∞}, inlet Mach number, molecular weight, Prandtl number, CFL number and number of time-steps, amongst other things. A separate feature of restarting calculations is also incorporated in the initialization subroutine.

4.2. Boundary conditions

After the aforementioned, the boundary conditions are to be applied. The boundary conditions are implemented based on the boundary type. Furthermore, it is here that the closest-to-boundary nodes data would be used. The following are the conditions to be applied:

4.2.1. Subsonic inflow

In case of a subsonic inflow, any disturbance, perturbation or condition applied to the flow downstream can travel back upstream to the inlet. Therefore, the condition for subsonic inlet boundary should be to "absorb" the disturbance, or take information from within the domain. Therefore, velocity at the inlet is updated to that of the closest non-boundary neighbor node, and temperature is calculated by subtracting the kinetic energy from the stagnation temperature:

$$T_i = T_0 - \frac{u^2+v^2}{2C_p} \qquad \textbf{(5.1)}$$

4.2.2. Supersonic inflow

Information or a perturbation can naturally travel fastest only at the speed of sound. Thus, in case of a supersonic inflow, even if an information travels upstream it would not be able to make it to the inlet, since the fluid, and thus information, is travelling faster downstream. Therefore, all variables at the inlet are simply updated to the initial flow conditions.

4.2.3. Outflow

Outlets inherently derive information from the flow upstream. Therefore, all variables at the outlet must be updated to that of the nearest non-boundary node's. It is necessary to note that the velocity component normal to the outflow should be updated with the modulus of the nearest node's velocity, so as to keep from backflow.

4.2.4. Symmetry

The idea of keeping a symmetry boundary condition is to work on only half of the actual domain and assume that the results would be symmetrical on the other side. In keeping with the same, it must be ensured that no information is transferred across the symmetry line, and thus all velocities and gradients of variables must be brought to zero. If the symmetry boundary is vertical then u must be brought to zero, and if horizontal, v must be brought to zero. In all other conditions the dot product of the normal vector of that boundary and the velocity vector should equal zero. To make all gradients zero, as was done in case of subsonic inflow, all other variables should be updated to that of the closest non-boundary node's.

4.2.5. Wall

The wall condition is one of the most used and ironically one of the most general boundary conditions. The only universal condition applicable to a wall is that all velocity normal to the wall is to be brought to zero. Beyond this, one may implement the condition with any other requisite. For example, the wall may be isothermal, and thus would require the temperature to be updated to a constant. The wall may be adiabatic, and thus would require that the normal temperature gradient be brought to zero. For that, the temperature must be updated to that of the closest non-boundary node's temperature value. The wall may be a no-slip wall, which would require all components of velocity at the wall to be brought to zero, or it may be a free slip wall, in which case the component parallel to the wall should be updated with the velocity of the nearest non-boundary node's.

4.3. Artificial dissipation

After the application of boundary conditions, the artificial dissipation is calculated. This scheme uses the Jameson Schmidt Turkel (JST) method of artificial dissipation. Details of the calculations are given in chapter 3. The method will be described briefly here. A loop is run for all nodes to calculate the numerator and the denominator that'd be used to calculate the artificial viscosity. Here again, the concept of neighbor nodes is employed to find out the pressure of every neighboring node to a given node. After this the values of $\nabla^2\varphi$ and $\nabla^4\varphi$ are calculated, and finally the dissipation term is arrived at. This dissipation term is then to be used in the main calculation that is to be done in the time-step iterations.

4.4. Time stepping

Time steps are the periods of time in which the solution is advanced through. In this research, a time step is calculated as

$$\Delta t \leq \frac{\Delta x}{|u|+c} \qquad \textbf{(5.2)}$$

Where,

Δt stands for the time step, Δx stands for minimum edge length of the elements, |u| stands for modulus of velocity in the element and c stands for speed of sound.

Furthermore, c can be written as,

$$c = \sqrt{\gamma RT}$$

Where γ is the specific heat ratio of the fluid, and R is is the gas constant for the fluid.

Therefore, this can be refined to

$$\Delta t \leq \frac{\Delta x}{|u|+\sqrt{\gamma RT}} \qquad \textbf{(5.3)}$$

For explicit solution, the value of Δt cannot be any arbitrary value, indeed, it must be less than or equal to some maximum value. This max value is usually estimated by

$$\Delta t < min\left[\frac{\Delta x}{|u|}, \frac{\Delta y}{|v|}\right]$$

Or

$$N_{cfl} = \Delta t * min\left[\frac{\Delta x}{|u|}, \frac{\Delta y}{|v|}\right]^{-1}$$

Where N_{cfl} is the Courant Friedrich Lewy or CFL number, which means that, Δt must be less than the time required for a wave or information to propagate between two adjacent nodes.

4.5. Flux vector calculation

After the time-step is calculated, the flux vectors *f* and g are to be calculated. $\vec{F}$ components are to be calculated as in equns. (2.8) and (2.9). The exact equations that are employed in this scheme is what'd be illustrated in this section. Again, recounting that equn.(2.10) is to be solved at an elemental level, the *f* and g are to be evaluated element-wise. Thus an element-wise loop is run to evaluate each of the four parts to both *f* and g. The following are the elemental variables evaluated:

$$\frac{\partial u}{\partial x}_e = \frac{\partial N_1}{\partial x}u_1 + \frac{\partial N_2}{\partial x}u_2 + \frac{\partial N_3}{\partial x}u_3$$

$$\frac{\partial v}{\partial x}_e = \frac{\partial N_1}{\partial x}v_1 + \frac{\partial N_2}{\partial x}v_2 + \frac{\partial N_3}{\partial x}v_3$$

$$\frac{\partial u}{\partial y}_e = \frac{\partial N_1}{\partial y}u_1 + \frac{\partial N_2}{\partial y}u_2 + \frac{\partial N_3}{\partial y}u_3$$

$$\frac{\partial v}{\partial y}_e = \frac{\partial N_1}{\partial y}v_1 + \frac{\partial N_2}{\partial y}v_2 + \frac{\partial N_3}{\partial y}v_3$$

$$\frac{\partial T}{\partial x}_e = \frac{\partial N_1}{\partial x}T_1 + \frac{\partial N_2}{\partial x}T_2 + \frac{\partial N_3}{\partial x}T_3$$

$$\frac{\partial T}{\partial y}_e = \frac{\partial N_1}{\partial y}T_1 + \frac{\partial N_2}{\partial y}T_2 + \frac{\partial N_3}{\partial y}T_3$$

$$\rho u H_e = \frac{\sum_{i=1}^{3}(\rho_i e_i + P_i)u_i}{3};\ \rho v H_e = \frac{\sum_{i=1}^{3}(\rho_i e_i + P_i)v_i}{3}$$

$$\rho u_e = \frac{\sum_{i=1}^{3}\rho_i u_i}{3};\ \rho v_e = \frac{\sum_{i=1}^{3}\rho_i v_i}{3}$$

$$\rho u_e^2 = \frac{\sum_{i=1}^{3}\rho_i u_i^2}{3};\ \rho v_e^2 = \frac{\sum_{i=1}^{3}\rho_i v_i^2}{3};\ \rho u v_e = \frac{\sum_{i=1}^{3}\rho_i u_i v_i}{3}$$

$$P_e = \frac{\sum_{i=1}^{3}P_i}{3};\ T_e = \frac{\sum_{i=1}^{3}T_i}{3}$$

$$u_e = \frac{\sum_{i=1}^{3}u_i}{3};\ v_e = \frac{\sum_{i=1}^{3}v_i}{3}$$

This code implements the Sutherland scheme for calculating the viscosity:

$$\mu = 11.848 \times 10^{-8}\sqrt{m} \times {T_e}^{0.6}$$

Where *m* stands for molar weight.

Shear stress is calculated as

$$\tau_{xx_e} = \frac{2\mu}{3}\left(2\frac{\partial u}{\partial x_e} - \frac{\partial v}{\partial y_e}\right)$$

$$\tau_{yy_e} = \frac{2\mu}{3}\left(2\frac{\partial v}{\partial y_e} - \frac{\partial u}{\partial x_e}\right)$$

$$\tau_{xy_e} = \mu\left(\frac{\partial v}{\partial x_e} + \frac{\partial u}{\partial y_e}\right)$$

Finally, *f* and *g* are calculated as

$$f_e = \begin{Bmatrix} \rho u_e \\ \rho u_e^2 + P_e - \tau_{xx_e} \\ \rho u v_e - \tau_{xy_e} \\ \rho u H_e - k\frac{\partial T}{\partial x_e} - u_e\tau_{xx_e} - v_e\tau_{xy_e} \end{Bmatrix}$$

$$g_e = \begin{Bmatrix} \rho v_e \\ \rho u v_e - \tau_{xy_e} \\ \rho v_e^2 + P_e - \tau_{yy_e} \\ \rho v H_e - k\dfrac{\partial T}{\partial y}_e - v_e \tau_{xy_e} - u_e \tau_{yy_e} \end{Bmatrix}$$

In the above equations, the subscript *e* stands for elemental level variable, and *i* stands for nodal level variable.

4.6. Time step iterations

After the first set of flux variables are calculated, we proceed to commence the time-step iterations of this solver. First, the value R from equn.(2.17) is evaluated, by running an element-wise loop. . R itself will contain nodal data, even though the loop is run element-wise, since the flux variables are at an elemental level. It is important to note here that though the values of flux variables are being calculated element wise, the equations will be solved for at the nodal level. This is so because all the unknown variables in *U* are defined at a nodal level.

$$R_i = \sum\nolimits_{e_i} \Omega\left(\frac{\partial N_j}{\partial x} f_j + \frac{\partial N_j}{\partial y} g_j\right)$$

Where *i* stands for node number, e_i stands for all elements that share that node, *Ω* stands for the area of those elements and *j* stands for the index that node has with respect to the element. For example, if nodes 5, 110 and 32 make up the nodes of element 1, and in that order, then R would contain at index 5, 110 and 32, the values of the above equations evaluated at j=1, 2 and 3 respectively, with respect to element 1. Here is an actual part of the code to further elucidate:

```
for (int e=1; e<=elements; e++){
        for (int m=1; m<=4; m++){
                                R[n1[e]][m]+=elarea[e]*(f[e][m]*b1[e]+g[e][m]*c1[e]);
                                R[n2[e]][m]+=elarea[e]*(f[e][m]*b2[e]+g[e][m]*c2[e]);
                                R[n3[e]][m]+=elarea[e]*(f[e][m]*b3[e]+g[e][m]*c3[e]);
                        }
                        }
```

Where elarea stands for element area, and b_i and c_i are the shape function derivatives, and n_1, n_2 and n_3 are the nodes of that element and in that order. One would notice a second loop running four times. That is to evaluate every one of the four R's for the four variables encapsulated in U.

4.7. Convergence iterations

After having evaluated R from equn.(2.17), we start with the convergence iterations to gain precision of results. First, we calculate the second term on the right hand side of the equation. As mentioned earlier, the lumped mass matrix itself is a scalar multiple of the identity matrix. In our case, the values for M_l and M_c are calculated to be the following:

$$M_c = \frac{\Omega}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$M_l = \frac{\Omega}{12}\begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

$$\therefore\ M_c - M_l = \frac{\Omega}{12}\begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}$$

Thus, when multiplied with change in flux variables, we have

$$(M_c - M_l)\delta U = \frac{\Omega}{12}\begin{bmatrix} -2\delta U_1 + \delta U_2 + \delta U_3 \\ \delta U_1 - 2\delta U_2 + \delta U_3 \\ \delta U_1 + \delta U_2 - 2\delta U_3 \end{bmatrix}$$

Where the subscript in δU simply stands for the node number or node index with respect to the element. It is reminded again that this matrix though being calculated at an elemental level, is stored at the nodal level. The procedure is similar to what was done for R.

After this term is calculated as well, we proceed to actually calculating the change in the variables. For this, a node-wise loop is run. The right hand side is created by simply adding the two terms R and $(M_c - M_l)\delta U$. As for the left hand side, the unknown variables are multiplied to the lumped mass matrix, which is a scalar multiple of the identity matrix. Therefore in essence, they are multiplied to a scalar multiple only. In our case, this multiple is $\frac{\Omega}{3}$. This is fairly obvious from the lumped mass matrix formula above. All that is left to be done is to add the artificial dissipation term to R, multiply the dissipated R with the time-step, then add this to the second term. Then we assemble all values of LHS and RHS of the domain, before dividing the net RHS with $\frac{\Omega total}{3}$. The final equation would look like

$$\delta U_{i+1} = \frac{\sum_{i=1}^{n} \Delta t(R_i + D_i) + (M_c - M_l)\delta U_i}{\sum_{j=1}^{e} \frac{\Omega_j}{3}}$$

This solution is run for as many times as is one's wish by which the order of change must reduce. This scheme runs convergence iterations ten times every time-step, so as to gain reduction in order by two to three orders.

Once the convergence iterations are done with, all the primitive variables are calculated out of the flux variables, and the process of applying boundary conditions, figuring out a dissipation term, calculating new flux variables and calculating a new time-step is carried out. This entire process is carried out by as many number of times the user wishes to move forward in time.

5. RESULTS AND VALIDATION

5.1. Overview

The solver once coded into a computing language, is to be validated against standard results to test the legitimacy of the solver. In this study, the code is validated for three problems: flow past a cylinder for subsonic conditions, shock tube problem for transonic conditions and flow past a blunt body at Mach 6.57 for supersonic conditions. In certain cases, the validation is quantitative and others it is qualitative.

5.2. The shock tube problem

The shock tube problem is a classic test case for the validation of compressible flow schemes and codes. It is one of the few compressible flow problems that has an analytical solution available (e.g. Hirsch, 1990) that can be compared to numerical and experimental results. The problem can be characterized by the sudden rupture of a diaphragm in a long one-dimensional tube separating two initial gas states at different pressures and densities. The state after the rupture of the diaphragm (t > 0) consists of three major compressible flow phenomena; a shock wave propagating downstream at a speed equal to $|\vec{u}|+c$ followed by a contact discontinuity traveling with the fluid at a velocity of $|\vec{u}|$ and an expansion fan (rarefaction wave) traveling upstream at a rate of $|\vec{u}|-c$. The interesting aspect of the shock tube problem is that all of the above major flow phenomena associated with compressible flow are concurrently present.

The standard initial conditions for running this test are:

$$\begin{Bmatrix}\rho_l\\ P_l\\ \vec{u}_l\end{Bmatrix}=\begin{Bmatrix}8.0\\ 10.0\\ 0.0\end{Bmatrix};\ \begin{Bmatrix}\rho_R\\ P_R\\ \vec{u}_R\end{Bmatrix}=\begin{Bmatrix}1.0\\ 1.0\\ 0.0\end{Bmatrix}$$

Where ρ, P and $\vec{u}$ have their usual meanings and L and R subscripts stand for left and right of the diaphragm. In our case, the length of the setup is kept at 100 m and height of 2 m, and the diaphragm is placed in the middle at 50 m. The entire domain is discretized into several elements and data is collected for the mid-mid line, i.e. y=1 m. The only boundary condition applied in this problem is a no-slip boundary wall on all four boundaries of the domain. Results are obtained for t=0.2 s after diaphragm removal.

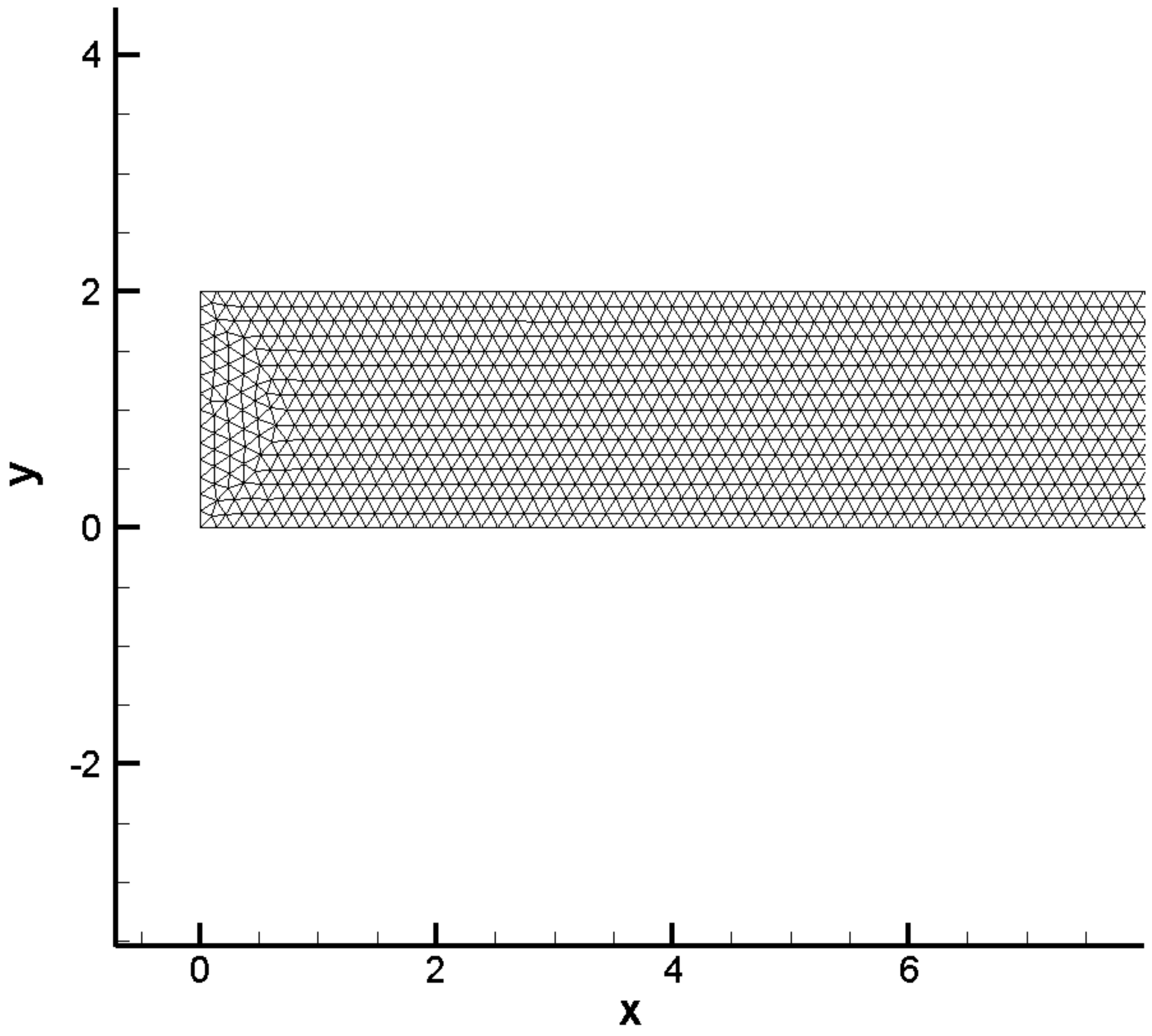

Fig.1. Mesh close-up for shock tube

5.2.1. Grid independence

First we establish that the results are going to be independent of the number elements in the grid. For this the entire domain is discretized into nearly 22000 elements (coarse), then conditions are applied and is solved for after t=0.2s. Then the results are collected along y=1. Next, the same domain is discretized into early 44000 elements (fine), and results are generated. We then superimpose the results and expect them to be same. This is illustrated in fig. 2 and fig.3. The mesh with 44000 elements is illustrated in fig.1 so as to appreciate the scale of discretization. As can be seen the density and pressure plots for both the meshes yield almost concurrent results. This thus establishes the grid independence of the computer code. It is important to note that once demonstrated, it is redundant to establish grid independence for the other two problems. Therefore, this demonstration is deemed sufficient.

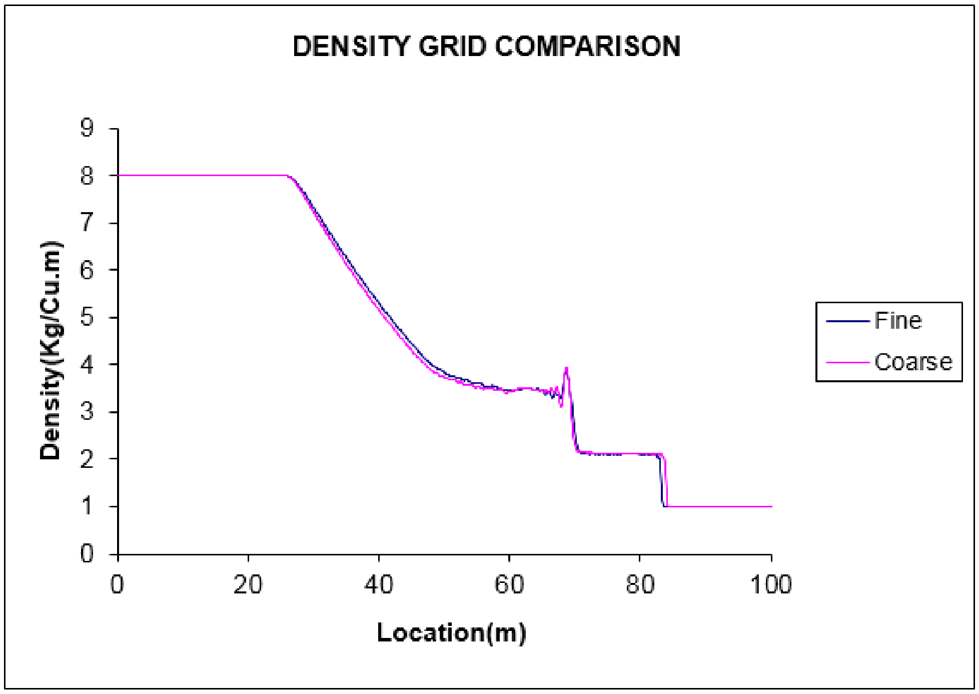

Fig. 2. Density grid comparison

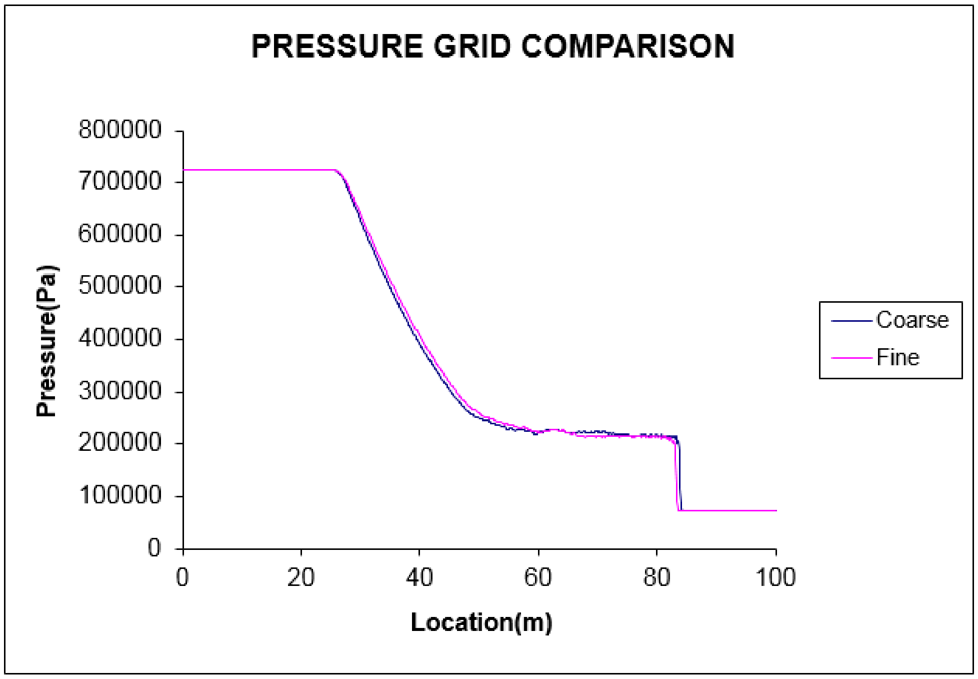

Fig. 3.Pressure grid comparison

5.2.2. Results

Next we compare the results obtained from the fine mesh with analytical results available.

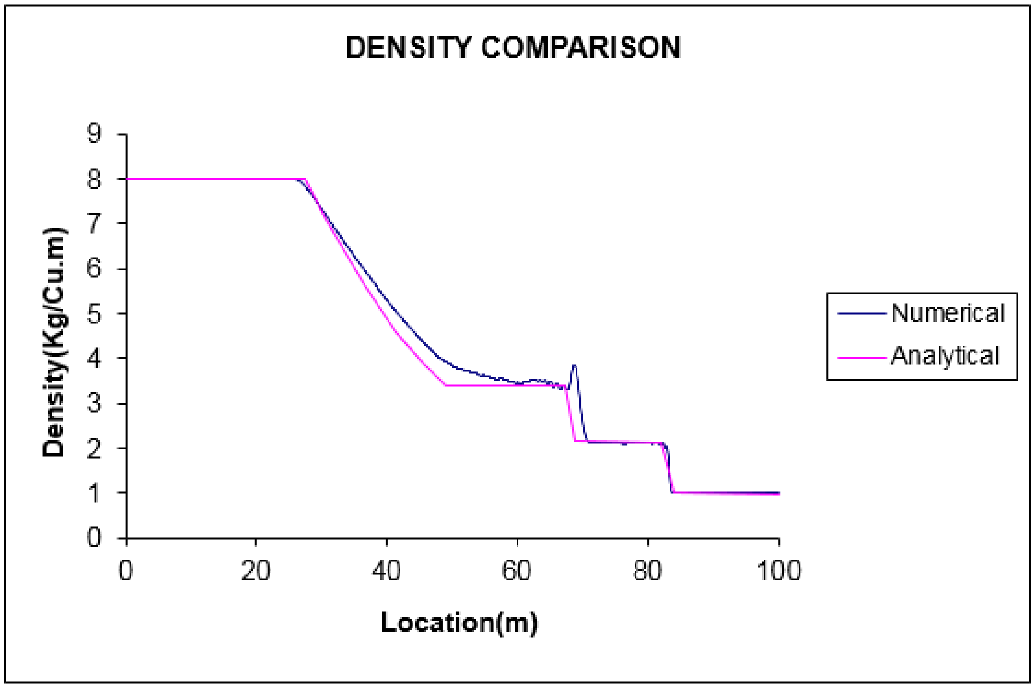

Fig. 4. Density comparison

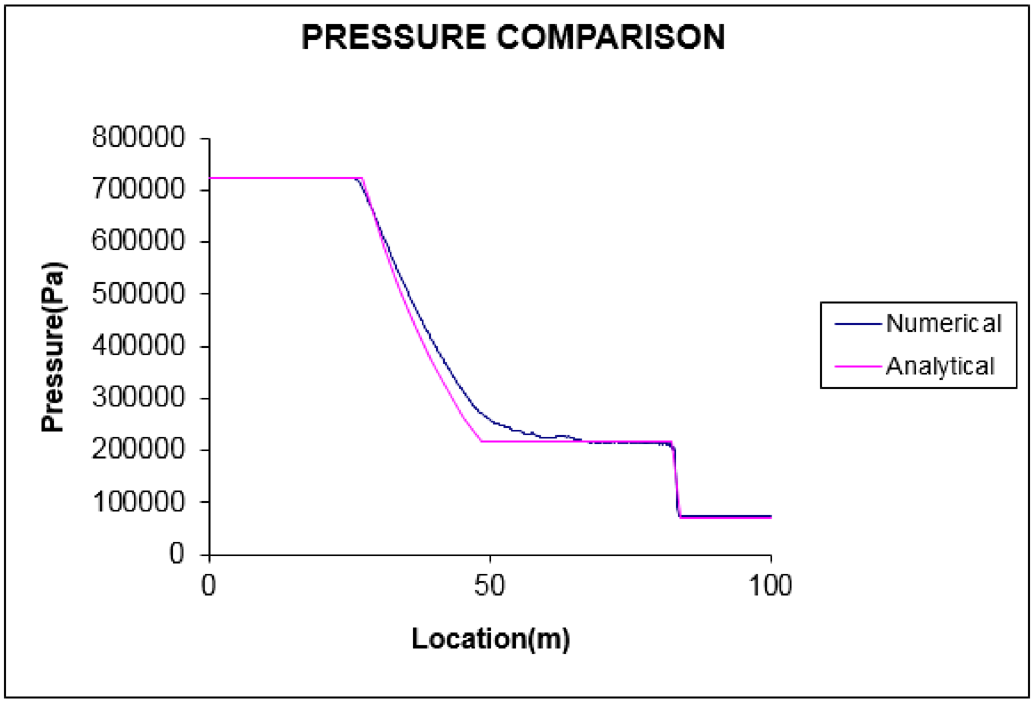

Fig. 5. Pressure comparison

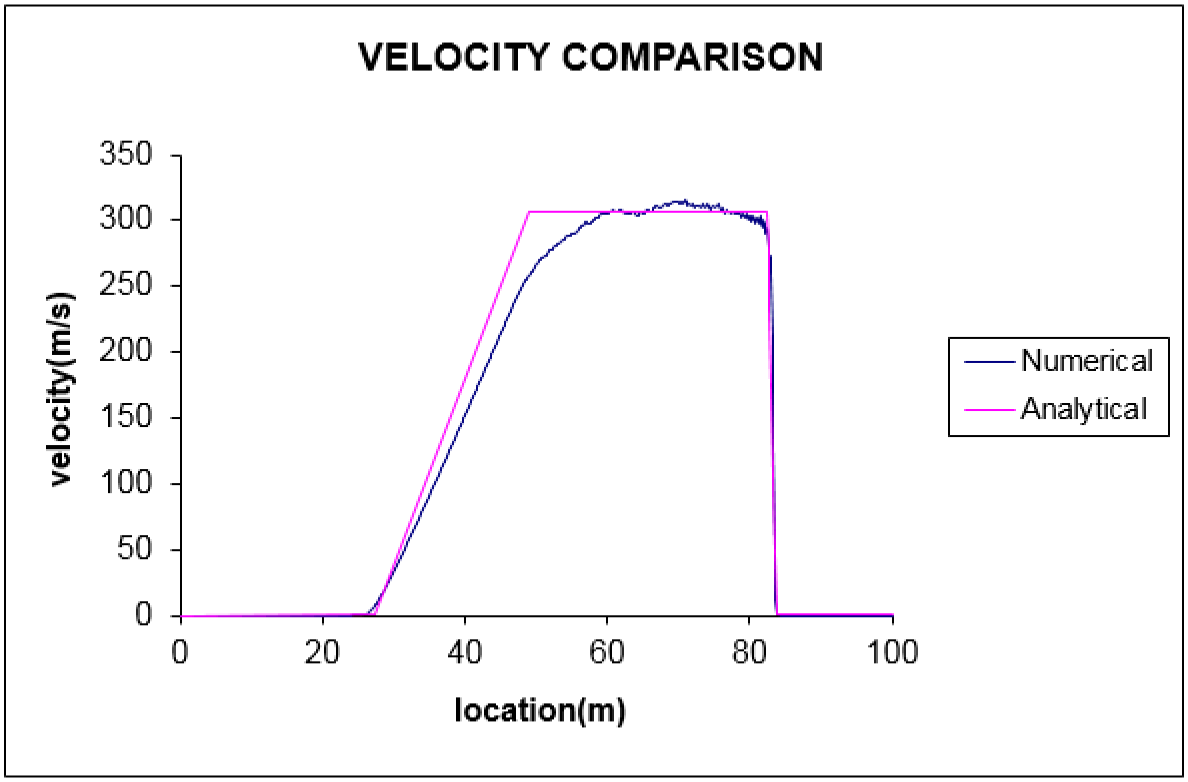

Fig. 6. Velocity comparison

Fig. 4, 5 and 6 see an amiable agreement between the numerically calculated results and the analytical results already available. Undeniably, one may observe a few departures from ideality, but there is always room for improving the technique and finding even better results. The aim was to illustrate that the scheme works under reasonably accepted margins of error.

5.3. Flow past a cylinder

This is a straightforward qualitative test for the code. Here an elongated domain is drawn around a cylinder and is subjected to subsonic flows at different speeds. Ultimately the attempt here is to capture the formation of the Von Karman vortex street. The domain is discretized into 22600 elements, with a greater density of elements around the cylinder. Fig. 7 illustrates the same. The initial conditions are standard conditions with 1atm pressure and 300K temperature. The boundary conditions here are subsonic inlet, outlet, and no-slip wall.

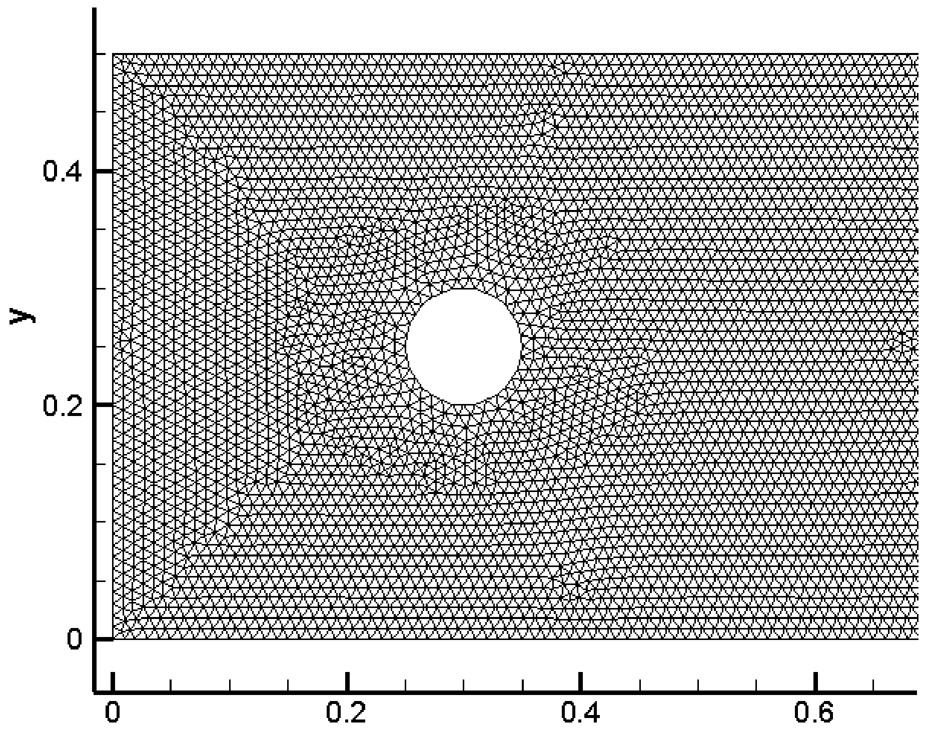

Fig. 7. Mesh close-up of cylinder

5.3.1. Results

The flow is simulated for three Reynolds numbers: 200 2000 and 20000. Shown here are the vorticity contours of that flow.

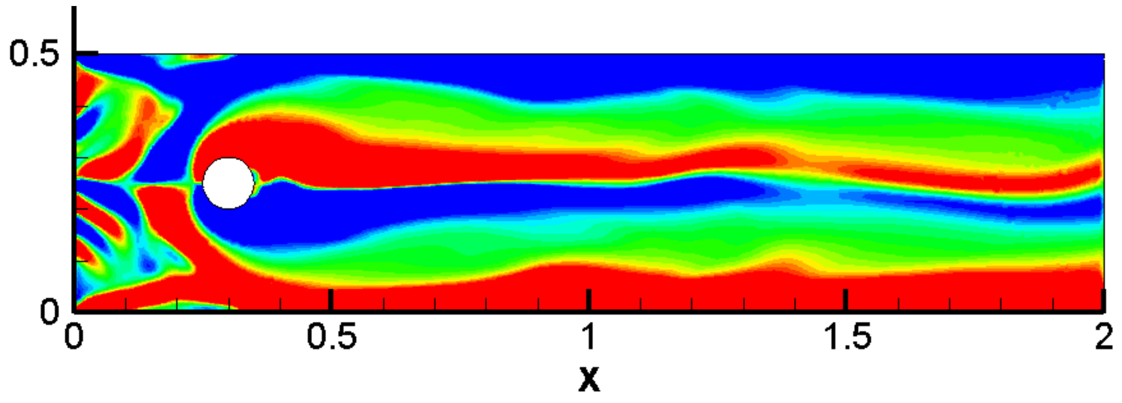

Fig. 8. Re=200

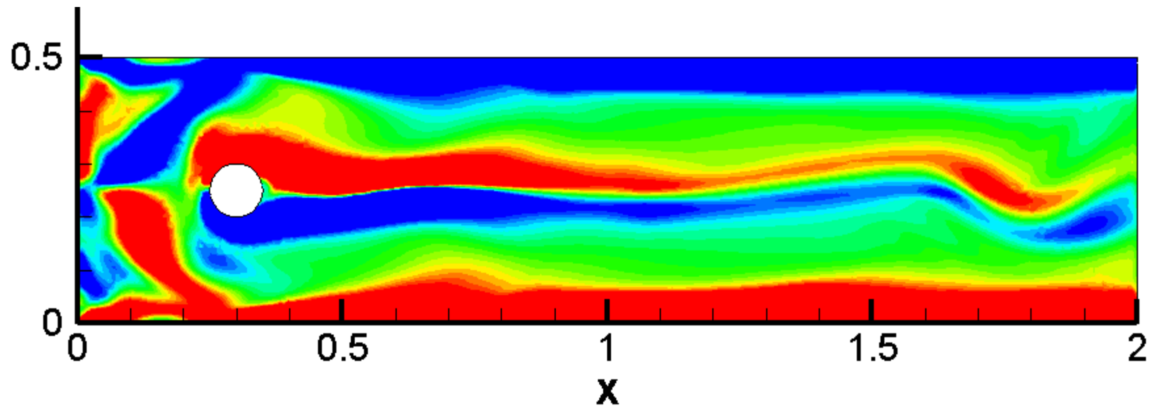

Fig. 9. Re=2000

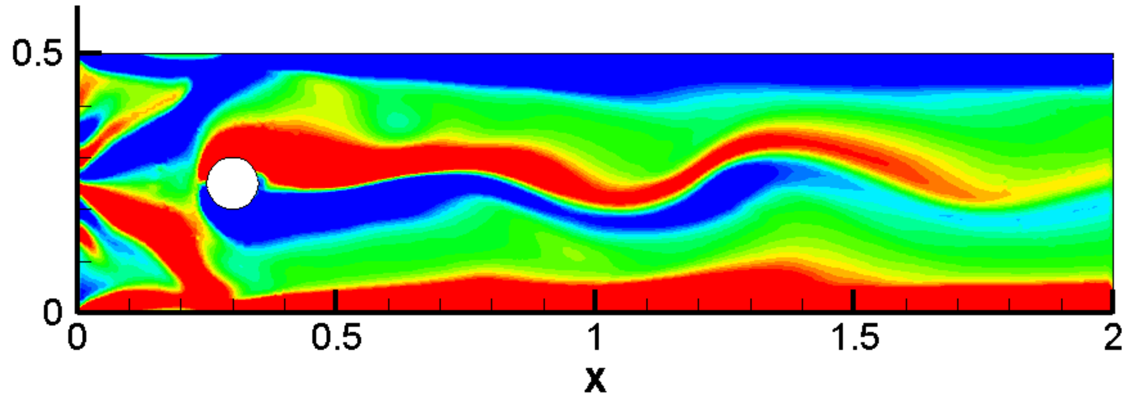

Fig. 10. Re=20000

We notice a gradual increment in the intensity of the vortices as we increase the Reynolds number. This demonstration is also important because this proves that not only does the code work for steady flow, but is able to show results for transient flow as well.

5.4. Supersonic flow past blunt body

This is an important case as it establishes the legitimacy of the code for supersonic and hypersonic study. In this problem, a blunt boy of certain dimensions is subjected to an incoming flow of Mach 6.57, amongst other conditions. Then the solver is employed to solve for the same and the contours of various variables are presented here as results. This is a steady state solution. Therefore, calculations are stopped when the order of change in the flux variables is sufficiently low ($\sim10^{-21}$).

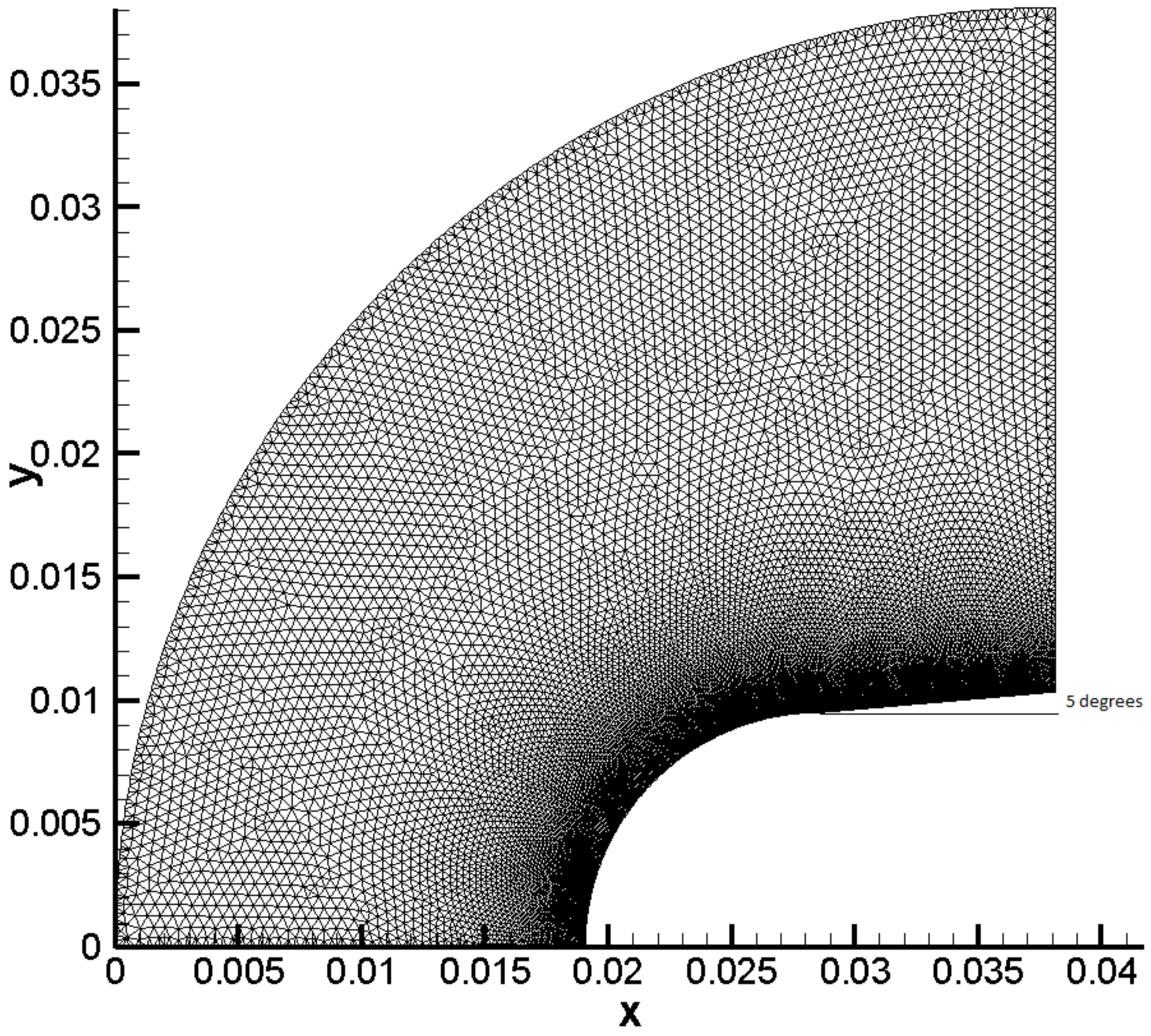

Fig. 11. Blunt body mesh

A symmetry line is assumed along y=0. There are 30500 elements in this mesh. The peculiar shape of the inlet is to aid in getting better Mach contours. The initial conditions are :

P_0 =101325 Pa T_0 =300 K   M=6.57 γ=1.38   Prandtl=0.7

After these conditions are imposed, the simulations are run for a sufficient number of iterations before yielding results.

5.4.1. Results

The results in the form of contours are as follows.

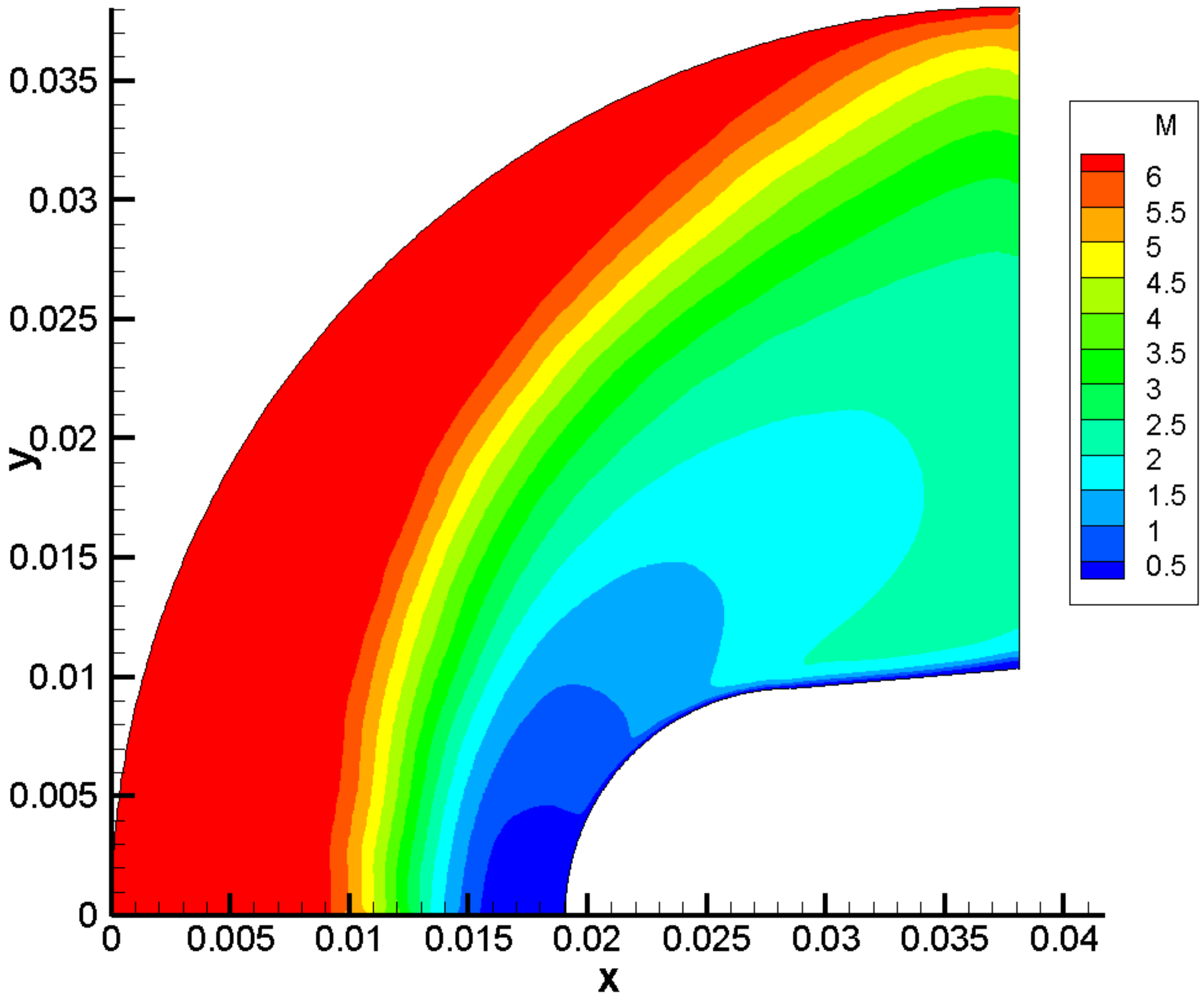

Fig. 12. Mach contours

Fig. 12. Shows the successful creation of a "bow shock" structure which is characteristic for such flows and geometries. The other contours confirm the same.

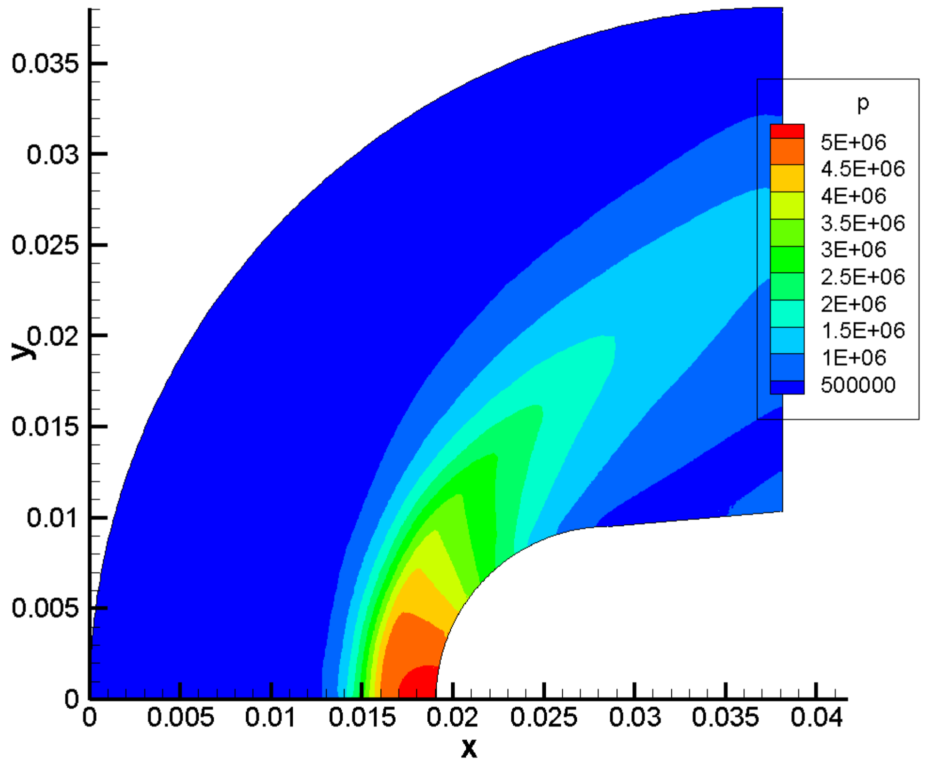

Fig. 13. Pressure contours

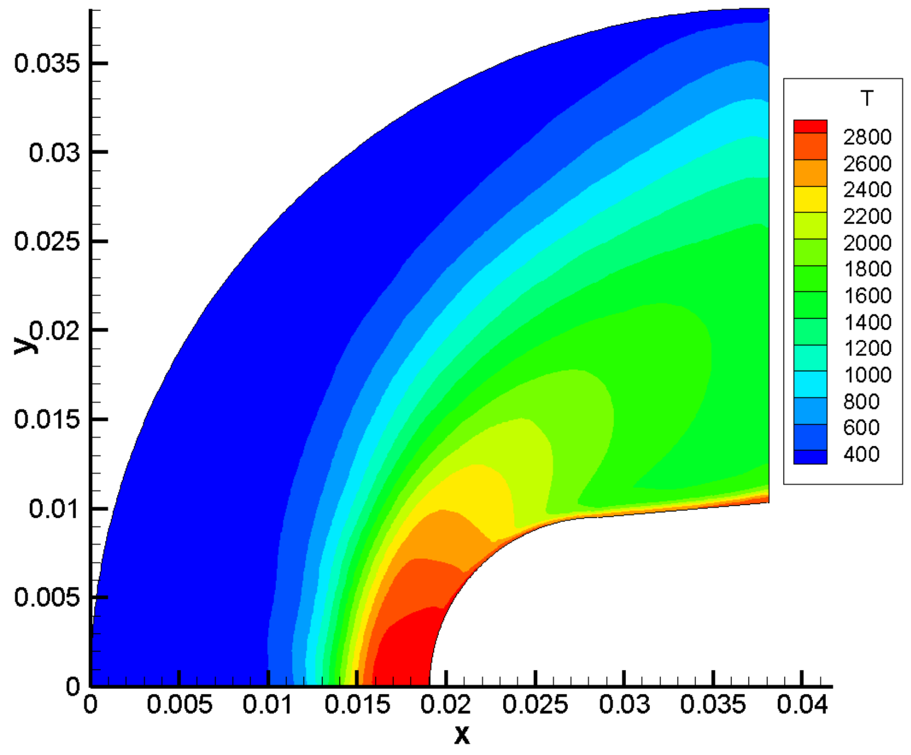

Fig. 14. Temperature contours

6. CONCLUSION

In this study, a solver is developed which consists of a rapid convergence scheme, which also grants better precision. The code is based upon the finite element method, and employs the Galerkin method of weighted residuals to solve for the equations. This code works for compressible and incompressible, viscous and inviscid, steady and transient conditions. The code is validated for three cases symbolic of the three kinds of flow based on speed, i.e. subsonic flow verified by flow pas a cylinder, transonic flow verified by the shock tube problem, and supersonic flow verified by flow past a blunt body.

This scheme is part of a larger scheme that involves a pressure correction method. Therefore this research can be taken forward in many ways, and there is much scope for further studies in this direction.

APPENDIX A: General Properties of Linear Triangular Finite Elements

A.1 Properties of Shape Functions for Linear Triangular Finite Elements

The linear triangular element shown in Figure 6-1 has straight sides and three nodes, one at each corner. A consistent labeling of the nodes is a necessity and proceeds in a counter-

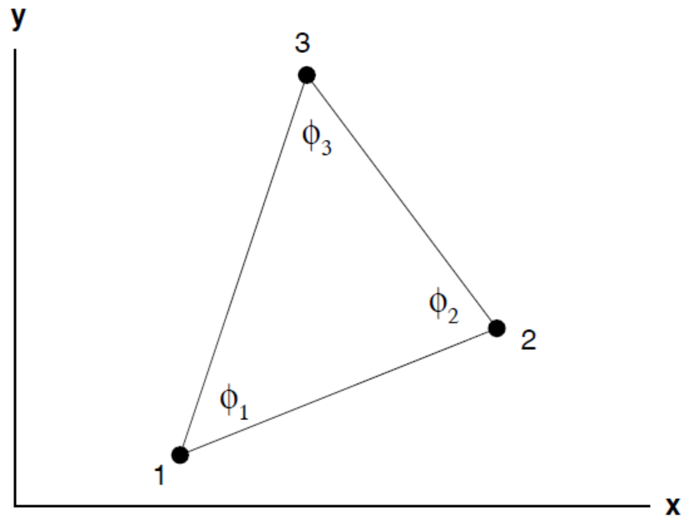

Figure 6-1 Parameters for the linear triangular element.

clockwise fashion in accordance with the right-hand rule of vectors. The labeling of node 1 is arbitrary. The nodal values of ϕ are ϕ_1, ϕ_2, and ϕ_3 whereas the nodal coordinates are (x_1, y_1), (x_2, y_2), and (x_3, y_3). The variable ϕ is independent of the spatial coordinates of the nodes. The following discussion of finite element shape functions for linear triangles is paraphrased from Segerlind (1984).

The variation of ϕ in the domain can be defined by the interpolation polynomial

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y \tag{0.1}$$

with the nodal conditions

$$\phi = \phi_1 \text{ at } x = x_1, y = y_1$$
$$\phi = \phi_2 \text{ at } x = x_2, y = y_2$$
$$\phi = \phi_3 \text{ at } x = x_3, y = y_3.$$

Substitution of these conditions into equation (0.1) produces the system of equations

$$\begin{aligned}\phi_1 &= \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 \\ \phi_2 &= \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 \\ \phi_3 &= \alpha_1 + \alpha_2 x_3 + \alpha_3 y_3\end{aligned} \qquad \textbf{(0.2)}$$

which can be recast as

$$\begin{Bmatrix}\phi_1\\ \phi_2\\ \phi_3\end{Bmatrix} = \begin{bmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{bmatrix}\begin{Bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{Bmatrix}. \qquad \textbf{(0.3)}$$

The determinant of the coefficient matrix in equation (0.3) is

$$\det\begin{bmatrix}1 & x_1 & y_1\\ 1 & x_2 & y_2\\ 1 & x_3 & y_3\end{bmatrix} = 2A$$

where A is the area of the triangle. Solving the system of equations (0.3) for ϕ_1, ϕ_2, and ϕ_3 yields

$$\begin{aligned}\alpha_1 &= \frac{1}{2A}\left[(x_2 y_3 - x_3 y_2)\phi_1 + (x_3 y_1 - x_1 y_3)\phi_2 + (x_1 y_2 - x_2 y_1)\phi_3\right] \\ \alpha_2 &= \frac{1}{2A}\left[(y_2 - y_3)\phi_1 + (y_3 - y_1)\phi_2 + (y_1 - y_2)\phi_3\right] \\ \alpha_3 &= \frac{1}{2A}\left[(x_3 - x_2)\phi_1 + (x_1 - x_3)\phi_2 + (x_2 - x_1)\phi_3\right].\end{aligned} \qquad \textbf{(0.4)}$$

Substituting equation (0.4) into equation (0.1) and rearranging produces an equation for ϕ in terms of three shape functions, N_1, N_2, and N_3, and the nodal values of ϕ, ϕ_1, ϕ_2, and ϕ_3,

$$\phi = N_1\phi_1 + N_2\phi_2 + N_3\phi_3 \qquad \textbf{(0.5)}$$

where

$$N_1 = \frac{1}{2A}[a_1 + b_1 x + c_1 y] \qquad \textbf{(0.6)}$$

$$N_2 = \frac{1}{2A}[a_2 + b_2 x + c_2 y] \qquad \textbf{(0.7)}$$

$$N_3 = \frac{1}{2A}[a_3 + b_3 x + c_3 y] \qquad \textbf{(0.8)}$$

and

$$a_1 = x_2y_3\text{-}x_3y_2,\ b_1 = y_2\text{-}y_3, \text{ and } c_1 = x_3\text{-}x_2$$
$$a_2 = x_3y_1\text{-}x_1y_3,\ b_2 = y_3\text{-}y_1, \text{ and } c_2 = x_1\text{-}x_3$$
$$a_3 = x_1y_2\text{-}x_2y_1,\ b_3 = y_1\text{-}y_2, \text{ and } c_3 = x_2\text{-}x_1.$$

Thus, the independent variable ϕ is related to the nodal values by a set of shape functions that are linear in x and y. This means that the derivatives of ϕ are constant within an element. For example, differentiating equation (0.5) with respect x gives

$$\frac{\partial \phi}{\partial x} = \frac{\partial N_1}{\partial x}\phi_1 + \frac{\partial N_2}{\partial x}\phi_2 + \frac{\partial N_3}{\partial x}\phi_3 \qquad \textbf{(0.9)}$$

with

$$\frac{\partial N_i}{\partial x} = \frac{b_i}{2A} \quad i = 1, 2, \text{ and } 3.$$

Therefore,

$$\frac{\partial \phi}{\partial x} = \frac{1}{2A}[b_1\phi_1 + b_2\phi_2 + b_3\phi_3]. \qquad \textbf{(0.10)}$$

Because b_1, b_2, and b_3 are elemental constants and ϕ_1, ϕ_2, and ϕ_3 are independent of the space coordinates, the derivative has a constant value in a linear triangular finite element.

A natural coordinate system for linear triangular finite elements is area coordinates, which are denoted by L_1, L_2, and L_3. These coordinates give the ratio of the area of a sub triangular region to the area of the complete triangle. Consider point B as shown in Figure 6-2. The first

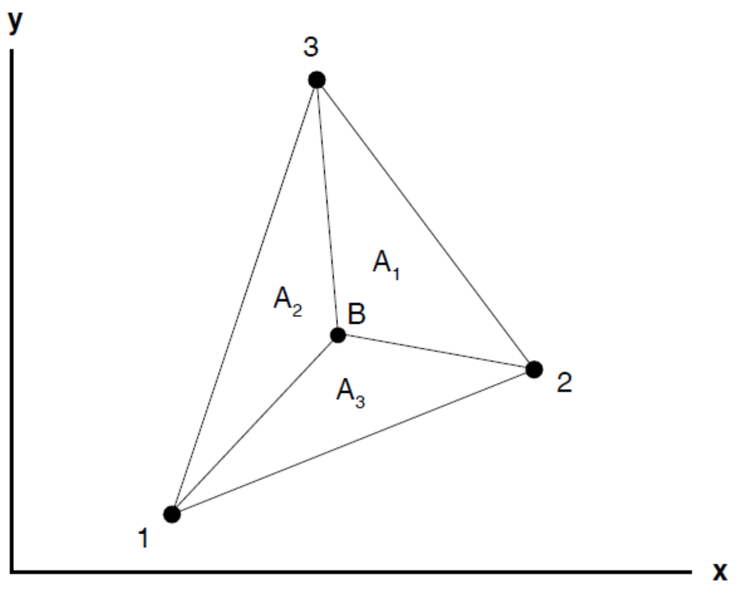

Figure 6-2 A triangular element divided into areas corresponding to the area coordinates.

area coordinate is found by forming the ratio

$$\frac{A_1}{A} = L_1 \qquad \mathbf{(0.11)}$$

which is the ratio of the area of the sub triangular region, denoted by A_1 in Figure 6-2, by the total area A of the element. Similar equations can be written for L_2 and L_3 giving

$$L_2 = \frac{A_2}{A} \text{ and } L_3 = \frac{A_3}{A}. \qquad \mathbf{(0.12)}$$

Because $A_1 + A_2 + A_3 = A$,

$$L_1 + L_2 + L_3 = 1. \qquad \mathbf{(0.13)}$$

Equation (0.11) can be recast into another form. Multiplying the numerator and denominator by 2 gives

$$\frac{2A_1}{2A} = L_1 . \qquad \textbf{(0.14)}$$

Using the determinant for the region corresponding to L_1 produces

$$\det\begin{bmatrix} 1 & x & y \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} = 2A_1 \qquad \textbf{(0.15)}$$

or

$$2A_1 = (x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y \qquad \textbf{(0.16)}$$

where x and y are the coordinates of B in Figure 6-2. Substituting equation (0.16) into equation (0.14) yields

$$L_1 = \frac{1}{2A}[(x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y]. \qquad \textbf{(0.17)}$$

Equation (0.17) is identical to equation (0.6). Thus,

$$N_1 = L_1$$

and similarly for L_2 and L_3 shows that

$$N_2 = L_2 \;\text{ and }\; N_3 = L_3 .$$

Therefore, the area coordinates for the linear triangular element are identical to the shape functions and the two sets of quantities can be interchanged.

The advantage of using the area coordinate system is the existence of an integration formula that simplifies the evaluation of area integrals (Eisenberg and Malvern, 1973). This integral equation is

$$\int_A L_1^i L_2^j L_3^k \, dA = 2A \frac{(i)!(j)!(k)!}{(i+j+k+2)!}, \tag{0.18}$$

where i, j, and k are exponential powers.

A.2 Evaluation of Basic Integrals Common to Linear Triangular Finite Elements

Equation (0.18) allows the various finite element integrals found in Chapters 2 an 3 to be exactly integrated for linear triangles. Therefore, the time consuming (and less accurate) numerical integration (Gauss quadrature) can be avoided. The following finite element volume integrals are evaluated on an elemental basis with equation (0.18):

$$\int_\Omega d\Omega = 2A \frac{(0)!}{(2)!} = A \tag{0.19}$$

$$\int_\Omega \{N\} d\Omega = 2A \frac{1}{(3)!} \begin{Bmatrix} (1)! \\ (1)! \\ (1)! \end{Bmatrix} = \frac{A}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$$

$$\int_\Omega \{N\}[N] d\Omega = 2A \frac{1}{(4)!} \begin{bmatrix} (2)! & (1)! & (1)! \\ (1)! & (2)! & (1)! \\ (1)! & (1)! & (2)! \end{bmatrix} = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{0.20}$$

$$\begin{aligned} \int_\Omega \{N\}[\vec{\nabla} N] d\Omega &= \int_\Omega \{N\} d\Omega [\vec{\nabla} N] = 2A \frac{1}{(3)!} \begin{Bmatrix} (1)! \\ (1)! \\ (1)! \end{Bmatrix} [\vec{\nabla} N] \\ &= \frac{A}{3} \begin{bmatrix} \vec{\nabla} N_1 & \vec{\nabla} N_2 & \vec{\nabla} N_3 \\ \vec{\nabla} N_1 & \vec{\nabla} N_2 & \vec{\nabla} N_3 \\ \vec{\nabla} N_1 & \vec{\nabla} N_2 & \vec{\nabla} N_3 \end{bmatrix} \end{aligned} \tag{0.21}$$

$$\int_{\Omega}\{\vec{\nabla}N\}\bullet[\vec{\nabla}N]\,d\Omega=\int_{\Omega}d\Omega\{\vec{\nabla}N\}\bullet[\vec{\nabla}N]=2A\frac{(0)!}{(2)!}\{\vec{\nabla}N\}\bullet[\vec{\nabla}N]$$
$$=A\begin{bmatrix}\vec{\nabla}N_1\bullet\vec{\nabla}N_1 & \vec{\nabla}N_1\bullet\vec{\nabla}N_2 & \vec{\nabla}N_1\bullet\vec{\nabla}N_3\\ \vec{\nabla}N_2\bullet\vec{\nabla}N_1 & \vec{\nabla}N_2\bullet\vec{\nabla}N_2 & \vec{\nabla}N_2\bullet\vec{\nabla}N_3\\ \vec{\nabla}N_3\bullet\vec{\nabla}N_1 & \vec{\nabla}N_3\bullet\vec{\nabla}N_2 & \vec{\nabla}N_3\bullet\vec{\nabla}N_3\end{bmatrix} \qquad \textbf{(0.22)}$$

Note that shape function derivative matrices may be brought outside of integrals as they are defined as elemental constants for linear triangular finite elements. The following two boundary integrals are evaluated as line integrals.

$$\int_{\Gamma}\{N\}\,d\Gamma=\frac{l_b}{2}\begin{Bmatrix}1\\1\end{Bmatrix} \qquad \textbf{(0.23)}$$

$$\int_{\Gamma}\{N\}[N]\,d\Gamma=\frac{l_b}{6}\begin{bmatrix}2 & 1\\1 & 2\end{bmatrix} \qquad \textbf{(0.24)}$$

In equations (0.23) and (0.24), l_b is the length of the face of the element on the boundary.

APPENDIX B: GEOMETRY INTERPRETATION ALGORITHMS

The following algorithms are used to find out neighboring nodes to every node, finding out the closest non-boundary neighbor to every boundary node and for finding out boundary elements in a given two dimensional linear triangular element mesh.

B.1. Finding neighbor nodes

For finding out neighbor nodes, a grid file containing element-wise nodal connectivity must be made available.

Steps of algorithm is as follows:

1. Read nodes element-wise. For triangular elements, each one of the three nodes is a neighbor to the other two.
2. Record each node and it neighbors. Since the nodes are being read element-wise, there is a possibility of repetition of node reading. Therefore, check if the node has been recorded previously.
3. Check if the other two element nodes have been already read as a neighbor to this node.
4. Record the neighbor nodes accordingly.
5. Repeat for all elements.

B.2. Finding nodes closest to boundary nodes

Nodes closest to boundary nodes, that aren’t boundary nodes themselves, are found using the array obtained from the above mentioned neighbor-node algorithm, the list of boundary nodes and the following algorithm:

1. Use the array obtained from the above mentioned neighbor node algorithm to find out the neighbors of boundary nodes.
2. From the neighbors, sort out and separate the ones that are boundary nodes themselves.
3. Find out the distance between the node under question and each neighbor node that is not at the boundary, and keep storing the closest node. The node at the end of the iterations (being run for a single boundary node) is the closest neighbor.
4. Repeat for all boundary nodes.

B.3. Finding boundary elements

Boundary elements must be determined in order to implement the boundary integration term in equn.(2.15). Following are the steps:

1. Run a loop of elements. Inside this, run a loop of boundary nodes.
2. Check if any one of the three nodes is a boundary node.
3. If one of them is, run the boundary node loop inside again and check if anyone of the other two are boundary nodes as well.
4. If another one of them is, the element is a boundary element. Record the element. Also record the two nodes that form the boundary. Alternatively, use a code to identify the orientation of the element against the boundary. For example, if n_1 and n_2 form the boundary, record an orientation code 1, and 2 for n_2 and n_3 and so on.

REFERENCES

1. “An Efficient, Semi-Implicit Pressure-Based Scheme Employing a High-Resolution Finite Element Method for Simulating Transient and Steady, Inviscid and Viscous, Compressed Flows on Unstructured Grids,” Richard C. Martineau, Ray A. Berry, Idaho National Engineering and Environmental Laboratory, Bechtel BWXT Idaho, LLC April, 2003
2. “Computational fluid dynamics- An introduction”, J.D. Anderson Jr., third edition, 2009
3. “The Finite element Method- Basic concepts and applications”, Pepper and Heinrich, 1992
4. “Fundamentals of heat and mass transfer”, Frank P Incropera , David P. Dewitt WILEY INDIA Edition.
5. “Gas dynamics: The Riemann Problem and Discontinuous Solutions: Application to the Shock Tube Problem”, “An introduction to scientific computing”, 2007
6. “A new finite element approach for prediction of aerothermal loads - Progress in inviscid flow computations”, K. BEY, NASA, Langley Research Center, Hampton, VA; E.A. THORNTON, Old Dominion University, Norfolk, VA; P. DECHAUMPHAI, Old Dominion University, Norfolk, VA
7. “Finite elements in the solution of field problems”, Zienkiewicz and Cheung, The engineer, 1965
8. J. Donea, “A Taylor-Galerkin Method for Convective Transport Problems”, International Journal for Numerical Methods in Engineering, Vol. 20, 1984.
9. A. Jameson, W. Schmidt, and E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes, AIAA Paper 81-1259, 1981