SSSU 117
ISSN 0140 3818

UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING AND APPLIED SCIENCE

CONVECTIVE CELL APPROACH FOR SOLVING INCOMPRESSIBLE EULER FLOWS: EXPLICIT FLUX VECTOR SPLITTING SCHEME USING ARTIFICIAL COMPRESSIBILITY

A.M.Wright and S.R.Turnock

Ship Science Report 117

May 1999

Convective cell approach for solving incompressible Euler flows: Explicit flux vector splitting scheme using artificial compressibility

A. M. Wright
S. R. Turnock
Department of Ship Science, University of Southampton

May 21, 1999

Abstract

Artificial compressiblity has been used to model three dimensional steady state incompressible fluid flow. The numerical scheme applies the finite volume method to the Euler equations via multi-stage explicit time integration and flux vector splitting spatial discretisation. To allow the modelling of complex geometries arbitrary polyhedra control volumes have been used, defined by connectivity of geometrical entities. In addition moving meshes are allowed and the provision for adaptive meshing is provided. Solutions of flow over a two dimensional hump and around a Wigley hullform have produced acceptable convergence histories but are as yet unvalidated as to their accuracy. Ongoing work includes the validation of data, the imposition of a free surface boundary and the ability to model unsteady flow.

Ship Science Report 117

Contents

List of Figures

Nomenclature

$\mathbf{a}$	convection speed
$\vec{A}$	Jacobian of flux vector with respect to conservative variables with components A, B, C
$\mathcal{B}$	artificial bulk viscosity factor
$\mathbf{c}$	speed of sound
e	internal energy per unit mass
E	total energy per unit mass
f^*	numerical flux
$\vec{f}_e$	external force vector
$\vec{\mathbf{F}}$	scalar flux vector with components **f, g, h**
g	gravity acceleration
h	enthalpy per unit mass
H	stagnation enthalpy per unit mass
$\vec{k}$	wave direction
K	projection of Jacobian matrix on propagation direction $\vec{k}$
$\mathcal{K}$	artificial compressibility factor
$\mathbf{L}$	Transformation matrix between characteristic and primative variables
$\mathbf{M}$	Transformation matrix between conservative and primative variables
$\vec{n}$	normal vector
p	static pressure
$\mathbf{P}$	Transformation matrix between characteristic and conservative variables
$\mathcal{P}$	preconditioning matrix for artificial compressibility
$\mathbf{Q}$	source term vector
s	entropy per unit mass
t	time
T	temperature
$\mathbf{U}$	conservative state vector
$\vec{v}$	velocity with Cartesian components u, v, w
V	volume
$\mathbf{V}$	primative variables vector
w_f	work done by external force
$\mathbf{W}$	characteristic variables vector
x, y, z	Cartesian coordinates

α	Runge-Kutta time integration coefficient
γ	specific heat ratio
λ	eigenvalue
Λ	eigenvector
ξ, η, ζ	curvilinear coordinates
ρ	density
τ	pseudo time
ψ	piezometric pressure (incorporating hydrostatic component)

Subscripts

A	artificial compressibility value
cv	contravarient value
g	grid/mesh value
m	moving mesh value
ND	non-dimensional value
t	partial derivative with respect to time
x, y, z	components in the x, y, z direction
x, y, z	partial derivatives with respect to x, y, z
∞	free stream condition
ξ, η, ζ	partial derivatives with respect to ξ, η, ζ

Superscripts

m	real time step number
n	pseudo-time step number
T	transposed (vector or matrix)

Symbols

$\frac{D}{Dt}$	substantial/total derivative
$\frac{\partial}{\partial t}$	partial derivative
∇	grad operator $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$
Δ	finite step

1 Introduction

The prediction and modelling of flow around complete marine vehicles has challanged hydrodynamicists for over a century. Only in the last decade, with the rapid increase in computer hardware, have such large simulations become possible [1, 2, 3]. Correct modelling of the free surface around marine vehicles subject to waves, current and wind is necessary for the accurate prediction of forces, wave resistance and the vessels performance in a seaway. In order to provide valuable data for designers to evaluate a design an accurate, robust and flexible flow computation scheme is needed. The flows modelled all have the characteristics of incompressibility, rotationality and viscosity in a bouyancy driven domain.While boundary methods of potental flow can model complex geometries with incompressible flow, neither rotationality nor viscosity can be modelled; lifting bodies require circulation confined to an infinitely thin wake sheet whose position is a priori unknown. Viscous boundary element methods are as yet restricted to low Reynolds numbers. Field methods on the other hand are advantageous in solving highly non-linear flows at an unlimited range of Reynolds numbers, providing a detailed model of the entire domain.

The principle difficulty in solving the Navier-Stokes equations is that they are a set of highly coupled non-linear partial differential equations. If the viscous terms are removed, providing the Euler equations (rotational flow), then the system of equations reduces from second order to first order. These equations are hyperbolic in nature for compressible flow and parabolic for incompressible flow. In the case of truely incompressible flow the eigenvalues of the solution tend towards infinity, leaving a set of parabolic equations which require more complex solution methods.

The third problem is the fundementally unsteady nature of the flow. Some form of time accurate scheme is required to achieve correct predictions of vessel motions and loadings exerted, which places extra emphasis and stringent control upon the numeric scheme.

The key to further advances in the accurate prediction of loading is in the ability to maintain a sufficiently low value of spatial discretisation error throughout the computational domain and at the same time define the position and motion of moving bodies and their intersection. This requires that significant distortions of the computational mesh are removed and that all significant physical effects are included in the numerical algorithm. The conceptual basis of the solution that has been developed is that of a convective finite volume cell; free to move according to imposed sea state, current and body motion; able to adapt to the local flow; and capable of accurately following a complex moving surface. To cope with these requirements the cellular topology must have the ability to choose the most appropriate cell topology for specific flow regimes. This requires an arbitrary description of a polyhedron via the connectivity of the geometric entites.

This report describes the development of a solver that models the Euler equations for incompressible flow in the presence of gravity and an arbitrarily moving domain. Chapter 2 details the fundamentals of the finite volume scheme. Chapter 3 defines

the underlying data structures and the geometrical entities used by the scheme. Chapters 4, 5, and 6 develop finite volume theory to be able to cope with the flow domain. The boundary conditions utilised are described in Chapter 7. The spatial and temporal discretisation of the numerical scheme is defined in Chapters 8 and 9 respectively, and the overall implementation methodology in Chapter 10. Flow solution results to date are presented in Chapter 11, and future developments in Chapter 12.

2 Mathematical representation of flow equations

2.1 Introduction

There are three techniques commonly in use today to solve discrete computational problems; finite difference, finite element and finite volume. The Finite Volume Method discretises the integral form of a conservation law directly in the physical space. Introduced into the field of numerical fluid dynamics independantly in 1971 and 1972 for the solution of two dimensional Euler flow, it presently has a wide range of applications and two distinct advantages. The method can take full advantage of an arbitrary mesh, allowing control volume shape and location manipulation, and the adoption of direct discretisation of the integral formulation ensures that the basic entities (mass, momentum and energy) will also remain conserved [4, page 237].

This Chapter introduces the governing equations of fluid dynamics and the method of discretising space into finite volumes is described.

2.2 Governing Equations

The Euler equations express a set of conservation laws for the quantities mass, momentum and energy. They are a reduced form of the full Navier-Stokes equations, describing inviscid flow where all viscous and thermal conductivity terms are neglected. This form of the governing equations is an improved approximation of fluid behaviour over the potential flow equations as it models rotationality.

Analytically, the primary advantage of the Euler equations over those models containing viscous terms is that the system of partial differential equations reduces from second to first order.

The time dependent Euler equations, in conservative form, may be written in vector notation as -

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \vec{\mathbf{F}} = \mathbf{Q} \tag{1}$$

where $\mathbf{U}$ is termed the state vector, $\vec{\mathbf{F}} = (\mathbf{f}, \mathbf{g}, \mathbf{h})$ are the flux vectors, and $\mathbf{Q}$ is a vector of source terms.

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wH \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} 0 \\ \rho f_{ex} \\ \rho f_{ey} \\ \rho f_{ez} \\ \rho W_f \end{bmatrix} \tag{2}$$

ρ is the density, u, v and w are the velocity components in the x, y, and z Cartesian co-ordinate directions, E is the specific total energy, p is the static pressure

and H is the specific stagnation enthalpy. In the source term vector $\vec{f}_e$ is the external force and W_f the work performed by the external force ($W_f = \rho \vec{f}_e \cdot \vec{v}$).

The Euler equations have still to be supplemented by the constitutive laws. The thermodynamic laws define the internal energy e or the enthalpy h as a function of only two other thermodynamic variables choosen between density ρ, pressure p, temperature T, entropy s or any other intensive variable.

For an ideal gas (compressible flow such as air), the equation of state is written as

$$\frac{p}{\rho} = RT \tag{3}$$

where R is the gas constant per unit mass and is equal to the universal gas constant devided by the molecular mass of the fluid. As a result the internal energy and enthalpy are only functions of temperature and have the following relation.

$$\gamma = \frac{C_p}{C_v} \tag{4}$$

where C_p is the specific heat coefficient under constant pressure and C_v the specific heat coefficient under constant volume.

$$e = C_v T = \frac{1}{\gamma - 1}\frac{p}{\rho} \tag{5}$$

$$h = C_p T = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} \tag{6}$$

$$H = E + \frac{p}{\rho} = h + \frac{\vec{v}^2}{2} \tag{7}$$

$$c^2 = \gamma RT = \frac{\gamma p}{\rho} \tag{8}$$

In the absence of heat sources the entropy equation for continuous flow variations reduces to

$$T\left(\frac{\partial s}{\partial t} + \vec{v} \cdot \vec{\nabla} s\right) = 0 \tag{9}$$

expressing that entropy, s, is constant along a flow path. Hence the Euler equations describe isentropic flows in the absence of discontinuities.

The equations form a set of coupled, non-linear partial differential equations, hyperbolic in nature, and dictated by the convection of wave-like properties. The different forms in which they can be represented are described in Appendix A.

The approach detailed within this work uses the conservative formulation in preference to a non-conservative form for two reasons. Numerical experiments and comparisions show that non-conservative formulations are generally less accurate than conservative ones, particularly in the presence of strong gradients. The second reason, of less importance to the type of flow being modelled, is that numerical source terms which tend to emerge in discountinuous flows can give rise to large errors

and incorrect shock intensities with a non-conservative formulation. Therefore, in order to obtain the correct discontinuities, it has been shown that it is necessary to discretise the conservative form of the flow equations [4, page 240][5].

2.3 Finite Volume Discretisation

The finite volume method discretises the integral form of a conservative law by applying it locally to discrete volumes. The summation of the local conservation laws results in the cancellation of fluxes across internal faces, re-establishing global conservation.

The integral form of the equations expresses an average change over a volume due to the flux through the surface of the volume, thus placing no restrictions upon the topology of the discrete volumes. The advantages of this method are

- There are no mesh topology nor placement of control point restrictions as with the Finite Difference method
- The conservative law can be solved directly in physical space, removing the requirement of transformation to and from curvilinear coordinates
- The spatial coordinates of geometrical entites are used only in the calculation of cell volume and face area

Due to these extra degrees of freedom the determination of the fluxes on control volume interfaces can be calculated by a variety of techniques, resulting in a versatile methodology. This versatility is a main reason for the popularity of the method, and a key factor in its selection for this work.

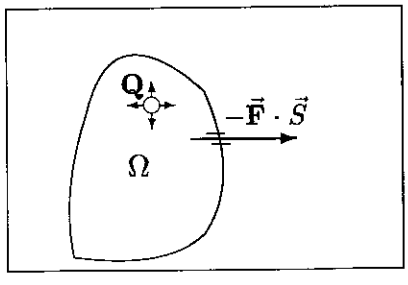

Figure 1: Arbitrary Volume

The integral conservation law, written for a discrete volume is

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{U} d\Omega + \oint_{S} \vec{\mathbf{F}} d\vec{S} = \int_{\Omega} \mathbf{Q} d\Omega \tag{10}$$

where Ω is a volume or arbitrary shape and size, as shown in Figure 1, and $\vec{S}$ is the outward facing surface contour. Therefore the time variation of the conservative

variables U depends purely upon the surface values of the fluxes and the summation of the source vector. Due to this, Equation 10 can be discretised, as shown in Equation 11, for a discrete volume Ω_i.

$$\frac{\partial}{\partial t}\left(\mathbf{U}_i, \Omega_i\right) + \sum_i \left(\vec{\mathbf{F}} \cdot \vec{S}\right) = \mathbf{Q}_i \Omega_i \tag{11}$$

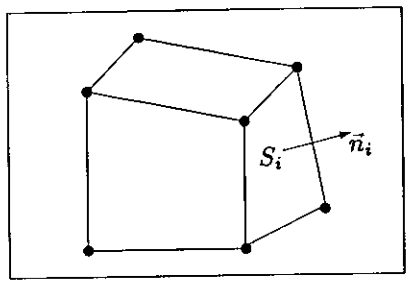

Figure 2: Discrete Control Volume

The first term in this formulation defines an average change in the conserved properties within the volume - i.e. the solution is piecewise constant. The following constraints upon the volumes must be satisfied in order to ensure conservation

- The domain volume must equal the sum of the discrete control volumes
- Overlapping of adjacent control volumes may only occur if the internal surface is shared
- Interface fluxes must be evaluated by a control volume independent formulae; this secures that fluxes on internal faces will cancel.

Due to its generality the finite volume method can handle any type of mesh, structured or unstructured. Differences between these meshes are discussed by Wright and Turnock [6]. The only geometrical definition that distinguishes a scheme is where the state variables are stored. They can either be stored at the vertices of a mesh, in a 'Cell Vertex' scheme, or at the centres of cells, in a 'Cell Centered' scheme.

2.4 Cell centered and cell vertex schemes

The most commonly used method is to store the state vector information at the cell centre, as no extra geometrical construction and calculations are required. The main disadvantage of such methods is that boundary conditions require an extrapolation of the variables onto the boundary face. This causes an extra computational cost and, more importantly, in areas of high flow gradients the interpolation can introduce numerical inaccuracies.

When the flow variables are stored at the vertices of the mesh a larger flexibility of control volume shape is possible; cells surrounding the vertex may be amalgamated, creating overlapping control volumes, or a dual mesh can be created on top of the original mesh. The second option is currently a more common approach (Rycroft [7]). One of the advantages of the cell vertex approach is the accuracy with which boundary condition values can be defined, due to the positioning of vertices on the boundaries. A second advantage stems from the fact that loss of accuracy due to skewed cells is higher for cell centered schemes than cell vertex schemes [8]. The derived conclusion of this is that non-uniform meshes would benefit from a cell vertex scheme (Rycroft [7], page 58).

As accurate surface pressures are of prime importance in marine applications (free surface deformation and hull resistance for example) and taking into account the requirement of adaptable and robust meshs a cell vertex scheme has been chosen in this work. This scheme will operate upon unstructured arbitrary control volumes; that is each control volume will have an arbitrary number of faces and be identified in an unstructured manner. Further details of this method of control volume definition are detailed by Rycroft [7] and Wright and Turnock [6].

3 Data structures and Geometrical mesh

The subject of data structure development and requirements is discussed and detailed in a previous technical report by Wright and Turnock [6]. As a result this Chapter will only describe the mesh requirements of the solver, the key geometrical entities and the resultant data structure used.

3.1 Geometrical entities

As specified in earlier work (Wright and Turnock [9]) there are four basic entites from which a mesh is created - nodes, edges, faces and cells. An edge can be uniquely defined by nodes, a face by edges and a cell by faces. The entire mesh can be uniquely defined by the cells.

Earlier work (Rycroft and Turnock [10]) utilised all these entites, linking them sequentially. Subsequent investigations and algorithm developments have removed the neccesity for explicit definition of cells and edges. Instead the use of nodes and faces alone, when joined via the correct connectivity allows the full geometrical definition. The algorithm utilises triangular faces, the union of which defines the faces of the control volume, as shown in Figure 3. The face area is a summation of the triangular segment areas and the facial normal is an area weighted average of the segment normals.

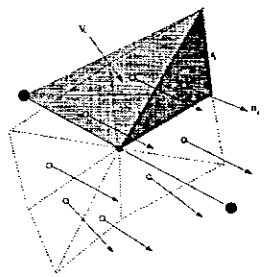

Figure 3: Control volume face definition

From these faces control volumes can be created, with the Global Conservation Law of surface area conservation being implicitly adhered to. The volume of the cells can be calculated from the triangular face segments, in an extension of the facial area calculation. Each triangular segment is defined as the base of a tetrahedron, with the apex being the control volume data point (i.e. the grid node). The summation of the volumes of these tetrahedra equals the volume of the cell. Figure 3 shows the planar area of a triangular segment and the volume subtended to it from one of the grid nodes adjacent to the face. It should be noted that this triangular segement subtends the grid node on the other side of the face, and hence a tetrahadral volume will have to be calculated for it as well.

If the mesh is contructed from a range of topologies there is no limit on the

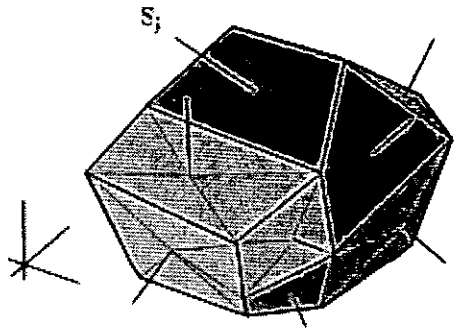

Figure 4: Polygon control volume

number of faces connected to a given vertex. Thus the control volumes associated with such a mesh may be complex polygons, as shown in Figure 4.

3.2 Mesh connectivity

Due to the unstructured nature of the data, and the arbitray polyhedral shaping of each control volume to allow for local mesh refinement (see Wright and Turnock [6]), it was decided that the flow solver will operate using an algorithm based upon a loop through all faces within the mesh. This allows all geometrical properties to be calculated and the flux terms to be evaluated. Because a cell vertex scheme has been chosen, as detailed in Chapter 2.4, the flux to/from a control volume requires that the face know which control volumes it is attached to.

To allow the construction of the face, and the calculations of areas and volumes, the faces have a connection to the 'construction' nodes that define the limits of the face. These construction nodes are in the position of cell and facial centres of the original mesh.

The construction nodes in turn require connection to the grid nodes (hence forth termed flow nodes) to allow correct interpolation of flow properties at boundaries but more importantly to aid the local remeshing process, and the subsequent reallocation of entity connections. In conjunction with this the flow nodes have connections to adjacent flow nodes to ensure the geomentrical Global Conservation Law is maintained throughout mesh adaptation as well as aiding the efficiency of the remesh procedures.

Figure 5 details all these connections graphically.

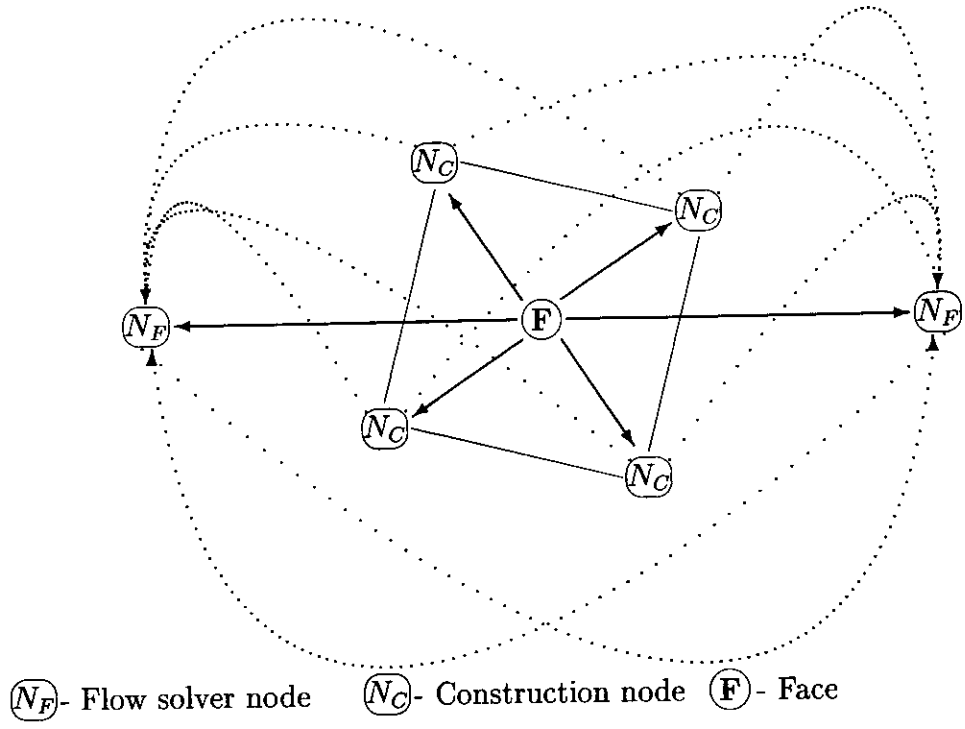

Figure 5: Connectivity of geometrical mesh

3.3 Data structure

In conjunction with the development of the basic data structure, and interconnectivity, code development has concentrated on determining the most efficient manner of accessing the data. A combination of one dimensional lists of data (stacks) and non-contiguous linked lists have been utilised with memory pointers to produce a structure that allows efficient access to data as well as minimising memory overheads. The development and details of this structure are detailed in a previous technical report (Wright and Turnock [6]). The manner of accessing the data structure is shown in Figure 6. Memory pointers are denoted by arrows, and stacks by segmented boxes. It should be noted that the stack of memory allocated to storing facial data is over-sized, to allow for the creation and destruction of faces during grid adaptation without costly global reallocation. A binary tree data structure has been devised for the flow node section of the data structure to allow flow solver node creation and destruction by remeshing. As a node is split, it is 'deactivated' (denoted by the graphical representation of a node being covered), and pointers to its 'children' created. These children are not direct members of the stack accessed by the overall grid structure, but via the parent node. Figure 7 demonstrates how this process can be repeated for several 'generations' of children.

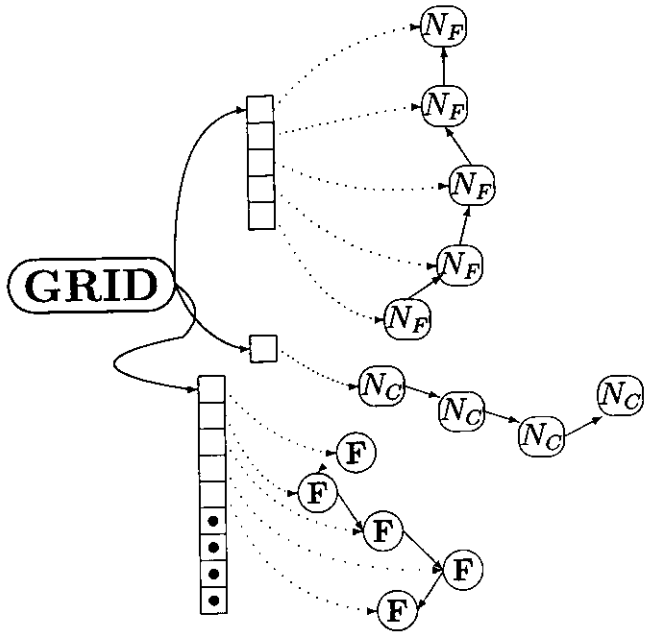

Figure 6: Data structure of geometrical mesh

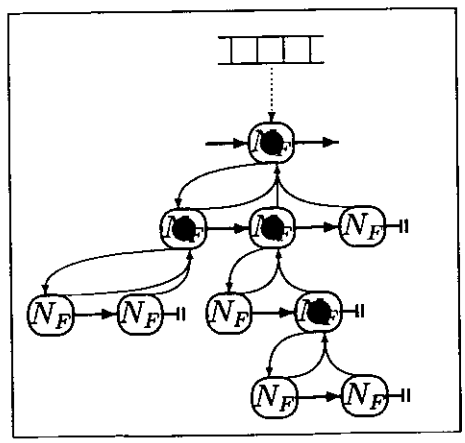

Figure 7: Binary tree data structure of flow solution nodes

4 Artificial Compressibility

The fundamental problem in the prediction of incompressible flow is the lack of a time dependant derivative in the continuity equation. This is due to the density having a constant value. In turn, a time-independent constraint must be imposed upon the momentum equations to ensure a divergence free velocity field. An additional problem is that the eigenvalues resulting from the system of Euler equations become infinite in the limit of incompressible flow; this is because for an incompressible fluid the speed of a pressure wave is infinite. Thus the standard methods for computing compressible flows cannot be used.

4.1 Theoretical development

There are methods widely used that take the divergence of the momentum equations and solve implicit equations at each time step for the fundamental variable fields (pressure and velocities) such that continuity is conserved (Hino [11], Miyata et al [12] and Tahara et al [13]). This method is however expensive due to the requirement of solving the implicit equations by an iterative method and also calculating the divergence of the momentum equations in a curvilinear coordinate system.

An alternative method is that suggested by Chorin in 1967 [14]. In the original paper Chorin describes the problems with the current methods, and highlights problems with each one. The incompressible Euler equations are listed as -

$$\begin{aligned} \nabla \cdot \vec{v} &= 0 \\ \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} &= -\frac{1}{\rho} \nabla p \end{aligned} \tag{12}$$

where $\vec{v}$ is the velocity vector, p is the pressure, and ρ is the density

Chorin suggested the addition of an artificial time dependent to the continuity equation. The Euler equations now took the form -

$$\begin{aligned} \frac{1}{\rho_o} \frac{\partial p}{\partial t} + \mathbf{c}^2 \nabla \cdot \vec{v} &= 0 \\ \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} &= -\frac{1}{\rho} \nabla p \end{aligned} \tag{13}$$

ρ_o is the true density and $\mathbf{c}$ is an arbitrary real parameter

Chorin put forward the hypothesis that if, as the calculation progresses, the solution of Equation 13 converges to a steady solution (i.e. one which does not depend on t) then it is a steady solution of Equation 12 and does not depend upon $\mathbf{c}$, the artificial speed of sound.

Chorin also stated that it was necessary for the artificial Mach number to be below unity for the set of equations to hold.

The rest of Chorin's original 1967 publication is concerned with the creation of a suitable finite difference scheme and verification that the method is applicable. Due

to the relatively early publication of the method little computational validation could be undertaken, and indeed the full scope of possibilities for the method unrealised.

The next paper of note that develops Chorin's method is Rizzi and Eriksson [15].The method is detailed in a similar manner as by Chorin, but the main application of the method was to rotational flow. This form of flow was chosen because of the widespread presence of vorticity in realistic flows. With potential models this requires prior knowledge of the position of the wake sheet; a complicated procedure for all but the most trivial of situations. The problems of solving an incompressible flow are highlighted, with the emphasis being upon the effect of wave speed upon the stability of the hyperbolic set of equations. The use of artificial compressibility is described as a pseudo-temporal evolution for the pressure which is hyperbolic and which converges to the true steady state value.

The method is described in some detail by Rizzi and Eriksson, the aspects discussed being hyperbolicity in conjunction with curvi-linear coordinates and also the value used for the artificial speed of sound, $\mathbf{c}$. By matrix manipulation the only variable that the numerical stability was dependent upon was the ratio of $\frac{\mathbf{c}^2}{|\vec{v}|^2}$. From this the condition that $\mathbf{c}^2 = max\,(0.3, r(\vec{v}\cdot\vec{v}))$ with $1 \leq r \leq 5$ has been adopted.

The CFL condition for local stability is also discussed, as are discontinuities. There are several discontinuity situations discussed, but none are given any special significance. More detail of the boundary conditions are given however, and are advanced as being of prime importance. Solid wall and far field boundary conditions are discussed. The far field boundary conditions are specified as a form of Engquist's hierarchical series of absorbing boundary conditions (Engquist and Majda [16]). This is basically a form of characteristic boundary equations.

The inclusion of artificial viscosity is promoted, due to the need to counteract non-linear aliasing associated with central differencing, to control the cascading of energy from large scale to small scale and also to ensure the existence of a steady state. Rizzi and Eriksson use a linear fourth order differencing term for each of the three spatial directions.

Rizzi and Eriksson verified their model with two dimensional irrotational and rotation and three dimensional rotational case studies. All are discussed thoroughly.

Ramshaw and Mousseau [17] develop the method of artificial compressibility further with the introduction of an accelerated solution via the use of artificial bulk viscosity. Due to sound waves being compressive in nature the most effective method of damping is bulk viscosity. The pressure gradient term of the standard Euler equations $(-\frac{1}{\rho}\nabla p)$ is replaced by $-\frac{1}{\rho}\nabla q$, with q as defined in Equation 14.

$$q = p - \rho\mathcal{B}\nabla\cdot\vec{v} \tag{14}$$

Thus the constant $\rho\mathcal{B}$ is an artificial bulk viscosity. The value of $\mathcal{B}$ was given a value to minimise the time needed to reach steady state. It is claimed that a value of $\mathcal{B} = 0.2\frac{\nabla^2}{\nabla t}$ gave very good results. The scheme devised by Ramshaw and Mousseau was validated on simple two dimensional test cases; mainly a variety of channel

flows. Improvements to the residual reductions are charted, and the conclusions drawn are that variations of this technique should be worthy of consideration in a wide range of computational flow methods.

The most recent publication that develops the concept of artificial compressibility is Farmer, Martinelli and Jameson [1].The technique is used within a finite volume multigrid Euler scheme for the solution of three dimensional fully non-linear ship wave problems. The adapted Euler equations, equivalent to set of Equations 13, are defined as -

$$\frac{\partial \mathbf{U}}{\partial t} + \mathcal{P}\left(\nabla \cdot \vec{\mathbf{F}}\right) = 0 \tag{15}$$

where $\mathbf{U}$ is the conservative vector and $\vec{\mathbf{F}} = (\mathbf{f}, \mathbf{g}, \mathbf{h})$ are the flux vectors. The preconditioning matrix, $\mathcal{P}$, incorporates the artificial compressibility factor, $\mathbf{c}$, as used in set of Equations 13.

The choice for $\mathbf{c}$ is taken as $\mathbf{c}^2 = \mathcal{K}(u^2 + v^2 + w^2)$. $\mathcal{K}$ is a constant in the order of unity. This formulation allows the artificial speed of sound to be large in areas of high velocity and low pressure to improve accuracy, while much smaller in regions of lower velocities. This choice of $\mathbf{c}$, it is stated, also greatly improves the behaviour of the boundary condition. Farmer et al discuss the effects of artificial compressibility upon upstream and downstream characteristic boundary conditions. At outflow boundaries the equation characteristics are required to be sufficiently upstream dominant, hence allowing zero gradient extrapolation. The majority of the rest of the publication is concerned with the finite volume scheme used and the coupling with the free surface algorithm.

Results obtained with this formulation were on the whole accurate, with only a little over prediction of the free surface height. It is noted that the pressure residual from the continuity equation gives a good indication of the error of divergence of mass. It is concluded that the method of solution created is accurate and up to ten times more efficient that methods reported in previous literature [1].

The method is presently in widespread use and has been adopted by the majority of projects developed in a recent EPSRC directed programme in marine CFD [18].

4.2 Theory

From the formulation of the Euler equations it can be seen that the density term can be made to appear on either the left hand side or the right hand side of the mass conservation equation. That is the conservative term in the mass conservation can be $\frac{p}{\rho}$ or it can be the pressure term alone.

This results in changes to the other conservative variables and to all the flux vectors. That is -

$$\frac{p_t}{\rho} + \mathbf{c}^2 u_x + \mathbf{c}^2 v_y = 0 \tag{16-a}$$

or

$$p_t + \mathbf{c}^2 \rho u_x + \mathbf{c}^2 \rho v_y = 0 \tag{16-b}$$

This results in the conservative momentum terms being either u and v or ρu and ρv respectively. Correspondingly the pressure terms in the momentum equations are either ρ or p, again respectively. In Ni's original paper [19] detailing his numerical scheme the latter approach was taken; that is the conservative vector was $\mathbf{U} = (p, \rho u, \rho v)^T$. However Chorin's method when defined by Rizzi and Eriksson [15], places the density upon the left hand side of the equation. With the density being a constant this causes the Jacobian to be of a simpler nature. Farmer et al [1] also utilise this formulation of the Euler equations.

Therefore the definition of the governing equations used in this work is as Equation 15 -

$$\frac{\partial \mathbf{U}}{\partial t} + \mathcal{P}\left(\frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z}\right) = 0 \tag{17}$$

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \quad \mathcal{P} = \begin{bmatrix} \mathbf{c^2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} u \\ u^2 + p \\ uv \\ uw \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} v \\ uv \\ v^2 + p \\ vw \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + p \end{bmatrix}$$

(Rizzi and Eriksson [15])

These values can be non-dimensionalised by dividing lengths and speeds by a representative distance L and the magnitude of the free stream velocity $|\vec{v}|$ respectively. Pressure is non dimensionalised by $\rho|\vec{v}|^2$.

By analogy with the equations of motion for a compressible fluid, the pressure can be related to the artificial speed of sound/ compressibility factor, $\mathbf{c}$, by an artificial density, ρ_A. These parameters are related as follows -

$$p = \mathbf{c}^2 \rho_A \tag{18}$$

When the temporal derivatives tend to zero the set of equations satisfy precisely the incompressible Euler equations, with the consequence that the correct pressure may be established using the artificial compressibility formulation, as stated by Equation 18. It is evident from this that the scheme is not time accurate, due to the non-physical representation of the domain during convergence. A logical conclusion is that local timestepping can be utilised to increase the solver efficiency.

It should be stated that the matrix $\mathcal{P}$ plays an important role in determining convergence rates and stability. $\mathcal{P}$ may be viewed as a device to create a well posed system of hyperbolic equations that are to be integrated to steady state in a manner similar to standard finite volume formulations. The artificial speed of sound can in turn be viewed as a relaxation parameter (Farmer et al [1]). The value of the artificial speed of sound is taken as

$$\mathbf{c}^2 = \mathcal{K}(u^2 + v^2 + w^2) \tag{19}$$

where $\mathcal{K}$ is a constant of order unity. Rizzi and Eriksson [15] stated that "when the ratio of $\mathbf{c}^2$ to $|\vec{v}|^2$ is greater than unity the pressure waves dominate .... and the

system is better conditioned". Thus it can be assumed that the artificial speed of sound is a locally varying constant, and the constant, K, must be greater than 1 yet small enough "so that convective and acoustic effects occur on similar time scales" (Ramshaw and Mousseau [17]).

If the altered Euler equations, 17, are written in a quasi-linear form then the eigenvalues are as stated in Equation 20. The full derivation of these values is given in Appendix B.

$$\mathbf{\Lambda} = \begin{bmatrix} \vec{v}\vec{n} & -- & -- & -- \\ -- & \vec{v}\vec{n} & -- & -- \\ -- & -- & \vec{v}\vec{n} + \mathbf{a} & -- \\ -- & -- & -- & \vec{v}\vec{n} - \mathbf{a} \end{bmatrix} \tag{20}$$

$$\text{where} \quad \mathbf{a}^2 = \vec{v}\vec{n}^2 + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2). \tag{21}$$

and n_x , n_y and n_z are local normal components in a Cartesian framework.

An important consideration is that the speed of propagation is not the speed of sound, $\mathbf{c}$, as with standard compressible flow, but instead $\mathbf{a}$.

The implementation of this theory in practise, and numerical values used in the pre-conditioning matrix are detailed in Chapter 10

5 Source terms

Compressible flow solvers in the main tend to neglect the source vector **Q**, equating the left hand side of Equation 11 to zero. This is due to the small magnitude of its members.

$$\mathbf{Q} = \begin{bmatrix} 0 \\ \rho f_{ex} \\ \rho f_{ey} \\ \rho f_{ez} \\ \rho W_f \end{bmatrix} \tag{22}$$

In Equation 22 the mass conservation term is zero anyway, and the body force (gravity) is of several orders smaller that the momentum fluxes. There are exemptions to this of course, such as in jet turbine analysis, where large quantities of momentum are imparted to the fluid, and in coupled boundary layer flow solvers, where momentum transfer is important.

Hydrodynamic analysis is however different, particularly when the free surface is being considered. Because the density of water is three orders of magnitude greater that that of air, the body force terms are crucial.

The method most often used to model these body force terms is to incorporate them with the pressure term; the static pressure, p, and the hydrostatic pressure term, $-\rho g z$, are combined to give a piezometric pressure, ψ.

$$\psi = p + \rho g z \tag{23-a}$$

$$\begin{aligned} \psi_{\text{ND}} &= \frac{p}{\rho|\vec{v}|^2} + \frac{\rho g z}{\rho|\vec{v}|^2}\frac{L}{L} \\ &= p_{\text{ND}} + \frac{z}{L}\frac{gL}{|\vec{v}|^2} \\ &= p_{\text{ND}} + z_{\text{ND}} Fr^{-2} \end{aligned} \tag{23-b}$$

In the non-dimensionalisation L is a representative distance (usually the length of the vessel), and Fr is the Froude number.

This term is placed within the adapted Euler equations (Equation 17) instead of p. The state and flux vectors now take the form shown in Equation 24. The source vector of the discrete conservation laws can thus remain empty.

$$\mathbf{U} = \begin{bmatrix} \psi \\ u \\ v \\ w \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} u \\ u^2 + \psi \\ uv \\ uw \end{bmatrix} \mathbf{g} = \begin{bmatrix} v \\ uv \\ v^2 + \psi \\ vw \end{bmatrix} \mathbf{h} = \begin{bmatrix} w \\ uw \\ vw \\ w^2 + \psi \end{bmatrix} \tag{24}$$

6 Adaptive and moving grids

6.1 Introduction

The adaptation and movement of a discretised grid is now recognised as a method of improving the solution accuracy of a computational flow solver [21, Section 5.8][22]. This is due to the fact that the grid is initially generated without a detailed knowledge of the flow behaviour. As a result the node distribution may not correspond to the solution requirements. Indeed, when very rapid solution changes occur, or when the computational domain evolves in time, it is nearly impossible to design a unique and adequate grid for the entire computational solution; this requires a numerical scheme to modify the grid between iterations and time steps. Another reason for the movement of grid node and cell positions is that due to domain boundaries moving position; either in a prescribed, periodic manner (e.g. an oscillating wing) or in an iterative, flow dependent manner (for example the free surface of water).

6.2 Theoretical development

One of the earliest papers found that discusses the application and implementation of moving grids is Thomas and Lombard [23]. A new finite difference method is introduced that adheres to the Global Conservation Law (GCL), an aspect that had often been missing from finite difference schemes before. The standard form of the Euler equations is stated, as are several finite volume derivations. The GCL is defined by Equation 25.

$$\frac{\partial}{\partial t}\int_V dV = \int_S W_S \cdot dS \tag{25}$$

where V is the cell volume and W_S the local velocity at the boundary surface S. A boundary conforming curvilinear coordinate system is utilised, via the Jacobian transformation matrix, $\vec{A}$. It is noted that the numerical value of $\vec{A}$ must be consistent with the effective cell volume utilised in the numerical discretisation of the governing equations for accuracy.

The GCL has been formulated in curvilinear coordinates, and after simplification it can be expressed as Equation 26.

$$\vec{A}_t + \left(\hat{\xi}_t\right)_\xi + \left(\hat{\eta}_t\right)_\eta + \left(\hat{\zeta}_t\right)_\zeta = 0 \tag{26}$$

$$\left(\hat{\xi}_t\right) \equiv \vec{A}\xi_t \quad \text{etc}$$

$\vec{A}$ is the Jacobian, as before

$$\xi = \xi(x, y, z, t)$$
$$\eta = \eta(x, y, z, t)$$
$$\zeta = \zeta(x, y, z, t)$$

Implicit solutions to the inviscid Navier-Stokes equations are created, and the similarity of the devised finite difference scheme to finite volume methods is shown.

A limited number of results are provided for the case of a two-dimensional short sphere-cone and for supersonic three dimensional flow past a blunt nosed body.

Various methods and general concepts concerning the implementation of adapting a grid were documented in the following years, notable papers being Hindman et al [24] and Eiseman [25].

Hindman, Kutler and Anderson attempt to devise a two dimensional unsteady Euler equation solver applicable to most fluid dynamic situations. It uses the now common multi- blocking method, with each block containing a set region of the domain, with discontinuities as block boundaries.

The Euler equations are defined in the usual format (Equation 1) with the third dimension removed, leaving the state and flux vectors as defined below. The source vector is taken to be zero. Curvilinear coordinates are used, with the Jacobian of the mapping allowing transformation between systems. The effect of the Jacobian upon the maintenance of independence of the flow solution upon grid distribution is discussed, and general conditions derived, with their relationship to the GCL noted.

Two methods of grid movement and grid speed calculation are detailed. The first - originally used by Thomson et al [26] - requires the interior node points to satisfy a non-linear elliptic partial differential equation; this grid operator, G, defined as $G = \mathcal{G}_{\mathcal{A}}\frac{\partial^2}{\partial\xi^2} - 2\mathcal{G}_{\mathcal{B}}\frac{\partial^2}{\partial\eta\partial\xi} + \mathcal{G}_{\mathcal{C}}\frac{\partial^2}{\partial\eta^2}$ ($\mathcal{G}_{\mathcal{A}}, \mathcal{G}_{\mathcal{B}}, \mathcal{G}_{\mathcal{C}}$ are coefficients derived from the curvilinear coordinates). With a redefined boundary position the new node positions can be found by this elliptical grid refinement. It should be noted that due to the non-linearity of the operator the solution to the above equation is iteratively derived. The grid speeds (vital for correct application of the GCL) can be found via backward finite differences from the boundaries but this requires extra information from the surfaces and may be inconsistent with the scheme used to define the boundary movement in time. The method preferred by Hindman et al is to differentiate the above elliptical partial differential equations with respect to time. This yields a matrix equation of the form

$$\mathcal{S}\left(u_G, v_G\right) = -\vec{A}^2\left(P_t x_\xi + Q_t x_\eta + P_t y_\xi + Q_t y_\eta\right) \tag{27}$$

where P and Q are source terms to concentrate the grid distribution, $\vec{A}$ is the Jacobian of the mapping from (x, y) to (ζ, η) and (u_G, v_G) is the speed of the grid. S is a matrix whose terms involve derivatives with respect to (ζ, η) and the source terms P and Q. The resulting system is linear, which means a direct solution can be found. In addition the grid point locations can be determined by simple time integration of these computed speeds rather than from the iterative solution of the elliptical partial differential equations.

The standard MacCormack predictor corrector scheme is used to integrate the flow equations and the GCL equation in time. The necessity of the Jacobian computed from the new grid point locations being an accurate representation of the actual Jacobian is emphasised, and the error is calculated to be of the order Δt^3. A detailed algorithm to implement this scheme is provided and results for the case

of a reflection planar shock diffraction problem at a supersonic Mach number (4.71) and a ramp angle of 60^o provided.

The number of tabulated results is somewhat limited, but they are in close accord with experimental data.

Eiseman [25] uses an alternating direction method to adapt the numerical mesh to the physical problem. This extends the work promoted by the likes of Dwyer et al [27], where clustering of mesh points is gained from domain property gradient information. The benefits listed for this method are that the transformation is non-singular, and the algorithm is simple, fast and independent of the numerical model. Various forms of clustering, such as uniform arc length and curvature clustering are discussed, and the concept of an abstract surface solution introduced. Uniformity of arc length provides a poor solution when one very large peak in the solution is present, and curvature clustering is affected by discontinuities. Also, in higher dimensions there is no unique definition for either method. The concept of uniformity can be expressed by a range of parameters, but it was assumed that arc lengths and cell volumes were the two most effective, with the greatest ability for definition. Curvature clustering can also be separated into several aspects. Curvature can be split into geodesic and normal curvature; the easiest applied method is to use normal curvature directly.

The alternating direction method is obtained when a transformation of surface coordinates is iteratively created by separate transformations along the coordinate curves that are grouped by direction and cycled towards convergence. Curve arc length is taken to be the non- dimensionalising independent variable. With the grouping into curves a series of one dimensional transformations can occur, which results in correspondingly monotone and non-singular compositions. Along each curve in the i^{th} direction, the general transformation t_i as a function of arc length s_i is determined by the proportionality

$$\partial t_i \propto W_i(s_i)\partial s_i \tag{28}$$

relating differential elements ∂t_i to ∂s_i by means of a non-zero weight function, W.

Eiseman details the weighting function and the manner in which clustering is achieved, and expands Equation 28 to incorporate the quantities requiring resolution. Collectively, the application to all i-direction coordinate curves yields a full transformation that preserves non-singularity, assuming that transverse and i direction tangents are non-aligned. Upon alternating directions and, in general, upon cycling through all directions I a finite number of times, the resulting transformation is non-singular.

Problems associated with discretely defined curves made of a finite number of points within the framework of an overall algorithm are discussed. Problems can arise from the piecewise linear approximation of the curves, and is observed when there are closely spaced curves; the curves can intersect, and cross over each other, creating a grid singularity. Curve entanglement can also be encountered. Solutions to both these problems are presented. The choice of the weighting function is

detailed, and limitations described.

The method of alternating direction grid adaptation is primarily so that disturbances can be adequately resolved. In order to test it Eiseman [25] examined several artificially created disturbances. The resulting grid distributions are shown pictorially. There are no results for actual flow simulations but the method of grid generation and adaptation presented is robust and detailed.

There are few papers on moving/adaptive grids until the 1990's, when the topic has reappeared in a series of papers from a number of institutes and development groups. One of the more noticeable such sources is that at the Ecole Polytechnique de Montéral. The first paper of note upon moving grids was by Trépanier, Reggio, Zhang and Camarero [22]. A finite volume methodology applied to the solution of two dimensional axisymmetric Euler equations on arbitrary moving grids is presented. While little is added to the possible methods of moving the grid the publication does specify clearly how to incorporate a moving grid within a numerical flow solver.

The Euler equations are detailed, with the difference from Equation 1 being that the grid speed is incorporated in the flux vectors to comply with the GCL. That is -

$$\frac{\partial}{\partial t}\int_{V(t)} \mathbf{U}\eta dV + \oint_{S(t)} \vec{n}\vec{\mathbf{F}}\eta dS = \int_{V(t)} \mathbf{Q}\eta dV \tag{29}$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho\vec{v} \\ \rho E \end{bmatrix} \quad \vec{\mathbf{F}} = \begin{bmatrix} \rho(\vec{v} - \vec{v_g}) \\ \rho\vec{v}(\vec{v} - \vec{v_g}) + \mathbf{I}p \\ \rho(\vec{v} - \vec{v_g})E + \vec{v}\rho \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} 0 \\ \frac{\vec{e}p}{y} \\ 0 \end{bmatrix}$$

where nomenclature is as before, with the addition of subscript G denoting grid values and $\mathbf{I}$ being the unit tensor. Equation 29 is set in a radial framework, hence η is the distance y from the axis and $\vec{e}$ is the unit radial vector. Note that for plane flow η is 1 and $\vec{e}$ is taken to be 0.

The nature of the GCL is stated, and the implicit approach to maintaining it is detailed, along with the numerical discretisation and flux definition. This is Roes' flux-splitting scheme [28] of a Riemann solver.

The effect grid motion causes from a physical and a mathematical point of view is noted as that only the convectable variables are affected in a physical frame which mathematically the motion only modifies the eigenvalues of the Jacobian. In both cases any term with the velocity vector $\vec{v}$ in it becomes $(\vec{v} - \vec{v_g})$. Therefore the velocity of the faces of a control volume must be specified to calculate the new convective terms. This must be done in terms of the geometry. The introduction of the extra grid motion term is detailed for Roes scheme. This was then applied to a triangular control volume mesh.

Four numerical tests were carried out to validate the method described. The first used random grid motion to study the grid motion independence of the code. On a zero velocity field with random grid motion the maximum level of the $(\rho - 1)$ field was of the order 10^{-5}, verifying the code's independence from the addition of

the grid speed terms to the governing equations. The second and third test used linearly varying grid node velocities for one dimensional compression and expansion of a gas. Results shown are good, with only small discrepancies at the shock level. The final numerical test was for the equivalent of the one dimensional tests, but in two dimensions. Again results were close to analytical results.

In a later paper the same research group detail a method of grid movement. Trépanier, Reggio, Paraschivoiu and Camarero [29] detail a methodology for the computation of unsteady Euler flows in configurations with moving boundaries. While the earlier paper listed above was primarily concerned with the manner of incorporating grid movement within a flow solver this publication details the manner of grid-flow coupling via error estimation.

The same method of domain discretisation using unstructured triangular grids via a generalised version of Roes' approximate Riemann solver, taking into account grid movement was used; see Equation 29 and Trépanier et al [22]. Boundary conditions were enforced by using the idea of image cells at the boundaries in which the flow properties are set according to the type of boundary, and then the Riemann solver is applied to compute the flux across these boundaries. At walls all flow is tangential and at inlets/outlets characteristic boundary conditions are imposed, utilising the direction of the characteristics of the flow properties.

The grid management algorithm is described in three separate topics; the geometry induced grid, the flow induced grid and grid re-meshing due to poor quality. The geometry induced grid is defined as taking account of the effect of the moving curves via their velocity and also an indirect smoothing term based on nodal displacement. Thus the velocity of the grid nodes can be represented by $\vec{v_g} = v_{\vec{GG}} + v_{\vec{GS}}$ where the subscript GG denotes the geometrical grid velocity while the subscript GS is a smoothing term. This resulting grid speed, $\vec{v_g}$, is then used by the flow solver to compute the volumetric variations of the computational finite volume cells. The differing attributes of Dirichlet and Neumann curve-node interaction are discussed, and it is stressed that if only the nodes near the moving boundary are moved then a rapid degeneration of the grid near the boundary will occur. To minimise this the velocities of the nodes on the boundaries can be used to calculate the velocities of the internal nodes. It is noted that while the imposition of a Laplacian equation for each velocity component would provide a very smooth field but would also be extremely expensive computationally. The alternative promoted in this publication is to assign to the internal nodes the mean velocity of their direct neighbours. If this procedure is repeated for a few iterations then the result is a diffusion-like operator that smoothes out the large variations in grid velocity. The smoothing term, $v_{\vec{GS}}$, produces an additional smoothing of the transient grid evolution by averaging the position of its neighbours. The velocity of the grid node is then calculated be dividing the node translation by a time interval, which is related to the non-dimensional time scale of the problem. A point of caution is that the curves upon which the nodes are moved can become entangled; such problems can be eliminated during the re-meshing stage.

The flow induced grid adaptation created by Trépanier et al strives to equalise the local error for all of the computational cells. This is accepted as being the optimal grid for the numerical solution. The error estimated was the integration of the difference between the piecewise computed solution in each element and a piecewise linear solution. This method is described in greater detail by Ilinca, Camarero, Trpanier and Reggio [30]. A full listing of the equations and mathematical requirements is provided, as well as adding extra depth to some aspects of the earlier publication [29]. Once the error has been estimated for all given volumes it has to be spread evenly over the domain. Trépanier et al assume the error to be proportional to grid size, and the characteristics of the next grid are thus obtained by scaling.

Dynamic re-meshing is used to satisfy the requirements of the geometry and flow induced grids. Global re-meshing can be expensive so the system used consists of local actions to refine, coarsen, cure and/or smooth the evolving grid. The methods of refinement and coarsening are discussed, and a method for grid curing proposed. This is based upon a quantitative measure of grid quality involving cell area and edge lengths -

$$\text{QUALITY} = \frac{4\sqrt{3} \quad \text{AREA}}{|\text{EDGE}_1|^2 + |\text{EDGE}_2|^2 + |\text{EDGE}_3|^2} \tag{30}$$

Thus a triangle has a value of 1 if equilateral and 0 if degenerate. A watershed limit of 0.4 was deemed to define a bad triangle. The cure methods used were the swapping of diagonals and the deletion of bad triangles. The transfer of the grid adaptation solution transfer and the overall re-meshing algorithm are discussed, with the implicit conservation of information between meshes being noted and the various strategies involved in the re- meshing stated. The frequency of the re-mesh was determined by two conditions; a geometric re-mesh was performed each time the minimum time interval needed to reduce one of the triangle areas by half is reached while a flow re-mesh is carried out after a certain number of iterations (the values of which were not stated in the paper).

Results were obtained for an oblique shock reflection, a blast wave from a shock tube , an exploding pressure vessel and the operation of a circuit breaker. All were validated with experimental data and compared against previous simulations. The results indicate good potential for industrial applications with further developments especially in the error estimation.

Slater, Liou and Hindman [31] present an approach for the generation of dynamic grids utilising grid speeds computed from the time differentiation of a set of grid equations. The Euler equations are written in curvilinear coordinates, and the requirements of operating in both the Lagrangian and the Eulerian points of view are noted in a similar manner as previous papers. The equations used to compute the grid speeds are derived from the Euler - Lagrange equations in the variation manner utilised by Brackbill and Saltzman [32]. These equations relate the physical space to the parametric computational space. The Lagrangian is defined so as to include measures of smoothness, orthogonality and cell volume adaptation.The method of

defining these measures is discussed and the relative weightings used are defined. A matrix format is used throughout, with the base components being the Jacobian derivatives. The grid speed equations are obtained from the time derivation of the grid equations. Utilising the chain rule this is reduced to cross derivatives with respect to time and the computational space. These equations are discretised using second order central differences and solved using the Gauss-Seidel point relaxation method with Neumann boundary conditions to impose orthogonality.

Coupling of the flow and grid equations is achieved in time via an explicit two-stage Lax- Wendroff method, the manner of which is detailed, with a full algorithm detailed.

Results were obtained for a simple dynamic boundary (one edge of a rectangular domain rotating about a point) and for inviscid start-up flow in a nozzle. Effective values for numerical damping and iteration levels are picked out. For all cases the weighting function of the grid adaptation was dependant upon flow density (hence capturing shock waves with more accuracy). One unexpected effect of using dynamic grids is that integrating the grid speeds prior to solving the grid equations at the next time step does not provide a suitable first guess of the solution. Instead perturbations are introduced into the flow-field, and hence inaccuracies. Apart from this anomaly it was proven that the use of a dynamic grid was more efficient, due to the use of linear equations, as opposed to non-linear equations. The possible applications of the method to viscous flows are mentioned.

Another series of papers by the same author that explores the use of moving grids are those by Gaitonde and Fiddes [33, 34, 35]. In the three papers different aspects of the same research are detailed and discussed. A three dimensional moving mesh method for the calculation of unsteady transonic flows has been created. Each paper provides a small introduction to the topic, and cites the reason for the use of the Euler equations to be the accurate capturing of any shock waves.

All grid movement in this research is due to moving boundaries - naturally occurring in unsteady flows. Mesh adaptation to improve flow prediction accuracy is not dealt with. The basic method used was algebraic grid generation based on transfinite interpolation. An advantage of this technique is that the inertial speed of an arbitrary grid point can be found by the same interpolation scheme. The boundaries are assumed to have a prescribed motion, and at each step the transformation from the curvilinear computational domain to the physical domain is a vector-valued function parametric co-ordinates of the grid. The full algebraic calculations are defined, and the inclusion of stretching functions is covered. The grid speeds are found as the differential of the transformation function. It is assumed that the stretching coefficients are independent of time for the sake of simplicity.

The flow solver used was a finite volume implicit scheme, using both a cell centred and cell vertex method. The variations between these two methods is described in Chapter 2. It shall merely be stated at this point that Gaitonde and Fiddes use the standard Euler equations with the contra-variant velocities again being $(\vec{v} - \vec{v_G})$ as defined by Trépanier et al [22].

In the second paper of 1995 (reference [34]) Gaitonde and Fiddes make note of the GCL, stating that if the volumes are evaluated exactly in terms of the cell co-ordinates errors will be introduced by the moving grid. This is because the grid motion is only approximately solved in the integration scheme. In order to avoid these errors the GCL devised by Thomas and Lombard [23] was utilised.

In all three publications the validating data used was for AGARD foil test cases and a LANN wing undergoing non-rigid motion. Results are promising, proving the moving grid procedure to be simple, flexible and effective. Areas highlighted for further development were the effect of the various parameters and coefficients of the grid generation procedure upon the solution quality as well as the robustness of the method.

One of the most recent papers published that incorporates adaptive gridding with the Euler equations is Riemslagh and Dick [36]. A multigrid method is applied to unstructured meshes for steady state problems. The flux difference splitting method applied to unstructured grids is detailed, as are the implementation of boundary conditions. Polynomial flux difference splitting with second order corrections is applied in the model used. This is detailed much more precisely by Dick [37]. The boundary conditions used were an impermeable solid wall (the convective part of the flux vector was set to zero) and a far field condition where the cell on the external side was given free stream properties. Difficulties and hence modifications to the solid boundary condition were required at the finite trailing edge of a foil to maintain a correct Kutta condition.

The concept of grid adaptation used was to start with a completely uniform grid and locally adapt it to capture interesting flow features; in this research these were deemed to be shock waves, stagnation regions and the trailing edge of the foil (NACA 0012). The criteria used to refine the mesh were the local flow parameter gradients. Error estimation based criteria were not considered. The first parameter was based upon the pressure difference over an edge. If

$$|p_i - p_j||EDGE_{ij}| \geq \frac{|p_{max} - p_{min}||EDGE_{ref}|}{S_p} \tag{31}$$

Then $EDGE_{ij}$ would be refined by placing an extra node in the centre of the edge. P is the pressure, L the edge length and S a sensitivity parameter. The reference edge length was taken as the foil chord length. This criteria is used to capture shock waves and stagnation regions. A second criteria accounted for the entropy gradient in exactly the same formulation as the pressure gradient in Equation 31, with S_p bieng swapped for S_s. This criteria was used to trigger shock wave and tangential discontinuity capture. The sensitivity constants were chosen as S_p=250 and S_s=60.

Within the flow model three such adaptations were carried out, with smoothing occuring after each. On the finest multigrid mesh this adaptation caused an increase of nodes from 1944 to 8566 (an increase of 341%) while the coarsest mesh only had increase of 14% on an initial quantity of 158 nodes.

The lift coefficient results correlate well with reference results in the AGARD

report for inviscid flow methods, but there are some discrepancies in the drag readings. As with all other multigrid methods the increased efficiency is commented upon.

6.3 Theory

To allow moving grids with the Euler equations adapted for artificial compressibility (Equation 17) the concept of contravarient velocities must be added to the flux vector $\vec{\mathbf{F}}$, as demonstrated in Equation 29.

Thus the state vector and flux vector now take the form -

$$\mathbf{U} = \begin{bmatrix} \psi \\ u \\ v \\ w \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} (u - u_G) \\ u(u - u_G) + \psi \\ v(u - u_G) \\ w(u - u_G) \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} (v - v_G) \\ u(v - v_G) \\ v(v - v_G) + \psi \\ w(v - v_G) \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} (w - w_G) \\ u(w - w_G) \\ v(w - w_G) \\ w(w - w_G) + \psi \end{bmatrix} \tag{32}$$

All nomenclature is as before, with ψ, the piezometric pressure, replacing p as in accordance with Chapter 5.

Due to the manner in which the speed of propagation, $\mathbf{a}$, is related to the local velocity it also becomes dependant upon the contravarient velocity. For simplification a few other velocity terms are required -

$$\overrightarrow{v_{cv}} = \vec{v} - \overrightarrow{v_g} \tag{33-a}$$

$$\overrightarrow{v_m} = \frac{\overrightarrow{v_{cv}} + \vec{v}}{2} = \vec{v} - \frac{1}{2}\overrightarrow{v_g} \tag{33-b}$$

The propagation speed within moving grids is defined as -

$$\mathbf{a}_m^2 = \overrightarrow{v_m}\vec{n} + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2) \tag{34}$$

As a subsequent result of the flux vector changing, the eigenvalues have also changed from those defined by Equation 20. The eigenvalues for a moving mesh are defined by Equation 35.

$$\mathbf{\Lambda} = \begin{bmatrix} \overrightarrow{v_m}\vec{n} & -- & -- & -- \\ -- & \overrightarrow{v_m}\vec{n} & -- & -- \\ -- & -- & \overrightarrow{v_m}\vec{n} + \mathbf{a}_m & -- \\ -- & -- & -- & \overrightarrow{v_m}\vec{n} - \mathbf{a}_m \end{bmatrix} \tag{35}$$

The full derivation of these values and the definition of the split flux vector is in Appendix B.

6.4 Implications

As a result of non-stationary meshes a number of issues arise.

Facial area : Due to nodal movement, facial areas will vary from time-step to time-step and hence must be re-calculated each time.

Cellular volume : As with the faces the cellular volume will vary between time-steps, hence requiring a re-calculation each time. As a result local time-stepping is even more favourable than before, due to the global search required for a global time step.

Numerical stability and convergence : Due to the lowering of the propagation speed (refer to Equations 33-b and 34) the scheme will become more stable, hence a larger time-step can be used and convergence reached sooner. This is dependant upon the grid moving in the same direction as the flow. If however the grid speed is in a negative direction with respect to the flow then the time step will have to be reduced to maintain stability. Thus it is in the interest of convergence that as well as concentrating mesh points in areas of high gradients, the mesh convects with the flow, in the direction of the local streamline.

Mesh entanglement : With successive movements there is the possibility of mesh entanglement; that is either edges of a face crossing each other or faces of a volume intersecting. It is necessary therefore to initiate a regular grid quality check. The checks and rectification techniques used in conjunction with this flow solver code are detailed in previous technical reports by Wright and Turnock [9, 6].

7 Boundary Conditions

A computational domain is incapable of representing an infinite physical domain; the truncation of the domain introduces artificial boundaries. These boundaries must simulate the flow features to correctly model the internal flow field as well as ensuring a stable numerical method with acceptable convergence characteristics. The relative position of these boundaries to the disturbed flow within the domain in conjunction with the manner in which they are treated is a key influence upon the overall accuracy of the solution. As well as the various forms of far field boundaries the domain will also be bounded by natural solid boundaries, and deforming boundaries such as the water-air free surface. Both of these type of boundaries require special treatment to ensure that the physical properties are correctly simulated. An additional problem is that higher order stencils may not be fully defined in the region of any boundary, adding an extra requirement to the algorithm.

Presently three types of boundary conditions have been created; a solid impermeable wall, a far field condition, and a characteristic boundary condition (either inflow or outflow).

7.1 Solid boundary treatment

The physical solid boundary is most easily defined by its non-porous nature. This results in no normal velocity; this is expressed in Equation 36.

$$\vec{v} \cdot \vec{n} = 0 \tag{36}$$

The definition of a non-porous surface equates to no mass flux through it. Equation 36 does not achieve this alone however; mass flux terms across solid wall faces of control volumes are also set to zero. These two procedures can be summarised as follows:

1. Restrict the mass flux via removing any mass flow terms from the flux vectors on the solid wall faces. Thus the only contribution is from the pressure term.

$$\mathbf{f} = \begin{bmatrix} 0 \\ \psi \\ 0 \\ 0 \end{bmatrix} \mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ \psi \\ 0 \end{bmatrix} \mathbf{h} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \psi \end{bmatrix} \tag{37}$$

2. Impose Equation 36 by aligning the flow in the control volume with the boundary by redefining the velocity vector in a tangential direction.

$$\vec{v}_{new} = \vec{v}_t \left(\frac{|\vec{v}|}{|\vec{v}_n|} \right) \tag{38}$$

where $\vec{v}_t = \vec{v} \times \vec{n}$ and $|\vec{v}_n| = \vec{v} \cdot \vec{n}$

The easiest manner of implementing stage 1 is to reflect all the fluxes in the solid wall. This simply ensures all fluxes normal to the boundaries are cancelled by a flux of equal magnitude but opposite direction. Unfortunately the use of arbitrary polyhedrals on a three dimension scheme makes this method difficult to implement. A second approach that was taken instead was to apply zero mass flux to a face on the solid surface. The pressure on this face was calculated from a linear interpolation of the pressure of surrounding vertices.

A problem that has arisen due to the use of a cell vertex scheme is that a numerical error is introduced as a result of the inconsistency with the position of the vertex and the centre of the control volume. While not creating any serious problems this is an area requiring further consideration.

7.2 Open boundary treatment

By far the simplest is the far field condition. It is assumed that the flow is entirely tangential to the boundary; i.e. there is no disturbance from the external state, and there is no movement of mass or momentum from the domain outwards. Thus the only contributors to the residual at a far field boundary node are the internal cells; in effect the addition of any imaginary external cell is zero. Correspondingly, when the residual is calculated for the immediately internal cell, the flux term from the face of the control volume that is on the boundary is zero.

The characteristic boundary equations are the most technical and complex, but are correspondingly more robust and adaptable to a variety of flow situations. The basic concept is to produce a boundary that is non reflective, so that the calculated flow field is independent of the location of the far field boundaries. The exact formulation and derivation of the conditions can be seen in papers such as Engquist and Majda [16] and Giles [38]. As has been stated in previous Chapters, the behaviour of the Euler equations is hyperbolic with disturbances propagating in a wave like manner, as defined in Appendix A. The characteristic variables express the governing equations in a quasi-linear format, reducing them to a set of uncoupled equations, the gradients of which are the eigenvalues as defined in Chapter 4.2 and defined by Equation 20. In isentropic flow the Riemann variables are assumed to stay constant along the respective characteristic lines, and are termed the Riemann Invarients. These invarients can be used to evaluate new values on the open boundary based upon the propagation of information from up and downstream of the boundary. The domain from which invariant is obtained depends upon the sign of the respective eigenvalue, where the problem is posed in a normal direction to the boundary. The eigenvalues can thus be expressed as

$$\begin{aligned} \lambda_1 &= \vec{v}_n \\ \lambda_2 &= \vec{v}_n + a \\ \lambda_3 &= \vec{v}_n - a \end{aligned} \tag{39}$$

Artificially compressible flow is always subsonic so eigenvalues λ_1 and λ_2 will always travel downstream and eigenvalue λ_3 upstream.

The method of characteristics described can be automatically built into the three dimensional upwind scheme described in Chapter 8 by setting a left or right fluid state as the prescribed free stream or interior of the domain, depending upon the direction of the normal velocity to the boundary. Thus the numerical flux on open boundary faces are found through application of the same algorithm as is used for internal face fluxes.

8 Spatial discretisation

8.1 Introduction

The simplest spatial discretization schemes are based on central differencing and have a symmetry with respect to a change in sign of the Jacobian eigenvalues which does not discriminate between downstream and upstream influences. As a result the propagation of disturbances along characteristics is not numerically modelled. Upwind schemes introduces the physical properties of the flow into the numerical discretisation. A problem with second- (and higher) order central schemes is the generation of oscillations in the locality of disturbances which require artificial viscosity to be damped. The construction of schemes which take into account the the physical properties of the fluid and equations aim to prevent such undesired oscillations. The first level of introduction of physical characteristics is termed flux vector splitting, and introduces information as to the sign of the eigenvalues. The flux terms are split and discretised directionally, according to the direction of the associated propagation speeds.

8.2 Flux vector splitting of Euler equations

In characteristic form, the Euler equations are written as Equation 40; see Appendix A for detail.

$$\frac{\partial \mathbf{W}}{\partial t} + \Lambda \frac{\partial \mathbf{W}}{\partial x} \tag{40}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix}$$

The eigenvector, Λ, can be split into a positive and negative matrix, such that

$$\begin{aligned} \Lambda &= \Lambda^+ + \Lambda^- \\ |\Lambda| &= \Lambda^+ - \Lambda^- \end{aligned} \tag{41}$$

or

$$\begin{aligned} \lambda &= \lambda^+ + \lambda^- \\ |\lambda| &= \lambda^+ - \lambda^- \end{aligned} \tag{42}$$

Λ^+ has only positive eigenvalues, and Λ^- has only negative eigenvalues.

Thus to gain directionality in the conservative form of the Euler equations the eigenvalues must be incorporated in the Jacobian via the transformation matrices between the characteristic and conservative forms. If $\mathbf{K}$ is taken to be the generalised Jacobian of the flux components for the conservative form of the Euler equations

(i.e. $\mathbf{K} = n_x A + n_y B + n_z C$ where $A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ etc.), then eigenvalues can be found from the solution of $det|\mathbf{K} - \lambda \mathbf{I}| = 0$. This can be solved by either solving for $det|\mathbf{K} - \lambda \mathbf{I}| = 0$ or by solving for the eigenvalues of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ seperately, and then combining the results. Either way, the resultant eigenvalues are as shown in Equation 20.

If $\mathbf{P} = \frac{\partial \mathbf{U}}{\partial \mathbf{W}}$, the transformation matrix between forms of the equations, then the directionalised eigenvalues can be incorporated within the Jacobian, as shown in Equation 43.

$$\begin{aligned} \mathbf{W_t} + \mathbf{\Lambda U_x} &= 0 \\ \Rightarrow (\mathbf{P^{-1}U_t}) + \mathbf{\Lambda}(\mathbf{P^{-1}U_x}) &= 0 \\ \Rightarrow \mathbf{PP^{-1}U_t} + \mathbf{P\Lambda P^{-1}U_x} &= 0 \\ \Rightarrow \mathbf{U_t} + (\mathbf{P\Lambda P^{-1}})\mathbf{U_x} &= 0 \end{aligned}$$

$$\Rightarrow \mathbf{K} = \mathbf{P\Lambda P^{-1}} \tag{43}$$

It should be noted that the system of equations maintains hyperbolicity if the eigenvalues are real and the norms of the transformation matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ are uniformly bounded for arbitrary real n_x, n_y and n_z.

The matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ can be gained from the asymmetry of the $\mathbf{K}$ matrix; there exists a set of right eigenvectors and a set of left eigenvectors (r^j and l^j respectively) of $\mathbf{K}$ such that the rows of $\mathbf{P^{-1}}$ are equal to the left eigenvectors and the columns of $\mathbf{P}$ are equal to the right eigenvectors.

$$\begin{aligned} \text{i.e.} \quad (l_i)^j \mathbf{K}_{ik} &= \lambda_j (l_k)^j \ , \ \textit{summation in } i \\ \text{and} \quad \mathbf{K}_{ik} (r_k)^j &= \lambda_j (r_i)^j \ , \ \textit{summation in } k \end{aligned} \tag{44}$$

From this an upwind formulation can be obtained with the Jacobians

$$\begin{aligned} \mathbf{K^+} &= \mathbf{P\Lambda^+ P^{-1}} \\ \mathbf{K^-} &= \mathbf{P\Lambda^- P^{-1}} \\ \text{with} \quad \mathbf{K} &= \mathbf{K^+} + \mathbf{K^-} \end{aligned} \tag{45}$$

Hence one has

$$\begin{aligned} & \mathbf{F} \\ = \ & \mathbf{F^+} + \mathbf{F^-} \\ = \ & \mathbf{K^+ U} + \mathbf{K^- U} \\ = \ & \mathbf{P\Lambda^+ P^{-1}} + \mathbf{P\Lambda^- P^{-1}} \end{aligned} \tag{46}$$

As stated by Equation 44, the matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ can be gained from these eigenvalues combined with the matrix $\mathbf{K}$. By Equations 43 and 46, the split flux vectors $\mathbf{F^+}$ and $\mathbf{F^-}$ can be found. The derivation and definition of these matrices, for both stationary and moving meshes, can be seen in Appendix B.

8.3 First order discretisation

If the flux terms of the conservative form of the Euler equations (Equation 17) are split according to the direction of the characteristics, then

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0 \tag{47}$$

Taking the one dimensional problem as a case study, if $\mathbf{F}^+$ is discretised with a backward difference scheme and $\mathbf{F}^-$ discretised with a forward difference then the concept of the upwind scheme will be realised. For simplicity the subscript i defines the discrete spatial position while the superscript n defines the timestep.

$$\begin{aligned}
\mathbf{U}_i^{n+1} - \mathbf{U}_i^n &= -\frac{\partial t}{\partial x}(\mathbf{F}_i^+ - \mathbf{F}_{i-1}^+)^n - \tau(\mathbf{F}_{i+1}^- - \mathbf{F}_i^-)^n \\
\Rightarrow \Delta\mathbf{U}_i^n &= -\frac{1}{2}\frac{\partial t}{\partial x}\{\mathbf{F}_{i+1} - \mathbf{F}_{i-1}\} + \frac{1}{2}\frac{\partial t}{\partial x}\{|\mathbf{F}_{i+1}| - 2|\mathbf{F}_i| + |\mathbf{F}_{i-1}|\} \\
\Rightarrow \Delta\mathbf{U}_i^n &= -\frac{\partial t}{\partial x}(\mathbf{F}^*_{i+\frac{1}{2}} - \mathbf{F}^*_{i-\frac{1}{2}}) \qquad (48)
\end{aligned}$$

$$\text{where} \quad \mathbf{F}^*_{i+\frac{1}{2}} = \frac{\mathbf{F}_{i+1} + \mathbf{F}_i}{2} - |\mathbf{A}|\frac{\mathbf{U}_{i+1} - \mathbf{U}_i}{2} \tag{49}$$

This scheme is 1^{st} order accurate in both time and space. A higher order scheme (e.g. MUSCL) is not being implemented at this stage because of a number of reasons. The developmental stage of the code incorporated with the unstructured nature of the spatial grid and the unstable nature of higher order schemes makes the implementation of such a scheme difficult, while the use of adaptive meshing, as described by Wright and Turnock [6], negates the increased accuracy of a higher order scheme.

8.4 Implementation

The calculation of the individual split eigenvalues is accomplished in the same manner as that by Steger and Warming (1986) -

$$\lambda^{\pm} = \frac{\lambda \pm \sqrt{\lambda^2 + \varepsilon^2}}{2} \tag{50}$$

The ε is added because of the lack of continual differentiablity at zeros of the eigenvalue (i.e. sonic and stagnation points). It should be noted that for the case of artificial compressibility only stagnation points are of interest due to the flow being sub-(*pseudo*)sonic at all times.

Thus we have a first order scheme where the conservative vector and flux vector states at each of the nodes surrounding a face are combined to produce the flux through the face. This is purely one dimensional, but can be expanded into three dimensions by the adoption of the generalised Jacobian matrix $\mathbf{K}$ instead of $\mathbf{A}$ and $\vec{\mathbf{F}}$ instead of $\mathbf{F}$.

The entire flux into a cell is then the summation of the flux throught all surrounding faces, such as -

$$\Delta \mathbf{U}_x = -\frac{\Delta t}{\Delta V} \sum_{\alpha=1}^{no.faces} \mathbf{F}^*_\alpha Area_\alpha \tag{51}$$

9 Time Integration

9.1 Numerical schemes

The full Euler equations have hyperbolic characteristics, which lends them to time marching procedures where numerical waves are naturally convected. Both explicit or implicit schemes can be used; the numerical stencil used for updating a point explicitly does not include points on the new time level whereas an implicit stencil does include such points.

Due to this use of purely known data points, explicit schemes can be easily implemented whereas implicit schemes require a more complex solution method, typically a matrix solver. Implicit schemes are often used however because of stability considerations. Explicit schemes are always conditionally stable but implicit schemes have less conditions, and often are unconditionally stable. That is, the time step in explicit schemes is closely linked to the size of the mesh, usually resulting in a very small time step. Implicit schemes on the other hand can have much larger timesteps and hence reach a converged solution sooner.

Due to their relative ease to program, and ability to allow more efficient algorithms an explicit scheme has been used in this work. It is also noted that for unsteady schemes with grid adaption and motion explicit schemes are liabel to be more effective.

9.2 Time integration

As stated in Chapter 4, artificial compressibility destroys time accuracy, so a time accurate discretisation scheme is not required, nor is one which uses small time steps. Hence one stage explicit methods, such as Euler's method, are not used; instead a multistage method has been implemented, as used by Rizzi amd Eriksson [15] and Farmer et al [1]. An explicit fourth order Runge-Kutta scheme is used

$$\begin{aligned} U^0 &= U^{n+1} \\ U^1 &= U^0 + \alpha_1 \frac{\Delta t}{V_\Omega} R\left(U^0\right) \\ U^2 &= U^0 + \alpha_2 \frac{\Delta t}{V_\Omega} R\left(U^1\right) \\ U^3 &= U^0 + \alpha_3 \frac{\Delta t}{V_\Omega} R\left(U^2\right) \\ U^4 &= U^0 + \alpha_4 \frac{\Delta t}{V_\Omega} R\left(U^3\right) \\ U^{n+1} &= U^4 \end{aligned} \tag{52}$$

where R(U) is the flux integral over the volume, as defined by Equations 49 and 51, and the coefficients α_i are given as

$$\alpha_1 = 0.15 \quad \alpha_2 = 0.3275 \quad \alpha_3 = 0.57 \quad \alpha_4 = 1.0 \tag{53}$$

The maximum allowable time step to maintain a stable scheme requires that the Courant (CFL) number for this explicit time marching was taken to be $2\sqrt{2}$. This method was presented by Jameson et al [39] in 1981. A representative distance over which a wave is permitted to travel is found from a volume to area ratio. This combines with a combination of the numerical wave solution propagation speeds to give the allowable time step, as shown in Equation 54 below.

$$\Delta t = \omega \frac{CFL \cdot V}{\lambda_x S_x + \lambda_y S_y + \lambda_z S_z} \tag{54}$$

where ω is a safety factor, usually taken to be 0.5 and S_x, S_y and S_z are summations of the directionalised areas over the surface of the control volume V, as defined in Equation 55. The directionalised speeds of propagation (λ_x etc) are defined in Equation 56.

$$\vec{S} = \frac{1}{2} \sum_{i=0}^{n} |S_i \vec{n}_i| \tag{55}$$

$$\begin{aligned} \lambda_x &= |u| + a \\ \lambda_y &= |v| + a \\ \lambda_z &= |w| + a \end{aligned} \tag{56}$$

The scheme is accelerated towards steady state via local time stepping, where each control volume uses the local maximum time-step allowable.

9.3 Unsteady flow calculations

Because the method of artificial compressibility destroys time accuracy neither an implicit nor an explicit time integration scheme will allow unsteady flow calculations. To solve incompressible unsteady flow the normal technique is to use a pressure correction method (Miyata et al [40]). To keep the efficiency of the artificial compressibility method a dual time approach has been adopted. This approach has been described by Gaitonde [35]; an implicit discretisation is used in real time, but at each real time step the solution is marched to steady state through a pseudo-time via the explicit Runge Kutta method defined by Equation 52. To differentiate between real time and pseudo-time, t shall be used for real time and τ for pseudo-time. The superscript m is used to denote the real time step and n the pseudo-time step. The result of this is that the Runge Kutta scheme of Equation 52 develops into

$$\begin{aligned} U^0 &= (U^{m+1})^n \\ U^k &= U^0 + \alpha_k \frac{\Delta \tau}{V^{k-1}} R^* \left(U^{(k-1)} \right) \\ (U^{m+1})^{n+1} &= U^4 \end{aligned} \tag{57}$$

with

$$R^*(U^k) = \frac{3V^{m+1}U^k - 4V^mU^m + V^{m-1}U^{m-1}}{2\Delta t} + R^{m+1}(U^k) \tag{58}$$

With the introduction of this extra dimension, the stability criterion for the pseudo-time step, $\Delta\tau$, becomes

$$\Delta\tau = min\left[\omega\frac{CFL \cdot V}{\lambda_x S_x + \lambda_y S_y + \lambda_z S_z}, \frac{2\Delta t}{3}\right] \tag{59}$$

It should be noted that the volume terms are dependant upon the real time position (denoted by the superscript m) due to the presence of moving boundaries such as a deforming free surface or the movement of a vessel. Due to the implicit nature of the real time stepping procedure the step size, Δt, can be choosen purely upon accuracy considerations.

To date the ability to model unsteady flow is limited to two dimensions, but due to the structured manner of the program it can easily be adapted to the full three dimensional solver and is the subject of on going work

10 Implementation of scheme

10.1 Framework of code

A numerical scheme for solving three dimensional hydrodynamic flow is presented in Chapters 7, 8 and 9. While the data structure and connectivity list defined in Chapter 3 could be implemented in a variety of computer programming languages it has been decided that the environment created by C++ is most suitable. This language allows the gradual creation of extra routines and inheritence required by the developmental nature of this work. A modular construction has been used to assist the motivation towards versitility and generality of the code. Libraries of functions have been generated that operate at different levels of the data structure, yet allow efficient passing of relevant information. This modularisation also promotes the use of object orientated coding, where specific functions are associated with particular geometrical entites.

10.2 Program algorithms

After initialisation the volume of each vertex based cell is defined in a face based loop. In the steady state flow solution phase there is an outer loop to incorporate the four stage Runge Kutta temporal integration, inside which there is a facial loop to define the flux through each interface, followed by a nodal loop to update the state vectors by the resdiual defined in the face loop. The use of a face loop when calculating the upwind flux terms is the most efficient due to the flux being passed to the control volume on either side of the face simultaniously. The iterative loops are continued until convergence is attained. The convergence can be measured by the maximum mass residual, the maximum difference in residuals between time steps or by the number of iterative cycles completed. Algorithm 1 illustrates the operations involved.

Algorithm 1

1. *Input mesh*
2. *Input free stream flow field*
3. *Initialise* **FLAG** = FALSE
4. *For i = 1 to i = number of faces*
   (a) *Evaluate facial area and normal*
   (b) *Evaluate δ V associated with face on either side*
5. *Allocate memory for state vector properties and residuals*

6. *Do*

   (a) *For $i = 1$ to $i = 4$*

      i. *For $j = 1$ to $j =$ number of faces*

         - *Trace which members of binary tree (see Figure 7) on either side of face are active.*
         - *If internal face*
           A. *Evaluate numerican flux $\vec{F}$ on faces by flux vector splitting (see Chapter 8)*
         - *If solid wall face*
           A. *Evaluate numerican flux $\vec{F}$ with zero mass flow imposed (see Chapter 7.1)*
         - *If far field face*
           A. *Evaluate numerican flux $\vec{F}$ as zero*
         - *Distribute $\vec{F}$ to active nodes $\delta U_\Omega = \delta U_\Omega \pm \vec{F}$*

      ii. *End loop around faces*

      iii. *For $j = 1$ to $j =$ number of nodes*

         - *Trace which members of binary tree (see Figure 7) are active.*
         - *If node lies on characteristic boundary*
           A. *Update using characteristics*
         - *Else*
           A. *Evaluate δt*
           B. *Update conserved state vector variables $U_\Omega + \alpha_i \frac{\Delta t}{V_\Omega} \delta U_\Omega$*
           C. *Reset residual $\delta U_\Omega = 0$*

   (b) *End Runge Kutta timestepping loop*

   (c) *Calculate the maximum error*

   (d) *For $i = 0$ to $i =$ number of nodes*

      i. *Upate Runge Kutta reference state variables (U_0)*

   (e) *Check for convergence*

      i. *If $\delta U_{MAX} \leq \delta U_{CONVERGENCE}$* **FLAG** = TRUE

7. *While* **FLAG** = FALSE

8. *Output flow data*

9. *Deallocate dynamic memory*

10. *Exit*

In addition to this main algorithm there are regular checks upon mesh quality, from which mesh adaptation can occur, the methods of which are described in previous technical reports [9, 6].

The movement of nodes and ajoining mesh due to the movement of boundaries (such as the free surface) will occur after the Runge Kutta timestep.

The addition of unsteady flow calculations via pseudotime will result in another loop, outside the main convergence do/while loop, marching through real time steps.

11 Validation test cases

To test the convergence, stability and accuracy of the numerical scheme two typical cases were utilised. The geometrical meshs were all created using the in-house mesh generator *FLEXIMESH*, detailed in Ship Science Report 101, [41].

11.1 Flow over a circular arc hump

11.1.1 Geometric definition

The first test case is a three dimensional representation of a two dimensional problem. The domain is a simple channel of unit height with a circular arc obstruction on its lower surface, centrally placed along the length. The arc's height is 10% of its chord, with the free stream channel height equalling the chord. Figure 8 shows a wire frame model of the domain.

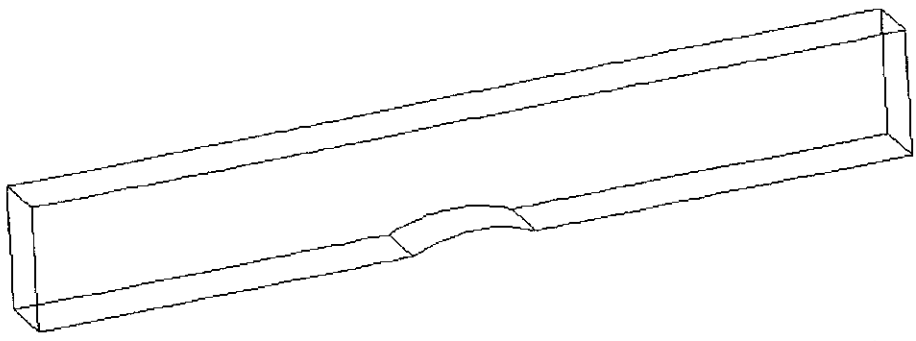

Figure 8: Wire frame of hump mesh

To test convergence cababilities two different meshes were used. Mesh H2 had 1650 node points while mesh H3 had 8690. Both meshes can be seen in Figure 9.

11.1.2 Flow domain definition

The flow was defined to have a velocity of 5 ms^{-1} in the x direction, with zero velocity components in the y and z directions. The fluid was declared to have a density of 1000 kgm^{-3}. The artificial compresibility constant, $\mathcal{K}$, was given a value of 1.5.

The upstream and downstream $y-z$ plane boundaries were declared as characteristic boundaries (see Chapter 7.2), while all other boundaries were defined as walls (see Chapter 7.1).

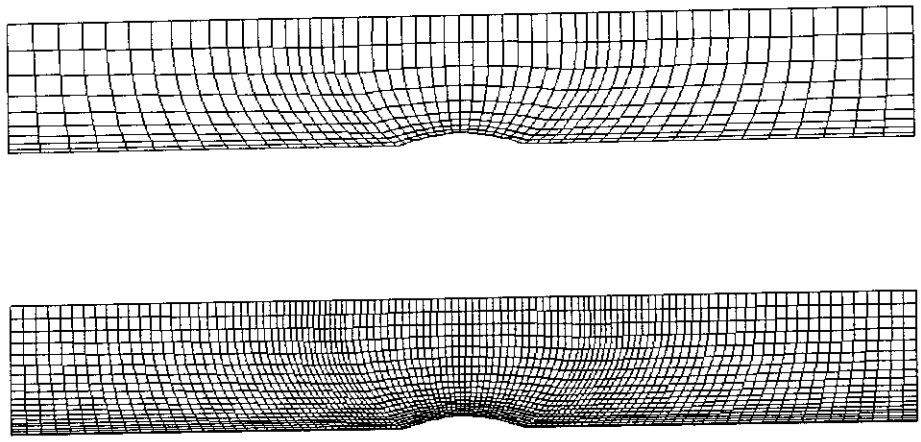

Figure 9: Hump meshes utilised to test effects of increasing number of nodes; H2 & H3 respectively

Convergence was assumed to have been reached when the average x momentum residual (non-dimensional) was below 1×10^{-8}

11.1.3 Results

Using the relatively coarser H2 mesh the model converged to a residual of 9.995×10^{-9} after 2124 iterations. The solution of the streamwise velocity field and the pressure field can be seen in Figures 10 and 11 respectively. The CPU time required for each timestep was 0.71 seconds, ± 0.05 seconds. Figure 15 details the convergence history.

When the finer H3 mesh was used, the CPU usage per timestep increased to 4.1 seconds (± 0.05 seconds) and the number of iterations required to reach the convergence criteria was 3835. Again the convergence history can be viewed in Figure 15. The solution to this flow can be seen in Figures 12 and 14 which again show the streamwise velocity and the pressure fields respectively.

The pressure variation upon the surface of the hump, for both meshes, can be seen in Figure 13.

As can be seen, the overall charactersitics of the flow field are correct, but a stagnation point does not fully form at the leading edge of the hump, and the low pressure/high velocity zone on the middle of the hump is not truely symmetric. Both of these indicate that the solution has still not fully converged. Further accelerational techniques such as multi-gridding should be used due to the rate of convergence tending towards zero, as shown in Figure 15. What is more promising is the improved solution accuracy with the finer mesh, as well as the smooth convergence histories, suggesting a stable and accurate numerical scheme.

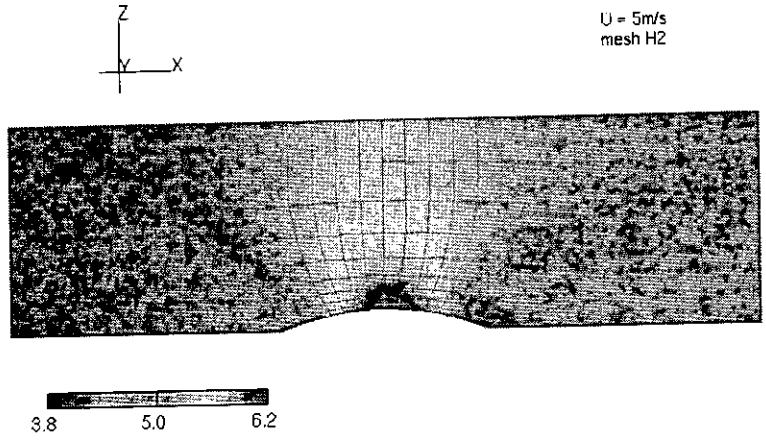

Figure 10: Streamwise velocity profile over coarse hump mesh (H2 mesh)

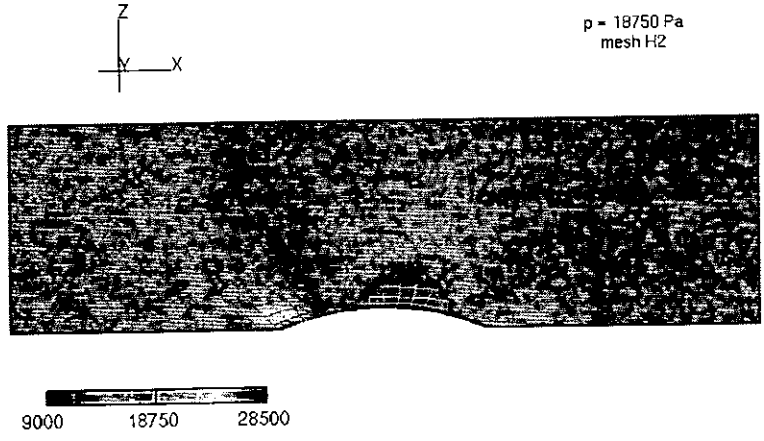

Figure 11: Pressure profile over coarse hump mesh (H2 mesh)

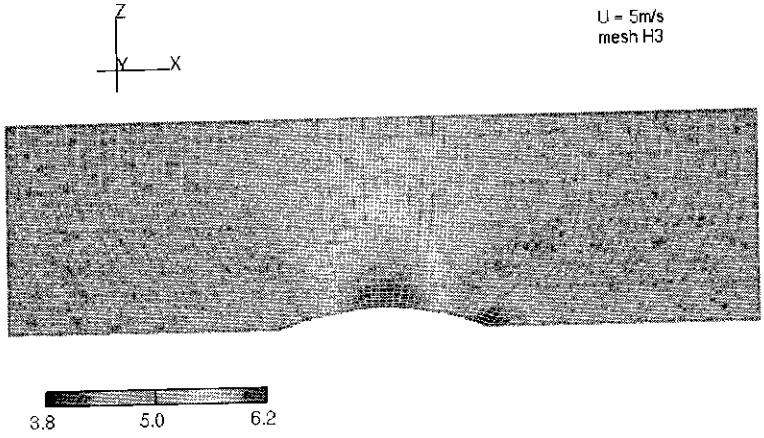

Figure 12: Streamwise velocity profile over fine hump mesh (H3 mesh)

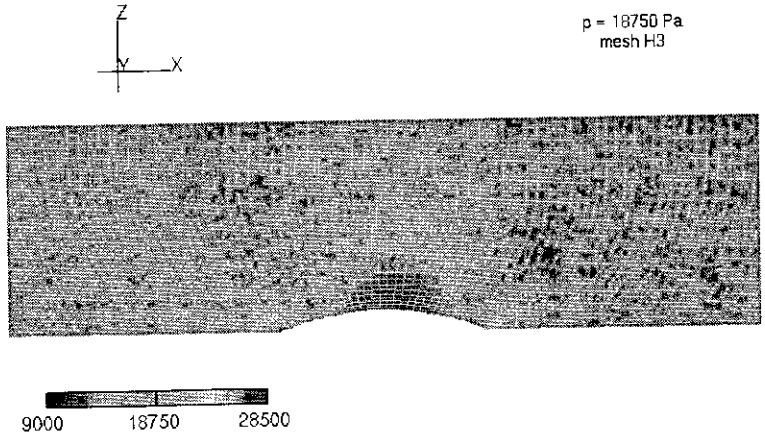

Figure 13: Pressure profile over fine hump mesh (H3 mesh)

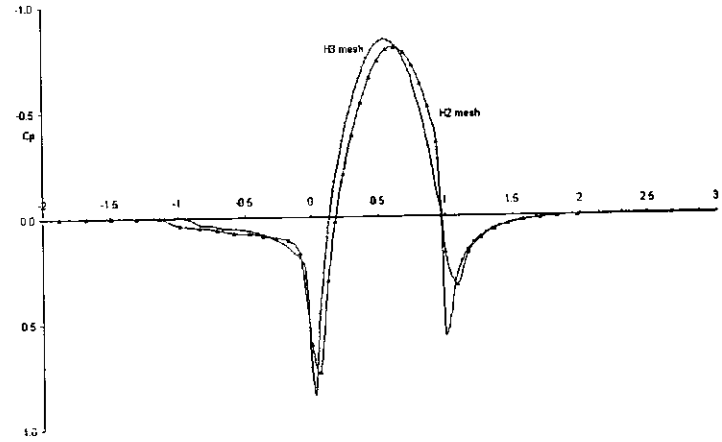

Figure 14: Pressure profile on surface of hump for both meshes

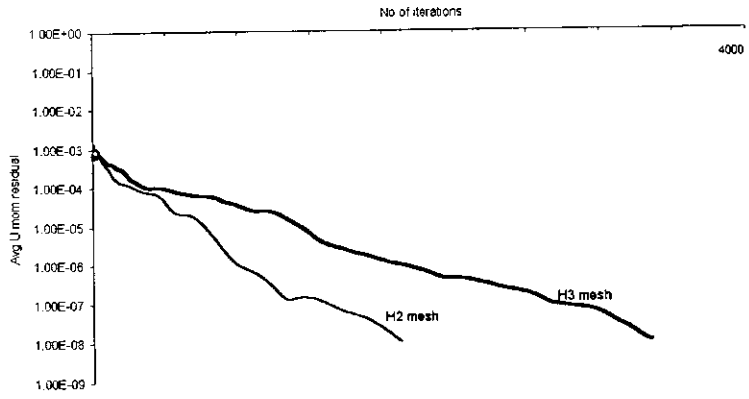

Figure 15: Convergence histories for the solution of flow over the hump grid for the two meshs, H2 and H3

11.2 Wigley hull

11.2.1 Geometric definition

The second example choosen was of the mathematical three dimensional Wigley hull form. The hull form is defined by equation 60.

$$y = \pm\frac{B}{2}\left\{1-\left(\frac{x}{L}\right)^2\right\}\left\{1-\left(\frac{z}{T}\right)^2\right\} \tag{60}$$

For this model $\frac{B}{L}$ was taken to be 0.1 and $\frac{T}{L}$ was defined as 0.0625. With a unitary half ships length (L), the domain taken for the model extends from $x = -2L$ to $x = 2L$, $y = 0$ to $y = 0.75L$ and from $z = 0$ to $z = 0.75L$. Figure 16 shows a wireframe of the domain, and Figure 17 shows the grid utilised, consisting of 11448 nodes.

11.2.2 Flow domain definition

The flow was defined to have a velocity of 5 ms^{-1} in the x direction, with zero velocity components in the y and z directions. The fluid was declared to have a density of 1000 kgm^{-3}. The artificial compresibility constant, $\mathcal{K}$, was given a value of 1.5.

The upstream and downstream $y-z$ plane boundaries were declared as characteristic boundaries (see Chapter 7.2), while all other boundaries were defined as walls (see Chapter 7.1).

Convergence was assumed to have been reached when the average x momentum residual (non-dimensional) was below 1×10^{-8}

11.2.3 Results

The model converged to a x momentum residual of 9.9892×10^{-9} after 2459 iterations. Figure 18 shows the velocity field at three horizontal cuts through the domain; at the top of the domain (z = 0), on the keel-line of the hull (z = 0.0625) and at twice the hull depth (z = 0.125). The convergence history of the solution can be seen in Figure 19, as can the convergence history for the same model when global timestepping was used.

Figure 18 shows a converged solution with full three dimensional flow; the only possible error has been the placement of the wall parallel to the hull too close to the disturbance. A blockage effect is visible, causing the flow disturbance to be carried further from the hull form than would be seen in an infinite domain.

Figure 19 shows the vast improvement in efficiency of solution when local timestepping is adopted. In addition the convergence traces again show the stable nature of the numerical scheme.

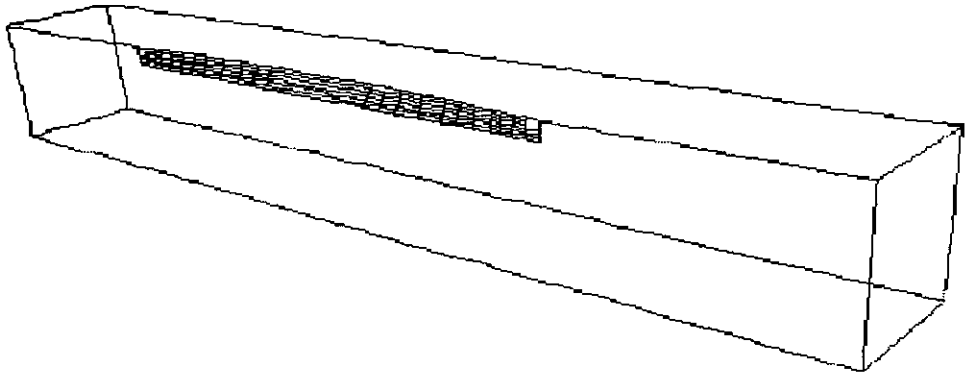

Figure 16: Wire frame of mesh around a Wigley hull form

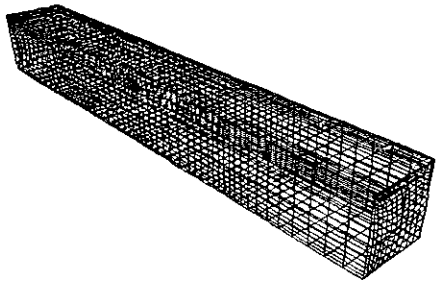

Figure 17: Mesh defined around the Wigley hullform

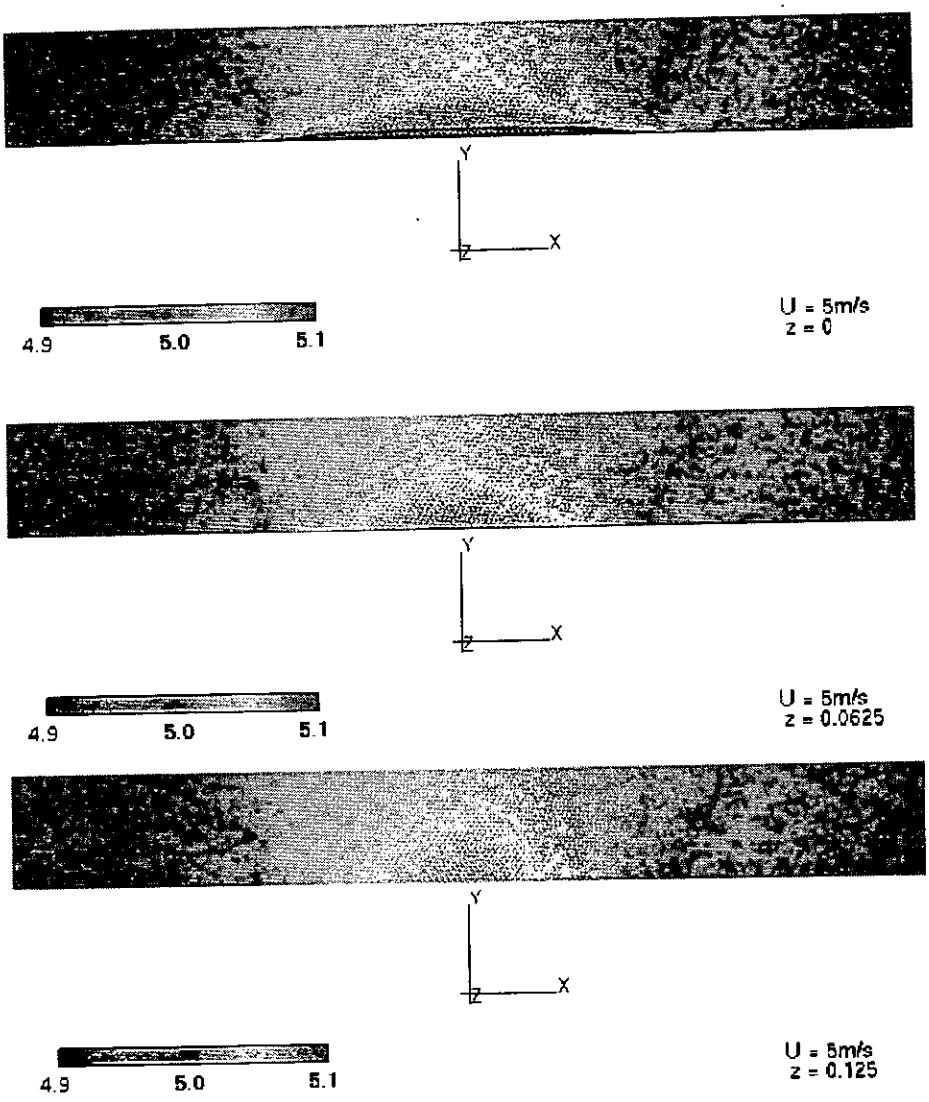

Figure 18: Streamwise velocity at three horizontal cuts through the grid (W1 mesh)

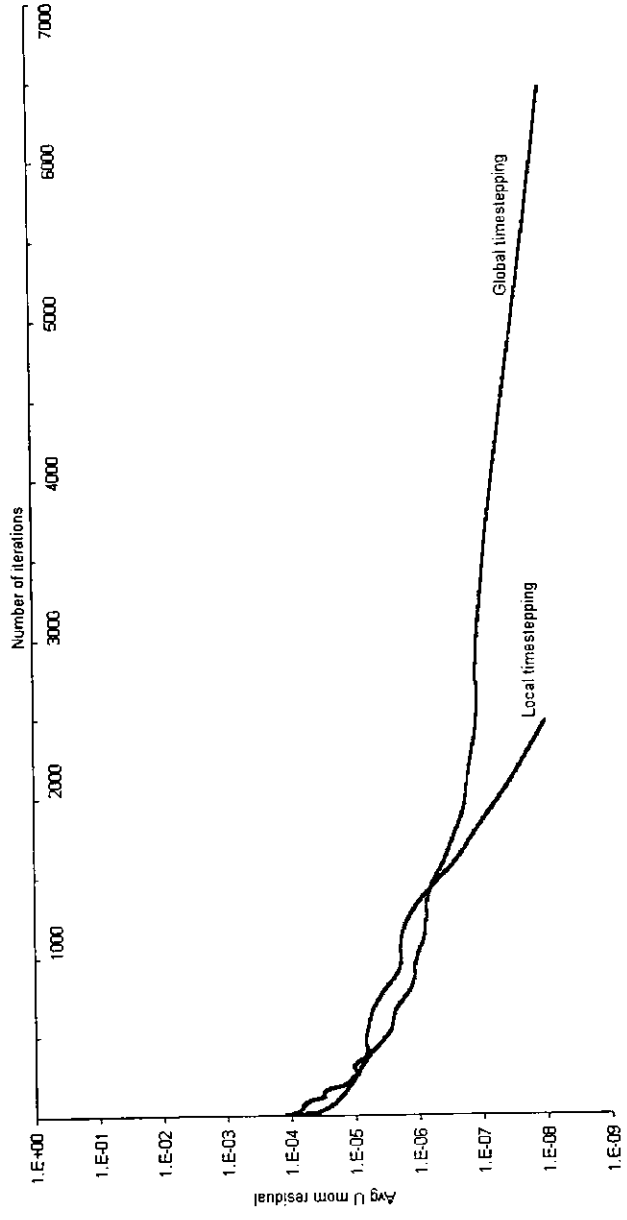

Figure 19: Convergence history for the solution of flow around the Wigley hullform, using different timestepping methods

12 Conclusions and Future developments

A flow solver has been created that models incompressible flow around three dimensional objects. A finite volume methodology is used, with arbitrarily shaped control volumes, in a flux vector splitting framework. The incompressible nature of the flow is solved via artificial compressibility which allows the use of standard compressible flow techniques such as local time stepping.

Initial code verification and validation has demonstrated the robustness and adaptability of the code, showing acceptable convergence histories and CPU usage. Further initiatives to improve efficiency, such as multi-gridding, will decrease the computational requirements further.

There are five areas that are being developed to increase the capabilities of the code.

Validation and verification : The primary goal at present is validation and verification of the results obtained from the solver, to gain a measure of the accuracy of the solver and its reliability to model flow, both qualitatively and quantatively. This work has already been initiated; case studies include a two-dimensional circlular arc hump, three dimensional foils and three dimensional hull forms.

Free surface : A free surface boundary condition has been devised and is presently being coupled to the code. It utilises bi-cubic parametric splines to model the surface shape, which allows the calculation of local gradients and curvatures. This scheme is detailed in a seperate Ship Science Report (see Turnock and Wright [42]).

Higher order spatial discretisation : The most obvious method to improve the accuracy of the solution without increasing the mesh density is to use a higher order discretisation scheme. Presently only a first order scheme is utilised. The problems invovled with such algorithms are decreased stability and hgihly complex algorithms to gain the correct 'stencil' in the unstructured mesh used. The MUSCL scheme has been successfully used in a similar framework for fully compressible flow (Rycroft, [7]), and it is suggested that this work is a suitable starting point for such a development.

Unsteady flow : Pseudotime, already added to 2D flow (see Chapter 9.3), is presently to be added to the 3D solver. An implicit algorithm, as used by Gaitonde [35], is to be implemented in the near future.

Improved convergence : The most expedient manner in which convergence can be improved is via the use of multigridding. This technique has been used successfully elsewhere in similar codes [1], and the presence of a 'binary tree' like data structure (see Chapter 3) lends itself to the technique. The use of CV splitting and merging within the code (detailed in a previous technical report

[6]) results in the only required addition being the overall framework in which to switch between the various levels of the data structure.

References

[1] J. Farmer, L. Martinelli, and A. Jameson. "Fast multigrid method for solving incompressible hydrodynamic problems with free surfaces". *AIAA Journal*, 32(6):1175–1182, June 1994.

[2] C. Yang and R. Löhner. "Fully nonlinear ship wave calculation using unstructured grid and parallel computing". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 125–150, May 1998.

[3] M. Beddhu, M. Y. Jiang, D. L. Whitfield, L. K. Taylor, and A. Arabshahi. "CFD validation of the free surface flow around DTMB Model 5415 using Reynolds Averaged Navier-Stokes Equations". In *Proceedings of Third Osaka colloquium on advanced CFD applications to ship flow and hull form design*, pages 373–393, May 1998.

[4] C. Hirsch. *"Numerical Computation Of Internal And External Flows"*, volume 1. J. Wiley and sons, 1988. ISBN 0-471-92385-0.

[5] P. D. Lax. "Weak solutions of non-linear hyperbolic equations and their numerical computation". *Comm. Pure and Applied Mathematics*, 7:154–193, 1954.

[6] A. M. Wright and S. R. Turnock. "Efficient multi-level adaptation methods for unstructured polyhedral computational meshes". Ship Science Report 109, Department of Ship Science, University of Southampton, 1999.

[7] N. C. Rycroft. *"An adaptive three dimensional, finite volume, Euler solver for distributed architecture using arbitrary polyhedral cells"*. PhD thesis, University of Southampton, 1998.

[8] E. Turkel. "Accuracy of schemes with non-uniform meshes for compressible fluid-flows". *Applied numerical Mathematics*, 2(6):529–550, 1986.

[9] A. M. Wright and S. R. Turnock. "Techniques for assessing the flow and spatial quality of arbitrary 3d computational meshes". Ship Science Report 108, Department of Ship Science, University of Southampton, 1999.

[10] N. C. Rycroft and S. R. Turnock. "Hybrid cell finite volume Euler solutions of flow around a main-jib sail using an IBM SP2". In *Proceedings of Parallel CFD 97, Manchester*, May 1997.

[11] T. Hino. "Computation of Free Surface Flow around an Advancing Ship by the Navier-Stokes equations". In *Proceedings 5th International Conference on Numerical Ship Hydrodynamics*, pages 103–117, 1989. National Academy Pres, Washington D.C.

[12] H. Miyata, T. Sato, and N. Baba. "Difference solution of a viscous flow with free-surface wave about an advancing ship". *Journal of Computational Physics*, 72:393–421, 1987.

[13] Y. Tahara, F. Stern, and B. Rosen. "An Interactive Approach for Calculating Ship Boundary Layers and Wakes for Nonzero Froude Number". *Journal of Computational Physics*, 98(1):33–53, 1992.

[14] A. J. Chorin. "A numerical method for solving incompressible viscous flow problems". *Journal of Computational Physics*, 2:12–26, 1967.

[15] A. Rizzi and L.E. Eriksson. "Computation of inviscid incompressible flow with rotation". *Journal of Fluid Mechanics*, 153:275 – 312, 1985.

[16] B. Engquist and A. Majda. "Absorbing boundary conditions for the numerical simulation of waves". *Mathematical of computation*, 31(139):629–651, July 1977.

[17] J. D. Ramshaw and V. A. Mousseau. "Accelerated artificial compressibility method for steady state incompressible flow calculations". *Computers and Fluids*, 18(4):361–367, 1990.

[18] J. M. R. Graham. "Marine CFD final workshop; Status of Programme". Concise summary of all six projects undertaken within the EPSRC funded directed programme into marine CFD, 1999.

[19] R. H. Ni. "A multiple-grid scheme for solving the Euler equations". *AIAA Journal*, 20(11):1565–1571, November 1982. presented as AIAA paper81-1025.

[20] C. Hirsch. *"Numerical Computation Of Internal And External Flows"*, volume 2. J. Wiley and sons, 1988. ISBN 0-471-92452-0.

[21] J.D. Anderson. *"Computational Fluid Dynamics - The basics with applications"*. McGraw Hill Inc., 1995. ISBN 0-07-113210-4.

[22] J. Y. Trepanier, M. Reggio, H. Zhang, and R. Camarero. "A finite volume method for the euler equations on arbitrary lagrangian-Eulerian grids". *Computers and fluids*, 20(4):309–409, 1991.

[23] P. D. Thomas and C. K. Lombard. "Geometric Conservation Law and its application to flow computations on moving grids". *AIAA Journal*, 17(10):1030–1037, October 1979.

[24] R. G. Hindman and P. Kutlerand D. Anderson. "Two dimensional unsteady Euler equation solver for arbitrarily shaped flow regions". *AIAA Journal*, 19(4):424–431, April 1981.

[25] P. R. Eiseman. "Alternating direction adaptive grid generation". *AIAA Journal*, 23(4):551–560, April 1985.

[26] J. F. Thomson, F. C. Thames, and C. M. Mastin. "Automatic Numerical Generation of Body fitted Curvilinear Coordinate Systemfor Field containing any number of arbitrary Two-Dimensional Bodies ". *Journal of Computational Physics*, 15:299–319, 1974.

[27] H. A. Dwyer, M. D. Smooke, and R. J. Kee. "Adaptive gridding for Finite difference Solutions to Heat and Mass tranfer problems". In J. F. Thomson, editor, *Numerical grid Generation*, pages 339–356. North-Holland Publishing Co., 1982.

[28] P. L. Roe. "approximate Reimann solvers, parameter vectors, and differencing schemes". *Journal of Comp. Physics*, 43:357–372, 1981.

[29] J. Y. Trepanier, M. Reggio, M. Paraschivoiu, and R. Camarero. "Unsteady Euler solutions for arbitary moving bodies and boundaries". *AIAA Journal*, 31(10):1869–1876, October 1993.

[30] A. Ilinca, R. Camarero, J. Y. Trepanier, and M. Reggio. "Error estimator and adaptive moving grids for finite volumes schemes". *AIAA Journal*, 33(11):2058–2065, November 1995.

[31] J. W. Slater, M. S. Liou, and R. G. Hindman. "Approach for dynamic grids". *AIAA Journal*, 33(1):63–68, January 1995.

[32] J. U. Brackbill and J. S. Saltzman. "Adaptive zoning for Singular Problems in Two Dimensions". *Journal of Computational Physics*, 46(2):342–368, 1982.

[33] A. L. Gaitonde and S. P. Fiddes. "A three-dimensional moving mesh method for the calculation of unsteady transonic flows". *Aeronautical Journal*, pages 150–160, April 1995. paper 2011.

[34] A. L. Gaitonde and S. P. Fiddes. "A comparision of a cell-centre method and a cell-vertex method for the solution of the two-dimensional unsteady Euler equations on a moving grid". In *Proceddings Instn Mech Engineers*, volume 209, pages 203–213, 1995.

[35] A. L. Gaitonde. "A dual time method for the solution of the unsteady Euler equations". *Aeronautical Journal*, pages 283 – 291, October 1994.

[36] K. Riemslagh and E. Dick. "A multigrid method with unstructured adaptive grids for steady Euler equations". *Journal of Computational and Applied Mathematics*, 67:73–93, 1996.

[37] E. Dick. "Second order formulation of a multigrid method for steady Euler equations trough defect-correction". *Journal of Computational and Applied Mathematics*, 35:159–168, 1991.

[38] M. B. Giles. "Nonreflecting boundary conditions for Euler equation calculations". *AIAA Journal*, 28(12):2050–2058, December 1990.

[39] A. Jameson, W. Schmidt, and E. Turkel. "Numerical solution of the euler equations by finite volume methods using Runge Kutta time stepping schemes". AIAA Paper 81-1259, 1981.

[40] H. Miyata, A. Kanai, T. Kawamura, and J-C. Park. "Numerical simulation of three-dimensional breaking waves". *Journal Marine science and technology*, 1:183–197, 1996.

[41] N. C. Rycroft and S. R. Turnock. "3-D Multiblock Grid Generator; FLEXIMESH". Ship Science Report 101, Department of Ship Science, University of Southampton, 1997.

[42] S. R. Turnock and A. M. Wright. "A free surface definition using adaptive bicubic patches". Ship Science Report, Department of Ship Science, University of Southampton, 1999.

A Various forms of the Euler equations

The Euler equations describe the conservation of mass, momentum and energy, with the omission of viscous and thermal conduction effects resulting in the description of what is known as inviscid flow. Three different forms of the equations are commonly used to define the system of equations, revealing their mathematical properties. These properties in turn impose the type of numerical approach that can be used to solve the equations. This section concentrates upon the definition and transformation between these forms of the Euler equations; the conservative, primitive and characteristic forms.

A.1 Conservative form of Euler equations

The Euler equations can most easily be defined as a set of conservative laws and as such can be derived using an arbitrary volume, Ω, fixed in space. Variation of a scalar quantity per unit volume U, bounded by a closed surface S can be expressed as a summation of internal and surface sources, and the transport across the bounding surface. The last of these mechanisms is termed a 'flux'.

Due to the neglection of viscous terms all diffusive terms are also neglected, leaving the system of equations dominated by the convective transport of mass and momentum.

The mathematical expression of this is

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega = -\oint_S \vec{F}\cdot d\vec{S} + \int_{\Omega} Q_v d\Omega + \oint_S \vec{Q}_S \cdot d\vec{S} \tag{61}$$

where $\vec{S}$ is outward facing, and Q_V and Q_S are the volume and surface sources respectively, as shown in Figure 20.

The conservation of mass momentum and energy can be expressed using the general integral conservation law given in Equation 61, generating a set of five coupled partial differential equations for three dimensional flow. Equation 62 describes the conservation of mass, where ρ is the density of the fluid and $\vec{v}$ is the velocity vector.

$$\frac{\partial}{\partial t}\int_{\Omega} \rho d\Omega = -\oint_S \rho\vec{v}\cdot d\vec{S} \tag{62}$$

Equation 63 defines the conservation of momentum, where $\vec{f_e}$ are external forces acting on the control volume, and p is the static pressure.The conservation of total energy E is defined in Equation 64 as a summation of a convective energy term and sources which model the work done by the external forces $\vec{f_e}$ and the forces working on the surface of the volume.

$$\frac{\partial}{\partial t}\int_{\Omega} \rho\vec{v} d\Omega + \oint_S \rho\vec{v}\left(\vec{v}\cdot d\vec{S}\right) = -\int_{\Omega} \rho\vec{f_e} d\Omega - \oint_S p d\vec{S} \tag{63}$$

$$\frac{\partial}{\partial t}\int_{\Omega} \rho E d\Omega + \oint_S \rho E\vec{v}\cdot d\vec{S} = -\int_{\Omega} \left(\rho\vec{f_e}\cdot\vec{v}\right) d\Omega - \oint_S p\vec{v}\cdot d\vec{S} \tag{64}$$

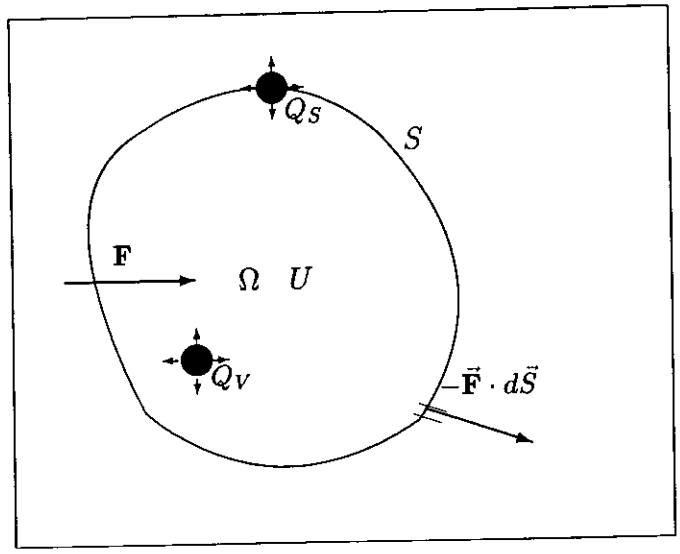

Figure 20: Arbitrary volume, containing sources and surface fluxes

Equation 64 can be simplified by introducing the stagnation enthalpy, as defined in Equation 7. Thus the energy conservation law is now expressed in the mode commonly recognised form as given in Equation 65.

$$\frac{\partial}{\partial t}\int_{\Omega} \rho E d\Omega + \oint_S \rho H \vec{v} \cdot d\vec{S} = \int_{\Omega} \left(\rho \vec{f_e} \cdot \vec{v}\right) d\Omega \tag{65}$$

These equations can be written in vector form as

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{U} d\Omega + \oint_S \mathbf{F} \cdot d\vec{S} = \int_{\Omega} \mathbf{Q} d\Omega \tag{66}$$

where $\mathbf{U}$ is the conservative state vector $\left\{ \begin{array}{c} \rho \\ \rho\vec{v} \\ \rho E \end{array} \right\}$ as defined in Equation 2, as are the flux vector $\vec{F}$ and the source vector $\mathbf{Q}$.

The result of the control volume analysis is the integral conservative form of the Euler equations. A differential form of the equations may also be found by studying an infinitesimal fluid particle of size dx by dy by dz. Expressions for the conserved variables on each of the faces as well as the change in the variables over the fluid particle may be found. Alternatively the integral form can be manipulated in order to obtain the differential form.

Because the volume Ω is fixed in space the limits of integration in Equation 66 are constant. This allows the time derivative to be placed inside the integral and for Gauss' theorum to be applied to the surface integral. As a result the integral form of the Euler equations takes the form defined in Equation 67.

$$\int_\Omega \frac{\partial \mathbf{U}}{\partial t} d\Omega + \int_\Omega \vec{\nabla} \cdot \mathbf{F} d\Omega = \int_\Omega \mathbf{Q} d\Omega$$
$$\rightarrow \int_\Omega \left[\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \mathbf{F} - \mathbf{Q} \right] d\Omega = 0 \tag{67}$$

It is clear that for the integral to equal zero it must be zero everywhere in the domain due to the arbitrary nature of the volume. Thus Equation 67 leads to the differential form of the conservative Euler equations -

$$\frac{\partial \mathbf{U}}{\partial t} + \vec{\nabla} \cdot \mathbf{F} = \mathbf{Q} \tag{68}$$

It should be noted that Gauss' theorem implies that the fluid properties are continuous. Real flows however may contain discontinuities such as shock waves and thus the integral form of the conservative Euler equations (Equation 66) which do not make this assumption is deemed the more fundementally correct.

A.2 Non-Conservative form of Euler equations

The equations derived in Section A.1 are known as the Conservative equations, after the variables in which they are expressed (**U**). If the same derivation method is followed for a volume travelling with the fluid, such that the mass of fluid within the volume remains constant but the surface area and volume vary, the Non-Conservative form of the Euler equations may be found. These equations are defined in terms of the primative variables (**V**).

This form of equations can be reached through direct derivation or by manipulation of the Conservative form. The mass conservation law (Equation 62) can be written in the differential form as

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \rho \vec{v} = 0 \tag{69}$$

Applying the product rule to the dot product term and with the use of the substantial (or total) derivative $\frac{D}{Dt}$ the mass conservation law may be rewritten as

$$\frac{D\rho}{Dt} + \rho \vec{\nabla} \cdot \vec{v} = 0 \tag{70}$$

The substantial derivative describes the rate of change as the fluid element moves throught space; it sums the local time dependant changes (denoted by $\frac{\partial}{\partial t}$) and the the changes that occur because the fluid particle moves around in space. The integral form of the mass conservation law in non-conservative form is defined as

$$\frac{D}{Dt} \int_\Omega \rho d\Omega = 0 \tag{71}$$

The momentum and energy conservation equations may also be written in non-conservative form as

$$\rho \frac{D\vec{v}}{Dt} = -\vec{\nabla} p + \rho \vec{f_e} \tag{72}$$

$$\rho \frac{D}{Dt}\left(e + \frac{\vec{v}^2}{2}\right) = -\vec{\nabla} \cdot (\vec{v}p) + \rho \vec{f_e} \cdot \vec{v} \tag{73}$$

As a result the primitive variables, $\mathbf{V}$ are defined as -

$$\mathbf{V} = \begin{Bmatrix} \rho \\ \vec{v} \\ p \end{Bmatrix} \tag{74}$$

The presence of x, y, and z derivatives on the right hand side of the non-conservative forms results in weak conservation; only single derivatives exist in the conservative forms, thus resulting in strong conservation. Across a discontinuity there exists only small or no changes to the flux variables $\vec{\mathbf{F}}$ whereas the primative variables such as p are discontinuous. This continuous nature of the fluxes in the conservative form leads to greater stability and hence more common usage in computations of the Euler equations.

A.3 Quasi-linear form of Euler equations

In order to investigate the mathematical properties of the system of Euler equations it is neccesary to write these equations in a quasi-linear format. This is because a system of quasi-linear partial differential equations of the first order will be hyperbolic if its homogenous part admits wave-like solutions. The Euler system of equations contains only first-order derivatives and if the external forces $\vec{f_e}$ are indepandant of the flow gradients the system of Euler equations is of first order in the variables $\mathbf{U}$.

The quasi-linear form of Equation 68 is written as

$$\begin{aligned} \frac{\partial \mathbf{U}}{\partial t} + \left(\frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{U}}\right) \cdot \vec{\nabla}\mathbf{U} &= \mathbf{Q} \\ \frac{\partial \mathbf{U}}{\partial t} + \vec{\mathbf{A}} \cdot \vec{\nabla}\mathbf{U} &= \mathbf{Q} \end{aligned} \tag{75}$$

or explicitly as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{U}}{\partial y} + \mathbf{C}\frac{\partial \mathbf{U}}{\partial z} = \mathbf{Q} \tag{76}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are the three Jacobian matrices of the flux vector $\vec{\mathbf{F}}$. The flux components of the standard Euler equations have the property of being homogenous functions of degree 1 of the conservative varibale vector $\mathbf{U}$ for fluids satisfying the relation

$$p = \rho \mathcal{F}(e) \tag{77}$$

This implies that

$$\vec{\mathbf{F}}(\lambda \mathbf{U}) = \lambda \vec{\mathbf{F}}(\mathbf{U}) \quad \text{for any } \lambda \tag{78}$$

and by differencing with respect to λ and setting $\lambda = 1$ one obtains the relation

$$\vec{\mathbf{F}}(\mathbf{U}) = \frac{\partial \vec{\mathbf{F}}}{\partial \mathbf{U}}\mathbf{U} = \vec{\mathbf{A}}\mathbf{U} \tag{79}$$

If isentropic flow is assumed the Jacobian matrices $\vec{\mathbf{A}}$ may be written in terms of the primative, or non-conservative, variables, $\mathbf{V}$. The Euler equations are thus expressed as

$$\frac{\partial \mathbf{V}}{\partial t} + \left(\vec{\tilde{\mathbf{A}}} \cdot \vec{\nabla}\right)\mathbf{V} = 0 \tag{80}$$

In order to gain further insight into the mathematical properties of the equations it is neccesary to find the eigenvalues of the Jacobians. Using Cramer's rule, the eigenvalues are found from

$$|\vec{\mathbf{A}} - \lambda \mathbf{I}| = 0 \tag{81}$$

where $\mathbf{I}$ is the identity matrix and λ is defined as an eigenvalue of the matrix $\vec{\mathbf{A}}$.

It should be noted that for the standard Euler equations it is easier to obtain the eigenvalues of the system when they are written in non-conservative form, as a function of $\mathbf{V}$ due to the much simpler structure of the Jacobians.

A.4 Characteristic form of Euler equations

If all eigenvalues are distinct and real then it can be concluded that the system of equations are hyperbolic, and exhibit wave-like solutions. An extension of this concept is to think of the equations as a series of waves propagating properties throughout the time-space domain. Further to this, the waves can be described as lines or surfaces along which certain properties remain constant. The domain on one side of the line/surface represents the region already affected by the wave, while the region on the other side of the line/surface represents the domain which is as yet non affected. These are the zones of 'influence' and 'dependence' respectively. The lines/surfaces are known as 'characteristic lines', the slopes of which are given by the eigenvalues of the Jacobians.

By directionalising the three dimensional equations using an arbitrary direction $\vec{k}$, the properties in three dimensions may be examined. The Jacobian matrix corresponding to this direction can be written as

$$\mathbf{K} = \vec{\mathbf{A}} \cdot \vec{k} \tag{82}$$

If $\vec{k}$ is the unit vector in any direction, the eigenvalues give the speed of propagation in the physical plane and the slopes of the characteristic surfaces $S(\vec{x}, t) = constant$ in the $(\vec{x}, t)$ plane, shown in Figure 21. Since $\vec{k}$ is arbitrary an infinite number of characteristic surfaces can be found, the envelope of which produce the 'Mach' conoid defining the zones of dependance and influence at a point in the $(\vec{x}, t)$ plane.

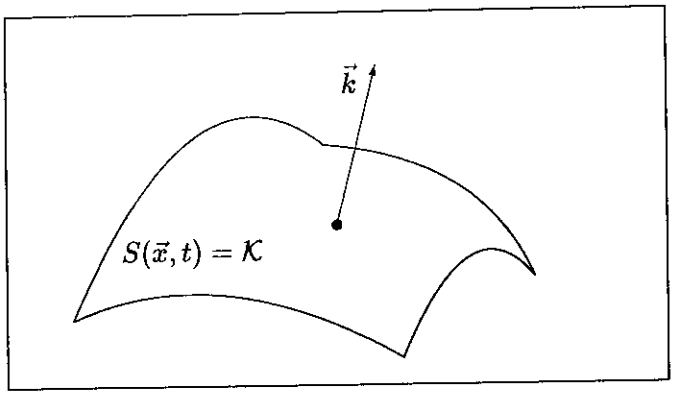

Figure 21: Characteristic surface

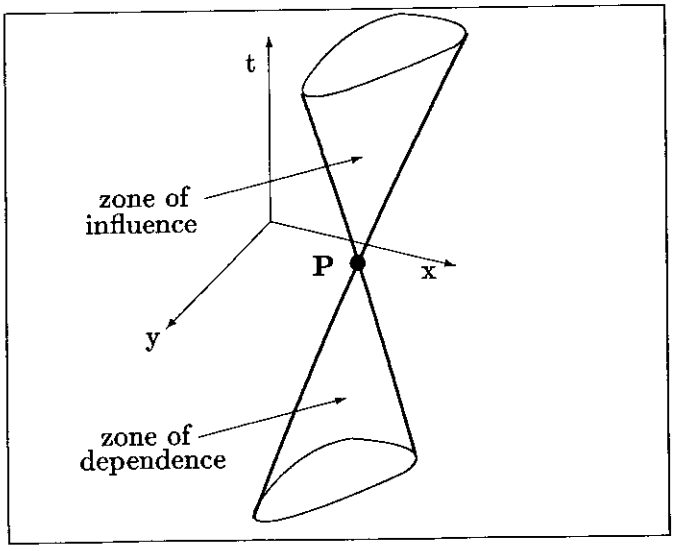

Figure 22: Zones of dependence and influence

The existence of these characteristic surfaces allows the system of equations to be reformulated such that they only contain derivatives along the surfaces, since the behaviour of the system is dictated by the properties which are propagated with the surfaces. The advantage to this formulation is that the number of spatial dimensions in which the equations are written reduces by one. The re-formulation results in a form containing only derivatives in directions lying on the surface, and can be constructed from a linear combination of the original equations. The transformation equation for λ_i, known as the compatability relationship, can be written as

$$\tilde{l}^i \frac{\partial \mathbf{V}}{\partial t} + \tilde{l}^i \left(\vec{\tilde{\mathbf{A}}} \cdot \vec{\nabla} \right) \mathbf{V} = \tilde{l}^i \tilde{\mathbf{Q}} \tag{83}$$

where l^i, the arbitrary coefficients of the linear combination, are found to be the left eigenvectors of the matrix $\tilde{\mathbf{K}}$.

A.5 Transformation between forms

The Jacobian matrix of the transformation from the conservative to the non-conservative variables is defined as

$$\mathbf{M} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \tag{84}$$

and its evaluation requires the explicit formulation of the fluid constitutive relations. An important point is that the definition of the non-conservative Jacobians does not require an explicit definition of the fluid constituative relations, and therefore has a larger validity range; that is they are not necessarily connected to a perfect gas assumption as is the case for the conservative Jacobians.

The relations between the conservative and the non-conservative Jacobians $\vec{\mathbf{A}}$ and $\vec{\tilde{\mathbf{A}}}$ can be expressed through a similar transformation with matrix $\mathbf{M}$. Applying Equation 84 to Equation 75 produces

$$\mathbf{M} \frac{\partial \mathbf{V}}{\partial t} + \vec{\mathbf{A}} \mathbf{M} \cdot \vec{\nabla} \mathbf{V} = \mathbf{Q} \tag{85}$$

Identifying with the non-conservative form, Equation 80 gives the relation

$$\vec{\tilde{\mathbf{A}}} = \mathbf{M}^{-1} \vec{\mathbf{A}} \mathbf{M} \quad \text{or} \quad \vec{\mathbf{A}} = \mathbf{M} \vec{\tilde{\mathbf{A}}} \mathbf{M}^{-1} \tag{86}$$

It can be seen from Equation 83 that a matrix $\mathbf{L}^{-1}$ can be constructed from the left eigenvectors $\tilde{l}^i$. That is, the i^{th} line of $\mathbf{L}^{-1}$ is the left eigenvector $\tilde{l}^i$. Thus, grouping all the eigenvalues together leads to

$$\mathbf{L}^{-1} \tilde{\mathbf{K}} = \Lambda \mathbf{L}^{-1} \tag{87}$$

Since the matrix $\tilde{\mathbf{K}}$ is not symmetric there exists a set of right eigenvectors $\tilde{r}^j$ associated with the eigenvalues λ^j. These column vectors are defined by

$$\tilde{\mathbf{K}} \tilde{r}^j = \lambda_j \tilde{r}^j \tag{88}$$

The matrix of right eigenvectors is the inverse of the matrix $\mathbf{L}^{-1}$ of the left eigenvectors.

With the introduction of the matrices $\mathbf{L}$ and $\mathbf{L}^{-1}$ one can write the compatability equations in a compact form

$$\left(\mathbf{L}^{-1}\partial_t + \mathbf{L}^{-1}\tilde{\vec{\mathbf{A}}} \cdot \vec{\nabla}\right)\mathbf{V} = \mathbf{L}^{-1}\tilde{\mathbf{Q}} \tag{89}$$

The matrix $\mathbf{P}$ defined by

$$\mathbf{P}^{-1} = \mathbf{L}^{-1}\mathbf{M}^{-1} \quad \text{or} \quad \mathbf{P} = \mathbf{ML} \tag{90}$$

plays the same role with respect to the conservative variables as the matrix $\mathbf{L}$ with the primative variables.

The compatibility relations (Equation 83 or Equation 89) lead to the introduction of the characteristic variables $\mathbf{W}$. They are defined as a column vector by the relation vaild for arbitrary variations δ

$$\begin{aligned} \delta\mathbf{W} &= \mathbf{L}^{-1}\delta\mathbf{V} \\ \delta\mathbf{V} &= \mathbf{L}\delta\mathbf{W} \end{aligned} \tag{91}$$

and

$$\begin{aligned} \delta\mathbf{W} &= \mathbf{P}^{-1}\delta\mathbf{U} \\ \delta\mathbf{U} &= \mathbf{P}\delta\mathbf{W} \end{aligned} \tag{92}$$

The relationship between the three sets of variables can be summarised as shown in Figure 23

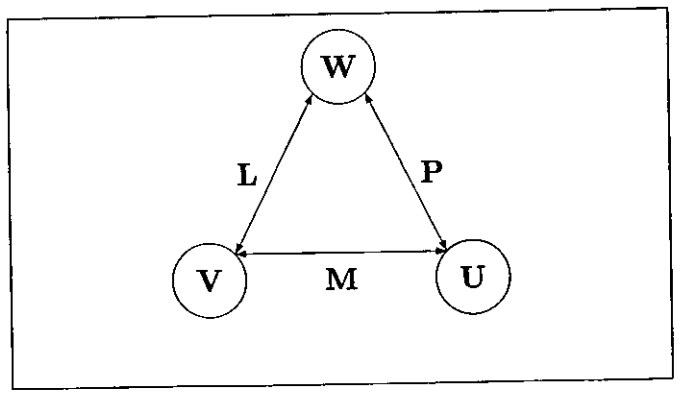

Figure 23: Relation between the conservative, primative and characteristic variables

B Derivation of characteristics of Euler equations

To gain directionality in the conservative form of the Euler equations the eigenvalues must be incorporated in the Jacobian via the transformation matrices between the characteristic and conservative forms. If $\mathbf{P} = \frac{\partial \mathbf{U}}{\partial \mathbf{W}}$, and $\mathbf{K}$ is taken to be the generalised Jacobian of the flux components for the conservative form of the Euler equations (i.e. $\mathbf{K} = n_x A + n_y B + n_z C$ where $A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}}$ etc.), then -

$$\begin{aligned} \mathbf{W_t} + \mathbf{\Lambda U_x} &= 0 \\ \Rightarrow (\mathbf{P^{-1}U_t}) + \mathbf{\Lambda}(\mathbf{P^{-1}U_x}) &= 0 \\ \Rightarrow \mathbf{PP^{-1}U_t} + \mathbf{P\Lambda P^{-1}U_x} &= 0 \\ \Rightarrow \mathbf{U_t} + (\mathbf{P\Lambda P^{-1}})\mathbf{U_x} &= 0 \end{aligned}$$

$$\Rightarrow \mathbf{K} = \mathbf{P\Lambda P^{-1}} \tag{93}$$

It should be noted that the system of equations maintains hyperbolicity if the eigenvalues are real and the norms of the transformation matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ are uniformly bounded for arbitrary real n_x, n_y and n_z.

The matrices $\mathbf{P}$ and $\mathbf{P^{-1}}$ can be gained from the asymmetry of the $\mathbf{K}$ matrix; there exists a set of right eigenvectors and a set of left eigenvectors (r^j and l^j respectively) of $\mathbf{K}$ such that the rows of $\mathbf{P^{-1}}$ are equal to the left eigenvectors and the columns of $\mathbf{P}$ are equal to the right eigenvectors.

$$\begin{aligned} i.e.\ (l_i)^j \mathbf{K}_{ik} &= \lambda_j (l_k)^j \ ,\ summation\ in\ i \\ \text{and}\quad \mathbf{K}_{ik}(r_k)^j &= \lambda_j (r_i)^j \ ,\ summation\ in\ k \end{aligned} \tag{94}$$

From this an upwind formulation can be obtained with the Jacobians

$$\begin{aligned} \mathbf{K^+} &= \mathbf{P\Lambda^+ P^{-1}} \\ \mathbf{K^-} &= \mathbf{P\Lambda^- P^{-1}} \\ with\ \mathbf{K} &= \mathbf{K^+} + \mathbf{K^-} \end{aligned} \tag{95}$$

Hence one has

$$\mathbf{F} = \mathbf{K^+U} + \mathbf{K^-U} = \mathbf{F^+} + \mathbf{F^-} \tag{96}$$

B.1 Artifical compressibility formulation

The state vector **U** and the flux vector, $\vec{F}$ = **(f, g, h)** are taken as -

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \mathbf{c}^2 u \\ u^2 + p \\ uv \\ uw \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \mathbf{c}^2 v \\ uv \\ v^2 + p \\ vw \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} \mathbf{c}^2 w \\ uw \\ vw \\ w^2 + p \end{bmatrix}$$

Definition of generalised Jacobian matrix, **K**;

$$\mathbf{K} = n_x A + n_y B + n_z C \tag{97}$$

where

$$A = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} \quad B = \frac{\partial \mathbf{G}}{\partial \mathbf{U}} \quad C = \frac{\partial \mathbf{H}}{\partial \mathbf{U}} \tag{98}$$

Thus

$$\mathbf{A} = \begin{bmatrix} \frac{\partial \mathbf{F_1}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{F_1}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{F_1}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{F_1}}{\partial \mathbf{U_4}} \\ \frac{\partial \mathbf{F_2}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{F_2}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{F_2}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{F_2}}{\partial \mathbf{U_4}} \\ \frac{\partial \mathbf{F_3}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{F_3}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{F_3}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{F_3}}{\partial \mathbf{U_4}} \\ \frac{\partial \mathbf{F_4}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{F_4}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{F_4}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{F_4}}{\partial \mathbf{U_4}} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{c}^2 & 0 & 0 \\ 1 & 2u & 0 & 0 \\ 0 & v & u & 0 \\ 0 & w & 0 & u \end{bmatrix} \tag{99}$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial \mathbf{G_1}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{G_1}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{G_1}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{G_1}}{\partial \mathbf{U_4}} \\ \frac{\partial \mathbf{G_2}}{\partial \mathbf{U_1}} & \ddots & & \vdots \\ \frac{\partial \mathbf{G_3}}{\partial \mathbf{U_1}} & & \ddots & \vdots \\ \frac{\partial \mathbf{G_4}}{\partial \mathbf{U_1}} & \cdots & \cdots & \frac{\partial \mathbf{G_4}}{\partial \mathbf{U_4}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \mathbf{c}^2 & 0 \\ 0 & v & u & 0 \\ 1 & 0 & 2v & 0 \\ 0 & 0 & w & v \end{bmatrix} \tag{100}$$

$$\mathbf{C} = \begin{bmatrix} \frac{\partial \mathbf{H_1}}{\partial \mathbf{U_1}} & \frac{\partial \mathbf{H_1}}{\partial \mathbf{U_2}} & \frac{\partial \mathbf{H_1}}{\partial \mathbf{U_3}} & \frac{\partial \mathbf{H_1}}{\partial \mathbf{U_4}} \\ \frac{\partial \mathbf{H_2}}{\partial \mathbf{U_1}} & \ddots & & \vdots \\ \frac{\partial \mathbf{H_3}}{\partial \mathbf{U_1}} & & \ddots & \vdots \\ \frac{\partial \mathbf{H_4}}{\partial \mathbf{U_1}} & \cdots & \cdots & \frac{\partial \mathbf{H_4}}{\partial \mathbf{U_4}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \mathbf{c}^2 \\ 0 & w & 0 & u \\ 0 & 0 & w & v \\ 1 & 0 & 0 & 2w \end{bmatrix} \tag{101}$$

Therefore -

$$\mathbf{K} = \begin{bmatrix} 0 & \mathbf{c}^2 n_x & \mathbf{c}^2 n_y & \mathbf{c}^2 n_z \\ n_x & \vec{v}\vec{n} + u n_x & u n_y & u n_z \\ n_y & v n_x & \vec{v}\vec{n} + v n_y & v n_z \\ n_z & w n_x & w n_y & \vec{v}\vec{n} + w n_z \end{bmatrix} \tag{102}$$

Eigen values of this generalised matrix can be found from the solution of $det|\mathbf{K} - \lambda \mathbf{I}| = 0$ or by solving for the eigenvalues of **A**, **B** and **C** seperately, and then combining the results. The second of these methods has fewer terms and allows the eigenvalues of one dimensional and two dimensional flows to be selected; hence it is this method that is followed.

$$det|\mathbf{K}-\lambda\mathbf{I}|=0$$

$$\Rightarrow\begin{vmatrix} -\lambda & \mathbf{c}^2 n_x & \mathbf{c}^2 n_y & \mathbf{c}^2 n_z \\ n_x & \vec{v}\vec{n}+un_x-\lambda & un_y & un_z \\ n_y & vn_x & \vec{v}\vec{n}+vn_y-\lambda & vn_z \\ n_z & wn_x & wn_y & \vec{v}\vec{n}+wn_z-\lambda \end{vmatrix}=0 \tag{103}$$

Seperating into component matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and solving for their eigenvalues gives

$$det|\mathbf{A}-\lambda\mathbf{I}|=0$$

$$\Rightarrow\begin{vmatrix} -\lambda_x & \mathbf{c}^2 & 0 & 0 \\ 1 & 2u-\lambda_x & 0 & 0 \\ 0 & v & u-\lambda_x & 0 \\ 0 & w & 0 & u-\lambda_x \end{vmatrix}=0$$

$$\Rightarrow\frac{1}{\lambda_x^2}\begin{vmatrix} \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 1 & 2u-\lambda_x \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 1 & 0 \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 1 & 0 \end{vmatrix} \\ \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 1 & v \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & u-\lambda_x \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & 0 \end{vmatrix} \\ \begin{vmatrix} -\lambda_x & \mathbf{c}^2 \\ 0 & w \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & 0 \end{vmatrix} & \begin{vmatrix} -\lambda_x & 0 \\ 0 & u-\lambda_x \end{vmatrix} \end{vmatrix}=0$$

$$\Rightarrow\begin{vmatrix} \frac{\lambda_x^2-2u\lambda_x+\mathbf{c}^2}{\lambda_x} & 0 & 0 \\ -\frac{v}{\lambda_x} & -\frac{(u-\lambda_x)}{\lambda_x} & 0 \\ -\frac{w}{\lambda_x} & 0 & -\frac{(u-\lambda_x)}{\lambda_x} \end{vmatrix}=0$$

$$\Rightarrow\frac{\lambda_x^2-2u\lambda_x+\mathbf{c}^2}{\lambda_x^2}\left(\frac{(\lambda_x-u)(\lambda_x-u)}{\lambda_x^2}-0\right)=0$$

$$\Rightarrow\left(\lambda_x^2-2u\lambda_x+\mathbf{c}^2\right)(\lambda_x-u)(\lambda_x-u)=0$$

$$\lambda_{x1}=u \tag{104-a}$$

$$\lambda_{x2}=u \tag{104-b}$$

$$\lambda_{x3}=u+\sqrt{u^2+\mathbf{c}^2} \tag{104-c}$$

$$\lambda_{x4}=u-\sqrt{u^2+\mathbf{c}^2} \tag{104-d}$$

Similarly for the solutions of $det|\mathbf{B}-\lambda\mathbf{I}|=0$ and $det|\mathbf{C}-\lambda\mathbf{I}|=0$, the eigenvalues for the y and z directions are

$$\lambda_{y1}=v \tag{105-a}$$

$$\lambda_{y2}=v \tag{105-b}$$

$$\lambda_{y3}=v+\sqrt{v^2+\mathbf{c}^2} \tag{105-c}$$

$$\lambda_{y4}=v-\sqrt{v^2+\mathbf{c}^2} \tag{105-d}$$

$$\lambda_{z1} = w \tag{106-a}$$

$$\lambda_{z2} = w \tag{106-b}$$

$$\lambda_{z3} = w + \sqrt{w^2 + \mathbf{c}^2} \tag{106-c}$$

$$\lambda_{z4} = w - \sqrt{w^2 + \mathbf{c}^2} \tag{106-d}$$

These results combine to give -

$$\begin{aligned} \lambda_1 &= un_x + vn_y + wn_z \\ &= \vec{v}\vec{n} \end{aligned} \tag{107-a}$$

$$\begin{aligned} \lambda_2 &= un_x + vn_y + wn_z \\ &= \vec{v}\vec{n} \end{aligned} \tag{107-b}$$

$$\begin{aligned} \lambda_3 &= un_x + vn_y + wn_z + \sqrt{(un_x)^2 + (vn_y)^2 + (wn_z)^2 + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2)} \\ &= \vec{v}\vec{n} + a \end{aligned} \tag{107-c}$$

$$\begin{aligned} \lambda_4 &= un_x + vn_y + wn_z - \sqrt{(un_x)^2 + (vn_y)^2 + (wn_z)^2 + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2)} \\ &= \vec{v}\vec{n} - a \end{aligned} \tag{107-d}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \vec{v}\vec{n} & -- & -- & -- \\ -- & \vec{v}\vec{n} & -- & -- \\ -- & -- & \vec{v}\vec{n} + a & -- \\ -- & -- & -- & \vec{v}\vec{n} - a \end{bmatrix} \tag{108}$$

$$where\ a^2 = \vec{v}\vec{n}^2 + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2). \tag{109}$$

As stated by eqn (94), the matrices $\mathbf{P}$ and $\mathbf{P}^{-1}$ can be gained from these eigenvalues combined with the matrix $\mathbf{K}$.

Using the right eigenvectors of $\mathbf{K}$ -

$$\mathbf{K}_{ik}(r_k)^j = \lambda_j(r_i)^j \ ,\ summation\ in\ k$$

$$\begin{aligned} j = 1, 2\ , \quad \mathbf{c}^2 n_x r_2 + \mathbf{c}^2 n_y r_3 + \mathbf{c}^2 n_z r_4 &= \vec{v}\vec{n} r_1 \\ n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) &= 0 \\ n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) &= 0 \\ n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) &= 0 \\ \Rightarrow \vec{n} r_1 + \vec{v}(n_x r_2 + n_y r_3 + n_z r_4) &= 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow r_1 &\equiv 0 \\ and\ (n_x r_2 + n_y r_3 + n_z r_4) &\equiv 0 \end{aligned}$$

$$Taking\ r_3 = 0,\ r_4 = -\frac{n_x}{n_z} r_2$$

$$Taking\ r_4 = 0,\ r_3 = -\frac{n_x}{n_y} r_2$$

There is of course the third option of taking $r_2 = 0$, but as will be shown later, the choice of which two eigenvector components are taken to be zero is irrelevant.

$$\begin{aligned} j = 3\ , \quad \mathbf{c}^2 n_x r_2 + \mathbf{c}^2 n_y r_3 + \mathbf{c}^2 n_z r_4 &= (\vec{v}\vec{n} + a) r_1 \\ n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) &= a r_2 \\ n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) &= a r_3 \\ n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) &= a r_4 \end{aligned}$$

if r_1 *is made arbitrary (the norm of* $\vec{n}$ *doesn't matter),*

$$\begin{aligned} \Rightarrow r_1 &= \mathbf{c}^2 a \\ \Rightarrow r_2 &= \mathbf{c}^2 n_x + u(\vec{v}\vec{n} + a) \\ \Rightarrow r_3 &= \mathbf{c}^2 n_y + v(\vec{v}\vec{n} + a) \\ \Rightarrow r_4 &= \mathbf{c}^2 n_z + w(\vec{v}\vec{n} + a) \end{aligned}$$

$$\begin{aligned} j = 4\ , \quad \mathbf{c}^2 n_x r_2 + \mathbf{c}^2 n_y r_3 + \mathbf{c}^2 n_z r_4 &= (\vec{v}\vec{n} - a) r_1 \\ n_x r_1 + u(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_2 \\ n_y r_1 + v(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_3 \\ n_z r_1 + w(n_x r_2 + n_y r_3 + n_z r_4) &= -a r_4 \end{aligned}$$

if r_1 *is made arbitrary again,*

$$\begin{aligned} \Rightarrow r_1 &= -\mathbf{c}^2 a \\ \Rightarrow r_2 &= \mathbf{c}^2 n_x + u(\vec{v}\vec{n} - a) \\ \Rightarrow r_3 &= \mathbf{c}^2 n_y + v(\vec{v}\vec{n} - a) \\ \Rightarrow r_4 &= \mathbf{c}^2 n_z + w(\vec{v}\vec{n} - a) \end{aligned}$$

Therefore,

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & \mathbf{c}^2 a & -\mathbf{c}^2 a \\ -n_z & -n_y & u(\vec{v}\vec{n} + a) + \mathbf{c}^2 n_x & u(\vec{v}\vec{n} - a) + \mathbf{c}^2 n_x \\ 0 & n_x & v(\vec{v}\vec{n} + a) + \mathbf{c}^2 n_y & v(\vec{v}\vec{n} - a) + \mathbf{c}^2 n_y \\ n_x & 0 & w(\vec{v}\vec{n} + a) + \mathbf{c}^2 n_z & w(\vec{v}\vec{n} - a) + \mathbf{c}^2 n_z \end{bmatrix} \tag{110}$$

From which its inverse can be found -

$$\mathbf{P^{-1}} = \begin{bmatrix} \frac{n_z\vec{v}\vec{n}-w\vec{n}^2}{n_x a^2} & -\frac{w\vec{v}\vec{n}+n_z c^2}{a^2} & -\frac{n_y(w\vec{v}\vec{n}+n_z c^2)}{n_x a^2} & \frac{(un_x+vn_y)\vec{v}\vec{n}+(n_x^2+n_y^2)c^2}{n_x a^2} \\ \frac{n_y\vec{v}\vec{n}-v\vec{n}^2}{n_x a^2} & -\frac{v\vec{v}\vec{n}+n_y c^2}{a^2} & \frac{(un_x+wn_z)\vec{v}\vec{n}+(n_x^2+n_z^2)c^2}{n_x a^2} & -\frac{n_z(v\vec{v}\vec{n}+n_y c^2)}{n_x a^2} \\ -\frac{\vec{v}\vec{n}-a}{2a^2c^2} & \frac{n_x}{2a^2} & \frac{n_y}{2a^2} & \frac{n_z}{2a^2} \\ -\frac{\vec{v}\vec{n}+a}{2a^2c^2} & \frac{n_x}{2a^2} & \frac{n_y}{2a^2} & \frac{n_z}{2a^2} \end{bmatrix} \tag{111}$$

It should be noted at this point that if the eigenvector compoments r_2 and r_4 were taken to be zero during the calculation of the first and second column of the matrix $\mathbf{P}$ then a $\frac{1}{n_y}$ term would appear in the $\mathbf{P}^{-1}$ matrix instead of $\frac{1}{n_x}$. The process is the same for r_2 and r_3 with n_z respectively. The choice of which two components are reduced to zero is irrelevant due to the multiplication of $\mathbf{P}$ and $\mathbf{P}^{-1}$ during the calculation of $\mathbf{K}$ and $\mathbf{F}$.

By equations (93), (96),

$$\begin{aligned} \mathbf{F}^{\pm} &= \mathbf{K}^{\pm}\mathbf{U} \\ &= \mathbf{P}\mathbf{\Lambda}^{\pm}\mathbf{P}^{-1}\mathbf{U} \end{aligned}$$

All of these matrices are now known. Thus $\mathbf{K}^{\pm}$ and $\mathbf{F}^{\pm}$ can be defined as shown in Equations 112 and 113 respectively.

$$
\mathbf{K}^{\pm} = \begin{bmatrix}
\frac{1}{2a}\left\{\begin{array}{c} -\lambda_3^{\pm}(\vec{v}\vec{n}-a) \\ +\lambda_4^{\pm}(\vec{v}\vec{n}+a) \end{array}\right\} & \frac{\mathbf{c}^2}{2a}n_x(\lambda_3^{\pm}-\lambda_4^{\pm}) & \frac{\mathbf{c}^2}{2a}n_y(\lambda_3^{\pm}-\lambda_4^{\pm}) & \frac{\mathbf{c}^2}{2a}n_z(\lambda_3^{\pm}-\lambda_4^{\pm}) \\
\frac{1}{a^2}\left\{\begin{array}{c} \lambda_1(n_x\vec{v}\vec{n}-u) \\ -\frac{1}{2\mathbf{c}^2}\lambda_3\left(\vec{v}\vec{n}-a\right)\begin{pmatrix} u\left(\vec{v}\vec{n}+a\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \\ -\frac{1}{2\mathbf{c}^2}\lambda_4\left(\vec{v}\vec{n}+a\right)\begin{pmatrix} u\left(\vec{v}\vec{n}-a\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} \lambda_1\begin{pmatrix} (vn_y+wn_z)\vec{v}\vec{n} \\ +(n_y^2+n_z^2)\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_x}{2}\lambda_3\left(u\left(\vec{v}\vec{n}+a\right)+n_x\mathbf{c}^2\right) \\ +\frac{n_x}{2}\lambda_4\left(u\left(\vec{v}\vec{n}-a\right)+n_x\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_y\left(u\vec{v}\vec{n}+n_x\mathbf{c}^2\right) \\ +\frac{n_y}{2}\lambda_3\left(u\left(\vec{v}\vec{n}+a\right)+n_x\mathbf{c}^2\right) \\ +\frac{n_y}{2}\lambda_4\left(u\left(\vec{v}\vec{n}-a\right)+n_x\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_z\left(u\vec{v}\vec{n}+n_x\mathbf{c}^2\right) \\ +\frac{n_z}{2}\lambda_3\left(u\left(\vec{v}\vec{n}+a\right)+n_x\mathbf{c}^2\right) \\ +\frac{n_z}{2}\lambda_4\left(u\left(\vec{v}\vec{n}-a\right)+n_x\mathbf{c}^2\right) \end{array}\right\} \\
\frac{1}{a^2}\left\{\begin{array}{c} \lambda_1(n_y\vec{v}\vec{n}-v) \\ -\frac{1}{2\mathbf{c}^2}\lambda_3\left(\vec{v}\vec{n}-a\right)\begin{pmatrix} v\left(\vec{v}\vec{n}+a\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \\ -\frac{1}{2\mathbf{c}^2}\lambda_4\left(\vec{v}\vec{n}+a\right)\begin{pmatrix} v\left(\vec{v}\vec{n}-a\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_x\left(v\vec{v}\vec{n}+n_y\mathbf{c}^2\right) \\ +\frac{n_x}{2}\lambda_3\left(v\left(\vec{v}\vec{n}+a\right)+n_y\mathbf{c}^2\right) \\ +\frac{n_x}{2}\lambda_4\left(v\left(\vec{v}\vec{n}-a\right)+n_y\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} \lambda_1\begin{pmatrix} (un_x+wn_z)\vec{v}\vec{n} \\ +(n_x^2+n_z^2)\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_y}{2}\lambda_3\left(v\left(\vec{v}\vec{n}+a\right)+n_y\mathbf{c}^2\right) \\ +\frac{n_y}{2}\lambda_4\left(v\left(\vec{v}\vec{n}-a\right)+n_y\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_z\left(v\vec{v}\vec{n}+n_y\mathbf{c}^2\right) \\ +\frac{n_z}{2}\lambda_3\left(v\left(\vec{v}\vec{n}+a\right)+n_y\mathbf{c}^2\right) \\ +\frac{n_z}{2}\lambda_4\left(v\left(\vec{v}\vec{n}-a\right)+n_y\mathbf{c}^2\right) \end{array}\right\} \\
\frac{1}{a^2}\left\{\begin{array}{c} \lambda_1(n_z\vec{v}\vec{n}-w) \\ -\frac{1}{2\mathbf{c}^2}\lambda_3\left(\vec{v}\vec{n}-a\right)\begin{pmatrix} w\left(\vec{v}\vec{n}+a\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ -\frac{1}{2\mathbf{c}^2}\lambda_4\left(\vec{v}\vec{n}+a\right)\begin{pmatrix} w\left(\vec{v}\vec{n}-a\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_x\left(w\vec{v}\vec{n}+n_z\mathbf{c}^2\right) \\ +\frac{n_x}{2}\lambda_3\left(w\left(\vec{v}\vec{n}+a\right)+n_z\mathbf{c}^2\right) \\ +\frac{n_x}{2}\lambda_4\left(w\left(\vec{v}\vec{n}-a\right)+n_z\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_y\left(w\vec{v}\vec{n}+n_z\mathbf{c}^2\right) \\ +\frac{n_y}{2}\lambda_3\left(w\left(\vec{v}\vec{n}+a\right)+n_z\mathbf{c}^2\right) \\ +\frac{n_y}{2}\lambda_4\left(w\left(\vec{v}\vec{n}-a\right)+n_z\mathbf{c}^2\right) \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} \lambda_1\begin{pmatrix} (un_x+vn_y)\vec{v}\vec{n} \\ +(n_x^2+n_y^2)\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_z}{2}\lambda_3\left(w\left(\vec{v}\vec{n}+a\right)+n_z\mathbf{c}^2\right) \\ +\frac{n_z}{2}\lambda_4\left(w\left(\vec{v}\vec{n}-a\right)+n_z\mathbf{c}^2\right) \end{array}\right\}
\end{bmatrix} \tag{112}
$$

$$
\mathbf{F}^{\pm} = \left[\begin{array}{c}
\frac{p}{2a}\left\{\lambda_4^{\pm}(\vec{v}\vec{n}+a)-\lambda_3^{\pm}(\vec{v}\vec{n}-a)\right\}+\frac{c^2\vec{v}\vec{n}}{2a}\left\{\lambda_3^{\pm}-\lambda_4^{\pm}\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_x\vec{v}\vec{n}-u\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}-a)(u(\vec{v}\vec{n}+a)+n_x\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}+a)(u(\vec{v}\vec{n}-a)+n_x\mathbf{c}^2)\right\} \\
+\frac{u\lambda_1^{\pm}}{a^2}\left\{(vn_y+wn_z)\vec{v}\vec{n}+(n_y^2+n_z^2)c^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{a^2}\left\{u\vec{v}\vec{n}+n_xc^2\right\} \\
-\frac{wn_z\lambda_1^{\pm}}{a^2}\left\{u\vec{v}\vec{n}+n_xc^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(u(\vec{v}\vec{n}+a)+n_xc^2)+\lambda_4^{\pm}(u(\vec{v}\vec{n}-a)+n_x\mathbf{c}^2)\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_y\vec{v}\vec{n}-v\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}-a)(v(\vec{v}\vec{n}+a)+n_y\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}+a)(v(\vec{v}\vec{n}-a)+n_y\mathbf{c}^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{a^2}\left\{v\vec{v}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{v\lambda_1^{\pm}}{a^2}\left\{(un_x+wn_z)\vec{v}\vec{n}+(n_x^2+n_z^2)c^2\right\} \\
-\frac{wn_z\lambda_1^{\pm}}{a^2}\left\{v\vec{v}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(v(\vec{v}\vec{n}+a)+n_yc^2)+\lambda_4^{\pm}(v(\vec{v}\vec{n}-a)+n_y\mathbf{c}^2)\right\} \\
\\
\frac{p}{a^2}\left\{\lambda_1^{\pm}(n_z\vec{v}\vec{n}-w\vec{n}^2)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}-a)(w(\vec{v}\vec{n}+a)+n_z\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{c}^2}(\vec{v}\vec{n}+a)(w(\vec{v}\vec{n}-a)+n_z\mathbf{c}^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{a^2}\left\{w\vec{v}\vec{n}+n_z\mathbf{c}^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{a^2}\left\{w\vec{v}\vec{n}+n_z\mathbf{c}^2\right\} \\
+\frac{w\lambda_1^{\pm}}{a^2}\left\{(un_x+vn_y)\vec{v}\vec{n}+(n_x^2+n_y^2)\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2a^2}\left\{\lambda_3^{\pm}(w(\vec{v}\vec{n}+a)+n_z\mathbf{c}^2)+\lambda_4^{\pm}(w(\vec{v}\vec{n}-a)+n_z\mathbf{c}^2)\right\}
\end{array}\right] \tag{113}
$$

B.2 Moving mesh formulation

When moving meshes are introduced the flux vectors alter to accomodate the Global Conservation Law by the introduction of the contravarient velocities. As a result the state and flux vector are defined as

$$\mathbf{U} = \begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix} \mathbf{f} = \begin{bmatrix} \mathbf{c}^2(u - u_G) \\ u(u - u_G) + p \\ v(u - u_G) \\ w(u - u_G) \end{bmatrix} \mathbf{g} = \begin{bmatrix} \mathbf{c}^2(v - v_G) \\ u(v - v_G) \\ v(v - v_G) + p \\ w(v - v_G) \end{bmatrix} \mathbf{h} = \begin{bmatrix} \mathbf{c}^2(w - w_G) \\ u(w - w_G) \\ v(w - w_G) \\ w(w - w_G) + p \end{bmatrix}$$

The method of calculating the eigenvectors and the various matrices is the same as defined in Appendix B.1. The eigenvector, Λ_{MM}, is defined in Equation 114. Matrices $\mathbf{P}_{MM}$ and $\mathbf{P}^{-1}_{MM}$ are defined in Equations 118 and 119 respectively. Finally the directionalised jacobian, $\mathbf{K}^{\pm}_{MM}$, and the directionalised flux vector, $\mathbf{F}^{\pm}_{MM}$, are defined by Equations 120 and 121 respectively.

$$\mathbf{\Lambda}_{MM} = \begin{bmatrix} \overrightarrow{v_g}\vec{n} & -- & -- & -- \\ -- & \overrightarrow{v_g}\vec{n} & -- & -- \\ -- & -- & \overrightarrow{v_m}\vec{n} + \mathbf{a}_m & -- \\ -- & -- & -- & \overrightarrow{v_m}\vec{n} - \mathbf{a}_m \end{bmatrix} \tag{114}$$

$$\text{where} \quad \mathbf{a}^2_m = \overrightarrow{v_m}\vec{n} + \mathbf{c}^2(n_x^2 + n_y^2 + n_z^2) \tag{115}$$

Because of the use of two distinct velocity terms, $\vec{v}$ and $\overrightarrow{v_g}$ defining the flow speed and the mesh speed respectively, there are a number of extra velocity terms used -

$$\overrightarrow{v_{cv}} = \vec{v} - \overrightarrow{v_g} \tag{116}$$

$$\overrightarrow{v_m} = \vec{v} - \frac{1}{2}\overrightarrow{v_g} \tag{117}$$

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & \mathbf{c}^2\left(\mathbf{a}_m + \frac{1}{2}\overrightarrow{v_g}\vec{n}\right) & -\mathbf{c}^2\left(\mathbf{a}_m - \frac{1}{2}\overrightarrow{v_g}\vec{n}\right) \\ -n_z & -n_y & u(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m) + \mathbf{c}^2 n_x & u(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m) + \mathbf{c}^2 n_x \\ 0 & n_x & v(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m) + \mathbf{c}^2 n_y & v(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m) + \mathbf{c}^2 n_y \\ n_x & 0 & w(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m) + \mathbf{c}^2 n_z & w(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m) + \mathbf{c}^2 n_z \end{bmatrix} \tag{118}$$

$$\mathbf{P^{-1}} = \begin{bmatrix} \frac{n_x\vec{v}\vec{n}-w}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{w\overrightarrow{v_{cv}}\vec{n}+n_x c^2}{\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{n_y(w\overrightarrow{v_{cv}}\vec{n}+n_x c^2)}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{(un_x+vn_y)\overrightarrow{v_{cv}}\vec{n}+(n_x^2+n_y^2)\mathbf{c}^2}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ \frac{n_y\vec{v}\vec{n}-v}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{v\overrightarrow{v_{cv}}\vec{n}+n_y c^2}{\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{(un_x+wn_z)\overrightarrow{v_{cv}}\vec{n}+(n_x^2+n_z^2)\mathbf{c}^2}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & -\frac{n_z(v\overrightarrow{v_{cv}}\vec{n}+n_y c^2)}{n_x\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ \frac{\vec{v}\vec{n}\left(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m\right)+\mathbf{c}^2}{2\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)\mathbf{a}_m\mathbf{c}^2} & \frac{n_x\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_y\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_z\left(\mathbf{a}_m-\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \\ -\frac{\vec{v}\vec{n}\left(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m\right)+\mathbf{c}^2}{2\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)\mathbf{a}_m\mathbf{c}^2} & \frac{n_x\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_y\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} & \frac{n_z\left(\mathbf{a}_m+\frac{1}{2}\overrightarrow{v_g}\vec{n}\right)}{2\mathbf{a}_m\left(a^2-\overrightarrow{v_g}\vec{n}\vec{v}\vec{n}\right)} \end{bmatrix} \tag{119}$$

$$
\mathbf{K}^{\pm} = \left[\begin{array}{cccc}
\frac{1}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_3^{\pm}\phi_{\mathcal{B}}(\vec{v}\vec{n}\phi_{\mathcal{B}} - a^2) \\ +\lambda_4^{\pm}\phi_{\mathcal{C}}(\vec{v}\vec{n}\phi_{\mathcal{C}} + a^2) \end{array}\right\} & \frac{\mathbf{c}^2 n_x}{2\mathbf{a}_m\phi_{\mathcal{D}}}(\lambda_3^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}} - \lambda_4^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}}) & \frac{\mathbf{c}^2 n_y}{2\mathbf{a}_m\phi_{\mathcal{D}}}(\lambda_3^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}} - \lambda_4^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}}) & \frac{\mathbf{c}^2 n_z}{2\mathbf{a}_m\phi_{\mathcal{D}}}(\lambda_3^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}} - \lambda_4^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}}) \\
\frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1(n_x\vec{v}\vec{n} - u) \\ -\frac{\lambda_3\left(\vec{v}\vec{n}\phi_{\mathcal{B}} - a^2\right)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ -\frac{\lambda_4\left(\vec{v}\vec{n}\phi_{\mathcal{C}} + a^2\right)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1\left(\begin{array}{c} (vn_y + wn_z)\overrightarrow{v_{cv}}\vec{n} \\ +(n_y^2 + n_z^2)\mathbf{c}^2 \end{array}\right) \\ +\frac{n_x\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_x\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_1 n_y\left(u\overrightarrow{v_{cv}}\vec{n} + n_x\mathbf{c}^2\right) \\ +\frac{n_y\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} \left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_y\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{a^2}\left\{\begin{array}{c} -\lambda_1 n_z\left(u\overrightarrow{v_{cv}}\vec{n} + n_x\mathbf{c}^2\right) \\ +\frac{n_z\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_z\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} u\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_x\mathbf{c}^2 \end{pmatrix} \end{array}\right\} \\
\frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1(n_y\vec{v}\vec{n} - v) \\ -\frac{\lambda_3\left(\vec{v}\vec{n}\phi_{\mathcal{B}} - a^2\right)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \\ -\frac{\lambda_4\left(\vec{v}\vec{n}\phi_{\mathcal{C}} + a^2\right)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_1 n_x\left(v\overrightarrow{v_{cv}}\vec{n} + n_y\mathbf{c}^2\right) \\ +\frac{n_x\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_x\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1\left(\begin{array}{c} (un_x + wn_z)\overrightarrow{v_{cv}}\vec{n} \\ +(n_x^2 + n_z^2)\mathbf{c}^2 \end{array}\right) \\ +\frac{n_y\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_y\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_1 n_z\left(v\overrightarrow{v_{cv}}\vec{n} + n_y\mathbf{c}^2\right) \\ +\frac{n_z\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_z\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} v\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_y\mathbf{c}^2 \end{pmatrix} \end{array}\right\} \\
\frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1(n_z\vec{v}\vec{n} - w) \\ -\frac{\lambda_3(\vec{v}\vec{n}\phi_{\mathcal{B}} - a)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ -\frac{\lambda_4(\vec{v}\vec{n}\phi_{\mathcal{C}} + a)}{2\mathbf{a}_m\mathbf{c}^2}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_1 n_x\left(w\overrightarrow{v_{cv}}\vec{n} + n_z\mathbf{c}^2\right) \\ +\frac{n_x\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_x\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} -\lambda_1 n_y\left(w\overrightarrow{v_{cv}}\vec{n} + n_z\mathbf{c}^2\right) \\ +\frac{n_y\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_y\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\} & \frac{1}{\phi_{\mathcal{D}}}\left\{\begin{array}{c} \lambda_1\left(\begin{array}{c} (un_x + vn_y)\overrightarrow{v_{cv}}\vec{n} \\ +(n_x^2 + n_y^2)\mathbf{c}^2 \end{array}\right) \\ +\frac{n_z\lambda_3\phi_{\mathcal{C}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} + \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \\ +\frac{n_z\lambda_4\phi_{\mathcal{B}}}{2\mathbf{a}_m}\begin{pmatrix} w\left(\overrightarrow{v_m}\vec{n} - \mathbf{a}_m\right) \\ +n_z\mathbf{c}^2 \end{pmatrix} \end{array}\right\}
\end{array}\right] \tag{120}
$$

where $\phi_{\mathcal{D}} = a^2 - \overrightarrow{v_g}\vec{n}\vec{v}\vec{n}$, $\phi_{\mathcal{B}} = \mathbf{a}_m + \frac{1}{2}\overrightarrow{v_g}\vec{n}$ and $\phi_{\mathcal{C}} = \mathbf{a}_m - \frac{1}{2}\overrightarrow{v_g}\vec{n}$

$$
\mathbf{F}^{\pm} = \left[\begin{array}{c}
\frac{p}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\lambda_4^{\pm}(\vec{v}\vec{n}\phi_{\mathcal{C}}+a^2)-\lambda_3^{\pm}(\vec{v}\vec{n}\phi_{\mathcal{B}}-a^2)\right\}+\frac{c^2\vec{v}\vec{n}}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\lambda_3^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}}-\lambda_4^{\pm}\phi_{\mathcal{B}}\phi_{\mathcal{C}}\right\} \\
\\
\frac{p}{\phi_{\mathcal{D}}}\left\{\lambda_1^{\pm}(n_x\vec{v}\vec{n}-u)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{B}}-a)(u(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_x\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{C}}+a)(u(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_x\mathbf{c}^2)\right\} \\
+\frac{u\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{(vn_y+wn_z)\overrightarrow{v_{cv}}\vec{n}+(n_y^2+n_z^2)\mathbf{c}^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{u\overrightarrow{v_{cv}}\vec{n}+n_xc^2\right\} \\
-\frac{wn_z\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{u\overrightarrow{v_{cv}}\vec{n}+n_xc^2\right\} \\
+\frac{\vec{v}\vec{n}}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\lambda_3^{\pm}\phi_{\mathcal{C}}(u(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_x\mathbf{c}^2)+\lambda_4^{\pm}\phi_{\mathcal{B}}(u(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_x\mathbf{c}^2)\right\} \\
\\
\frac{p}{\phi_{\mathcal{D}}}\left\{\lambda_1^{\pm}(n_y\vec{v}\vec{n}-v)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{B}}-a)(v(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_y\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{C}}+a)(v(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_y\mathbf{c}^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{v\overrightarrow{v_{cv}}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{v\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{(un_x+wn_z)\overrightarrow{v_{cv}}\vec{n}+(n_x^2+n_z^2)\mathbf{c}^2\right\} \\
-\frac{wn_z\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{v\overrightarrow{v_{cv}}\vec{n}+n_y\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\lambda_3^{\pm}\phi_{\mathcal{C}}(v(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_y\mathbf{c}^2)+\lambda_4^{\pm}\phi_{\mathcal{B}}(v(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_y\mathbf{c}^2)\right\} \\
\\
\frac{p}{\phi_{\mathcal{D}}}\left\{\lambda_1^{\pm}(n_z\vec{v}\vec{n}-w)\right. \\
-\frac{\lambda_3^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{B}}-a)(w(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_z\mathbf{c}^2) \\
\left.-\frac{\lambda_4^{\pm}}{2\mathbf{a}_m\mathbf{c}^2}(\vec{v}\vec{n}\phi_{\mathcal{C}}+a)(w(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_z\mathbf{c}^2)\right\} \\
-\frac{un_x\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{w\overrightarrow{v_{cv}}\vec{n}+n_z\mathbf{c}^2\right\} \\
-\frac{vn_y\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{w\overrightarrow{v_{cv}}\vec{n}+n_z\mathbf{c}^2\right\} \\
+\frac{w\lambda_1^{\pm}}{\phi_{\mathcal{D}}}\left\{(un_x+vn_y)\overrightarrow{v_{cv}}\vec{n}+(n_x^2+n_y^2)\mathbf{c}^2\right\} \\
+\frac{\vec{v}\vec{n}}{2\mathbf{a}_m\phi_{\mathcal{D}}}\left\{\lambda_3^{\pm}\phi_{\mathcal{C}}(w(\overrightarrow{v_m}\vec{n}+\mathbf{a}_m)+n_z\mathbf{c}^2)+\lambda_4^{\pm}\phi_{\mathcal{B}}(w(\overrightarrow{v_m}\vec{n}-\mathbf{a}_m)+n_z\mathbf{c}^2)\right\}
\end{array}\right] \tag{121}
$$