GENERALIZED SPATIAL DISCRETIZATION TECHNIQUES FOR SPACE-MARCHING ALGORITHMS

by

William Dandridge McGrory

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

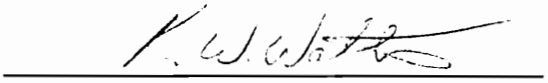
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

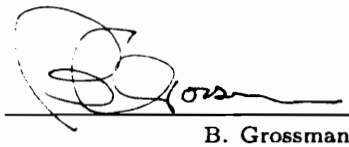
in

Aerospace Engineering

APPROVED:

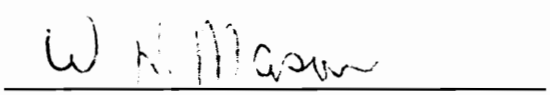


R. W. Walters, Chairman



B. Grossman

J. A. Schetz



W. Mason

W. L. Neu

January, 1991

Blacksburg, Virginia

GENERALIZED SPATIAL DISCRETIZATION TECHNIQUES FOR SPACE-MARCHING ALGORITHMS

by

William Dandridge McGrory

Committee Chairman: Robert W. Walters

Aerospace Engineering

(ABSTRACT)

Two unique spatial discretizations employing generalized indexing strategies suitable for use with space-marching algorithms are presented for the numerical solution of the equations of fluid dynamics. Both discretizations attempt to improve geometric flexibility as compared to structured indexing strategies and have been formulated while considering the current and future availability of unstructured grid generation techniques. The first discretization employs a generalized indexing strategy utilizing triangular elements in the two dimensions normal to the streamwise direction, while maintaining structure within the streamwise direction. The second discretization subdivides the domain into a collection of computational blocks. Each block has inflow and outflow boundaries suitable for space marching. A completely generalized indexing strategy utilizing tetrahedra is used within each computational block. The solution to the flow in each block is found independently in a fashion similar to the cross-flow planes of a structured discretization. Numerical algorithms have been developed for the solution of the governing equations on each of the two proposed discretizations. These spatial discretizations are obtained by applying a characteristic-based, upwind, finite volume scheme for the solution of the Euler equations. First-order and higher spatial accuracy is achieved with

these implementations. A time dependent, space-marching algorithm is employed, with explicit time integration for convergence of individual computational blocks. Grid generation techniques suitable for the proposed discretizations are discussed. Applications of these discretization techniques include the high speed flow about a 5° cone, an analytic forebody, and a model SR71 aircraft.

ACKNOWLEDGEMENTS

The author would like to express his deepest appreciation for all of the advice, support, and encouragement given to him by his advisor, Dr. Robert W. Walters. In addition, if not for Dr. Walters' enthusiasm in the classroom while the author was an undergraduate, he likely would not have pursued a career in the field of computational fluid dynamics. The author would also like to thank Dr. Bernard Grossman for all of his assistance over the past five years. He has thoroughly enjoyed the lectures Dr. Grossman has given, and thankful for all of the private assistance he has offered. The author would like to thank Dr. Joseph A. Schetz, Dr. William Mason, and Dr. Wayne L. Neu for their criticism of this dissertation. The author would like to thank all of his peers for innumerable and invaluable discussions which have contributed to the author's overall understanding of his research. Finally, the author would like to thank his parents for all of the encouragement and support which has been provided throughout his education.

Support for this work was provided by NSF grant ISI-8861052 and by NASA grant NAG-1-776.

TABLE OF CONTENTS

NOMENCLATURE

A	cross-sectional area
A	Jacobian Matrix
C_p	Nondimensionalized pressure coefficient
D	Diagonal Matrix
$\vec{F}$	Vector of the flux of conserved variables, Q
L	Lower Triangular Matrix
P	vertex node or point
P^K	Polynomial of degree K
Q	Vector of conserved variables
$\langle Q \rangle$	volume average of the conserved variables, Q
R	Residual calculated use in time integration
S	Surface area of volume, V
U	Upper Triangular Matrix
V	Volume
a	Speed of sound
c_x, c_y, c_z	Direction cosines in the x, y, z coordinate directions
e_0	Total energy per unit mass
f, g, h	Components of F in the three cartesian coordinate directions
$\hat{f}$	Component of F normal to the surface S
$\hat{i}, \hat{j}, \hat{k}$	unit normals in the three cartesian coordinate directions
i, j, k, n, m	indeces distinguishing different nodes or elements
$\hat{n}$	Unit normal to surface, S
p	Pressure

t	time
u, v, w	Cartesian velocity components
$\bar{u}$	Contravariant velocity
x, y, z	Cartesian coordinates
Δ	Forward difference operator
∇	Backward difference operator
α	Constants specified for Runge-Kutta explicit time integration
β	Parameter controlling the degree of limiting in the minmod limiter
γ	Ratio of specific heats
κ	Parameter controlling accuracy of *kappa* formulation
ϕ	Angle in conical coordinates formed between cone centerline and the line from a point to the cone vertex
ρ	density
ξ, η, ζ	Computational coordinates for structured discretizations

LIST OF FIGURES

1.0 INTRODUCTION

In the field of computational fluid dynamics (CFD), the rapid evolution of computer hardware in terms of both increased memory and CPU speed has brought about the desire to more accurately and efficiently model the forces acting on realistic and hence geometrically complex aerodynamic configurations. The topologies now being considered for numerical simulation are far more complex than those studied as recently as five years ago. Unfortunately, the vast majority of the advanced algorithm research effort has been centered around conventional finite-difference discretizations on structured grids. It has become painfully apparent that the generation of a single structured mesh about a complete aircraft is not acceptable.

In the past few years, the finite-element and finite-volume technologies have become standard tools of the CFD community. These methods have the significant advantage of being able to readily describe complex geometries in three dimensions through the use of unstructured meshes. For instance, figure 1 depicts two different discretizations of a cross-flow plane of a model SR71 aircraft. The structured quadrilateral grid suffers from highly skewed elements which result from connectivity requirements and the complexity of the boundary. The triangular discretization is not limited by any connectivity requirements with the result that no highly skewed elements are required. In addition, the high degree of control in the placement of elements makes adaptive mesh refinement very attractive which turns out to be very important for high speed compressible flow computations due to the presence of shock waves. However, the underlying algorithms used on these arbitrary grids are not nearly as advanced as their conventional mesh counterparts.

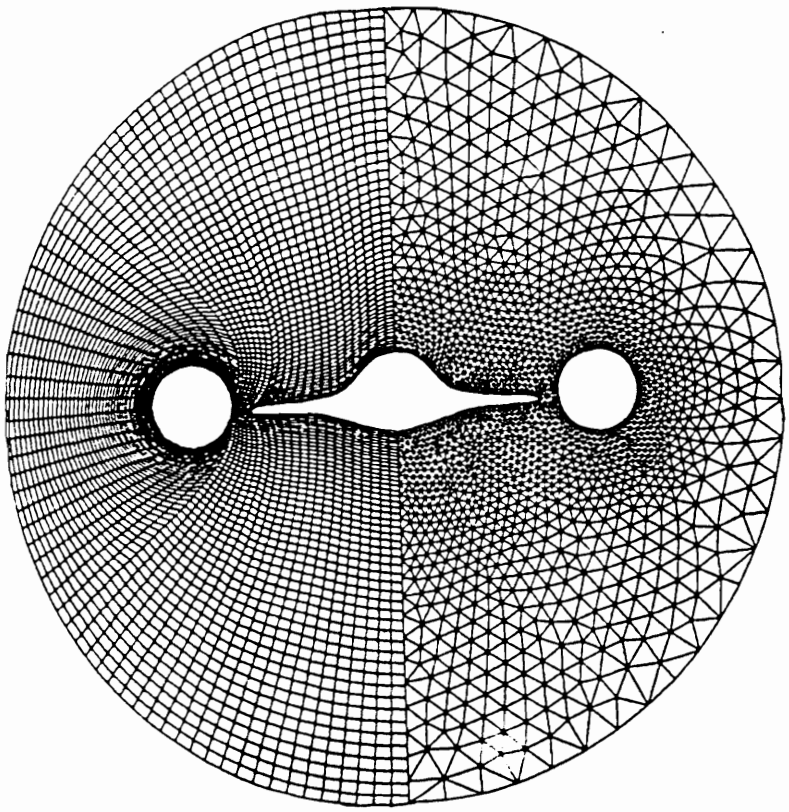


Figure 1: Comparison of quadrilateral and triangular grids about model SR71 cross section.

There are several motivations for developing alternative algorithms for use on unstructured grids in CFD simulations, particularly in the supersonic and hypersonic flight regimes. The renewed interest in hypersonic flight vehicles has stimulated a need for accurate and efficient computations at very high speeds. Flow about a model SR71 high performance aircraft at Mach 3.5 as depicted in figure 2 is one example. Another prime example is the National Aerospace Plane (NASP) in that the aerodynamic design of such a vehicle depends crucially on the engine-airframe integration. This in turn implies that both the external flowfield and the internal flowfield must be solved simultaneously. This greatly increases the geometric complexity of the problem and the grid generation process. Thus the topological complexity of these vehicles dictates the need for more sophisticated discretization techniques, but the lack of efficient flow solvers inhibits the use of finite-element methods on unstructured grids. Primarily, a reduction in the CPU execution time of the flow solver on arbitrary grids is needed. When the flight Mach numbers are greater than one, very efficient space-marching procedures have been developed for structured discretizations which are more than an order of magnitude faster than the conventional iterative methods commonly used for subsonic flow calculations[1,2,3]. However, there are no space-marching formulations available for computations on unstructured meshes. The numerical efficiency problem becomes apparent when one considers the fact that a single calculation about a very simple model of a hypersonic vehicle on the Cyber 205 vector supercomputer at the NASA Langley Research Center without a space-marching procedure required 21 hours of execution time[4] whereas the corresponding space-marching technique required less than 2 hours on the same grid[5]. Combining this with the fact that even the most efficient unstructured finite-element method (FEM) typically requires at least twice the CPU time of a conventional solver[6] (hence, at least 40 hours for the

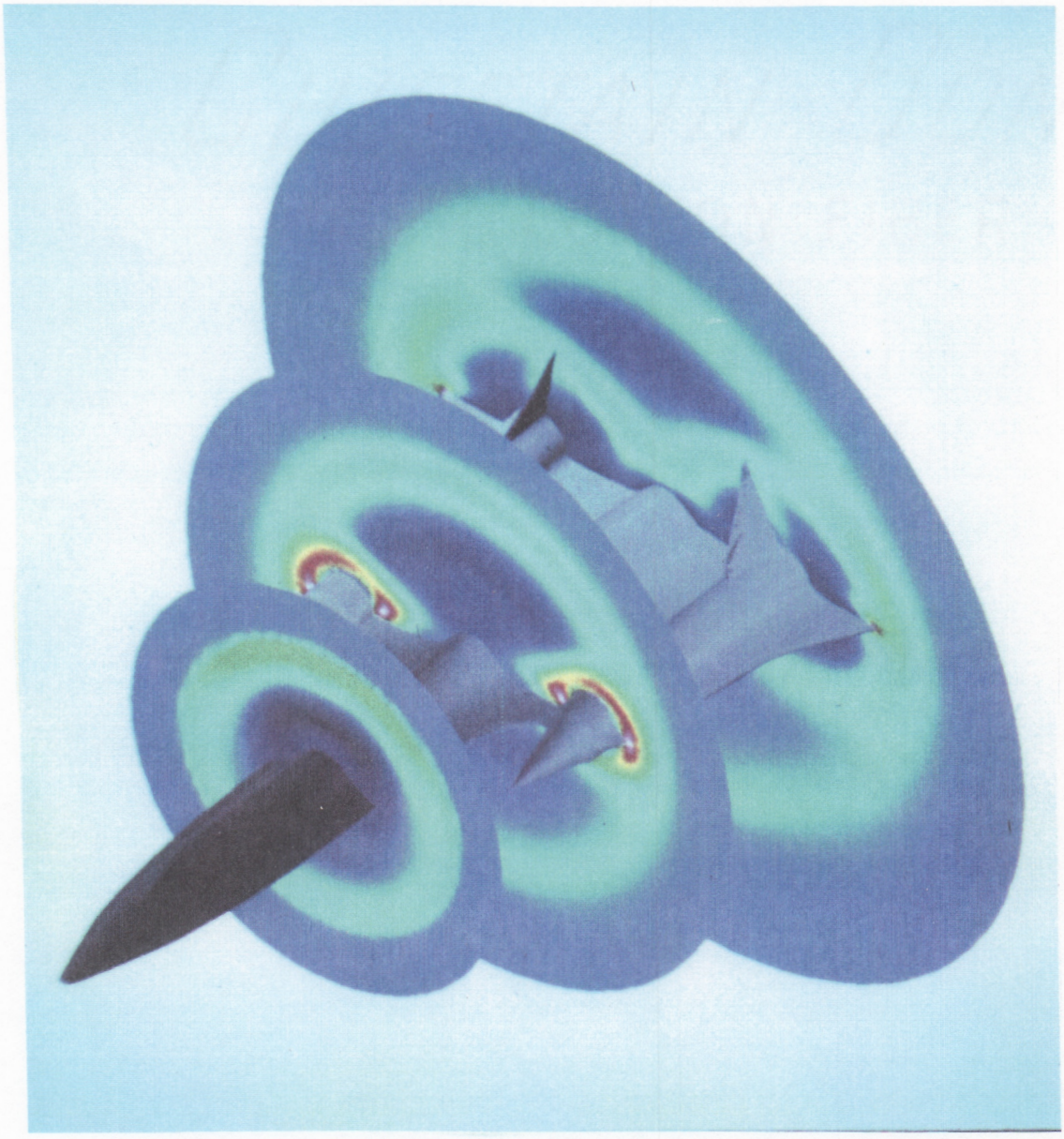


Figure 2: Pressure contours about three cross-flow planes of model SR71.

model problem) results in infrequent use of these schemes during the design process. Therefore, a substantial effort should be placed on developing algorithms capable of combining the efficiency of the marching methods with the geometric flexibility of the finite-element methods.

1.1 Background

1.1.1 General

Over the past two decades, the field of algorithm development for the numerical solution of the Navier Stokes equations has advanced significantly. Early algorithms include MacCormacks' explicit predictor-corrector method and its implicit counterpart[7,8], and implicit finite-difference schemes based on central difference approximations due to Briley and McDonald[9] and Beam and Warming[10]. Generalized time differencing, time linearization, and the so-called 'delta' form were incorporated ensuring steady-state solutions independent of the time step and at the same time yielding an unconditionally stable numerical method based on linear stability theory. Approximate factorization techniques were developed around the pioneering work of Douglas and Gunn[11] and Peaceman and Rachford[12] in order to efficiently solve the very large linear problem that arises from the linearization procedure. In the mid-to-late seventies, researchers at the NASA Ames Research Center coupled the now famous Beam and Warming algorithm with a computational mapping so that arbitrary domains could be readily analyzed provided that adequate grids were available to the user[13,14]. During this same time period, A. Jameson at the Courant Institute at NYU proceeded to develop explicit Runge-Kutta schemes for the time advancement along with residual smoothing and multi-grid methods for convergence acceleration[15,16]. In all of the above approaches, due to the presence of shock waves which manifest themselves as true discontinuities in the inviscid

limit, artificial viscosity had to be added to these algorithms in order to prevent the unbounded growth of high frequency waves. Structured grid generation techniques based on simple algebraic methods and on the numerical solution of differential equations were developed and used in conjunction with these flow solvers. These codes are in wide use in government and industry today.

A completely different approach for representing the spatial derivatives of the inviscid equations had been considered and developed to some extent by the Russian mathematician S. K. Godunov as early as 1959[17] and discussed by Peyret and Taylor[18]. It was based on the exact solution to the well known one-dimensional Riemann problem. However, at that time, the expense associated with the additional computational effort of solving the Riemann problem could not be justified. Moreover, it was not known how to extend the approach to second-order spatial accuracy or higher. Interest waned until the seventies and early eighties during which time a variety of mathematicians made remarkable progress in this area. Bram van Leer discovered how to increase the spatial accuracy of the method and a number of researchers developed approximate solutions to the Riemann problem making the approach much more computationally attractive. Notable among this group were the contributions of Chakravarthy[19,20], Harten[21,22], Osher and Solomon[23], Roe[24,25,26], Steger and Warming[27], Thomas[28,29], Van Leer[30,31], Woodward and Colella[32], and Yee[33]. Collectively, these schemes are referred to as upwind or flux-split schemes and have in common the fact that they model the propagation of information in accordance with the characteristic theory of hyperbolic systems of equations. During the Eighties, these schemes were extended to encompass the Navier-Stokes equations in generalized coordinates and a number of codes were developed for general use by the aircraft industry.

1.1.2 Space Marching on Structured Discretizations

Currently, space-marching methods represent the most efficient approach for obtaining steady-state solutions to the Euler or Parabolized Navier-Stokes equations provided that the constraints necessary for space marching are met. These methods are practical for high speed problems in which the inviscid flow is supersonic. When the solutions to the Navier Stokes equations are obtained over these same flow conditions, portions of the boundary layer are subsonic, thus effectively prohibiting the use of space-marching techniques. Vigneron devised a technique for weighting the contribution of the pressure gradient to the flux equations within the subsonic boundary layer[34]. When regions of subsonic flow are encountered, the weighting coefficient applied to the pressure term is calculated such that all of the characteristic waves are non negative. This has the effect of making the governing equations parabolic in this region when the unmodified equations are normally hyperbolic. With this approximation, the resulting Parabolized Navier-Stokes equations can be efficiently marched in space. Until recently, conservative space-marching techniques had been limited to first-order spatial accuracy in the marching direction. The advent and development of upwind or flux-split discretizations and the retention of the time derivative in the governing equations led to a family of higher-order accurate space-marching methods[1,2,3]. However, all of the marching methods currently in use were developed for structured grids and consequently, suffer the problem that complex geometries cannot be readily handled.

1.1.3 Zonal Interpolation Techniques

In order to partially alleviate the difficulty of discretizing a complex domain by a single set of ordered grid points, a number of researchers, inspired by the work of M.M. Rai[35,36,37], chose to break up the domain into a set of simpler regions

with the idea that generating a suitable and separate grid in each subregion of the domain is more viable. The composite grid is thus composed of a set of grids which are patched together along common interfaces. The individual grids are frequently referred to as zones and the overall approach is typically called a zonal or patched grid technique. Patched grid techniques have been successfully implemented with the governing equations of fluid dynamics and are basic features included in many of the production level codes in use today[5,38].

1.1.4 Generalized Indexing Strategies

On the other hand, the use of generalized indexing strategies on unstructured grids is far more powerful in terms of discretizing a complex flowfield. Generalized indexing stores the element and node connectivity in separate arrays. This allows the use of more arbitrary control volumes and eliminates the restriction of computational mapping of grid points to cartesian space used in conjunction with quadrilateral/hexahedral meshes. The French research team at INRIA[39] and the United Kingdom group at the University of Wales[40] have been pursuing this approach for perhaps the longest period of time in the CFD community. They routinely construct grids consisting of tetrahedra about space shuttle, fighter, and transport configurations. However, the algorithms commonly employed on these unstructured meshes are not as physically based as the characteristic based upwind schemes used on conventional grids.

In the past few years, researchers in the United States have begun to take notice of this work and have started to develop their own capability in this area. Very recently, upwind schemes on unstructured meshes have begun to be investigated. Of particular interest has been the search for more efficient time integration techniques and higher-order spatial accuracy. Slack and Whitaker have investigated both explicit and implicit time integration techniques for more efficient solution

of two-dimensional problems[41]. Barth has investigated techniques for performing multi-dimensional polynomial reconstruction of mean flow quantities as a means of increasing spatial accuracy for an generalized discretization[42,43]. One area of unstructured discretizations which promises to yield great increases in solution quality is that of solution adaptive grid refinement. Both Thareja et.al.[44], and Slack et.al.[45], have produced remarkable inviscid results on solution adaptive grids. However, no attempt has been made to exploit the fact that for high speed flows, space marching is possible and far more efficient numerically. This is understandable since the very nature of a three-dimensional unstructured grid makes the choice of a marching direction essentially meaningless.

1.1.5 Unstructured Grid Generation

Effective grid generation techniques have been critical to the success of generalized indexing strategies. In the past, the lack of effective discretization techniques necessary to obtain sufficient geometric flexibility has limited the acceptance of unstructured solution algorithms. However, two unstructured grid generation techniques have recently achieved some degree of acceptance in the CFD community. Both techniques discretize a domain into triangular elements in two dimensions, tetrahedral elements in three dimensions. The first approach is to generate points throughout the domain without regard to node connectivity. These nodes are then connected such that they form a Delaunay triangulation[46,47,48]. Various methods have been developed to provide the initial point distribution. Generation of nodes through the use of structured grid generators about portions of the domain has been popular[49]. Other methods which randomly introduce points have also been suggested. With these methods, points are added to the domain one at a time. As each point is added, it is tested against all other points already within the domain to see whether its addition violates some specified point distribution parameter. If it

does violate these conditions, it is removed from the list of nodes[50]. The second grid generation technique is the advancing front algorithm. This method generates nodes and element connectivity simultaneously. The element and node distribution are specified on some predefined background mesh. This algorithm was first introduced into the CFD community by Peraire[51], and also successfully applied by Löhner[52]. Both two and three-dimensional algorithms have been developed. However, the three-dimensional algorithm needs further effort to improve its robustness.

1.2 Scope of the Investigation

The purpose of this research is to investigate various discretization techniques which will allow the use of the numerically efficient space-marching algorithms while retaining the geometric flexibility of generalized indexing strategies. The simplest way to discretize a domain and ensure that no characteristic information may propagate upstream from a computational block is to require that all faces of the inflow and outflow cross-flow planes be oriented normal to the free-stream direction.

A typical discretization for a structured space-marching algorithm combines two-dimensional structured grids generated on separate cross-flow planes of the domain. The two-dimensional grids are oriented normal to the free-stream direction and are therefore suitable for use with the space-marching algorithm. These two-dimensional grids contain the same number of constant ξ lines and constant η lines such that when points of both constant ξ and η in two neighboring grids are combined, the resulting three-dimensional grid contains hexahedral volumes. If the component of velocity normal to each face of the cross-flow planes is always greater than the local speed of sound, the solution between two adjacent cross-flow planes will depend only upon values of that computational block and those values upstream of it, and the solution may be marched in space. This grid generation technique requires only the use of a two-dimensional structured grid generator.

Two approaches were chosen to extend the space-marching algorithm to generalized indexing strategies. In formulating these approaches, care has been taken to ensure the existence of cross-flow plane boundary faces which are parallel to the free stream. When considering alternative discretizations, thought must be given to the availability of discretization techniques.

The first discretization has been derived from a desire to use a two-dimensional unstructured grid generator. This is desired since there is a much wider availability of robust two-dimensional grid generators than robust three-dimensional grid generators. The first approach is to use a two-dimensional grid generator to discretize two neighboring planes with the same number of nodes and the same element connectivity in the same fashion as the structured discretization techniques. This method will retain the most structure to the discretization and will effectively increase geometric flexibility only in the cross-flow plane. While this method is thought to be the simplest to implement, it is not immediately possible since one cannot control the exact connectivity for a given geometry with currently available two-dimensional unstructured grid generators. With an unstructured grid there are no lines of constant ξ and η, the nodes are stored in a one-dimensional array while their elemental connectivity is kept in a second array. In order to apply this technique, it will be necessary to modify an existing two-dimensional grid generator to perform the task of generating consecutive grids of identical node distribution and element connectivity. Such a technique will result in control volumes which are prismatic in nature having three rectangular faces roughly parallel to the free stream, and two triangular faces normal to the free stream.

Due to the streamwise connectivity incorporated into this discretization, streamwise variations of the geometry will be somewhat limited. In order to alleviate this

restriction so that more complex geometries may be described, a zonal grid technique will be employed[35,36,37]. The domain will be split into two or more separate zones such that each zone will have two-dimensional grids created in the aforementioned manner. In order to incorporate this technique, it will be necessary to interpolate the values of the dependent variables from one zone to another. Even with the slight restriction on streamwise geometry, the resulting algorithm should still be capable of more accurately discretizing complex three-dimensional domains than existing structured algorithms.

While the first discretization technique attempts to discretize a domain with some degree of generalized indexing, it still attempts to retain as much structure as possible. The second discretization technique is formulated with the intent of retaining as little structure as possible. The necessary condition for use of a space-marching discretization is that the component of the Mach number normal to the inflow and outflow faces of each computational block be supersonic. In order to simplify the specification of inflow and outflow faces, all faces within the logical inflow and outflow planes will be planar. Thus a fully unstructured three-dimensional discretization may be employed between two adjacent cross-flow planes. This technique will only require that the faces of the outflow of one plane coincide with the inflow faces of the next plane. In this way no zonal interpolation will need to be employed. Each computational block between adjacent cross-flow planes will likely have a different number of elements within them, and different element connectivity. No influence in the element connectivity from one plane to the next will be imposed. Since a full unstructured discretization will be employed, suitable grid generation techniques will be required. Selection of such grid generation techniques will depend upon the robustness of the algorithm, and its ease of implementation.

Both discretization techniques have been implemented for the solution of the Euler equations of fluid dynamics. Issues concerning the spatial accuracy of these discretizations are discussed, as well as suitable time integration techniques within each computational block. Finally, several test cases have been used to demonstrate the flexibility of both of these discretizations.

2.0 GOVERNING EQUATIONS

In order to numerically obtain solutions about complex three-dimensional aircraft operating at hypersonic flight conditions it will be necessary to solve the complete Navier-Stokes equations which must include effects of nonequilibrium thermodynamics and chemical reactions. In addition suitable turbulence models must be employed. However, it is possible to make certain simplifying assumptions which will significantly reduce the complexity of the governing equations while retaining sufficient information to accurately assess the capabilities of different spatial discretizations.

2.1 The Euler Equations

If one assumes inviscid, adiabatic flow, one can derive the Euler equations which constitute the the conservation of mass, momentum and energy in a compressible, unsteady flow. If one carries out this derivation over a three-dimensional integral control volume, the following set of equations result:

$$\frac{\partial}{\partial t}\iiint_V Q \, dV + \iint_S \vec{F} \cdot \hat{n} \, dS = 0 \tag{2.1}$$

where

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e_0 \end{pmatrix} \tag{2.2}$$

and

$$\vec{F} = f\hat{i} + g\hat{j} + h\hat{k} \tag{2.3}$$

where

$$f = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho e_0 + p)u \end{pmatrix}, \quad g = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (\rho e_0 + p)v \end{pmatrix}, and \quad h = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw + p \\ \rho w^2 + p \\ (\rho e_0 + p)w \end{pmatrix} \tag{2.4}$$

where ρ is the density, $\langle u\hat{i}, v\hat{j}, w\hat{k}\rangle$ is the velocity vector, e_0 is the total energy per unit mass, p is the pressure, and $\hat{i}, \hat{j}, \hat{k}$ are orthogonal unit vectors in the x, y, z coordinate directions respectively. The vector $\vec{F}$ is the flux of mass, x, y, z momentum, and energy. The quantity $\vec{F} \cdot \hat{n}$ represents the flux per unit area out of the volume V, through the surface S when $\hat{n}$ is the locally outward pointing normal of S. This system of equations is closed by a perfect gas equation of state

$$p = (\gamma - 1)[\rho e_0 - \rho \frac{(u^2 + v^2 + w^2)}{2}] \tag{2.5}$$

where γ is the ratio of specific heats. Again, the assumption of a perfect gas equation of state greatly simplifies the governing equations while still retaining sufficient information to test the chosen spatial discretizations.

If the unit normal to the surface S is decomposed into its component directions,

$$\hat{n} = c_x\hat{i} + c_y\hat{j} + c_z\hat{k}, \tag{2.6}$$

The product of $\vec{F} \cdot \hat{n}$ may be written as

$$\hat{f} = \begin{pmatrix} \rho\bar{u} \\ \rho u\bar{u} + c_x p \\ \rho v\bar{u} + c_y p \\ \rho w\bar{u} + c_z p \\ (\rho e_0 + p)\bar{u} \end{pmatrix} \tag{2.7}$$

where $\bar{u}$ is the component of the velocity vector normal to the surface,

$$\bar{u} = c_x u + c_y v + c_z w. \tag{2.8}$$

2.2 Space-Marching Algorithm

A characteristic analysis of the one-dimensional Euler Equations reveals that information propagates from a point in three characteristic waves. These waves have speeds of u, $u \pm a$ where a is the local speed of sound. If the local velocity is everywhere greater than the local speed of sound, all information will propagate in the positive streamwise direction. One can take advantage of this fact when solving problems which satisfy these conditions. The solution for a particular point will only depend upon the solution of points upstream of it since information cannot propagate upstream. One can therefore completely space march the solution in the streamwise direction solving for each point as it is encountered.

This one-dimensional capability has be extended to multiple dimensions. The solution domain for a particular problem can be subdivided into planes such that each plane is oriented approximately normal to the predominant streamwise direction. If the local component of velocity normal to the inflow and outflow boundaries of each plane is greater than the local speed of sound, again, the solution in that plane will be independent of all planes downstream of it. Thus the solution of certain two and three-dimensional high speed problems can be marched in space in a fashion similar to the one-dimensional problem.

Many problems in two and three dimensions can be solved by employing a space-marching technique. These include certain internal flow calculations such as combustion inlets and nozzles, as well as certain high performance aircraft such as the National Aerospace Plane. The use of this algorithm offers significant savings in computational time as well as memory requirements. This algorithm is limited to solving aircraft which are comprised of slender fuselage and wing cross sections. The reason for this is that the algorithm will not support locally subsonic flow.

In addition to solving the Euler equations via a space-marching algorithm, it is also possible to modify the Navier Stokes equations to allow use of the technique. Normally, if the Navier Stokes equations were applied to the same class of problems, the flow would be everywhere supersonic except in a portion of the boundary layer. However, if the Vigneron technique[3,34] is applied, the characteristic eigenvalues of the system will all be positive, and the solution may be space marched. This technique results in a form of the governing equations known as the Parabolized Navier Stokes (PNS) equations.

2.3 Boundary Conditions

The solution of the integral governing equations requires the specification of initial conditions and boundary conditions. For steady-state problems, the initial conditions are set to free-stream conditions. Four distinct boundary conditions arise when flow conditions are suitable for the solution of the Euler equations with a space-marching algorithm.

The first boundary condition is related to supersonic inflow. In order to use the space-marching algorithm, the velocity normal to the inflow boundary must be everywhere greater than the local speed of sound. Since this has been assumed to be the case, the local flow conditions may be completely specified at this boundary. The contribution to the flux integral may then be calculated as a function of the specified local conditions.

The second boundary condition is that related to supersonic outflow. Again, since the governing equations are assumed to be suitable for space marching, the local velocity normal to the outflow boundary will be everywhere greater than the local speed of sound. Therefore, the local flow conditions may be completely specified by extrapolating information from the interior of the current plane. No information need be specified about conditions downstream of the outflow boundary.

The contribution to the flux integral may then be calculated as a function of the extrapolated local conditions.

The third boundary condition is used for a solid boundary. For this boundary condition, no flow through the boundary, i.e. $\bar{u} = 0$ is specified. In addition, isentropic flow, or $\partial s/\partial n = 0$, and adiabatic flow, $\partial h_0/\partial n = 0$ are specified. The final necessary equation is derived from the normal momentum equation and is $\partial p/\partial n = 0$ (assuming a large local radius of curvature). From these equations pressure, density and total enthalpy, h_0 may be extrapolated from the interior. Setting $\bar{u} = 0$ and evaluating $\hat{f}$ at the boundary results in

$$\hat{f}_b = \begin{pmatrix} 0 \\ c_x p_b \\ c_y p_b \\ c_z p_b \\ 0 \end{pmatrix} \tag{2.9}$$

where p_b is the pressure at the boundary, and $\langle c_x\hat{i}, c_y\hat{j}, c_z\hat{k}\rangle$ is the outward pointing normal to the solid boundary.

Often times the symmetry of a problem can be exploited to reduce the computational effort required in obtaining a solution. Thus the final boundary condition which will be implemented is that of symmetry. For this boundary condition, the flux will be calculated in a manner identical to that of interior flux calculations. With the flux-split algorithms which will be discussed in section 3.2, values of the flow quantities must be interpolated to each face from both sides of that face. At the symmetry boundary, the values of the flow quantities are only known on one side of each face such that the interpolation may only be performed from one side. The flow quantities on the remaining side can be calculated from the first side. Pressure, density and the components of velocity not normal to the face will be

identical to those of the first side. The component of velocity normal to the face will be opposite of the normal component of the first side.

$$\bar{u}_2 = -\bar{u}_1 \tag{2.10}$$

where subscripts 1 and 2 represent interpolations from the left and right states.

3.0 SPATIAL DISCRETIZATION

3.1 Introduction

A finite-volume spatial discretization has been chosen to implement this algorithm. Since it is derived from the integral conservation equations, this discretization has the capability of properly capturing all flow discontinuities. A finite-difference formulation derived from the differential form of the governing equations would not be appropriate since there are no logical computational coordinates, (ξ, η, ζ), as with a structured discretization. A finite-element formulation may be used with an unstructured discretization. However, finite-volume formulations in conjunction with flux-split algorithms have been shown to yield accurate results.

In the finite-volume formulation, the domain is subdivided into a discrete number of control volumes. The variation of the flow quantities is then approximated by some discrete function over each control volume. The integral equations are then solved simultaneously over each individual control volume. The particular functions describing the flow quantities of each control volume are used to evaluate the volume and surface integrals of the governing equations. The choice of these functions will determine the spatial accuracy of the particular implementation.

Typically, the volume average of Q, defined by,

$$\langle Q_i \rangle = \iiint_{V_i} Q \, dV / V_i \tag{3.1}$$

where the subscript i defines the control volume, is maintained for each control volume, and integrated in time. Thus the equation for $\langle Q \rangle$ at each volume becomes

$$\frac{\partial \langle Q_i \rangle}{\partial t} V_i = - \iint_{S_i} \vec{F} \cdot \hat{n} \, dS. \tag{3.2}$$

Since the only values of Q being maintained are the volume averaged quantities, the functions used to describe Q over the entire cell must use $\langle Q \rangle$ as the dependent variable. A further simplification will be to discretize the surface integral such that

$$\iint_{S_i} \vec{F} \cdot \hat{n}\, dS = \sum_j \iint_{S_{i,j}} \vec{F} \cdot \hat{n}\, dS \tag{3.3}$$

where the subscript j denotes the planar faces which surround volume i. The surface integral of each planar face is then approximated using interpolated values of Q.

When discretizing the computational domain for use with a space-marching algorithm, minimal constraints are placed upon the placement and orientation of control volumes which will enable the use of the space-marching algorithm. The volumes should be grouped in computational blocks bound by two cross-flow planes oriented approximately normal to the free-stream direction (figure 3). Figure 3 depicts five discrete blocks of a two-dimensional domain which have been individually discretized with triangular control volumes. Each face on the inflow and outflow cross-flow planes must be approximately normal to the free stream. Thus given the requirements for space-marching, each plane will have an inviscid solution independent of the downstream blocks while not influencing any blocks upstream of it.

With a structured discretization, each block typically has a control volume depth of one in the streamwise direction. Thus each volume of a structured space-marching grid has one face on the inflow of the plane, and one face on the outflow of the plane. This will minimize the number of cells in each block, thus improving the efficiency of the algorithm. However, it is not a requirement that each volume have faces on the inflow and outflow cross-flow planes. In fact, it is possible to have volumes which have no faces on either the inflow or the outflow planes. For a completely generalized indexing strategy, it probably will be necessary to have cells within each block which are completely interior to that block as demonstrated in Figure 3.

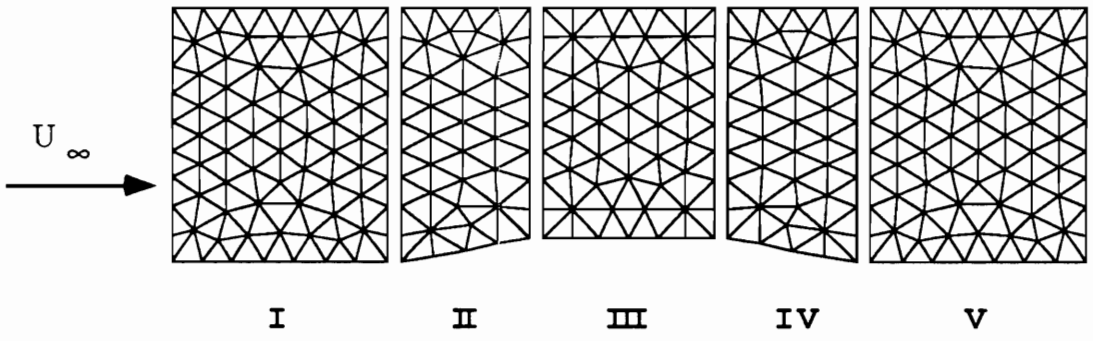


Figure 3: Triangular discretization of a two-dimensional domain suitable for space marching.

3.2 Flux Calculations

3.2.1 Flux-Split Algorithms

As stated in Section 3.1, pointwise approximations to the value of $\hat{f}$ on the faces of each control volume must be made. Due to the hyperbolic nature of the governing system of equations, information about Q at the face, necessary for the flux calculation, must be extrapolated from either side of that face. The Euler equations can be modeled as a system of waves traveling at characteristic speeds and carrying characteristic information. So-called upwind schemes attempt to extrapolate the characteristic information from the direction of these characteristic waves. Unsplit flux calculations which indiscriminately extrapolate information from all directions require the addition of artificial dissipation terms to control numerical instabilities of the solution. Researchers have shown that solutions to the Euler equations using flux-split algorithms yield more accurate results, especially for high speed flows, than unsplit flux calculations. In particular, the flux-difference-splitting (FDS) algorithm attributed to Roe yields excellent results for both inviscid and viscous calculations[53].

Roe developed a flux function which approximates the solution to the exact one-dimensional Riemann problem[24,25,26]. The Riemann problem models the time dependent interaction of two different flow states. The result of this interaction is three characteristic waves; a normal shock, a contact surface, and an expansion fan. Godunov first suggested solving the exact Riemann problem at every face of a finite-volume discretization[17,18]. A left state is extrapolated with flow quantities from the left and a right state is extrapolated with flow quantities from the right. These left and right states are then used as initial values for the Riemann problem. The state at the face just after the characteristic waves have formed is used to calculate

the flux at that face. The exact solution to the Riemann problem requires an iterative technique. Thus a solution algorithm employing an exact Riemann solver is computationally intensive[18]. Roe derived an approximate solution to the one-dimensional Riemann problem which gives explicit information about the flux across an interface after the characteristic waves have interacted. Thus this approach models the propagation of information from the Riemann problem without requiring an iterative solution.

Other flux-vector-split (FVS) algorithms attributed to Steger-Warming[27] and Van Leer[30,31] also yield good solutions to the Euler equations. They too require evaluation of the fluxes at the faces as a function of Q interpolated from the left and right states. However, the author chose to implement Roe's FDS scheme because it has been shown to yield better solutions with the viscous equations[53].

Roe's approximate Riemann solver was originally derived for the one-dimensional problem. If the normal components of the velocity are ignored, the one-dimensional formulation may be easily extended to multiple dimensions. The flux, $\hat{f}(Q_L, Q_R)$, is taken to be that flux normal to the face as described in equation 2.7. The values Q_L and Q_R are reconstructed from the two control volumes adjacent to the face and represent the two Riemann states.

In his derivation, Roe linearized the flux, $\hat{f}$, with respect to Q resulting in the Jacobian matrix

$$A = \frac{\partial \hat{f}}{\partial Q} \tag{3.4}$$

The Jacobian is then evaluated at some derived average of Q such that the Rankine-Hugoniot shock jump conditions are satisfied. Thus A is defined such that

$$\Delta \hat{f} = \tilde{A}(Q_L, Q_R)\Delta Q \tag{3.5}$$

where $\Delta Q = Q_R - Q_L$ and $\Delta F = \hat{f}_R - \hat{f}_L$. If the eigenvalues and eigenvectors of $\tilde{A}$ are found, $\tilde{A}$ may be expressed as

$$\tilde{A}(Q_L, Q_R) = \tilde{S}\tilde{\Lambda}\tilde{S}^{-1} \tag{3.6}$$

where Λ, $\tilde{S}$, and $\tilde{S}^{-1}$ represent the eigenvalues, and left and right eigenvectors respectively. Roe found the Roe averaged conditions to be:

$$\tilde{\rho} = (\rho_R \rho_L)^{1/2} \qquad \tilde{u} = \frac{\rho_L^{1/2} u_L + \rho_R^{1/2} u_R}{\rho_R^{1/2} + \rho_L^{1/2}} \tag{3.7}$$

$$\tilde{v} = \frac{\rho_L^{1/2} v_L + \rho_R^{1/2} v_R}{\rho_L^{1/2} + \rho_R^{1/2}} \qquad \tilde{w} = \frac{\rho_L^{1/2} w_L + \rho_R^{1/2} w_R}{\rho_L^{1/2} + \rho_R^{1/2}}$$

$$\tilde{h}_0 = \frac{\rho_L^{1/2} h_{0L} + \rho_R^{1/2} h_{0R}}{\rho_L^{1/2} + \rho_R^{1/2}}$$

where the subscripts R and L indicate a right and left fluid state, respectively. The Roe-averaged speed of sound ($\tilde{a}$) is computed from the Roe-averaged stagnation enthalpy. Roe uses these average quantities to compute the value of the flux at the interface. Knowledge of the underlying physics of the problem is used to upwind the components of the flux on the basis of the signs of the eigenvalues.

$$\hat{f}(Q_L, Q_R) = \frac{1}{2}\left(\hat{f}_L + \hat{f}_R - \tilde{S}\left|\tilde{\Lambda}\right|\tilde{S}^{-1}\Delta Q\right) \tag{3.8}$$

The term $\tilde{S}\left|\tilde{\Lambda}\right|\tilde{S}^{-1}\Delta Q$ in equation (3.8) may be separated into three components resulting from the three distinct eigenvalues.

$$|\Delta\hat{f}| = |\Delta\hat{f}_1| + |\Delta\hat{f}_2| + |\Delta\hat{f}_3| = \tilde{S}|\tilde{\Lambda}|\tilde{S}^{-1}\Delta Q$$

with

$$|\Delta\hat{f}_1| = \left|\tilde{U}\right| \begin{bmatrix} \Delta\rho - \Delta p/\tilde{a}^2 \\ \tilde{u}\left(\Delta\rho - \Delta p/\tilde{a}^2\right) + \tilde{\rho}\left[\Delta u - c_x\Delta\bar{U}\right] \\ \tilde{v}\left(\Delta\rho - \Delta p/\tilde{a}^2\right) + \tilde{\rho}\left[\Delta v - c_y\Delta\bar{U}\right] \\ \tilde{w}\left(\Delta\rho - \Delta p/\tilde{a}^2\right) + \tilde{\rho}\left[\Delta w - c_z\Delta\bar{U}\right] \\ (\tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2)/2\left(\Delta\rho - \Delta p/\tilde{a}^2\right) + \tilde{\rho}\left[\tilde{u}\Delta u + \tilde{v}\Delta v + \tilde{w}\Delta w - \tilde{U}\Delta\bar{U}\right] \end{bmatrix}$$

$$|\Delta \hat{f}_2| = \left|\tilde{U} + \tilde{a}\right| \left[\frac{\Delta p}{2\tilde{a}^2} + \frac{\tilde{\rho}\Delta\bar{U}}{2\tilde{a}}\right] \begin{bmatrix} 1 \\ \tilde{u} + c_x\tilde{a} \\ \tilde{v} + c_y\tilde{a} \\ \tilde{w} + c_z\tilde{a} \\ \tilde{h}_0 + \tilde{U}\tilde{a} \end{bmatrix} \tag{3.9}$$

$$|\Delta \hat{f}_3| = \left|\tilde{U} - \tilde{a}\right| \left[\frac{\Delta p}{2\tilde{a}^2} - \frac{\tilde{\rho}\Delta\bar{U}}{2\tilde{a}}\right] \begin{bmatrix} 1 \\ \tilde{u} - c_x\tilde{a} \\ \tilde{v} - c_y\tilde{a} \\ \tilde{w} - c_z\tilde{a} \\ \tilde{h}_0 - \tilde{U}\tilde{a} \end{bmatrix}$$

where $\Delta(\cdot) = (\cdot)_R - (\cdot)_L$ and $\Delta\bar{U} = c_x\Delta u + c_y\Delta v + c_z\Delta w$.

3.2.2 Flux Calculation at Planar Boundaries

The flux split algorithms require information about the state variables extrapolated from the left and right. However, no information is available about the state variables downstream of the outflow faces of the current computational block. This information is not necessary since all of the characteristic waves at the outflow faces will be directed downstream. Thus the values of the fluxes at the outflow faces may be calculated with values extrapolated from upstream. In addition, the values of the fluxes for the inflow faces will not depend upon values of Q for the current block. Hence, they may be calculated once from the previous block's solution and will remain fixed throughout the convergence of the current block. These values of $\hat{f}$ may be calculated from the upwind extrapolated values of Q and equation 2.7.

3.3 Decomposition of Domain into Pentahedra

3.3.1 Description of the Control Volume

As discussed in section 1.2, the first discretization which has been investigated attempts to retain as much structure as possible to its numbering strategy while still allowing the geometric flexibility of a generalized indexing strategy. If one considers a structured indexing technique, there are three computational indices,

(i, j, k), which correspond to the three logical directions, (ξ, η, ζ). Thus, element (i, j, k) is located logically in the same (j, k) location of the i plane as element $(i + 1, j, k)$ is in the $i + 1$ plane. With a completely generalized indexing strategy, there is only one computational index, n, and there is no logical direction associated with it. Thus there is nothing in the numbering strategy of n to indicate where element $n = n_1$ is located with respect to element $n = n_2$.

The idea behind the pentahedral discretization is to maintain two logical indices, (i, n), where the first index, i specifies the current computational block, and the second, n, specifies the element within the current block. With this indexing strategy, element (i, n) is located in the same location relative to the other elements within the i block as element $(i + 1, n)$ is located relative to the other elements in the $i + 1$ block. Note that as with elements in a completely generalized indexing strategy, elements in a current block need a connectivity array specifying their position relative to other elements within the current block. However, since element n in block i must be located in the same relative position as is element n in block $i + 1$, the same connectivity array may be used for each block.

The control volumes of each block may be generated by stacking two two-dimensional unstructured grids discretizing adjacent cross-flow planes. This will insure that the faces of the inflow and outflow planes will be properly aligned to allow space marching. A further constraint on the two-dimensional grid generation is that the generator be capable of producing grids with the same number of elements, and the same element connectivity. With this discretization, any two-dimensional generalized indexing strategy may be used. However, for the purpose of this investigation, a grid generator which constructs triangular elements was employed. The grid generation techniques are discussed in section 6.1. Figure 4 depicts a three

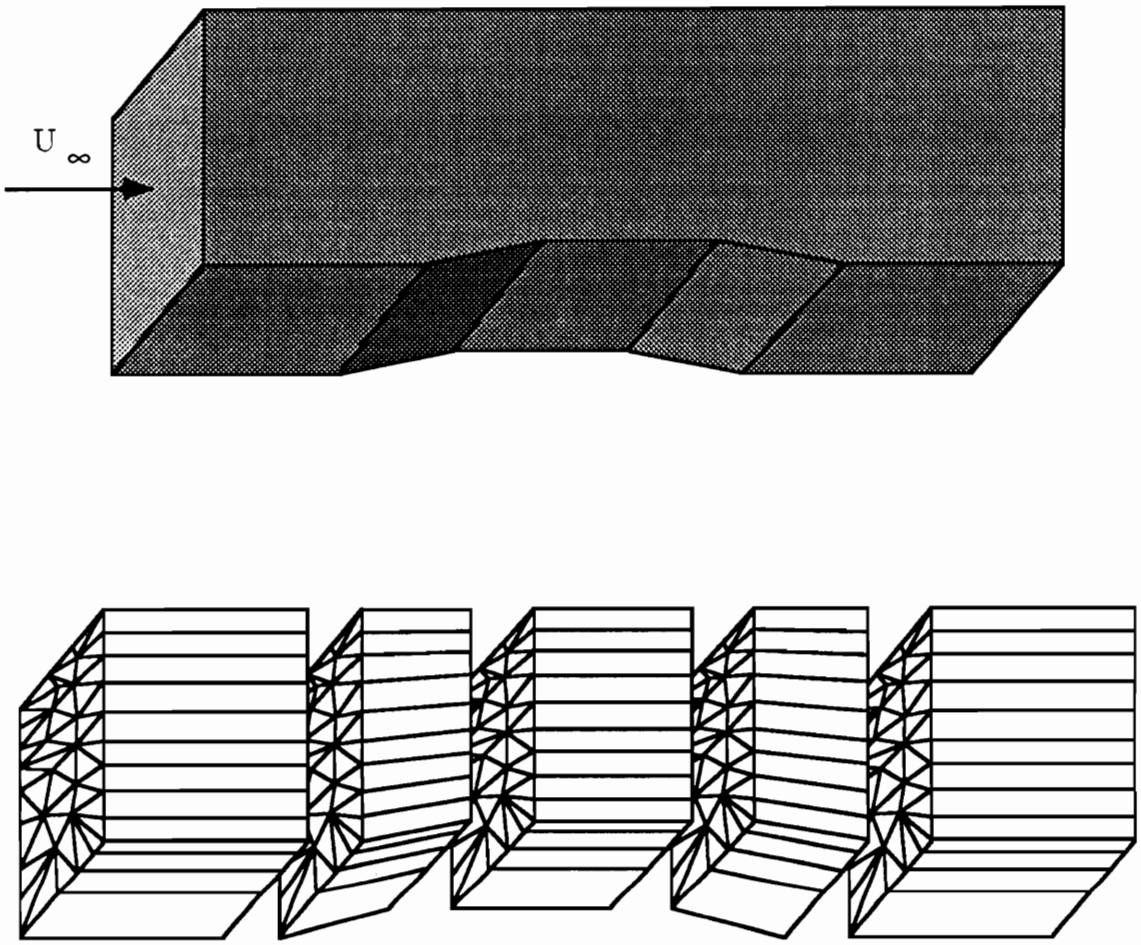


Figure 4: Pentahedral discretization of a three-dimensional nozzle consisting of five computational blocks.

dimensional nozzle subdivided into five computational blocks. Note that the cross-flow planes separating each block have identical connectivity such that pentahedral elements are formed within each block.

With this discretization, each cross-flow plane is essentially a two-dimensional generalized indexing grid with a finite depth. Therefore, discretization techniques used for two-dimensional algorithms may be incorporated within a cross-flow plane. With this fact in mind, a dual mesh sometimes referred to as a cell vertex discretization has been chosen within the cross-flow plane (figure 5) similar to the approach taken by Stoufflet[39]. In the cell vertex method, the vertices of the triangular elements form the approximate centroid of the volumes on the dual mesh. A centroidal dual mesh was chosen for the discretization within a plane. Each control volume is an n-sided polygon where n is the number of triangular elements surrounding each vertex. Each face of the polygon is formed by connecting the centroids of two neighboring triangular elements. The three-dimensional control volume (figure 6) is formed by giving a finite depth to the n-sided polygon. Thus the control volume is enclosed by two distinct types of faces. The first are the n-sided polygons at the inflow and outflow of the cross-flow plane (see figure 6). These faces separate the volumes among neighboring cross-flow planes. The second type of face is that separating the volumes within a single cross-flow plane (see figure 6). These faces are quadrilaterals formed by connecting the coinciding edges of the inflow and outflow faces.

Roe among others has indicated that control volumes consisting of more than three faces in two dimensions will yield more accurate results[54]. This is the motivation behind the use of a dual mesh. However, the complexity of a dual mesh on a completely unstructured three-dimensional mesh may preclude its use as an efficient discretization.

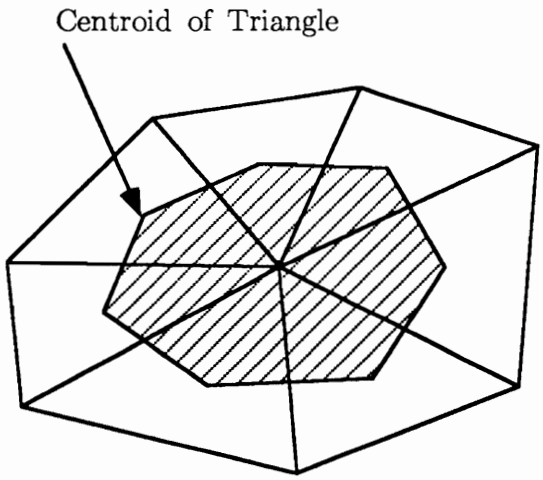


Figure 5: Example of a two-dimensional dual mesh

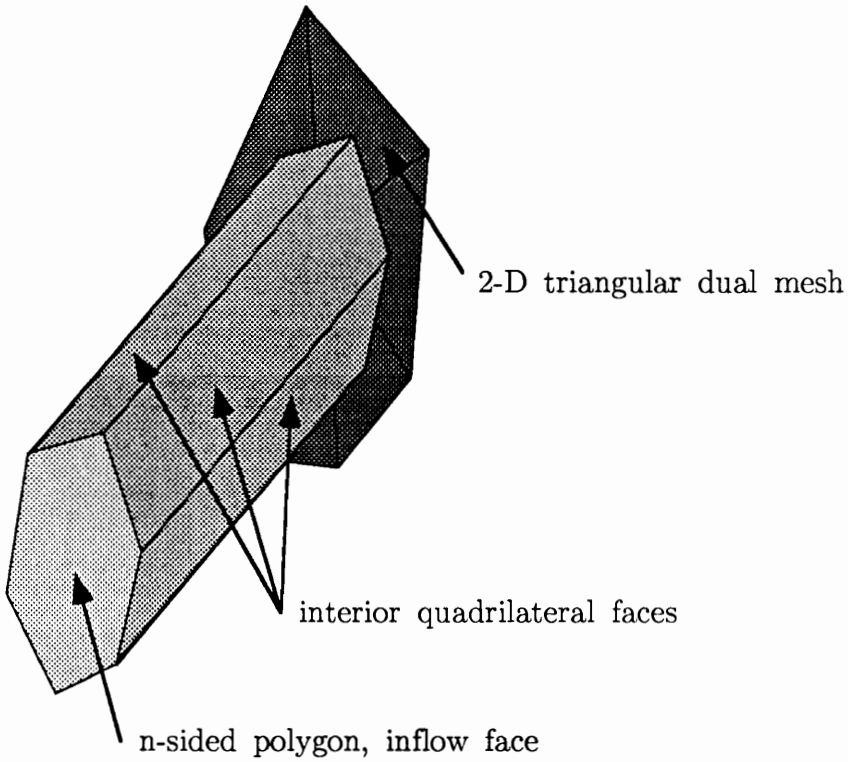


Figure 6: Example of the three-dimensional control volume resulting from stacked two-dimensional dual meshes.

3.3.2 Spatial Accuracy

Spatial accuracy of the discretization will be dependent upon the accuracy with which the integration of the flux is performed. The surface integral will be approximated using Gauss quadrature. For this particular discretization, a one point quadrature was used. Thus the flux calculation must be evaluated at the centroid of the face. Thus the surface integral for a control volume may be approximated by

$$\iint_{S_i} \vec{F} \cdot \hat{n}\, dS \approx \sum_j \hat{f}(Q_j)\, \Delta S_j \tag{3.10}$$

where Q_j is an approximation to the state variables at the centroid of face j extrapolated from neighboring cell averaged values. ΔS_j is the area of face j. For faces interior to each block, values of Q_{j_L} and Q_{j_R} for use with Roe's FDS algorithm may be obtained to first-order accuracy from the volume average values of the two elements adjacent to each face. Since this is a space-marching algorithm, the full, unsplit fluxes are used for the inflow and outflow faces to the block. These fluxes are evaluated using values of Q interpolated from upstream.

The spatial accuracy may be improved by constructing higher-order approximations of $Q_{L,R}$ used in the FDS scheme. The approach taken for this discretization was the use of the *kappa* formulation common among structured algorithms[55],

$$\begin{aligned} Q^-_{i+1/2} &= Q_i + \frac{1}{4}\{(1-\kappa)\bar{\nabla} \;+\; (1+\kappa)\bar{\Delta}\}\, Q_i \\ Q^+_{i+1/2} &= Q_{i+1} + \frac{1}{4}\{(1+\kappa)\bar{\nabla};+\,(1-\kappa)\bar{\Delta}\}\, Q_{i+1}. \end{aligned} \tag{3.11}$$

The $+$ and $-$ superscripts denote right and left biased interpolations respectively. κ controls the biasing and order of the interpolation and varies between $[-1, 1]$. This formulation includes the min-mod limiter where

$$\begin{aligned} \bar{\Delta} Q &= \text{minmod}(\Delta Q, \beta \nabla Q) \\ \bar{\nabla} Q &= \text{minmod}(\nabla Q, \beta \Delta Q) \end{aligned} \tag{3.12}$$

and

$$\text{minmod}(x, y) = \text{sgn}(x) * \max(0, \min[x\,\text{sgn}(y), y\,\text{sgn}(x)]) \tag{3.13}$$

with

$$\beta = (3 - \kappa)/(1 - \kappa) \tag{3.14}$$

The Δ and ∇ operators represent forward and backward differences respectively. The use of a limiter reduces the order of accuracy of the interpolation to first-order near regions of large gradients in order to suppress oscillations near discontinuities.

This method assumes a one-dimensional uniform distribution of points which lends itself well to structured discretizations. With the present discretization, it would only be possible to directly implement this formulation in the streamwise direction. A fully upwind, second-order interpolation ($\kappa = -1$) is performed in the streamwise direction as dictated by the marching algorithm. Note that the assumption must be made that pointwise values of Q at the cell centers are equivalent to $\langle Q \rangle$. Higher-order interpolations in the computational block are constructed in a manner similar to the two-dimensional formulation of Whitaker [56]. The interpolations are performed along vectors passing through the vertices directly to either side of the current face labeled points P_{n1} and P_{n2} in figure 7 .

Forward and backward differences are needed along this vector for the *kappa* formulation. The difference $Q_{n1'} - Q_{n1}$ is constructed by dotting the two-dimensional gradient of Q with the vector $\overrightarrow{P_{n1}P_{n1'}}$. The point $P_{n1'}$ is constructed such that

$$\overrightarrow{P_{n1}P_{n1'}} = \overrightarrow{P_{n2}P_{n1}}. \tag{3.15}$$

The gradient of Q is obtained by using Gauss's Theorem and integrating about the triangular cell containing the point P_{n1} and the vector $\overrightarrow{P_{n1}P_{n1'}}$. The difference $Q_{n2'} - Q_{n2}$ is derived in a similar fashion to that of $Q_{n1'} - Q_{n1}$.

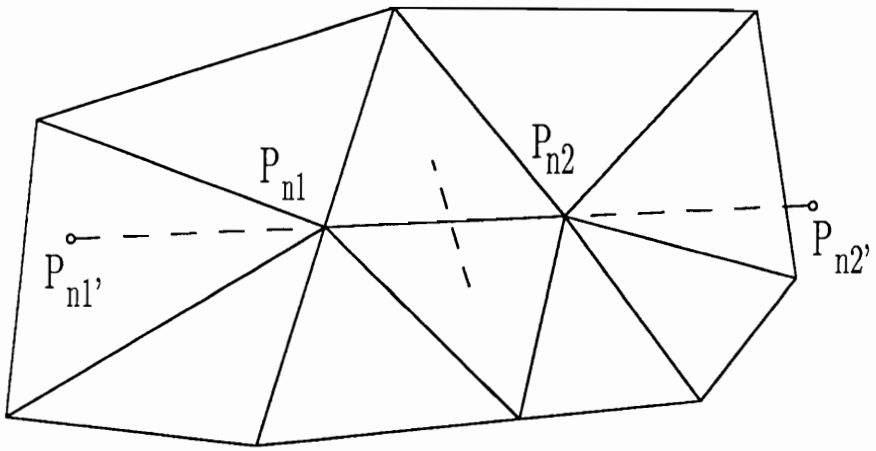


Figure 7: One-dimensional extrapolation of values within a cross-flow plane .

3.4 Decomposition of Domain into Tetrahedra

3.4.1 Description of the Control Volume

The second discretization attempts to achieve the opposite goal of the pentahedral discretization. This method retains as little structure as possible while still enabling the use of a space-marching algorithm. The discretization reduces to the requirement of planar boundaries at the inflow and outflow of each computational block, as in the two-dimensional grid of figure 3. This requirement still allows a completely generalized spatial discretization within the computational block bound by two adjacent cross-flow planes. Thus, there is really no limit to the types of control volumes which could be supported within a single plane. Several possible convex polyhedrons are presented in figure 8. Any of these control volumes may be supported, provided, of course, there is a grid generation algorithm available to construct them.

Note that there are no longer two logical indices describing each control volume. It would be possible to refer to a volume by the indices (i, n), where again i denotes the current block, and n denotes the element within block i. However, it is no longer possible to assume that element n_1 within block i_1 has the same orientation with respect to other elements within that block as does element n_1 within block i_2. It appears more logical to refer to all of the elements with a single index, n. The elements should then be ordered such that all of the elements of block one appear first, all of the elements of block two after the elements of block one, etc..

Although any number of different polyhedrons may be selected for this type discretization, the availability of grid generation techniques will most likely dictate the selection of the control volume. The most widely available three-dimensional

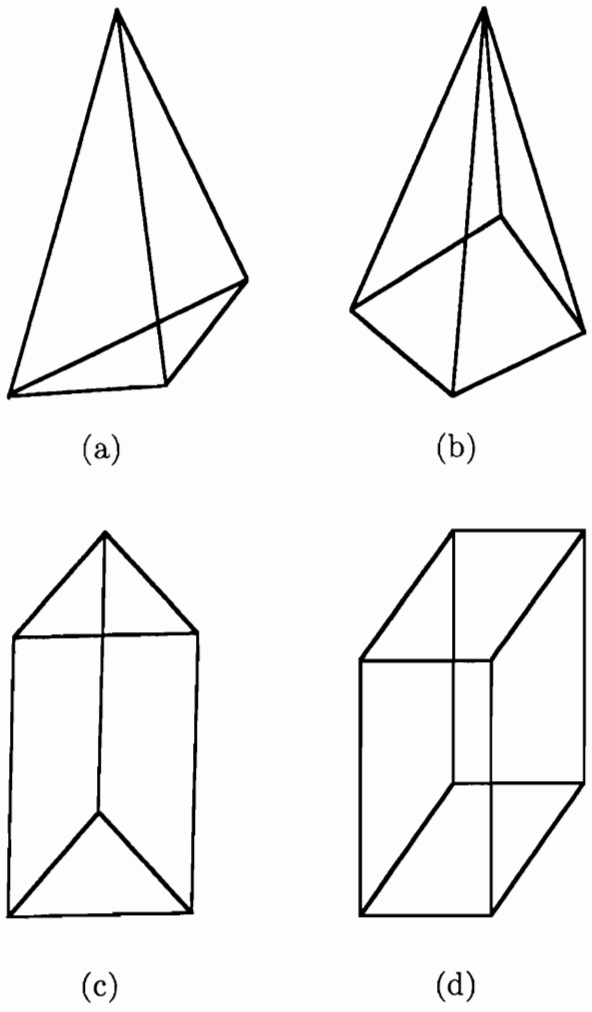


Figure 8: Simple three-dimensional convex polyhedra suitable for use as control volumes.

grid generation algorithms discretize a domain into tetrahedra (the first control volume in figure 8). Thus, the second discretization of this investigation employs the tetrahedron as its control volume. The dual mesh described in section 3.3.1 was not chosen for the three-dimensional discretization. The centroids of surrounding tetrahedra cannot be guaranteed to be coplanar. Thus, the centroidal dual described for two dimensions in section 3.3.1 cannot be extended to three-dimensional polygonal faces. Two other dual meshes might be chosen. The first, a median dual results in five to ten times the number of faces as compared to a tetrahedral mesh. Thus this discretization would be prohibitively expensive in terms of memory and execution time. The second dual mesh might be the Voronoi polygons resulting from the Delaunay triangulation of the grid points. While this discretization would result in the same number of faces as the tetrahedral mesh, the underlying mesh must be a Delaunay triangulation, a stipulation which cannot be guaranteed. It is the author's belief that a discretizaton of tetrahedral control volumes will be sufficient. Figure 9 depicts five computational blocks discretizing a three-dimensional nozzle. The inflow and outflow boundaries of each block are planar and each block is discretized with tetrahedral control volumes.

3.4.2 Spatial Accuracy

Spatial accuracy of the discretization will be dependent upon the accuracy with which the integration of the flux is performed. One-dimensional reconstructions will not result in true second-order or higher accurate solutions. Barth has formulated a multi-dimensional reconstruction of cell average data which results in true second-order and higher spatial accuracy[43]. Thus, this method of reconstruction is recommended for use with the current discretization. A multi-dimensional

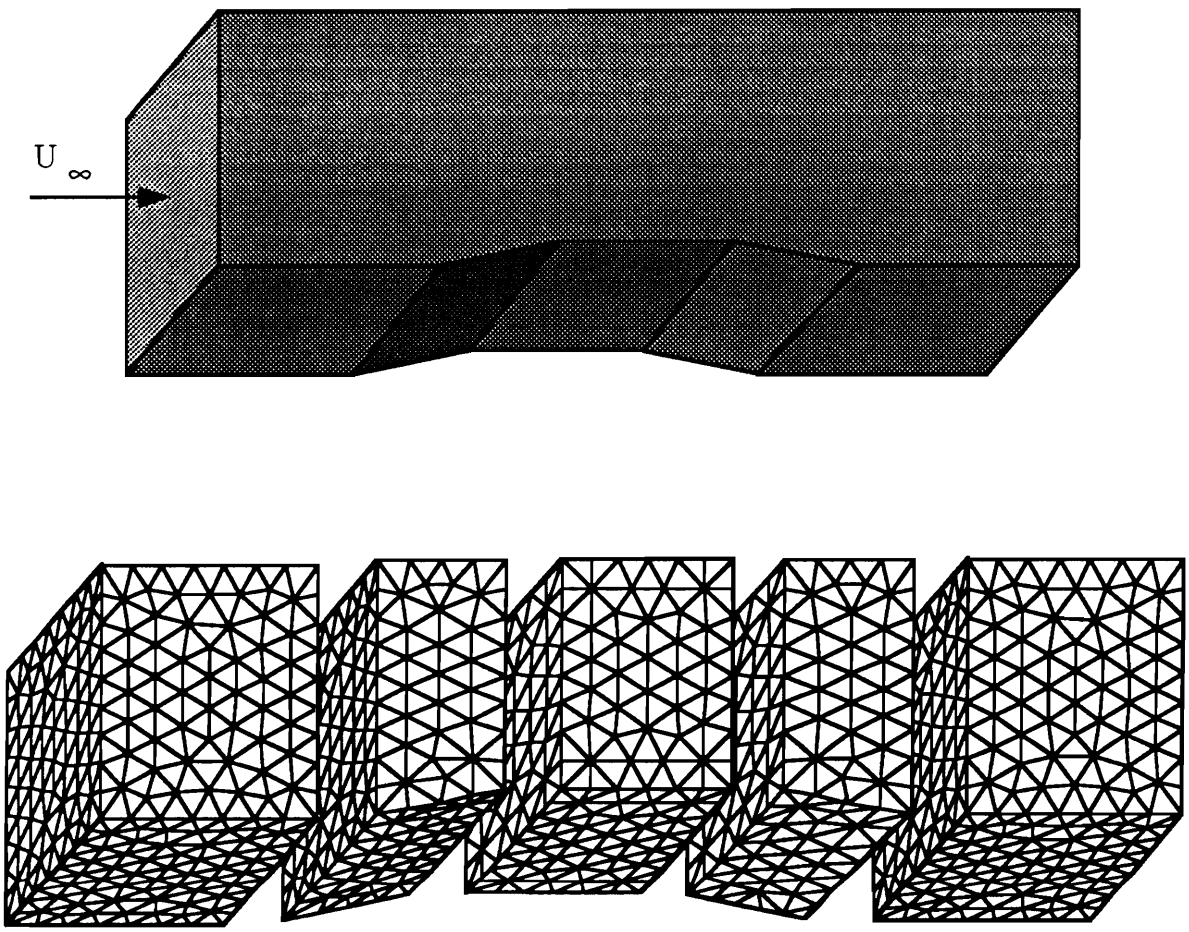


Figure 9: Tetrahedral discretization of a three-dimensional nozzle consisting of five computational blocks.

polynomial is constructed such that the integration of the polynomial over the tetrahedra recovers the volume average of that volume. A three-dimensional polynomial of degree K may be expressed as

$$P^K(x,y,z) = \sum_{i=0}^{K} \sum_{j=0}^{K-i} \sum_{k=0}^{K-j} a_{i,j,k} x^i y^j z^k \tag{3.16}$$

Thus, the coefficients of P^K must satisfy the following equation

$$\langle Q_i \rangle V_i = \iiint_{V_i} P^K(x,y,z)\, dV \tag{3.17}$$

There will be $(K+1)(K+2)(K+3)/6$ coefficients to a polynomial of degree K Thus $(K+1)(K+2)(K+3)/6 - 1$ additional equations must be specified in order to completely define the coefficients of P. Additional equations may be obtained by specifying neighboring tetrahedral volumes over which the polynomial must satisfy the mean of Q. Barth proposed a means for maintaining monotonicity for a linear reconstruction near regions of strong gradients which is described in reference 39.

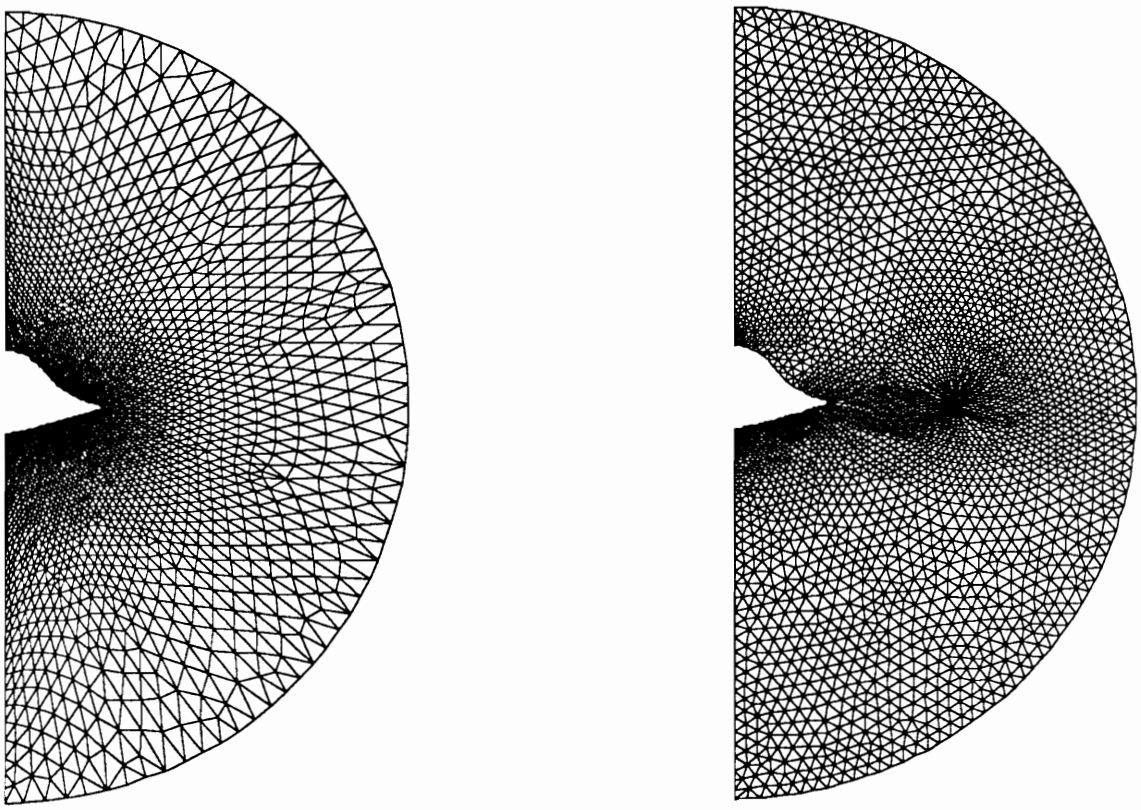
With the coefficients of the polynomial determined, values of Q may be reconstructed at all points of the faces. In order to maintain the same spatial accuracy of the solution, the integration of each triangular face must be performed to the same degree of accuracy of the reconstruction polynomial. Thus as the degree of K is increased, so must the degree of the quadrature performed in integrating each face.

4.0 ZONAL INTERPOLATION

Zonal discretization techniques have become popular for implementing three-dimensional structured algorithms with complex geometries[35,5]. The idea behind zonal techniques is to subdivide complex domains into a set of subdomains which may be simply discretized with a single structured mesh. Control volume faces at zonal boundaries may not have a one to one mapping from one neighboring zone to the next. Various techniques have been developed for structured algorithms which enable the mapping of the fluxes across the zonal interface from one zone to another.

The pentahedral discretization has some of the same geometric constraints that are inherent in a structured discretization. While this discretization allows virtually unlimited flexibility in the placement of nodes within a computational block, the method suffers from the same stiff requirement in the streamwise direction of structured discretizations. As with a structured discretization it will be very difficult if not impossible to discretize a domain which varies radically in the streamwise direction. However, even with well behaved domains it will often be necessary to modify the distribution of elements from one block to the next.

The need for use of multiple zones is demonstrated in figure 10 where the two grids shown discretize the same cross-flow plane of the model SR71 described later. This station is located just forward of the engine inlet diffuser. The grid in figure 10(a) is an appropriate discretization for a cross-flow plane of the forebody of this aircraft. The grid in figure 10(b) contains a clustering of points near the beginning of the diffusers. This clustering will be necessary to resolve the gradients which will occur at that location. It is clear from this picture that the faces of one zone do not coincide with those of the other, and that a method for constructing the fluxes at the inflow of the second zone is required.



(a) (b)

Figure 10: Two-dimensional triangular grids at a zonal interface

An interpolation procedure that is efficient to program and is conservative to within the truncation error of the discretization scheme has been chosen. The approach is to interpolate to find values of Q^2 at blocks i and $i-1$ where block i is the last block of zone one (superscripts indicate the zone to which the values refer). These values will be necessary to calculate a fully upwind interpolation to the inflow faces of block $i+1$. Note that only values of Q^1 are known at blocks i and $i-1$. The two-dimensional grids at the zonal interface are used to obtain the necessary interpolation coefficients. For example, values of Q associated with the faces of the grid in figure 10(a) will be used to interpolate to the values of Q at the faces of the grid in figure 10(b).

An assumption has been made that the value of Q at the triangle vertices is the same as $\langle Q \rangle$ for the cell vertex control volume. Therefore a technique for interpolating values from the vertices of one mesh to the vertices of another mesh is required. A quad-tree search algorithm is employed to find the triangular element from zone one which contains the vertex P_n^2 in zone two (figure 11). A linear interpolation of Q_{n1}^1, Q_{n2}^1 and Q_{n3}^1 between the vertices, P_{n1}^1, P_{n2}^1 and P_{n3}^1, is used to determine the value of Q_n^2 at P_n^2. An efficient search algorithm has been sought in order to reduce the cost associated with the zonal interpolation. Quad-tree technology[57] has been employed to rapidly locate the elements from zone 2 which are closest to the vertex for which a value is being interpolated. A description of the quad-tree structure and associated search algorithms may be found in Appendix B. This search algorithm greatly improves the interpolation performance as compared to a brute force type search algorithm.

Note that the underlying finite-volume algorithm is limited to at most second-order spatial accuracy. The linear interpolation of Q from one mesh to another is second-order accurate and thus within the truncation error of the solution algorithm.

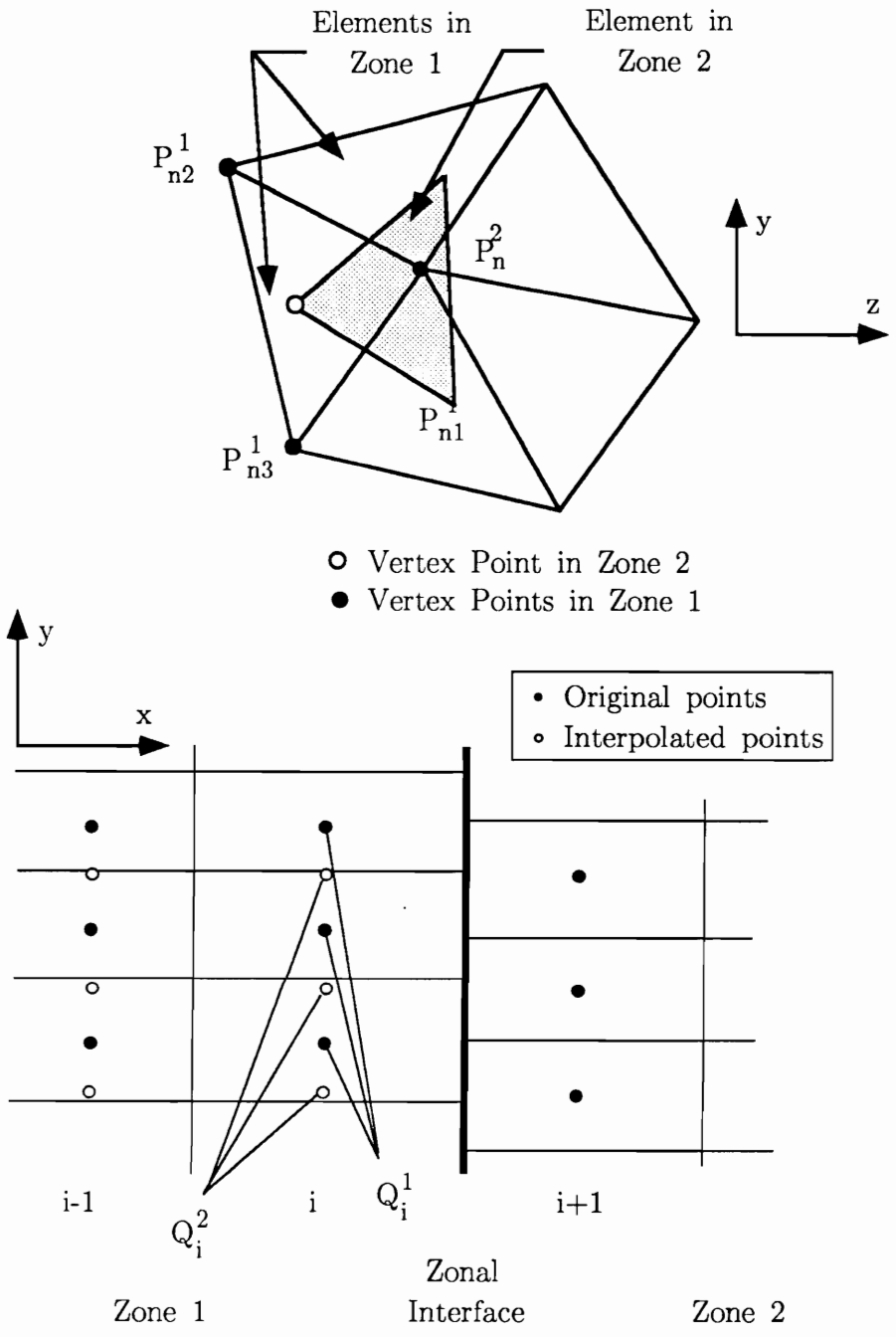


Figure 11: Interpolation from nodes in zone 1 to nodes in zone 2

If the spatial accuracy of the finite-volume algorithm were increased, it would be necessary to increase the accuracy of the zonal interpolation in order to maintain global accuracy of the solution

5.0 TIME INTEGRATION

5.1 Space-Marching Algorithm.

A time-dependent space-marching algorithm is used in the integration to the steady-state solution.[1,2,3] With this formulation, a single computational block is integrated in time. Since the characteristics do not travel opposite to the marching direction, the solution for the current block does not impose itself upon the solution of the upstream blocks. Equivalently, the current block is not effected by the solution downstream of the marching direction. This means that the individual blocks may be integrated in time separately. The flux terms on the right hand side due to the upstream terms will be homogeneous terms to the equations, and there will be no contribution from downstream blocks.

One advantage of the time dependent formulation is that spatial accuracy in the streamwise direction is limited only by the method of extrapolation of flow quantities from upstream. In addition, the classical method for space marching is derived from the time independent differential form of the governing equations. As has been previously discussed, generalized discretizations have little meaning when applied to differential equations. Finally, employing the time dependent form of the integral equations enables the use of any of the convergence acceleration techniques common to global, non space-marching iteration strategies.

5.2 Explicit Time Integration

A four-stage Runge-Kutta type scheme[58,15] has been chosen for the time integration of the individual blocks, i.e.,

$$\begin{aligned} Q_m^{(0)} &= Q_m^n \\ Q_m^{(1)} &= Q_m^{(0)} - \alpha_1 \Delta t R_m^{(0)} \\ Q_m^{(2)} &= Q_m^{(0)} - \alpha_2 \Delta t R_m^{(1)} \\ Q_m^{(3)} &= Q_m^{(0)} - \alpha_3 \Delta t R_m^{(2)} \\ Q_m^{(4)} &= Q_m^{(0)} - \alpha_4 \Delta t R_m^{(3)} \\ Q_m^{n+1} &= Q_m^n \end{aligned} \tag{5.1}$$

where

$$R_m^{(0)} = \sum^{j=nf_{int}} \iint_{S_j} \hat{f}(Q^{(0)})\, dS + \iint_{S_{out}} \hat{f}(Q^{(0)})\, dS + \iint_{S_{inf}} \hat{f}(Q)\, dS \tag{5.2}$$

With this notation, m represents the current cell and varies over all of the volumes in the current computational block. The term nf_{int} represents all faces interior to the current block which surround cell m. The subscript *out* represents any face lying on the outflow cross-flow plane. All of these fluxes are functions of Q in the current block at initial time (0). They also may be a function of the steady-state solution of the previous blocks. The subscript inf represents any face lying on the inflow cross-flow plane. All fluxes on the inflow plane will be functions of the steady-state solutions of the previous blocks. Thus, they need only be calculated once at the beginning of the iteration process. Non standard weighting coefficients experimentally chosen to yield faster convergence rates[59] are $\alpha_1 = 0.170$, $\alpha_2 = 0.273$, $\alpha_3 = 0.500$, $\alpha_4 = 1.000$. This Runge-Kutta type scheme has the advantage of being very simple and efficient to implement. In addition, this

strategy only requires the storage of two values of Q for each cell as opposed to the required storage of five values for the classic Runge-Kutta. If only the current blocks are maintained in memory, this integration strategy requires minimal storage, allowing low memory computers to solve large three-dimensional problems.

An attempt has been made to accelerate convergence with the explicit time integration through the use of local time stepping. The time step for each individual cell is calculated such that a constant Courant number is maintained over the entire domain. The time step, Δt will vary over the entire domain as a function of the local element size and flow conditions. A simple Von Neumann stability analysis for the one-dimensional Euler equations reveals the stability to be a function of the Courant number,

$$CFL = \frac{(|u| + a)\Delta t}{l} \tag{5.3}$$

where l is some characteristic length. This equation can be extended to cover three-dimensional volumes with the following equation

$$CFL = \frac{((|u| + a)A_x + (|v| + a)A_y + (|w| + a)A_z)\,\Delta t}{V} \tag{5.4}$$

Where A_x, A_y, A_z represent the projected areas of the volume V in the x, y, z coordinate directions respectively. Equation 5.4 can be solved for Δt as a function of the local geometry and flow conditions. The local time step is periodically recalculated to account for changing flow conditions.

5.3 Implicit Time Integration

While explicit time integration strategies are useful for their ease of implementation, slow convergence rates can destroy benefits obtained from employing a space-marching algorithm. This can become critical when attempting to solve the PNS equations. While the emphasis of this investigation was not placed upon time

integration strategies, it would seem impractical to attempt space-marching algorithms without knowledge of practical implicit time integration techniques. The following paragraphs describe implicit techniques which have been successfully implemented in two-dimensional global integration algorithms[41]. Since a time dependent space-marching algorithm has been employed, any of these global iteration techniques may be implemented.

The implicit algorithms discussed below are derived from the Euler implicit time integration algorithm expressed in delta form as

$$V\frac{\Delta Q}{\Delta t} = R^{N+1} \tag{5.5}$$

where $\Delta Q = Q^{N+1} - Q^N$. This nonlinear system of equations may be linearized and approximated as

$$A\Delta Q = R^N \tag{5.6}$$

where

$$A = (I\frac{V}{\Delta t} - \frac{\partial R^N}{\partial Q}) \tag{5.7}$$

If inversion of the complete matrix A is performed, quadratic convergence may be obtained. However, the exact linearization of A when Roe's FDS algorithm is employed is very costly to construct. Often an approximate linearization is constructed for use in A. When used with the relaxation techniques to be discussed, this approximation is often sufficient. However, quadratic convergence cannot be obtained when direct inversion of A is attempted.

On a randomly numbered unstructured mesh, the A matrix resulting from the linearization of the governing equations will be sparsely populated and have a variable bandwidth. It will be crucial to employ renumbering strategies such as reverse

Cuthill-McKee[60] in order to minimize the bandwidth and fill of these sparsely populated matrices. In a comparison of relaxation techniques in two-dimensions, a hybrid scheme using a combination of Runge-Kutta explicit time integration and LU decomposition combined with freezing of the LU proved most time efficient. Such an iteration strategy would be feasible for iteration of a single block in a three-dimensional problem.

In addition, the Point Jacobi and point Gauss-Seidel relaxation strategies may be directly implemented. These methods require no structure to the left hand side matrix. Their implementation is identical to that of a structured discretization. If one parses the left hand side matrix A into the sum of a block diagonal term D, a lower block triangular term, L, and an upper block diagonal term U such that

$$A = L + D + U \tag{5.8}$$

then the Jacobi iteration strategy is equivalent to solving the following system of equations.

$$D\Delta Q = R^N \tag{5.9}$$

As with the structured discretization, the Jacobi relaxation algorithm benefits from complete vectorization over the entire list of cells. The point Gauss-Seidel technique is equivalent to solving the following system of equations

$$(L + D)\Delta Q = R^N \tag{5.10}$$

The solution of this system of equations will not completely vectorize and consequently the Jacobi iteration strategy may converge faster to the steady state.

Finally, if a different cell numbering strategy is employed which attempts to place elements within a block tridiagonal matrix within the left hand matrix, a

relaxation technique similar to line Gauss-Seidel may be employed. With this strategy, the domain is subdivided into a set of blocks with many elements clustered along the tridiagonal. Each subdomain may be solved in a Gauss-Seidel sense, including the tridiagonal terms.

These relaxation strategies represent a nonexhaustive list of techniques for accelerating convergence in an implicit sense. While other techniques are available, these methods have been proven effective for two-dimensional global iteration algorithms. Note that the blocks of a three-dimensional space-marching algorithm are likely to contain the same order of magnitude of points as the two-dimensional problems.

6.0 GRID GENERATION

6.1 Generation of Pentahedra

The generation of the unstructured two-dimensional grids in the cross-flow planes is essential to the success of this algorithm. No currently available unstructured generator is capable of discretizing two unique two-dimensional domains such that the resulting grids will contain the same number of nodes and elements, and the same connectivity. With this in mind, a technique has been developed to take an existing two-dimensional unstructured grid about one domain and map its nodes such that the resulting grid discretizes a second domain. The mapping of points must be accomplished while maintaining the same number of elements and connectivity.

6.1.1 Hybrid Two-Dimensional Unstructured Algorithm

If the geometry does not vary radically between two domains, it is possible to generate a grid at the second plane by forcing the boundary points of the original grid to align themselves with the boundary of the second plane. In this way, the resulting grid has the same number of nodes as the first while maintaining the original connectivity. An example of such an initial grid is shown in figure 12(a) . This is a coarse grid generated about a cross-flow plane of the model SR71 problem and is suitable only for demonstration. This plane includes the mid section of the fuselage, the leading edge of the delta wing, and a section of the engine inlet. Figures 12(b) and 10(c) denote the two steps required to generate a second two-dimensional grid about a cross-flow plane forward of the first grid shown in figure 12(a). The second plane also includes the mid section of the fuselage, leading edge of the wing

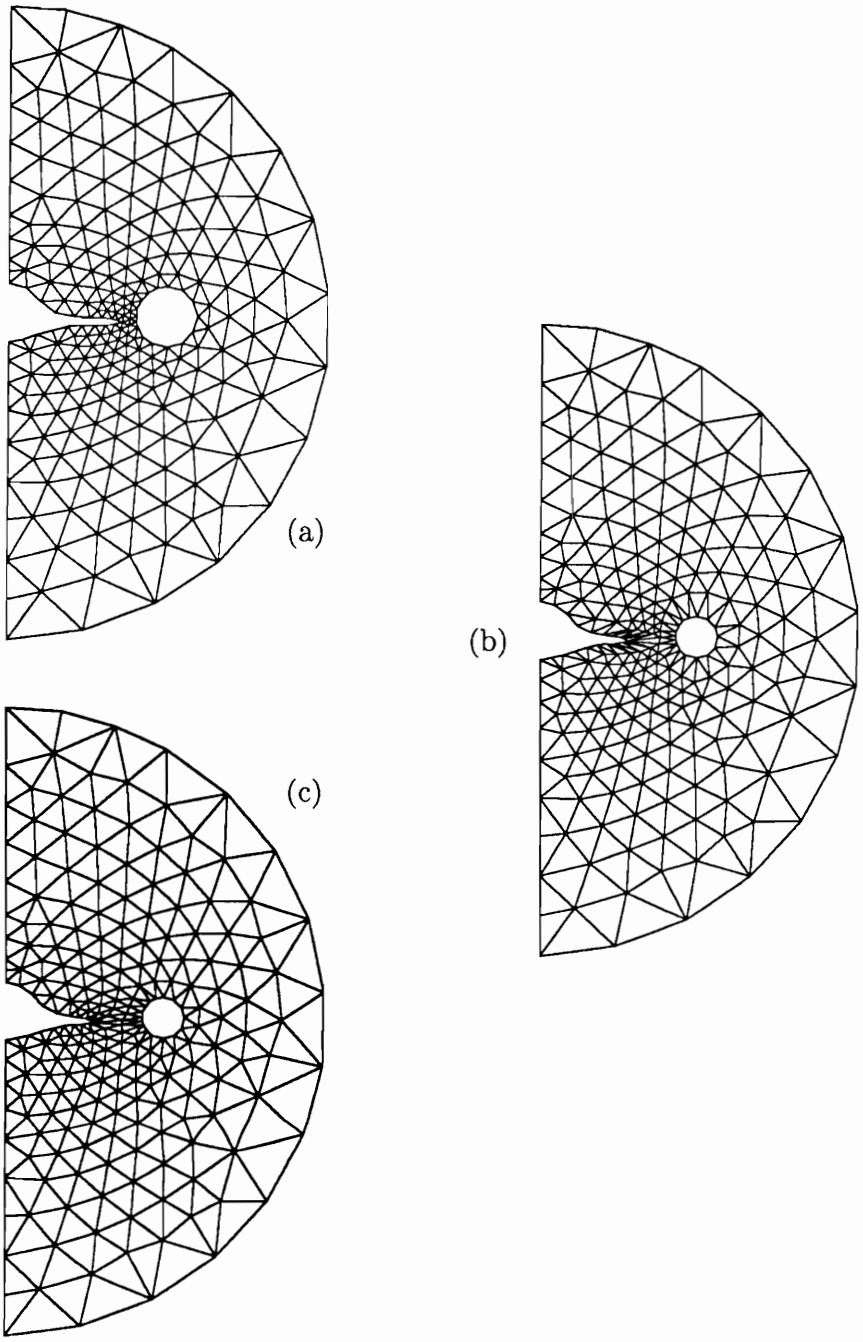


Figure 12: Construction of two-dimensional grids within a single zone

and inlet. However, the span of the wing is less than that of the first station, and the radius of the inlet is less at this location than the first.

The first step shown in figure 12(b) is to redistribute all of the boundary points of the grid in (a) such that they all fall on the boundary of the second plane. This step is accomplished while ensuring that the same number of points are maintained on each individual segment of the boundary. Thus the same number of points are kept on the top surface of the body-wing, as well as the top symmetry line, etc. In addition, the distribution is kept the same along each boundary segment. For instance, if there were clustering near the tip of the wing, this clustering would be maintained from one grid to the next. Examining figure 12(b), it is apparent that distortion of the triangular elements has occurred along the body and inlet. In order to reduce this distortion, the resulting grid is smoothed with a filtering algorithm. With this smoothing algorithm, the grid points will be more uniformly distributed such that they better describe the geometry of the current plane (figure 12(c)). The smoothing algorithm attempts to place each point at the center of the polygon formed by connecting all of the surrounding triangular elements (figure 13). Note that all nodes are redistributed with an explicit algorithm. The explicit algorithm is typically iterated upon five to ten times. With each iteration, the movement of each node is restricted to the bounding polygon. If this algorithm were iterated upon until convergence any prescribed clustering will be reduced. For example, clustering near boundaries for viscous calculations would be reduced if the smoothing algorithm were iterated upon to convergence. It is only desirable to reduce any local distortion caused by the redistribution of boundary nodes.

With the above described algorithm, a single unstructured grid can be generated for a single zone. This grid can then be used as a template for the remaining two-dimensional grids of that zone. Some observations about the selection of zone

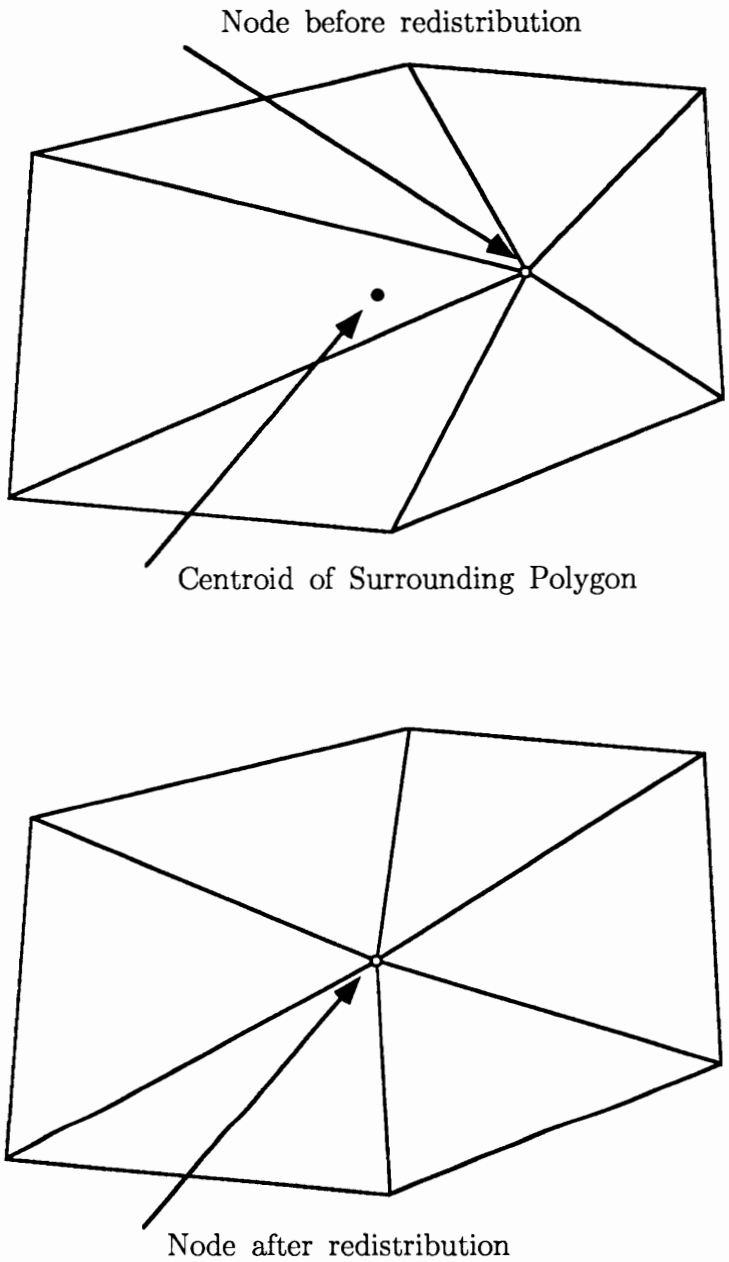
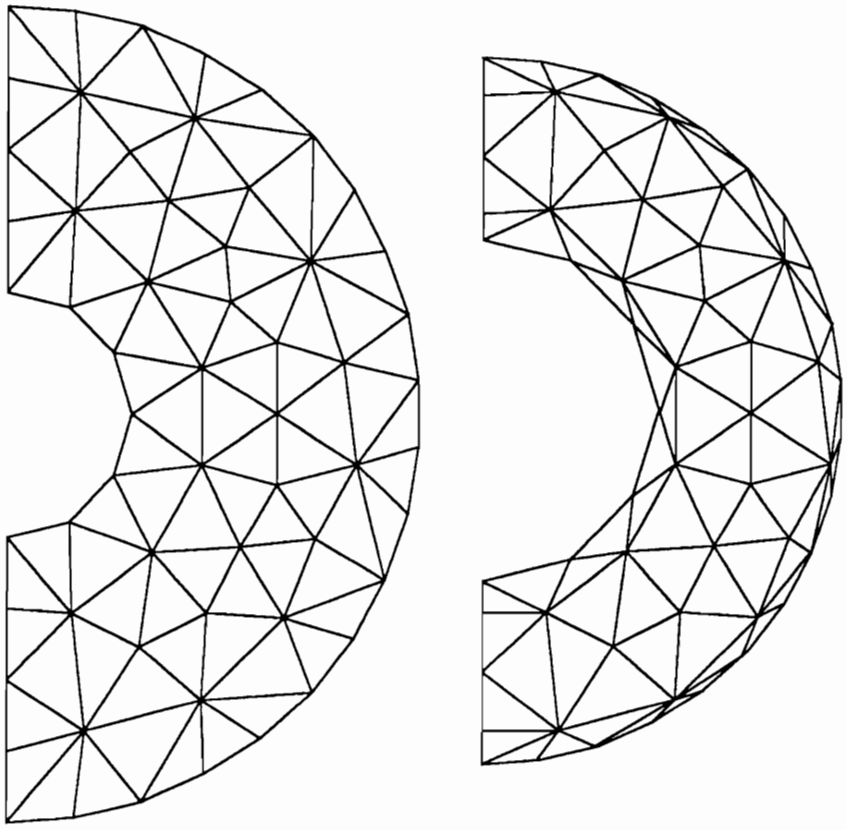


Figure 13: Demonstration of smoothing algorithm for pentahedral grid generation.

boundaries need to be considered. If the number of segments required to describe the boundaries of a plane change in the streamwise direction, a zonal boundary is necessary. For instance, a zonal boundary is required for the model SR71 at the vertex of the engine inlet, where at the upwind cross-flow plane, there is no evidence of the inlet, and across the zonal interface, the plane describes the vertex of the inlet. In addition, if the geometry has changed drastically between the initial grid and a candidate plane, a zonal boundary is appropriate. Consider a discretization of an aircraft with a delta wing. It would not be appropriate to use the same connectivity to describe a fuselage wing combination at the leading edge and a cross-flow plane near the trailing edge of the aircraft. In fact, for this discretization, several zones may be necessary to discretize a wing section.

The zonal interpolation described in Section 4 requires the discretization of the same plane with the connectivities of the two neighboring zones. For instance, if the new zone is required because of a change in the number of boundary segments used to describe the domain, the plane at the zonal interface must be describable with the number of boundary segments of the first zone and likewise the second zone.

Finally, experience has indicated that more suitable discretizations will result if boundary elements are extended, or elongated in the distortion of elements from the initial plane to the current cross section. This is opposed to the compression of elements. Boundary elements will be compressed when the area of the final domain being discretized is smaller than the area of the initial discretization and the boundary nodes of the final grid are moved into the original domain. Figure 14 represents an example of the case where the final domain area is smaller than the initial discretization area. The boundary elements of the resulting grid are distorted such that the area of each control volume is reduced. In this situation,



(a) (b)

Figure 14: Example of a poor choice for the parent two-dimensional grid for the pentahedral grid generation.

the smoothing algorithm does not redistribute interior nodes as well as it does when the boundary nodes are moved away from the initial domain. Thus, within a particular zone, careful consideration must be made as to the selection of the initial discretization plane.

6.1.2 Generation of Parent Two-Dimensional Grid

With the grid generation process described in section 6.1.1, a parent two-dimensional unstructured grid must be generated for each zone of a particular domain. There are a multitude of two-dimensional unstructured algorithms available to the which provide great flexibility when discretizing two-dimensional domains[46–52]. The effort in two-dimensions is much more mature than that in three-dimensions. Two very different algorithms seem to claim the most acceptance in the CFD community; Delaunay triangulation, and the advancing front algorithm.

Delaunay triangulation is a technique for specifying the connectivity of a given set of points such that the resulting triangular elements satisfy a set of well defined conditions. The triangulation has an associated dual mesh as described in section 3.3.1. However, the polygons of the dual mesh are not described by the centroidal dual mesh as discussed in that section. Instead, Voronoi polygons are described by the locus of points which are nearer to the node associated with that polygon than any other node. The lines which connect nodes of neighboring Voronoi polygons form the edges of the triangles of the underlying Delaunay triangulation (figure 15). The advantage of this triangulation is that the resulting triangular elements are as equiangular as is possible. This property is analogous to an orthogonal structured mesh. The disadvantage to this method of triangulation is that a separate algorithm must be used to specify the placement of nodes.

The advancing front algorithm generates node distribution and element connectivity simultaneously. Thus it represents a complete grid generation technique.

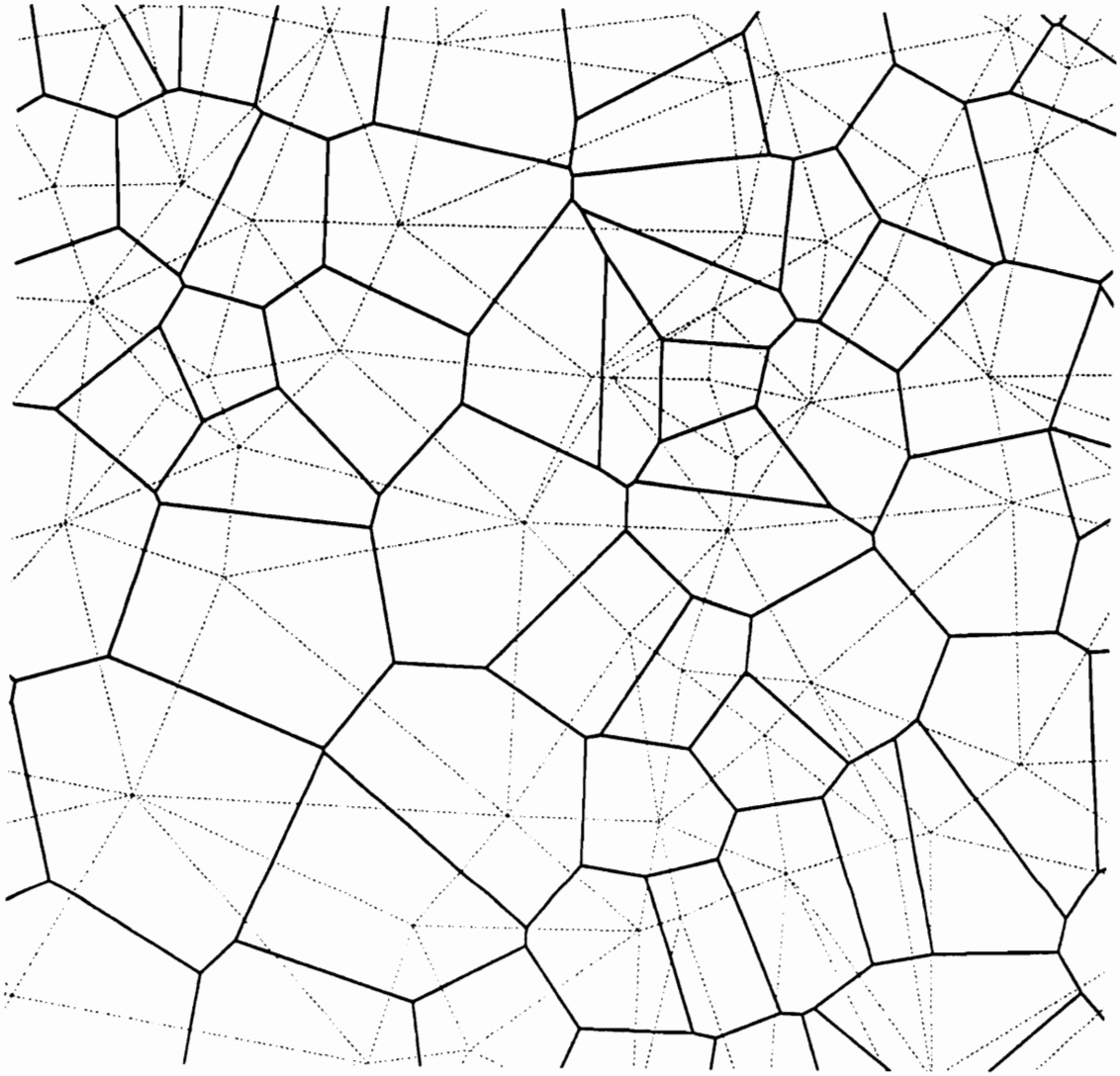


Figure 15: Voronoi polygons associated with a Delaunay triangulation.

A description of the two-dimensional algorithm is given in Appendix A. Basically, this algorithm generates nodes and elements in a scalar fashion. A background grid is maintained which has desired element size and orientation parameters at discrete points. These discrete values are then used to interpolate over the entire domain. Thus a new element is formed attempting to maintain the size and orientation at that point in the domain as specified by the background grid. The use of the background grid provides an enormous degree of flexibility in specifying element size and orientation. This capability enables the simple discretization of complex multiple bodied domains. In addition, the background grid is ideally suited to solution adaptive grid generation.

It is the authors belief that the advancing front algorithm holds the most promise for effective implementation on unstructured discretizations. Thus this discretization technique has been implemented as the grid generation for the parent plane of each zone within the three-dimensional domain. However, the Delaunay triangulation algorithm might also be implemented for use with this three-dimensional grid generation process.

6.1.3 Conversion of Structured Grid

Of great interest in this investigation is how well the unstructured discretizations compare to structured algorithms. While the unstructured discretizations enable the solution on more complex domains than structured algorithms, the algorithm ought to provide solution quality equivalent to structured algorithms on problems with simple domains. It is thus desirable to perform calculations on grids which have similar element and node distribution to structured grids. In this way a comparison may be made as to the solution quality of a similar structured algorithm.

Since the pentahedral elements are formed by stacking two-dimensional grids, all that is necessary to convert a structured three-dimensional grid to a structured

pentahedral grid is to convert the two-dimensional grids denoting the cross-flow planes to triangular discretizations. This may be accomplished by splitting each quadrilateral into two triangles. The result is that each hexahedral structured element becomes two pentahedral elements (figure 16). Note that in order to reduce grid induced error, the diagonal is alternated uniformly over the two-dimensional domain (figure 17).

6.2 Generation of Tetrahedra

As discussed in section 3.4.1, the motivation for the tetrahedral discretization is to allow the most freedom in geometric flexibility. Thus a fully generalized unstructured grid generator should be employed between the space-marching blocks. Ideally, the grid generation should be capable of discretizing the domain with a minimum number of elements in the streamwise direction. If possible the grid generation should introduce no nodes interior to the block. This is analogous to a structured space-marching grid where each cross-flow plane is iterated upon independently. As with the two-dimensional grid generation techniques, the methods of Delaunay triangulation and the advancing front algorithm seem to offer the greatest promise for meeting the requirements of three-dimensional unstructured grid generation in the field of CFD.

6.2.1 Advancing Front Grid Generation

For the reasons discussed in section 6.1.2, the advancing front algorithm is thought to offer the most benefits for three-dimensional unstructured grid generation. However, the three-dimensional algorithm is not at the same level of development as is the two-dimensional algorithm. While in principle, the three-dimensional algorithm is a direct extension from two-dimensions, several problems arise pertaining to the connectivity of tetrahedra which do not occur with the connectivity of

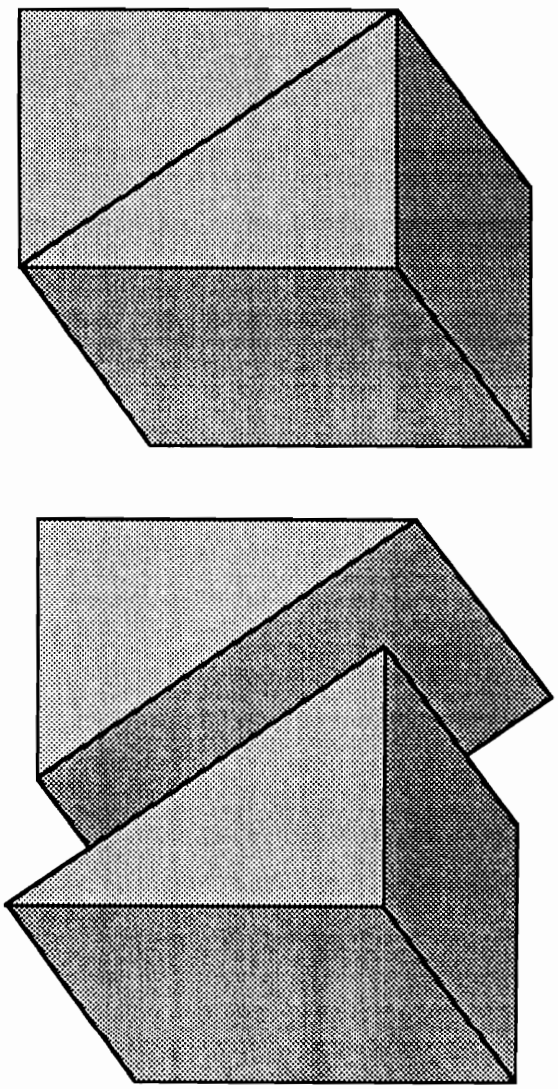


Figure 16: Discretization of hexahedral elements of a stuctured grid into the pentahedral elements for use with pentahedral discretization.

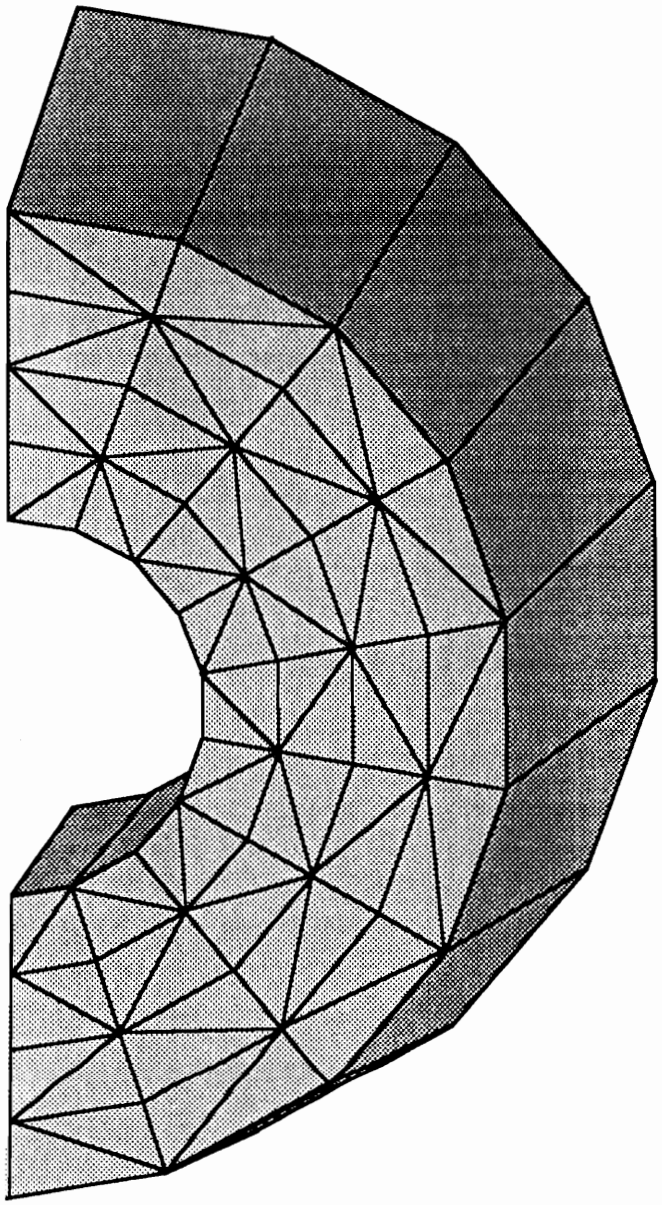


Figure 17: Conversion of a structured cross-flow plane to pentahedral elements.

triangles. At the current state of development, the three-dimensional algorithm is not as robust as the two-dimensional algorithm.

In order to pursue the use of the advancing front algorithm in three-dimensions, the two-dimensional algorithm described in Appendix A has been extended to three-dimensions. As discussed in section 6.2, optimally, the grid generation should introduce no new nodes into the interior section. In three-dimensions, the advancing front algorithm begins by discretizing the boundaries of the domain. With this completed, tetrahedral elements are introduced into the domain. If no interior points are to be added to the discretization, the algorithm must attempt to form elements using only nodes on the boundary of the block. However, the advancing front algorithm attempts to introduce a new node to the domain if it cannot find a suitable existing element with which to form a tetrahedra. An attempt was made to modify the advancing front algorithm such that it would not introduce a new node to the interior. Instead the selection criteria for the fourth node is reduced until an element can be formed. This attempt was not successful, and the modified algorithm was abandoned.

Successful attempts at discretizing a single computational block with minimal interior nodes have been made. At the current state of development, the three-dimensional grid generator is capable of discretizing blocks with approximately two levels of tetrahedra in the streamwise direction. Further improvements to the algorithm should enable the discretization of a block such that most tetrahedra span that block in the streamwise direction.

The three-dimensional algorithm in its current state of development can discretize a single block domain. Considerable work must be done to enable the advancing front algorithm to generate multiple block grids suitable for space marching. While these modifications are not conceptually difficult, there implementation

is tedious. In order to maintain geometric flexibility, the description of the domain should not be required to maintain information about the zonal interfaces. Thus provisions must be made to calculate the intersection of the domain with the inflow and outflow planes of a block. Each block must be discretized individually. However, the node and cell numbers of each block must be sequential. Connectivity information at the block interfaces must be passed from one block to the next. Thus, while the complete algorithm has not been implemented, there is no reason to believe that it will not be successful.

6.2.2 Delaunay Triangulation

Three-dimensional Delaunay triangulation algorithms are currently available[49] and seem to be more robust than current advancing front algorithms. Thus, this method of discretization may be more suitable for immediate use. As with the two-dimensional algorithm, this algorithm requires the separate generation of elements. A technique similar to the pentahedral grid generation could be employed. Two unstructured grids of differing connectivity could be generated at the inflow and outflow cross-flow planes of the current block. A three-dimensional Delaunay triangulation algorithm could then be used to discretize the nodes generated on the inflow and outflow planes. Again, the choice of a two-dimensional grid generation technique will depend upon user preference. The choice of the same node generation technique as used for the two-dimensional Delaunay triangulation would seem appropriate since no node connectivity is required for the two-dimensional grid. However, use of a scheme such as the advancing front algorithm has its advantages.

Solution adaptive grid generation has proven very effective in two-dimensions[45]. Its practical extension to three-dimensions promises dramatic improvements in solution quality. Delaunay triangulation algorithms, while efficient and robust, provide no simple means for solution adaptive node generation. Thus, if an advancing front

algorithm were used to generate node distribution on the inflow and outflow faces, some degree of solution adaptive capability might be obtained. Note that even with node points distributed optimally in a solution adaptive sense, the element distribution will be optimal in the Delaunay sense, which is not necessarily the same criteria.

6.2.3 Conversion of Structured Grid

As discussed in section 6.2.2, it is of interest to compare solutions on unstructured discretizations to structured discretizations with similar grid point distribution. Thus a means for subdividing the hexahedra of structured discretizations into the tetrahedra required for the tetrahedral discretization. On any given hexahedra, five tetrahedra is the minimum number necessary to describe the same volume (figure 18). In addition the diagonals on the exterior of each subdivided hexahedra must be swapped so that tetrahedra of adjacent hexahedra align themselves (figure 19). This alternating of diagonals must occur in each of the logical directions over the structured mesh. Note that for a given structured discretization, the corresponding tetrahedral mesh will have five times the number of control volumes.

In addition to a comparison of structured and unstructured algorithms, there is a need to easily generate grids on which to verify the unstructured discretization. No truly unstructured grid generation technique is currently available for the tetrahedral algorithm which is capable of discretizing multiple blocks for the space-marching algorithm. However, when using tetrahedral meshes derived from structured discretizations, the solution algorithm has absolutely no knowledge of the structure of the grid. As far as the solution algorithm is concerned, the grid is no different from any other tetrahedral discretization.

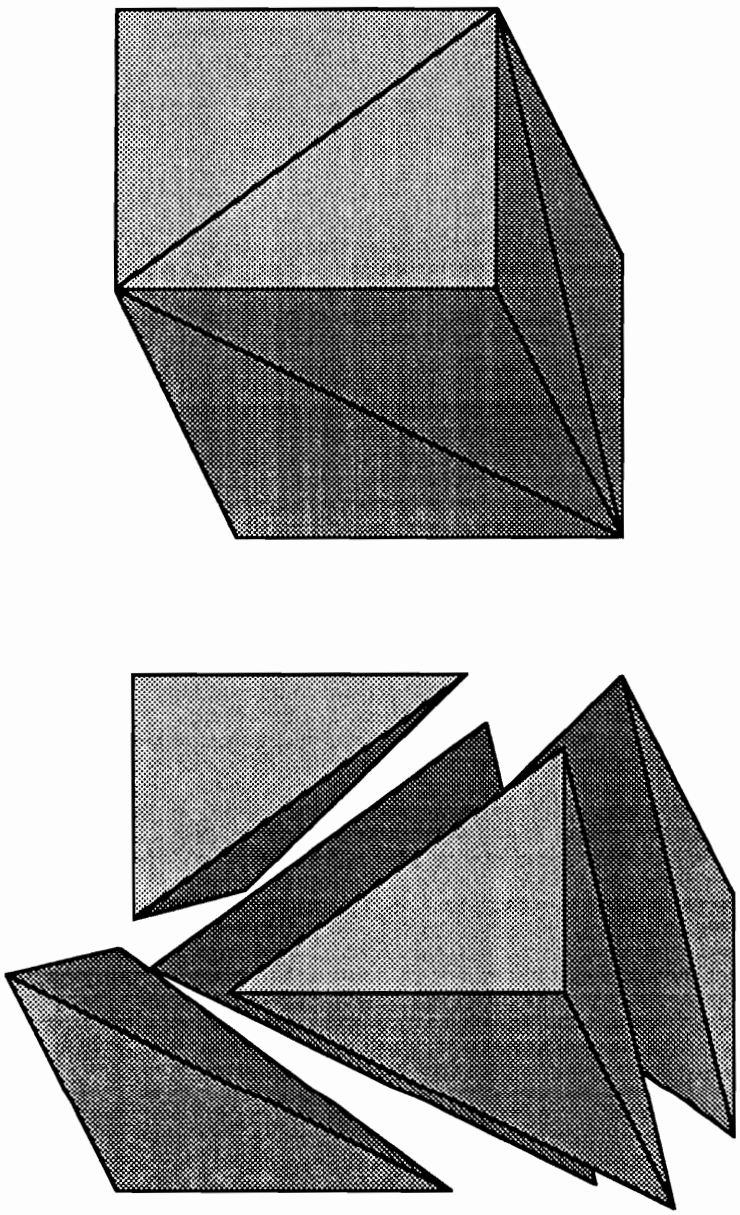


Figure 18: Discretization of hexahedral elements of a structured grid into the tetrahedral elements for use with a tetrahedral discretization.

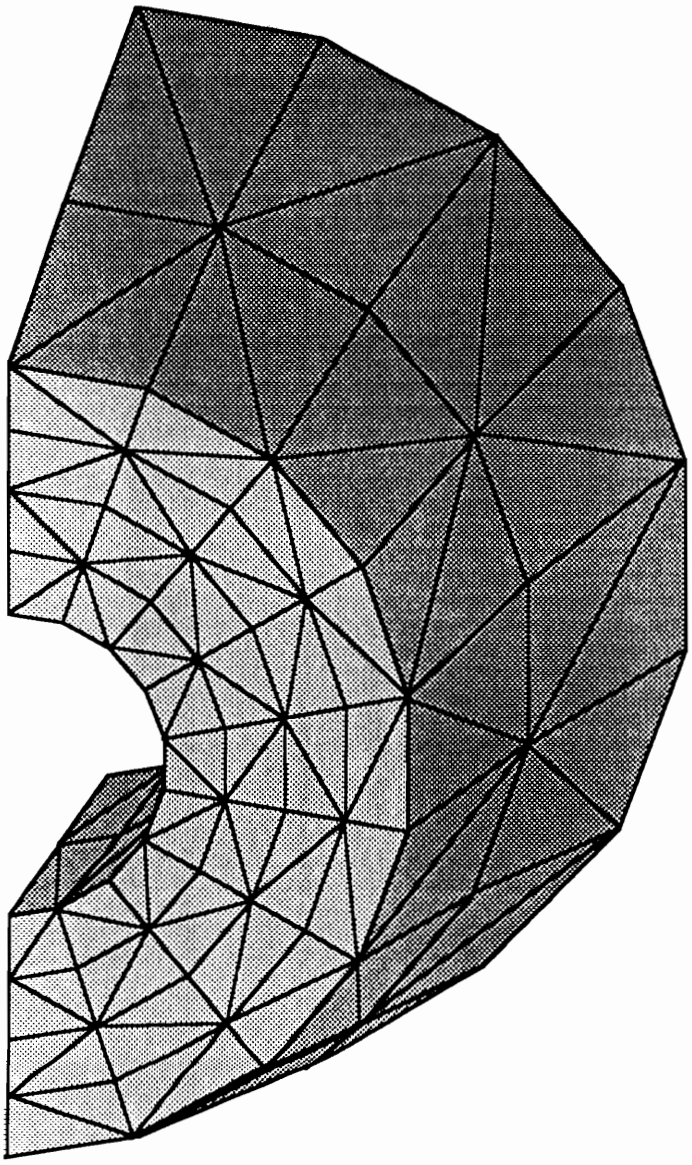


Figure 19: Conversion of structured cross-flow planes to tetrahedral elements.

7.0 COMPUTATIONAL RESULTS

Solutions have been obtained using the two proposed discretizations for three test cases. These test cases include the supersonic flow about a circular cone for which an analytic solution is known, flow about an analytic forebody for which experimental data is available, and flow about a model SR71 aircraft. These test cases combine to show the solution accuracy of the discretizations, and the geometric flexibility of the discretizations.

7.1 Five Degree Cone

The first test case is the flow about a 5° half angle cone in a Mach 5 free stream. This is a simple problem not meant to test the geometric flexibility but to verify each solution algorithm. The solution to the ordinary differential equation for the high speed flow about a cone following the work of Taylor and Maccoll[61] is presented for comparison.

For the pentahedral algorithm, a structured grid containing 40 by 40 by 1 elements was converted to a pentahedral mesh using the technique described in section 6.1.3. The resulting pentahedral mesh contained a single block of 3200 elements (figure 20). A value of $\kappa = 1/3$ is used in addition to the min-mod limiter for the linear reconstruction of flow quantities to the interior volume faces, resulting in a higher-order upwind biased interpolation in the cross-flow plane.

Analysis of inviscid flow over a cone proves that flow quantities are a constant along all lines emanating from the vertex of the cone. If the structured grid is constructed such that all lines through equivalent points in the inflow and outflow planes pass through the vertex of the cone, a conical boundary condition can be

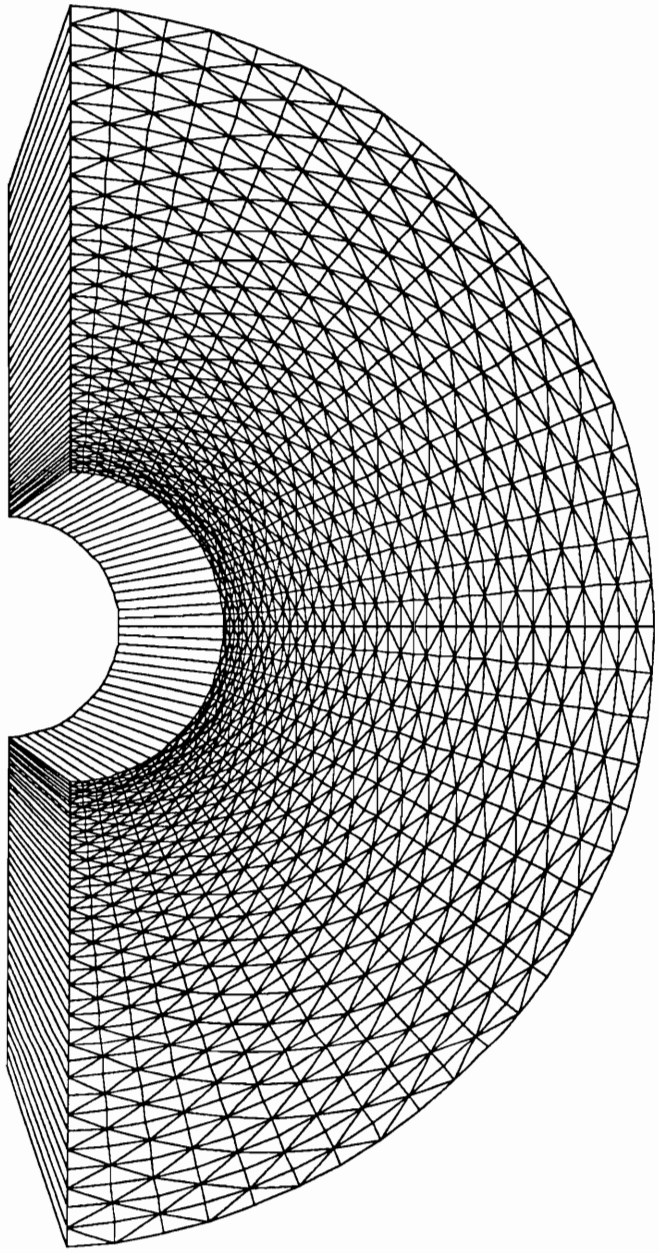


Figure 20 Cross-flow plane, pentahedral grid for 5° cone.

used to construct the flux at the inflow faces. Note that while the flux per unit area for the inflow and outflow faces will be identical, the areas of these faces will be different. Thus it is not necessary to march in space in order to get a solution to the cone flow. Flow quantities at other locations in the streamwise direction may be found by nondimensionalizing the radial distance of a point from the centerline of the cone. Flow quantities for a cone will be a function of the angle, ϕ, formed between the cone centerline and the line from the cone vertex and the point in question.

A 40 by 2 by 2 structured grid was used to generate the grid used for the tetrahedral mesh where the first dimension is the radial direction (figure 21). The resulting tetrahedral mesh contained a single block of 800 cells. There were effectively two elements in the streamwise direction. This structured grid was constructed to allow the periodic boundary condition necessary for conical flow. However, two elements were required in the streamwise direction to insure that the faces from the inflow and outflow planes would properly align. The solution for the tetrahedral discretization is limited at this time to first-order spatial accuracy.

Figure 22 shows a comparison of normalized pressure along a radial line emanating from the cone axis. Solutions from both tetrahedral and pentahedral discretizations are presented along with the exact, analytic solution. The figure shows good agreement between the numerical and analytical solutions. The computed solutions have correctly captured the oblique shock as well as predicted the correct value of pressure on the surface of the cone.

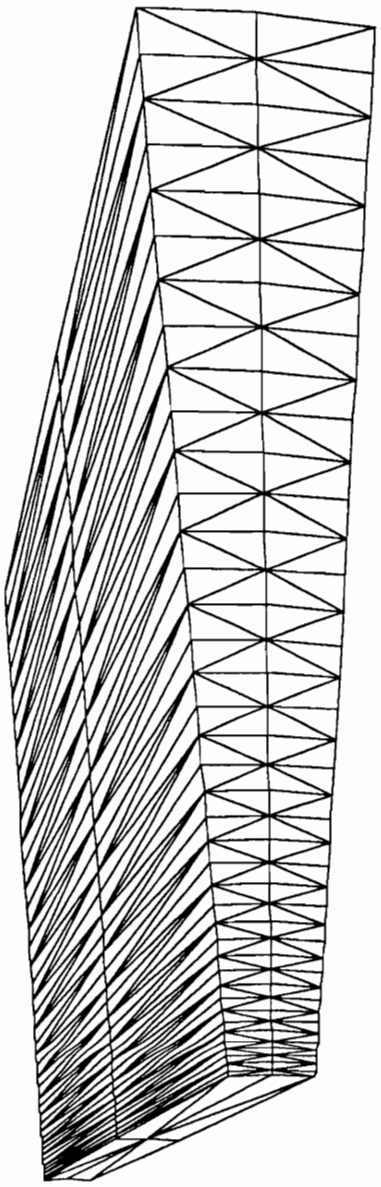


Figure 21: Cross flow plane, tetrahedral grid for 5° cone.

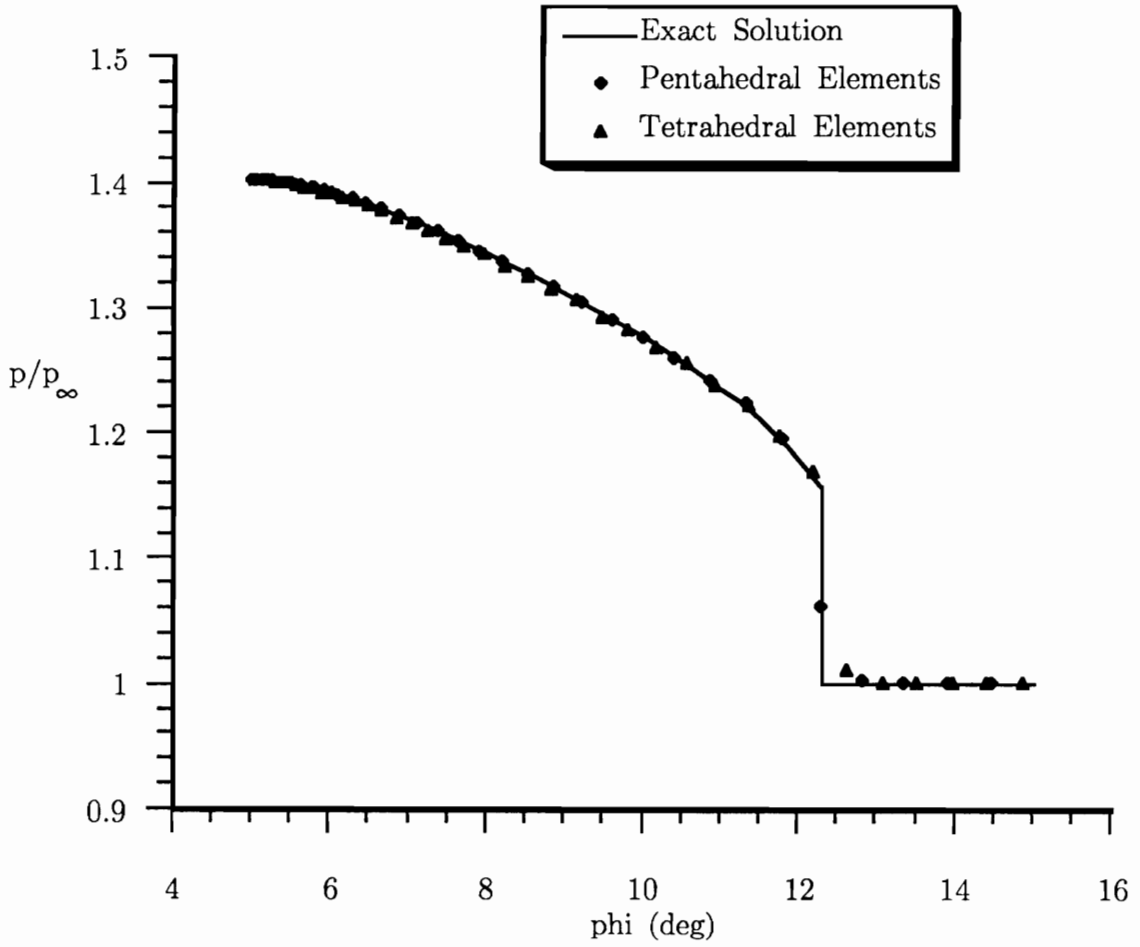


Figure 22: Pressure along radial line from 5° cone at Mach 5.

7.2 Analytic Forebody

A second test case is the flow about an analytic forebody which represents the cockpit region of a high-performance aircraft. Experimental data obtained from wind tunnel tests is available over a wide range of Mach numbers and angles of attack[62]. Flight conditions of Mach 1.7, 0° angle of attack and Reynolds number of three million based upon the body length were chosen for comparison with numerical calculations.

Two distinct pentahedral discretizations have been used to compute the solution about the analytic forebody. In both cases, a fully upwind second-order interpolation has been used in the free-stream direction with no limiting. A value of $\kappa = 1/3$ and MinMod limiting were used in the cross flow plane. The first discretization contains a single zone of forty computational blocks with 1800 cells in each block. This grid resulted from the conversion of a 30 by 30 by 40 structured grid. In addition, a two zone grid has been constructed employing the advancing front grid generation technique discussed in section 6.1. The first zone consists of 22 blocks with 1330 cells per block, while the second zone consists of 18 blocks with 1797 cells per block. The second zone is necessary to ensure sufficient points near the surface of the aft portion of the forebody. Figure 23 shows pressure contours about the symmetry plane and last cross-flow plane for the two zone unstructured discretization. In addition, the grid has been superimposed on the surface of the body and the last cross-flow plane. Note that the zonal interpolation yields smooth variation in contours from one zone to the next.

A single tetrahedral discretization was also used to compute the solution about the analytic forebody. First-order spatial accuracy was maintained. This discretization contained 40 blocks consisting of 4500 elements per block. Again, a 30 by 30

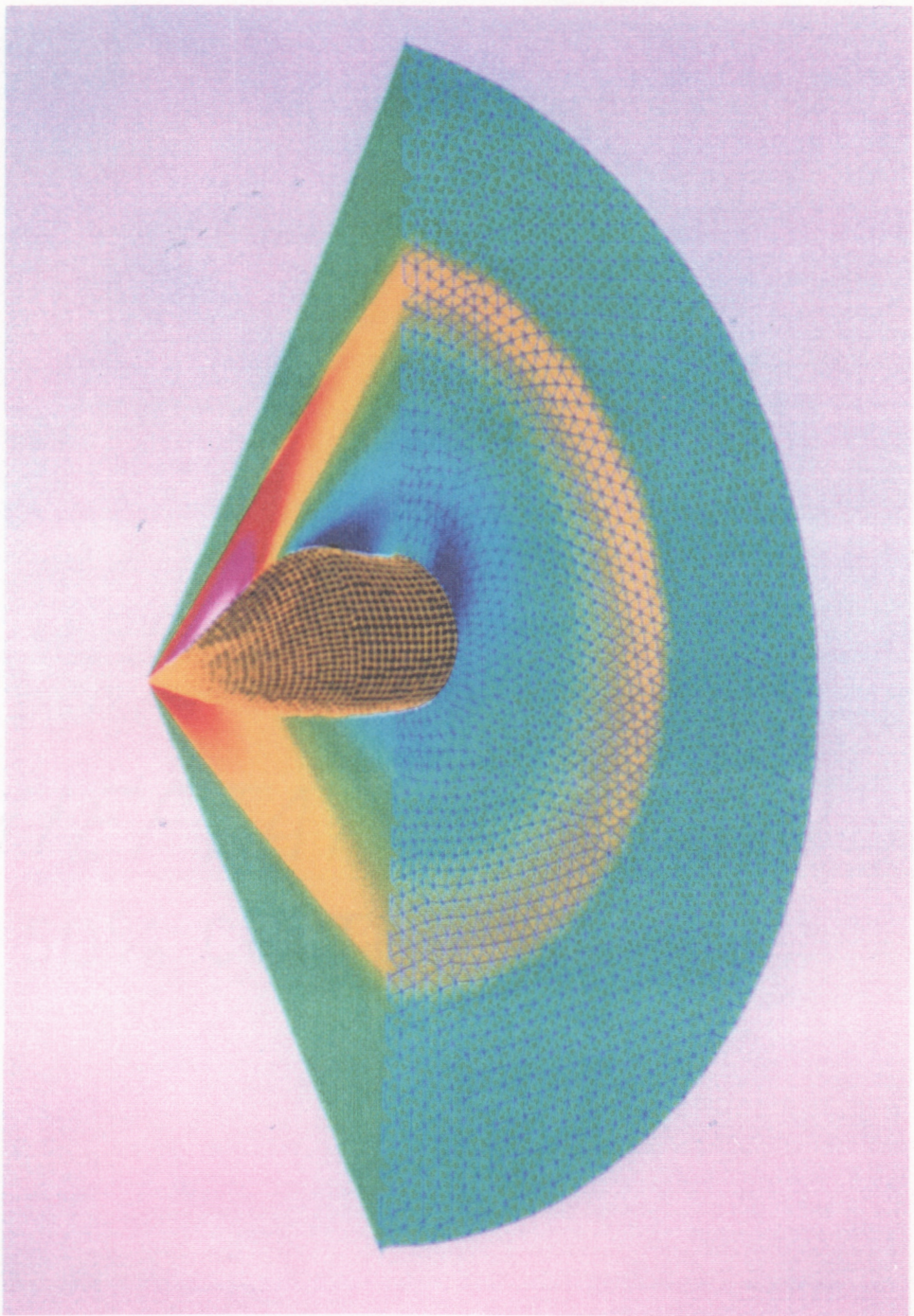


Figure 23: Pressure contours about the symmetry plane and the exit plane of an analytic forebody employing a two zone pentahedral discretization.

by 40 structured grid was used as a template for this discretization. Note that there is a $2\frac{1}{2} : 1$ difference in elements for the tetrahedral and pentahedral meshes resulting from the same size structured discretization. Thus, while both discretizations have the same grid point distribution, in reality the tetrahedral discretization is a finer mesh. Figure 24 shows pressure contours about the symmetry plane and last cross-flow plane for the tetrahedral discretization. In addition, the grid has been superimposed on the surface of the body and the last cross-flow plane.

Numerical data is available for a solution obtained on a structured grid[5] and is presented for comparison with experimental data and the unstructured numerical results. The pressure coefficient, C_p, is shown along the top and bottom surface of the body in figure 25. All numerical solutions show good agreement with experimental data. Most importantly, note that there is good agreement between the four numerical solutions. Thus, generalized discretizations can yield similar solution qualities to those of structured discretizations with similar grid point distribution. Figure 26 depicts the convergence history of each computational block of the tetrahedral discretization. Note that the number of iterations required to converge each block dramatically decreases as the solution progresses in the streamwise direction. While these values do not reflect optimal convergence rates, they do demonstrate the benefits of a space marching algorithm.

7.3 Model SR71

To demonstrate the geometric flexibility of the unstructured algorithms, the solution about a simplified model of the SR71 reconaissance aircraft has been calculated for the pentahedral mesh. This model includes a region of multiple elements in the streamwise direction at the start of the engine inlets, as well as multiple vertical tails. Both of these geometries cause difficulties with a structured discretization,

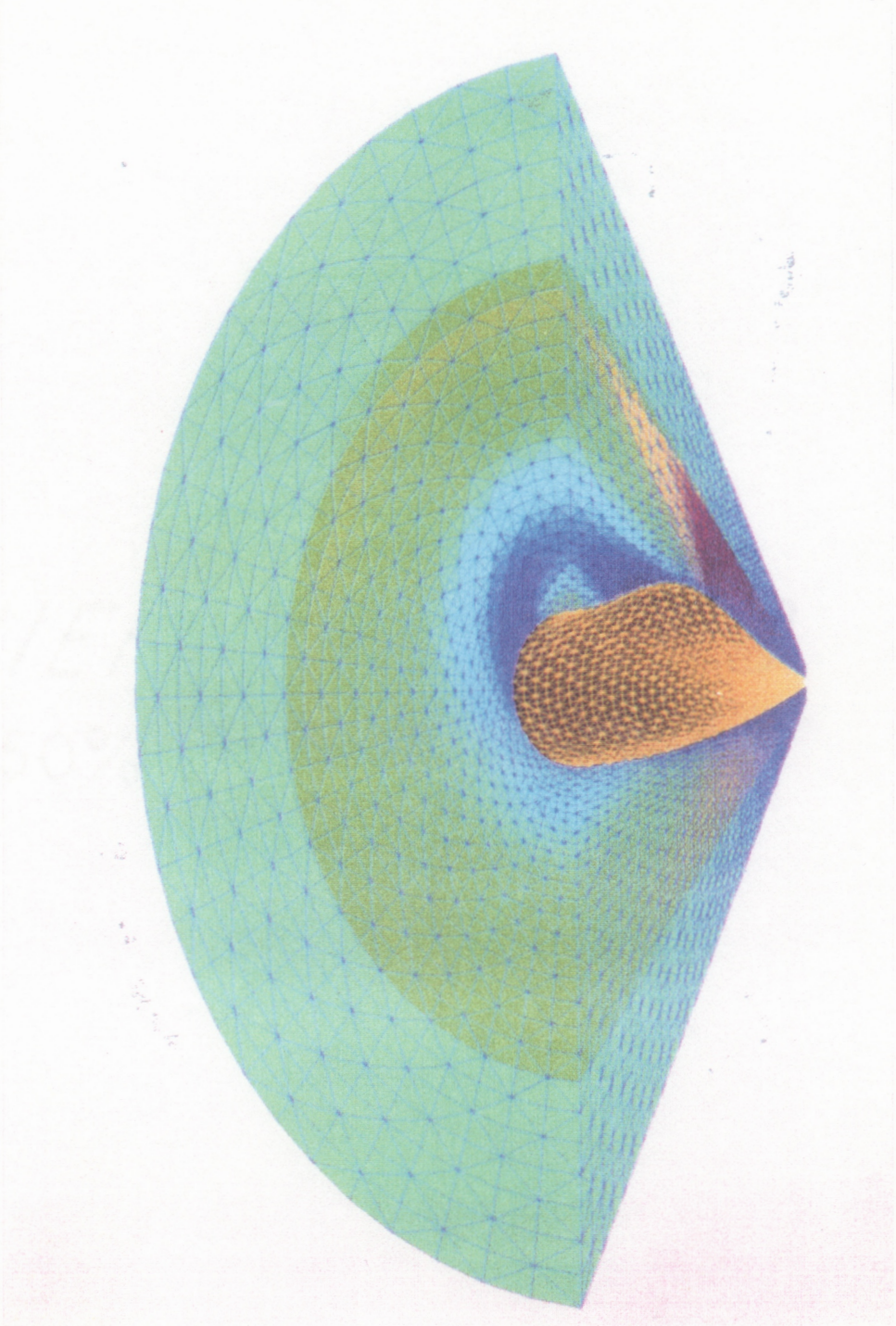


Figure 24: Pressure contours about the symmetry plane and the exit plane of an analytic forebody employing a tetrahedral discretization.

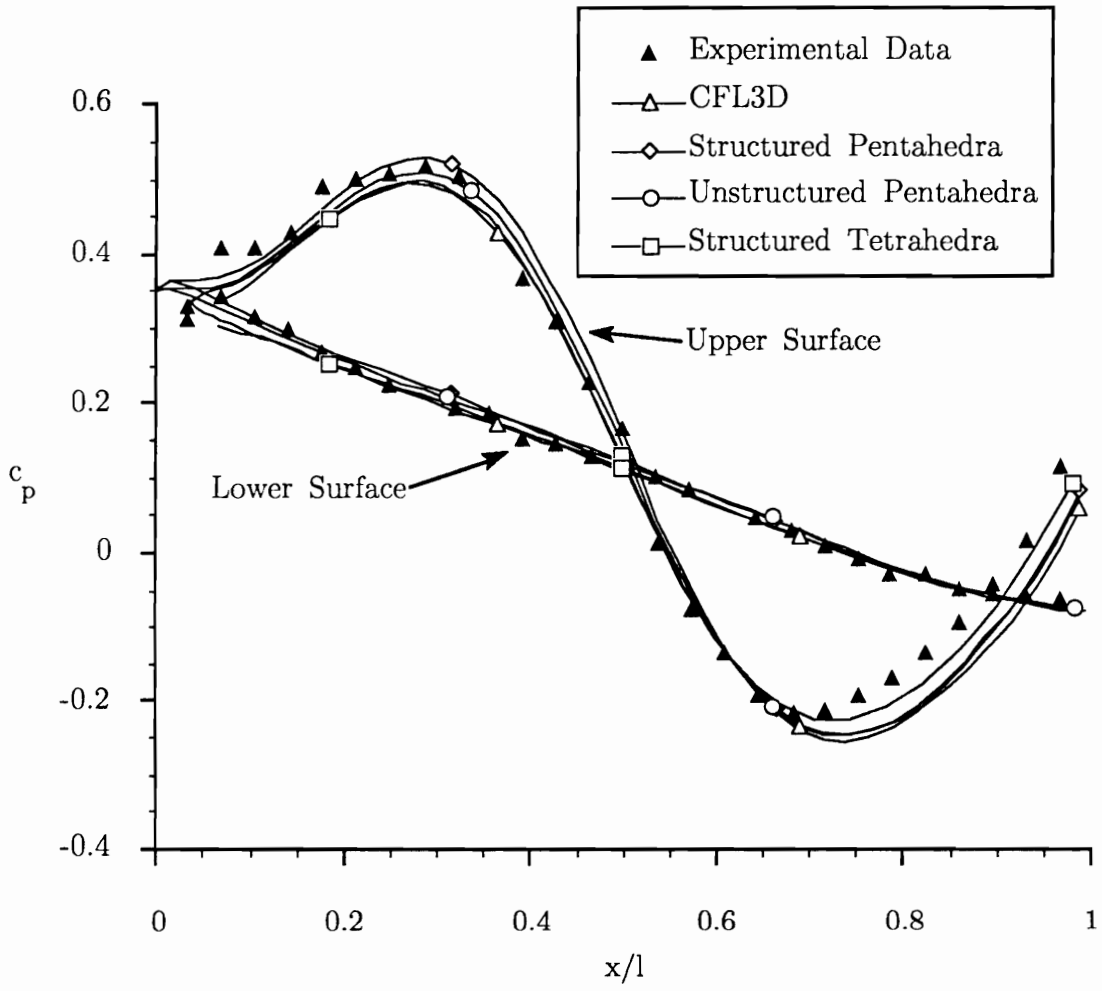


Figure 25: Longitudinal C_p distributions in the symmetry plane of an analytic forebody. $M = 1.7, \alpha = 0°$.

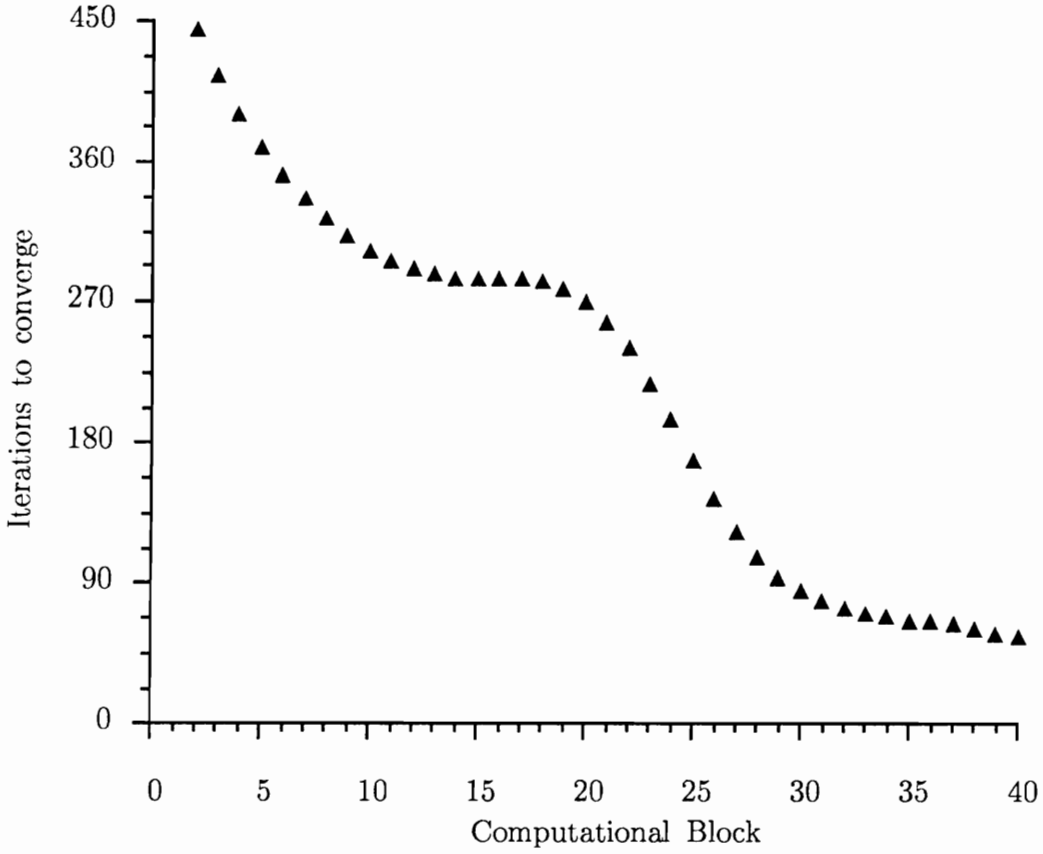


Figure 26: Convergence history based upon the computational block of a tetrahedral discretization of the analytic forebody.

while they impose no restrictions with an unstructured discretization. The solution is calculated in a Mach 3.5 free stream at a 0° angle of incidence relative to the root chord. Due to a current lack of sophisticated geometric modeling capabilities for boundary description and no current ability to handle space-marching grids for the three-dimensional advancing front algorithm, no solution was attempted using the tetrahedral discretization.

This pentahedral solution was obtained on a grid with a total of 42 blocks in ten separate zones; four zones in the forebody region, two in the multiple element, inlet region, two in the region forward of the vertical tail, and two to resolve the vertical tail section. The number of nodes in each block varied from 2338 in the first zone to a maximum of 5589 in the next to last zone. Figure 27 shows the grid on the surface of the body as well as in the last cross-flow plane. Note that there are discontinuities in the grid along the body at the zonal interfaces. Figure 28 presents pressure contours in the exit plane as well as the grid along the body. Figure 29 represents pressure contours along the surface of the body as well as in the exit plane. The grid at the exit plane has been superimposed on the figure. Figure 2 in section 1.2 represents pressure contours in three cross-flow planes along the body. In the final figures, all predominant flow features such as the bow shock and oblique shocks from the the nacelles may be seen.

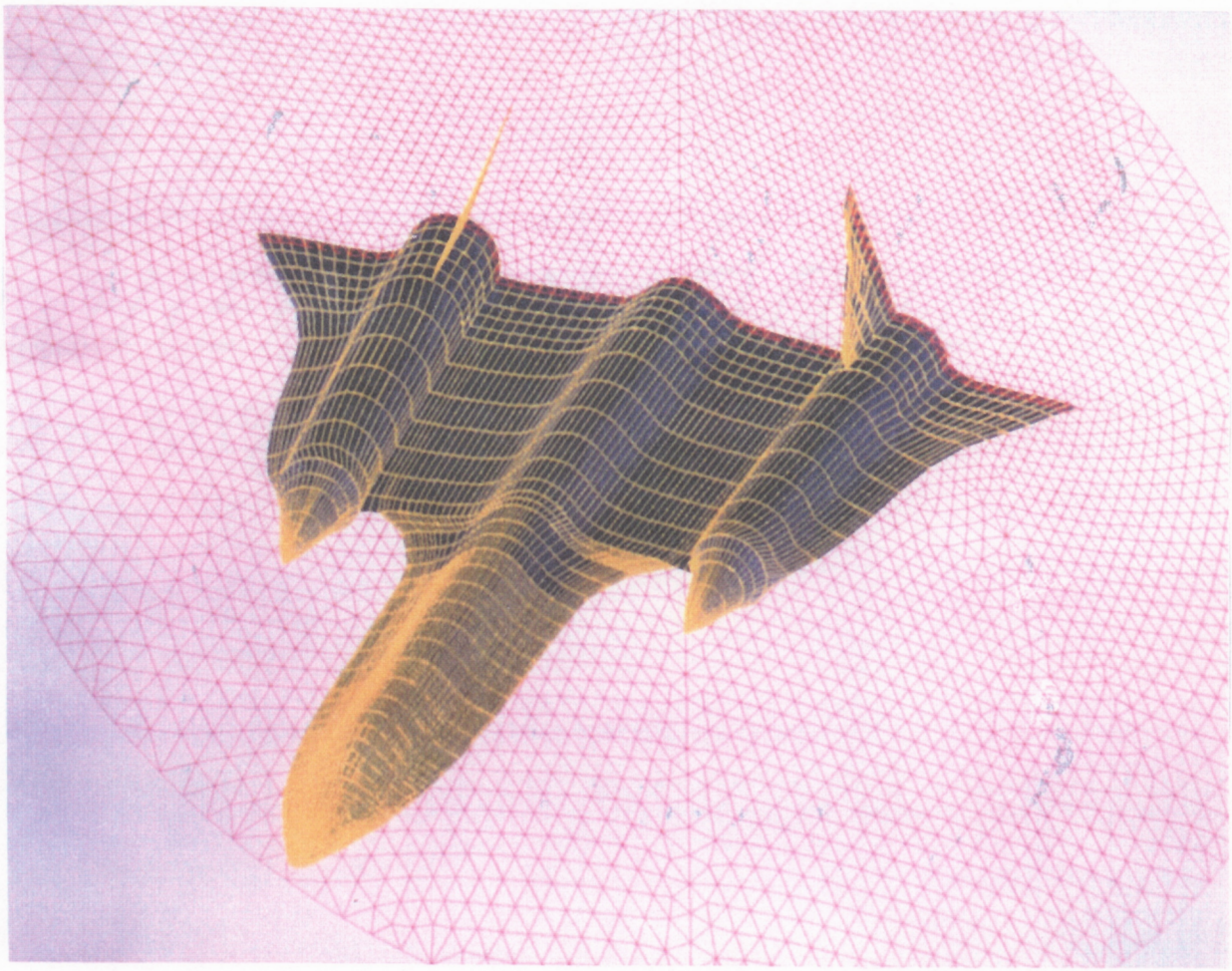


Figure 27: Grid on the surface and in the exit plane of a model SR71.

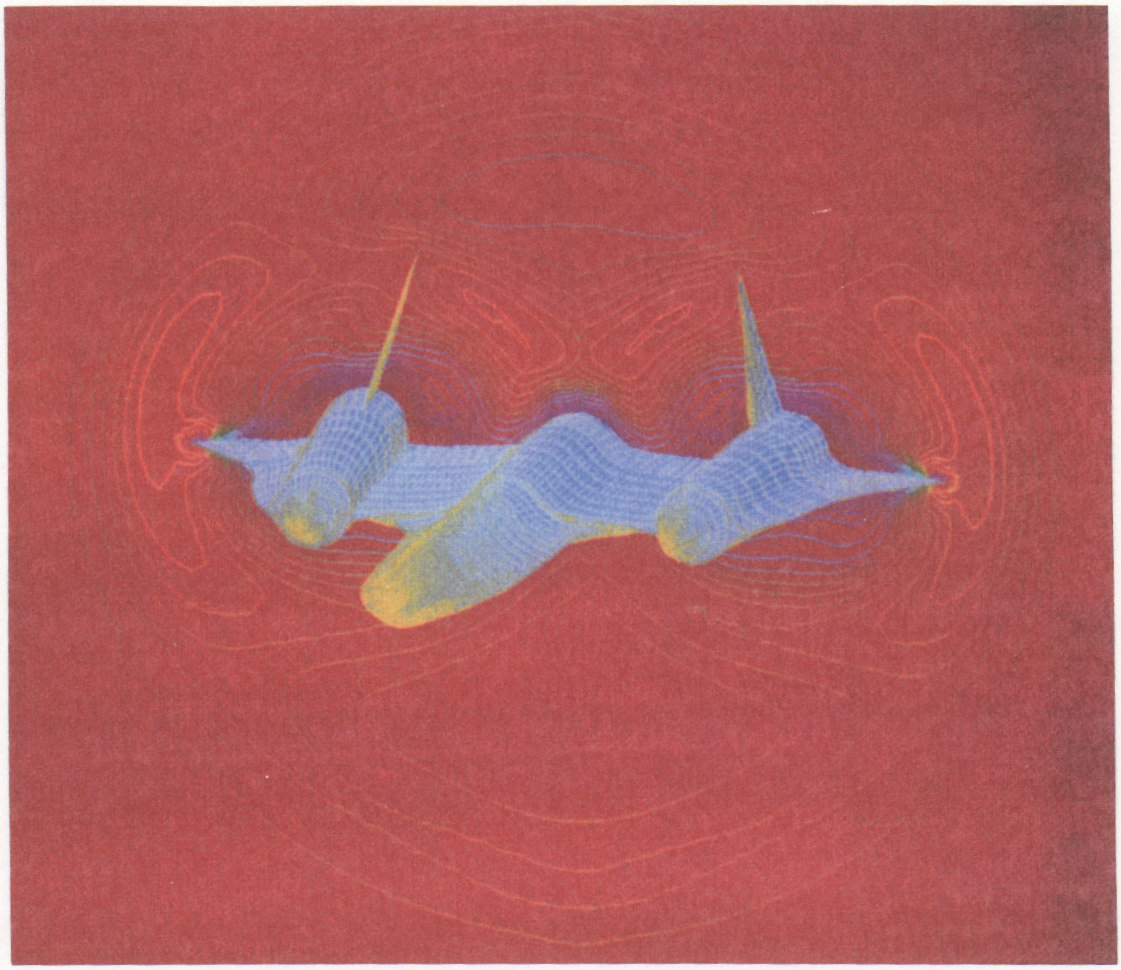


Figure 28: Line contours in the exit plane of a model SR71.

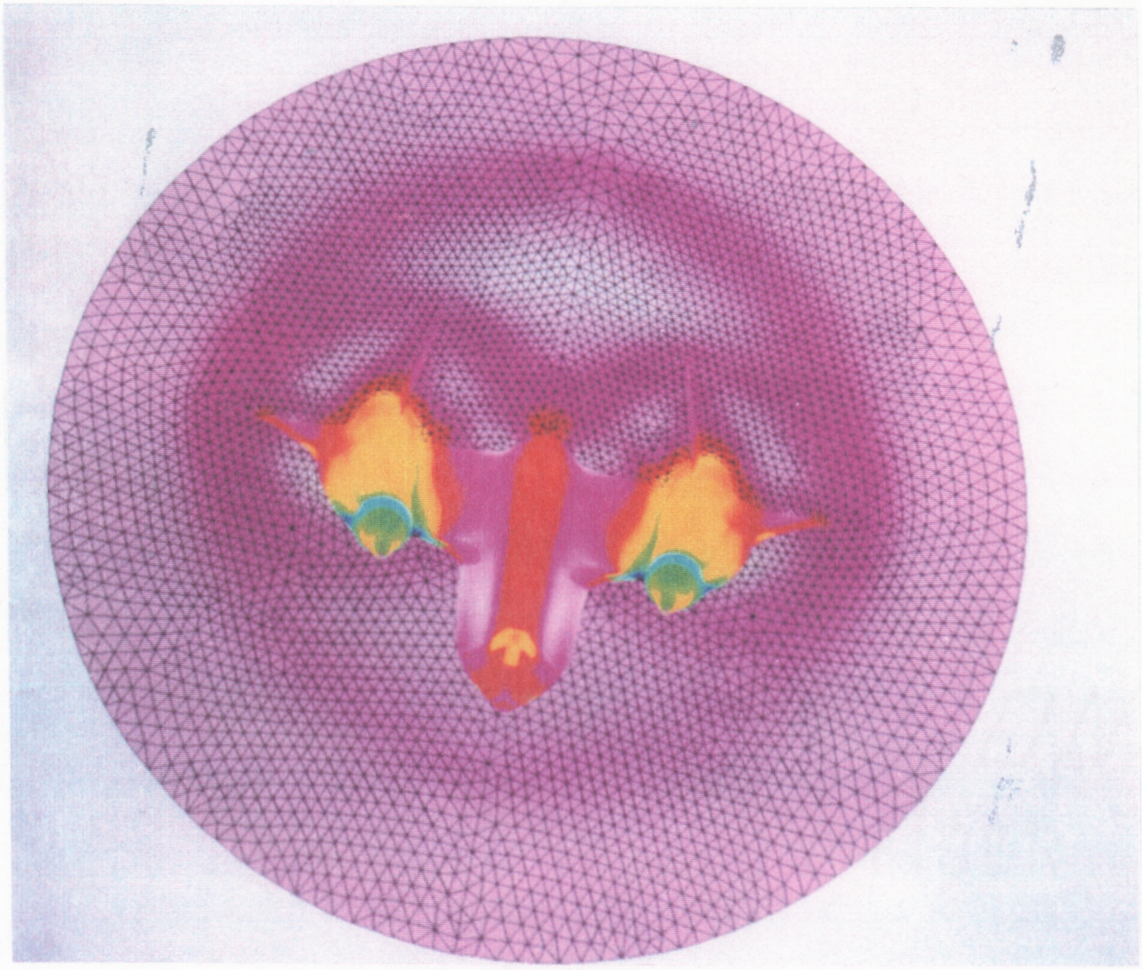


Figure 29: Pressure contours on the surface and in the exit plane of a model SR71.

8.0 CONCLUDING REMARKS

Implementations of space-marching algorithms in conjunction with generalized indexing strategies have been investigated. Two possible discretizations have successfully been implemented for the solution of the Euler equations. Successful extensions of first-order spatial accurate approximations to second-order have been accomplished for the pentahedral discretization. Solutions using unstructured discretizations with element distributions similar to structured discretizations compare favorably to solutions about the structured grids. Comparison of the one and two zone solutions of the analytic forebody with pentahedral discretization show that the interpolation procedure used is sufficient to ensure a smooth and accurate transition of the solution from one zone to the next. The solution about the model SR71 has demonstrated the capability of obtaining inviscid solutions about geometrically complex domains. The tetrahedral discretization has been successfully implemented, promising much greater geometric flexibility. However, three-dimensional unstructured grid generation is currently not sufficiently mature to provide discretizations which will demonstrate the true flexibility of the solution algorithm.

State of the art generalized indexing solution algorithms are still in their infancy. The areas of spatial accuracy, time integration, and grid generation all may benefit from improvements in implementation and efficiency. Given the discretization techniques described herein, most of these improvements in solution algorithms developed for global time integration strategies may be directly applied to the planar blocks of the space-marching discretization.

With the idea of an automated solution process which requires minimal effort in problem description from the user, some recommendations about the choice of

spatial discretization are made. The pentahedral discretization requires a large degree of user interaction in the grid generation process. There does not seem to be any way to mathematically relate the geometry of a particular plane to its suitability as a parent plane in the initial generation for a zone. In addition, there seems to be no way to quantify the location of all zonal interfaces. Therefore, the user must not only specify the number and location of zonal interfaces, they must also specify which plane within a zone with which to initialize the grid generation process. These requirements are not conducive to an automated grid generation environment. Finally, solution adaptive grid generation has received great attention recently as an efficient means for obtaining a high degree of spatial accuracy[45,51,52]. Generalized indexing provides a means for placing elements where they are required to obtain this spatial accuracy. However, with the pentahedral discretization, much flexibility in placing elements in the the streamwise direction is lost. Effectively, the only solution adaptive generation might be the clustering near solid boundaries required for viscous solutions. Thus, again a level of automation in the solution process is not available with the use of the pentahedral discretization. Thus, while this solution algorithm, grid generation combination currently allows greater geometric flexibility than the tetrahedral discretization, its further study is not recommended.

It is the author's opinion that the tetrahedral discretization solution algorithm, three-dimensional advancing front grid generation combination offers the most promise for an automated solution environment. With this combination, the specification of the location of planar interfaces could be a function of local element size in the streamwise direction. The width of the current computational block might be selected to minimize the depth of elements in the streamwise direction. Since the grid generation within a block is controlled solely by the advancing front grid generator, a solution adaptive algorithm could be employed within that block.

The element size in the streamwise direction would be controlled by the solution adaptive algorithm. Thus if the block size were a function of the element size in the streamwise direction, the depth of each block would be specified by the solution algorithm. With this situation, the user would only need to specify the geometric description of the entire domain, boundary conditions, and error tolerances for the solution adaptive algorithm. With this information, it is conceivable that the solution algorithm, grid generation combination could calculate all other necessary parameters and provide the user with a converged solution.

In order to realize the goal of automated solution algorithms several key areas require further development. The grid generation algorithms can benefit most by continued research. Currently available three-dimensional advancing front algorithms are not robust. They require tuning of certain parameters to allow discretization of different domains. This parameter tuning is contradictory to the concept of automation. In addition, geometric modeling of domain boundaries is crude. The capability to interpret complex boundaries composed of higher-order surfaces such as Bezier surfaces and B-splines will greatly enhance the geometric modeling capabilities of the grid generator. Finally, computational efficiency is an issue for the advancing front algorithm. The efficiency of many of the search routines used in this algorithm can probably be greatly enhanced through the use of parallelization. Note that the majority of code segments in the advancing front algorithm will not vectorize.

Another area which warrants continued research is the area of error calculation for use in the solution adaptive process. With solution adaptive calculations, the algorithm attempts to provide a uniform truncation error throughout the solution domain by controlling the element size and orientation. In order for this to occur, an accurate estimate of the truncation error must be made at discrete

points throughout the domain. Currently, a rather crude approximation derived from finite-difference expressions approximates the truncation error as a function of a single primitive variable such as pressure. While this approximation may be sufficient for some inviscid problems with ideal gas equations of state, it will not be a good approximation when more complex governing equations are considered. It is therefore the author's opinion that a more exhaustive investigation into suitable estimates for truncation error is warranted by the possible improvements in solution quality obtainable from solution adaptive algorithms. This investigation will not only benefit space-marching algorithms, but global iteration strategies as well.

Finally, the work presented thus far has been directed toward the solution of the Euler equations. While these equations can reveal useful information about flow quality, many calculations require the solution of the Navier Stokes equations in order to provide any necessary information. Preliminary investigations have been performed into the application of the Navier Stokes equations to generalized coordinates. However, again, the major stumbling block for this area seems to be grid generation. Many researchers claim that difficulties with round-off error in the advancing front algorithm prohibit the generation of elements with sufficiently large aspect ratios required to resolve viscous boundary layers on three-dimensional problems. With the resolution of this difficulty, the solution of the parabolized Navier Stokes equations should be possible with the spatial discretizations discussed here.

REFERENCES

1. Walters, R.W. and Dwoyer, D.L., "Efficient Solutions to the Euler Equations for Supersonic Flow with Embedded Subsonic Regions", NASA TP 2523, 1987.

2. Walters, R.W. and Dwoyer, D.L., "An Efficient Iteration Strategy for the Solution of the Euler Equations", AIAA Paper 85-1529, July 1985.

3. Newsome, R.W., Walters, R.W. and Thomas, J. L., "An Efficient Iteration Strategy for Upwind Relaxation Solutions to the Thin-Layer Navier-Stokes Equations," AIAA 87-1113-CP June, 1987.

4. Richardson, P.F. and Morrison, J.H., "Displacemant Surface Calculations for a Hypersonic Aircraft", AIAA Paper 87-1190, June 1987.

5. Walters, R.W., Reu, T., McGrory, W.D., Thomas, J.L., Richardson, P.F., "A Longitudinally-Patched Grid Approach with Applications to High Speed Flows", AIAA Paper 88-0715, Jan. 1988.

6. Löhner, R.,"Finite Elements in CFD: What Lies Ahead", *Int. J. Num. Meth. Eng.*, no. 24, pp. 1741-1756, 1987.

7. MacCormack, R.W., "The Effect of Viscosity in Hypervelocity Impact Cratering", AIAA Paper 69-354, 1969.

8. MacCormack, R.W., "A Numerical Method for Solving the Equations of Compressible Viscous Flow", *AIAA J.*, vol. 20, pp. 1275-1281, 1982.

9. Briley, W.R. and McDonald, H., "Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method", *J. Comp. Phys.*,vol. 24, pp. 372-397, 1977.

10. Beam, R.M. and Warming, R.F., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations", *AIAA J.*, vol. 16, pp. 393-402, 1978.

11. Douglas, J. and Gunn, J.E., "A General Formulation of Alternating Direction Methods", *Numerische Mathematik*, vol. 6, pp. 428-453, 1964.

12. Peaceman, D.W. and Rachford, H.H., "The Numerical Solution of Parabolic and Elliptic Differential Equations", *SIAM J.*, vol. 3, pp. 28-41, 1955.

13. Steger, J.L., "Implicit Finite Difference Simulation of Flow About Arbitrary Two-Dimensional Geometries", *AIAA J.*, pp.679-686, vol. 16, 1978.

14. Pulliam, T.H. and Steger, J.L.,"On Implicit Finite-Difference Simulations of Three Dimensional Flow", AIAA Paper 78-10, 1978.

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

16. Jameson, A. "Solution of the Euler Equations by a Multigrid Method", *Applied Mathematics and Computation*, vol. 13, pp. 327-356, 1983.

17. Godunov, S.K., "Finite-Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics", *Mat. Sbornik*, vol. 47, pp. 357-393, 1959. (In Russian)

18. Peyret, R., Taylor, T.D., **Computational Methods for Fluid Flow**, pp. 112-115, Springer-Verlag, New York, 1983.

19. Chakravarthy, S.R. and Osher, S., "Computing with High-Resolution Upwind Schemes for Hyperbolic Equations", *Lectures in Applied Mathematics*, vol. 22, pp.57-86, 1985.

20. Chakravarthy, S.R.,"Relaxation Methods for Unfactored Implicit Upwind Schemes", AIAA Paper 84-0165, 1984.

21. Harten, A., Lax, P.D. and Van Leer, B., "On Upstream Differencing and Godunov Type Schemes for Hyperbolic Conservation Laws", *SIAM Review*, vol. 25, pp. 35-61, 1983.

22. Harten, A., "High Resolution Schemes for Hyperbolic Conservation Laws", *J. Comp. Phys.*, vol. 49, pp. 357-393, 1983.

23. Osher, S. and Solomon, F., "Upwind Difference Schemes for Hyperbolic Systems of Conservation Laws", *Math. of Comp.*, vol. 38, pp. 339-374, 1982.

24. Roe, P.L., "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes", *J. Comp. Phys.*, vol. 43, pp. 357-372, 1981.

25. Roe, P.L., "Some Contributions to the Modelling of Discontinuous Flows", *Lectures in Applied Mathematics*, vol. 22, 1985.

26. Roe, P.L., "Characteristic Based Schemes for the Euler Equations", *Ann. Rev. Fluid Mech.*, vol. 18, pp. 337-365, 1986.

27. Steger, J.L. and Warming, R.F., "Flux Vector Splitting of the Inviscid Gas-dynamic Equations with Application to Finite-Difference Methods", *J. Comp. Phys*, vol. 40, pp. 263-293, 1981.

28. Thomas, J.L., Van Leer, B., and Walters, R.W., "Implicit Flux-Split Schemes for the Euler Equations", AIAA Paper 85-1680, 1985.

29. Thomas, J.L. and Walters, R.W., "Upwind Relaxation Algorithms for the Navier Stokes Equations", AIAA Paper 85-1501, 1985.

30. Van Leer, B., "Towards the Ultimate Conservative Difference Scheme. A Second Order Sequel to Godunov's Method", *J. Comp. Phys.*, vol. 32, pp. 101-136, 1979.

31. Van Leer, B., "Flux-Vector Splitting for the Euler Equations", *Lecture Notes in Physics*, vol. 170, pp. 507-512, 1982.

32. Colella, P. and Woodward, P.R., "The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulation", *J. Comp. Phys.*, vol. 54, pp. 174-201, 1984.

33. Yee, H., Warming, R.F. and Harten, A., "Implicit Total Variation Diminishing (TVD) Schemes for Steady-State Calculations", AIAA Paper 83-1902, 1983.

34. Vigneron, Y.C., Rakich, J.V. and Tannehill, J.C., "Calculation of Supersonic Viscous Flow over Delta Wings with Sharp Subsonic Leading Edges", AIAA Paper 78-1137, July 1978.

35. Rai, M.M., "A Relaxation Approach to Patched-Grid Calculations with the Euler Equations", AIAA Paper 85-0295, Jan. 1985.

36. Rai, M.M., "An Implicit, Conservative, Zonal-Boundary Scheme for Euler Equation Calculations", AIAA Paper 85-0488, Jan. 1985.

37. Hessenius, K.A., Rai, M.M., "Three Dimensional, Conservative, Euler Computations using Patched Grid Systems and Explicit Methods", AIAA Paper 86-1081, May 1986.

38. Walters, R.W., Cinnella, P., Slack, D.C., Halt, D., "Characteristic Based Algorithms for Flows in Thermo-Chemical Nonequilibrium", AIAA Paper 90-0393, Jan. 1990.

39. Stoufflet, B., Périaux, J., Fezoui, F., Dervieux, A., "Numerical Simulation of 3-D Hypersonic Euler Flows Around Space Vehicles using Adapted Finite Elements", AIAA Paper 87-0560, Jan. 1987.

40. Peraire, J., Peiro, J., Formaggia, L., Morgan, K., and Zienkiewicz, O.C., "Finite-Element Euler Computations in Three Dimensions", AIAA Paper 88-0032, 1988.

41. Whitaker, D. L., Slack, D. C., and Walters, R. W., "Solution Algorithms for the Two-Dimensional Euler Equations on Unstructured Meshes",AIAA 90-0697, Jan. 1990.

42. Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes", AIAA Paper 89-0366, Jan. 1989.

43. Barth, T. J., and Frederickson, P. O., "Higher Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction", AIAA Paper 90-0013, Jan. 1990.

44. Thareja, R.R., Morgan, K., Peraire, J., Peiro, J., "A Three-Dimensional Upwind Finite Element Point Implicit Unstructured Grid Euler Solver", AIAA Paper 89-0658, Jan. 1989.

45. Slack, D. C., Walters, R. W., and Löhner, R., "An Interactive Adaptive Remeshing Algorithm for the Two-Dimensional Euler Equations", AIAA Paper 90-0331, Jan. 1990.

46. Bowyer, A., "Computing Dirichlet Tessellations", *The Computer Journal*, vol. 24, No. 2, pp. 162-166, 1981.

47. Green, P. J. and Sibson, R., "Computing the Dirichlet Tesselation in the Plane", *The Computer Journal*, vol. 21, no. 2, pp. 168-173, 1977.

48. Guibas, L. and Stolfi, J., "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", *ACM Transactions on Graphics*, Vol. 4, No. 2, April 1985, pp. 74-123.

49. Baker, T. J., "Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", AIAA 87-1124-CP, 1987.

50. Kennon, S.R., "A Vectorized Delaunay Triangulation Scheme for Non-Convex Domains With Automatic Nodal Point Generation", AIAA Paper 88-0314, Jan. 1988.

51. Peraire, J., Vahdati, M., Morgan, K., Zienkiewicz, O.C., "Adaptive Remeshing for Compressible Flow Computations", *J. Comp. Phys.*, vol. 72, pp. 449-466, 1987.

52. Löhner, R., Parikh, P., "Generation of Three-Dimensional Unstructured Grids by the Advancing Front Method", AIAA Paper 88-0515, Jan. 1988.

53. Van Leer, B., Thomas, J.L., Roe, P.L., and Newsome, R.W., "A Comparison of Numerical Flux Formulas for the Euler and Naver-Stokes Equations", AIAA Paper 87-1104, June 1987.

54. Roe, P.L., "Error Estimates for Cell-Vertex Solutions of the Compressible Euler Equations", ICASE report 87-6, 1987.

55. Walters, R.W., Thomas. J.L., "Advances in Upwind Relaxation Methods",**State-of-the-Art Surveys on Computational Mechanics**, pp. 145-183, A.K. Noor and J.T. Oden, Eds., ASME, New York, 1989.

56. Whitaker, D.L., Grossman, B., Löhner, R., "Two-Dimensional Euler Computations on a Triangular Mesh Using an Upwind, Finite-Volume Scheme", AIAA Paper 89-0470, Jan. 1989.

57. Löhner, R., "Some Useful Data Structures for the Generation of Unstructured Grids", *Communications in Applied Numerical Methods*, Vol. 4, 1988, pp. 123-135.

58. van der Houwen, P., J., "Construction of Integration Formulas for Initial Value Problems", North Holland Publishing Conference, 1977.

59. Turkel, E. and Van Leer, B. "Flux-Vector Splitting and Runge-Kutta Methods for the Euler Equations", ICASE Report No. 84-27, June 1984, also in NASA CR-172415, 1984.

60. Duff, I.S., Erisman, A.M., and Reid, J.K., **Direct Methods for Sparse Matrices**, Oxford University Press, New York, 1986.

61 Taylor, G.I, and Maccoll, J.W., "The Air Pressure on a Cone Moving at High Speed", *Proc. Roy. Soc.* (London) ser. A, vol. 139, 1933, pp.278-311.

62. Townsend, J.C., Howell, D.T., Collins, I.K., and Hayes, C., "Surface Pressure Data on a Series of Analytic Forebodies at Mach Numbers from 1.70 to 4.50 and Combined Angles of Attack and Sideslip", NASA TM 80062, Jun., 1979.

APPENDICES

Appendix A: Two-Dimensional Advancing Front Algorithm

There are several motivations for selecting the advancing front algorithm for discretization of two and three-dimensional domains into triangles and tetrahedra. The algorithm generates both nodes and element connectivity simultaneously, eliminating the need for separate node generation and connectivity algorithms as with the Delaunay triangulation method. In generating nodes, the user has a great deal of flexibility in specifying placement in addition to the orientation of the elements. In the following implementation, the point distribution and cell orientation are specified on a background grid which is typically another triangular grid. With the above mentioned traits, the algorithm is ideally suited for solution adaptive grid generation. Finally, this algorithm is competitive in computer resource use with other generalized grid generation techniques.

On the other hand, several issues concerning the efficiency and robustness need to be addressed. This implementation is a scalar algorithm; the major routines are not vectorizable. However, many of the search routines might benefit from a parallel implementation. Some investigators claim that the highly distorted elements which can arise are not optimal in a Voronoi sense. However, the Delaunay triangulation cannot take into account the underlying physics of a problem in specifying connectivity, while the advancing front algorithm has this capability. At this time, the three-dimensional algorithm is not robust. Some researchers say that this algorithm is not capable of generating elements with sufficiently high aspect ratios for viscous flow calculations. However, this speculation has not been demonstrated.

The two-dimensional algorithm[40,44,51,52] requires the following information as input. All boundaries of the domain, both physical and far field must be described. In two-dimensions, the boundaries may be any number of line segments which, when

combined, form any number of closed curves. These line segments may consist of any type of analytically definable curve. Typically, linear segments and cubic splines are sufficient. The closed curves will define the exterior boundary and any discrete bodies within the domain. Some notation should be specified to be able to determine which side of each segment lies within the domain. For instance, the path of the exterior boundary must be in a counter-clockwise orientation, while any interior boundaries must follow a clockwise orientation. Suitable boundary description for the domain about two airfoils is shown in figure 30(a). This domain will be used as an example throughout this appendix.

In addition to the boundaries of the domain, point distribution and cell orientation must be specified throughout the domain. This is accomplished by specifying parameters at the nodes of a background grid which encompasses the entire domain (figure 30(b)). Values interior to these nodes are linearly interpolated from values at the nodes. The information necessary for 2-D specification of point distribution and cell orientation are a vector, $\vec{\alpha}$, indicating the direction of stretching for an element at that point, the element size, s, in the direction of $\vec{\alpha}$, and the ratio of element size, δ, between $\vec{\alpha}$ and the vector normal to $\vec{\alpha}$, figure 31(a). The boundary description and node, element distribution provide sufficient information to completely discretize a given domain.

The advancing front algorithm begins with no elements or nodes in the domain or on the boundaries. A generation front consisting of faces is first formed. This front initializes a dynamic boundary between the portion of the domain which has been discretized with elements, and that portion of the domain which has not. Since no elements are initially present, the initial generation front should consist of all faces which lie on the boundaries of the domain. Thus, the first step is to generate this initial front. One must place points or nodes on the boundaries one

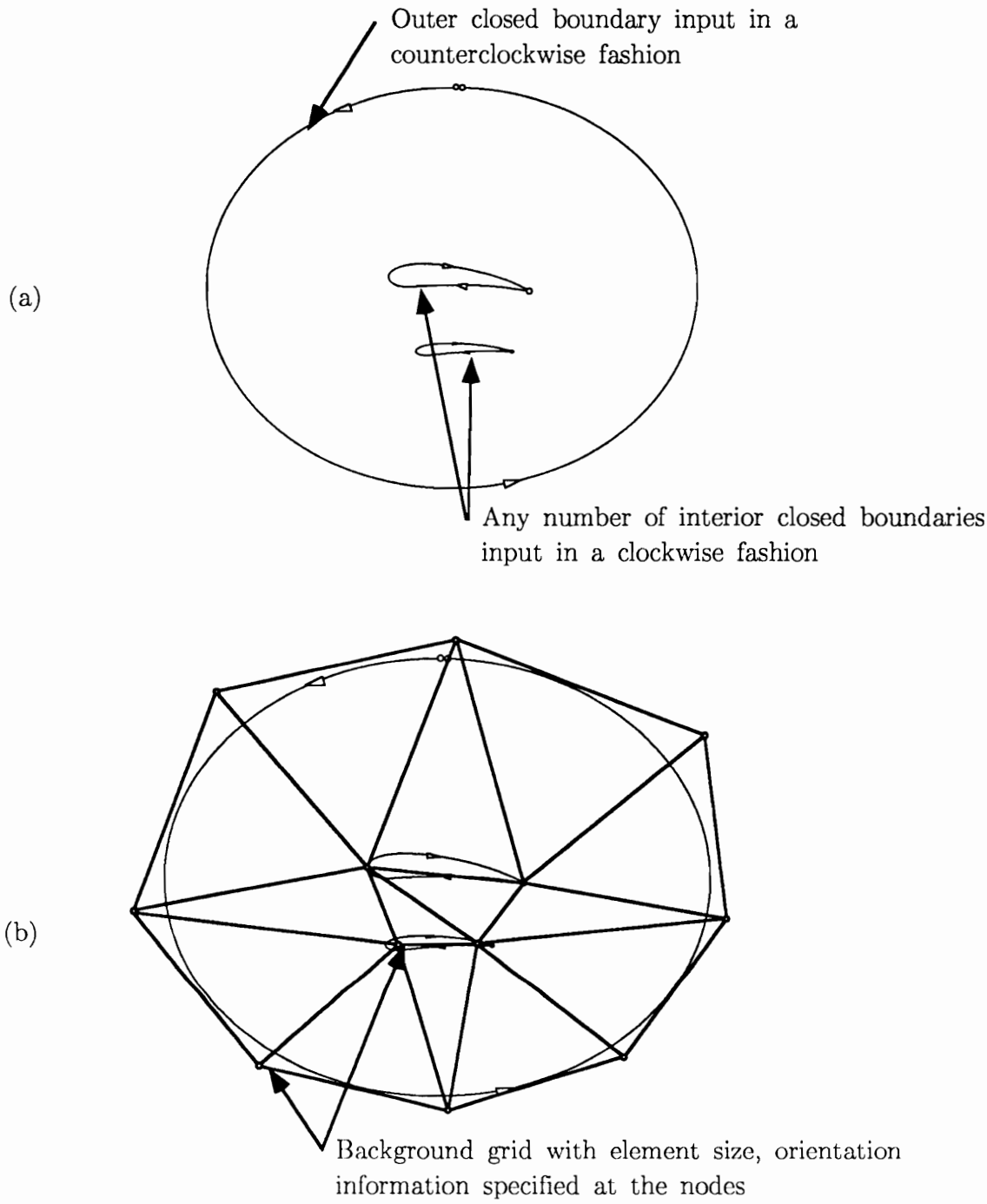
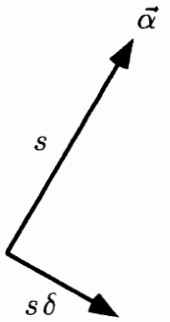
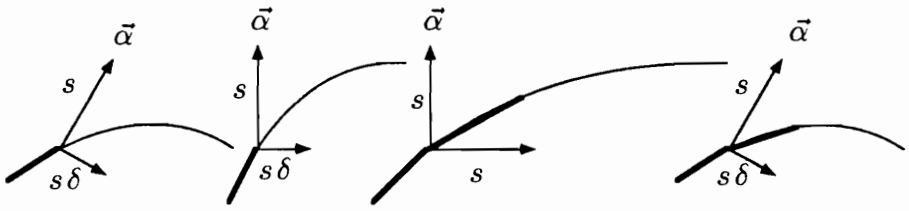


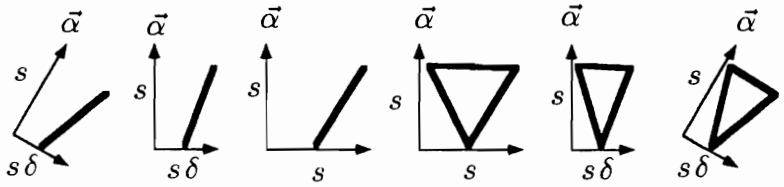
Figure 30: Required input for two-dimensional advancing front grid algorithm.



(a) Specification of element size and orientation



(b) Generation of a new face along a boundary segement.



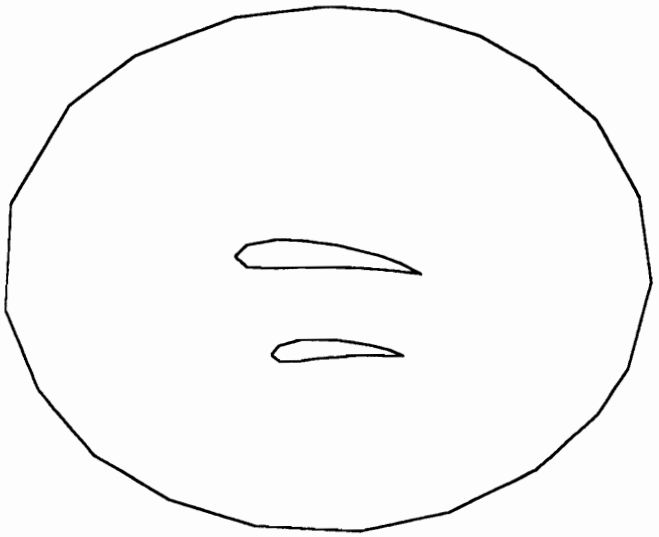
(c) Generation of a new element with an ideal point.

Figure 31: Element size, orientation paramaters, and their uses.

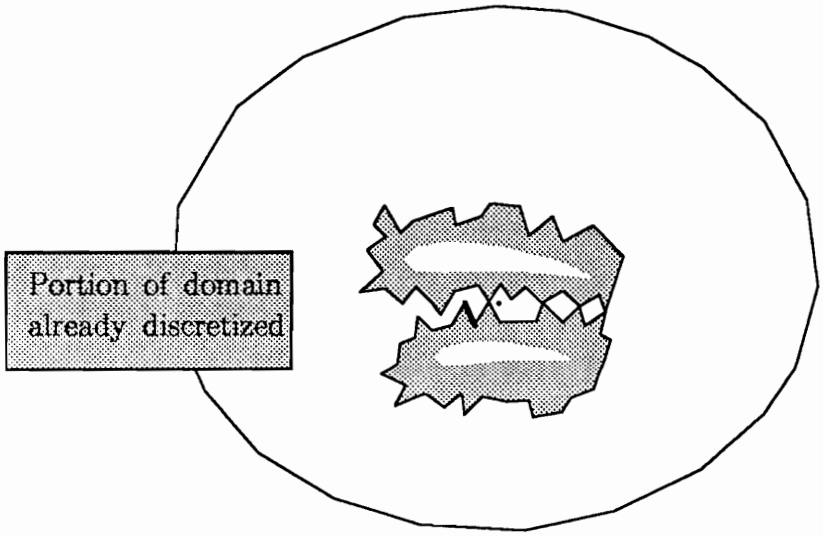
at a time according to the stretching and size information from the background grid. Neighboring nodes are connected to form boundary faces. The collection of all faces will form the initial front. In addition, each face on the generation front is ordered such that the vector from the first to the second point, when rotated 90° counterclockwise, will be directed toward the immediate area of the domain which has not yet been discretized. The other side of the face represents areas which have already been discretized or areas outside of the domain.

Each boundary segment, as described in the input parameters, is discretized individually by performing the following steps. First, a point is created at one end of each segment. At that point, element size information is found from the background grid. Then the local coordinates are unstretched coordinates. First rotate the coordinate system about the initial node until $\vec{\alpha}$ is directed along the "y" axis. The stretching is then removed from the "x" coordinate. In transformed coordinates, elements will ideally be equilateral triangles with sides of length s. Thus the next point is placed "s" units along the current segment from the first point. Note that the current segment is in unstretched coordinates also. Finally, the new point is transformed back into global, stretched coordinates (figure 31(b)). The new point becomes the next initial point and the addition of points continues until the end of the line segment is reached. This procedure is repeated for all segments which describe boundaries, both interior and exterior. Figure 32(a) represents the discretization of the boundaries which forms the initial generation front for the example problem.

Once the initial front has been generated, a new triangular element (and possibly a new node) is generated. One of the faces on the generation front will always be included in each new element. When a new element is generated, the portion of the domain already discretized changes, and thus the faces which comprise the



(a) Initial generation front.



(b) Intermediate generation front.

Figure 32: Generation fronts from example two-dimensional problem.

generation front change. Figure 32(b) represents the generation front at an intermediate stage of the grid generation process. The process of adding a new element is iterated upon until no faces are left on the generation front and the entire domain has been discretized.

Some criteria must be specified for the selection of the proper face on the generation front to eliminate from the front while adding a new element. In order to assure that all grid detail is maintained, small elements should be generated before large elements. Thus, the faces on the generation front should be sorted by face length. The smallest face on the current generation front should always be chosen. This will help to prevent large elements from being generated over regions dominated by small elements.

With a face selected for deletion, a third point used to form a new element must be selected. The ordering of the points of the face will indicate the side on which to generate the element. An ideal third point may be calculated in a manner similar in fashion to the creation of the boundary points. The stretching information at the current face is interpolated from the background grid. Using this information, the two end points of the current face are transformed into unstretched coordinates(figure 31(c)). A third point in transformed coordinates is then calculated. Note that ideally, all of the faces in a generation front will be of length s in local transformed coordinates, where s is the local ideal element size. Thus, ideal elements will be equilateral triangles with sides of length s. The ideal third point may be constructed in transformed coordinates using this information. With the third point constructed in transformed coordinates, it is only a matter of transforming this point back to global coordinates.

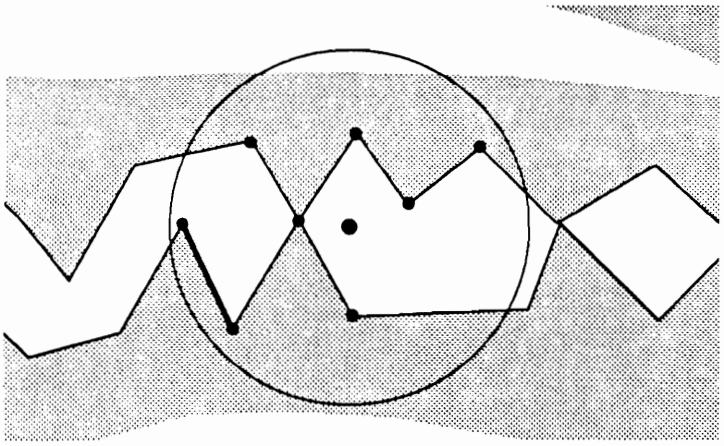
In many instances, the ideal point will be suitable for use in the grid. This would be the case if no existing points were near the ideal third point. However,

this will not always be the case. If new points were always added to the front, the algorithm would never close upon itself. Thus, provisions must be included to use existing points in the generation of elements. This is accomplished by selecting an existing point on the generation front near the ideal third point and using it as the true third point of the new element.

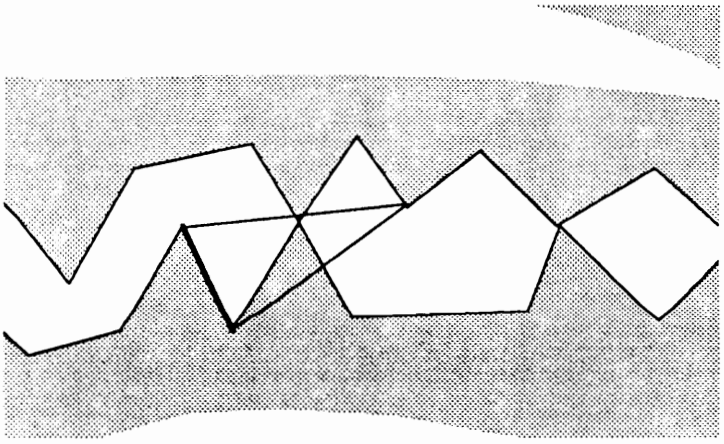
Thus after calculating an ideal third point, one must find all of the points on the front which are within some prescribed radius of the ideal third point (figure 33(a)). In specifying some search radius, it is likely that more than one point will satisfy this condition as demonstrated in figure 33(a). One must order the selected points according to how closely they match the third point. Starting with the point most closely matching the ideal third point, a check must be made to see if an element can be constructed without destroying the integrity of the front.

The integrity of the front will be destroyed if any of the faces on the generation front cross one another. Figure 33(b) depicts the use of a third point which is not suitable for the generation of a new element. However, unless some face crossing algorithm is employed to make sure that no intersections occur, nothing in the current selection process will prohibit the use of such an element. If the generation front is initially intact, it will only be necessary to check the newest faces against all of the other faces of the generation front. Note that no faces which lie interior to the front need be checked for intersection.

The ordered list of points must be checked sequentially until a point is found which allows the addition of an element without destroying the integrity of the front. Figure 33(c) depicts a valid existing point for the current face. If a point can be found, the generation front must be updated to reflect the deletion of the current face as well as any faces on the front which may intersect the new element and to

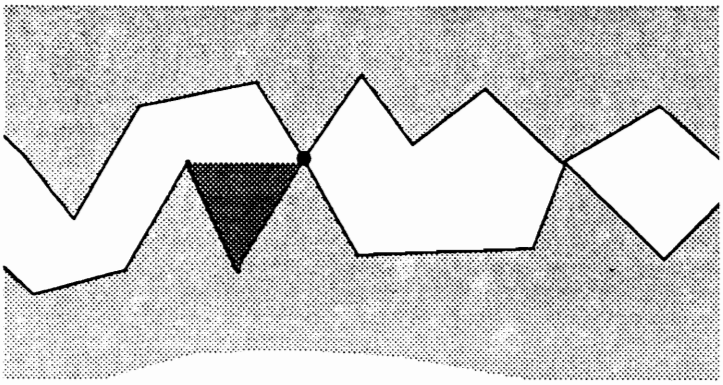


(a) Search for existing points near an ideal point

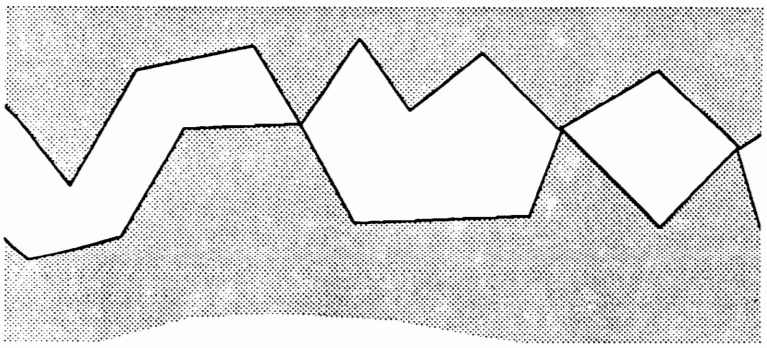


election of a bad point for the generation of a new

Figure 33: Intermediate steps of advancing front algorithm.



(a) Appropriate selection of an existing third point



(d) Generation front after update.

Figure 33: Intermediate steps of advancing front algorithm. (cont.)

reflect the addition of any new faces. Figure 33(d) represents the local generation front after the addition of the element described in figure 33(c).

If no existing point can be found the original ideal point must be checked to see if its use will cause destruction of the front integrity. If not, it may be used to generate a new element. If it is not possible to use the ideal third point, the search radius may be increased to find points close to the ideal point and the loop may be continued. In two-dimensions, it will always be possible to find an existing point on the front for use in generating a new element. However, its use might not form an element which closely matches the desired element orientation and distribution.

To reiterate, the process of selecting a face, generating an element adjacent to it, and updating the generation front is iterated upon until the entire domain has been discretized and no faces are left on the generation front.

The information presented thus far represents a skeleton algorithm. The selection of tolerances and the exact order of implementation vary from one implementation to the next. Listed below are examples of the choices available in any particular implementation of this algorithm.

The search radius for existing points close to the ideal point should be large enough to keep from adding unnecessary points yet small enough to approximately maintain the desired cell orientation and size. However, given these constraints, there is still flexibility in the selection of this tolerance. Slight variations in this value will result in distinct discretizations, all of which may satisfy grid quality constraints. Once the points close to the ideal point have been found, there are several options in ordering these points. This ordering can be keyed upon how close the point is to the ideal point, or how closely the resulting element matches the ideal element given the local stretching parameters. Another option would be to order points according to how close to being Delaunay the resulting triangulation would

be. If the ideal point cannot be used, some variations of the algorithm attempt to move the ideal element closer to the current front face in an effort to find a suitable point. This step would be included with the larger search for existing points.

Most variations on the basic algorithm will result in acceptable grids given the desired distribution parameters. However, these differences will result in different numbers of elements generated as well as the number of nodes introduced.

Since speed improvements for this algorithm cannot be obtained through the use of vectorization, careful attention must be given to the efficiency of the individual algorithms. Much searching must be done and thus considerable attention has been given to this area. Some of the uses for an efficient searching algorithm follow.

The element in the background grid which surrounds a particular point must be found. This search is necessary to interpolate local stretching parameters for a particular front face. An efficient means for locating all points in the vicinity of a given point must be available. This algorithm is necessary to find all points on the front near the ideal point mentioned above. In addition, one must find all faces on the front which intersect a given element. This routine is necessary to check for the integrity of the front when a new element is introduced.

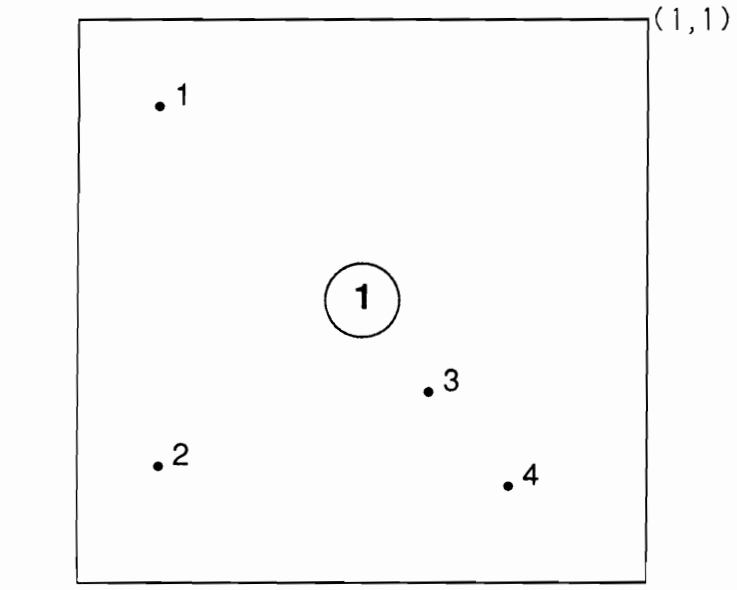
In the above algorithms, the searches can be performed in a two step fashion. In the first step, all points within a given rectangle are found in an efficient manner. Then a more thorough test of the resulting points or faces can be performed to satisfy the conditions of the particular algorithm. The quad-tree structure described in Appendix B has proved most useful in all of these algorithms.

Appendix B: Quad-Tree Storage Technology

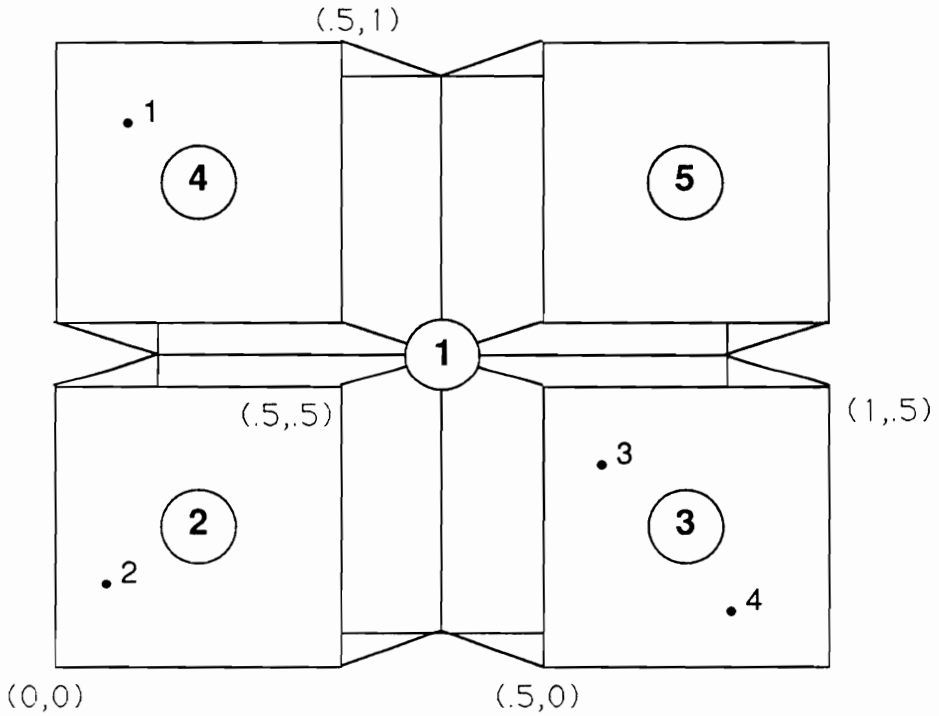
The quad tree is a storage structure which aids in sorting points in n-dimensional space[57]. After points have been placed in a quad-tree structure, efficient routines may be implemented which use this structure to aid in the search for points within n-dimensional volumes and points near specified n-dimensional points. Since there is no inherent logical connectivity to the generalized spatial discretizations as there is with a structured spatial discretization, these tree structures significantly aid in the search algorithms for near neighbors and face intersections necessary for solution algorithms and grid generation.

The quad-tree structure is most easily explained by describing the process of filling such a structure. The following example describes the filling of a two-dimensional quad tree. Points are placed one at a time into a structure which contains four storage locations(figure 34(a)). At initialization all of the four storage locations are empty. One by one the points are placed into empty storage locations until all four storage locations have been occupied.

One must note that the domain must have some minimum and maximum values in two dimensions which completely bound all of the points which will be placed into this structure. Typically, the grid generation is performed in coordinates normalized from $< 0, 1 >$ such that the original structure will be a unit square with origin at $(0, 0)$. With all four storage locations occupied, each dimension of the structure is divided by two. Thus in two dimensions there will be four subdivisions (figure 34(b)). Each of these four subdivisions becomes a new quad-tree structure with four storage locations of its own. The four points which were in the original structure are saved, and the four original storage locations are used to store the location of the four new structures. Finally the four original points are placed into the subquadrants which bound the particular point.

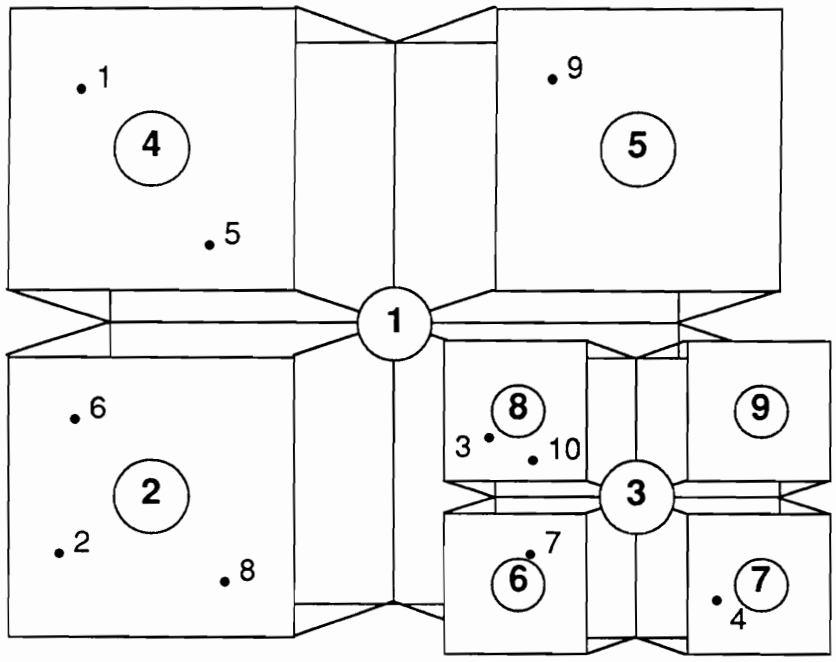


(a) Initial quad structure just before subdivision



(b) Initial quadrant just after subdivision.

Figure 34: Example of quad-tree structure.



(c) Quad tree structure after several subdivisions.

Figure 34: Example of quad-tree structure.(cont.)

With the four points properly placed, additional points can continue to be added. Since the initial quadrant has been subdivided, the points are placed in the respective bounding subquadrant. In a fashion similar to the original quadrant, when one of the subquadrants becomes filled (with four points) that quadrant is then subdivided into four additional quadrants, and their locations are stored in the subdivided quadrant's four storage locations. The four displaced points are then placed in the appropriate bounding subquadrant. The process of placing points and subdivision continues until all of the points have been placed in the structure. Figure 34(c) represents the quad-tree structure at some intermediate state.

This placement of the points into the quad-tree is a preprocessing step in preparation for search algorithms which will use the resulting structure. If the points which are to be sorted remain unchanged, this filling process need only be performed once. If the coordinates of a point change, this is equivalent to removing that point and adding a new point with the changed coordinates. If this is the case, the quad structure must be adjusted to account for the loss of the original point, and the new point must be added to the structure in the prescribed fashion.

With all points placed within the appropriate quad structures, search algorithms may be implemented. The first algorithm to be considered will be the search for a point near prescribed point. This algorithm may be used in the advancing front grid generator to find the background element surrounding a particular point. If the centroids of the elements in the background grid are placed in a quad-tree structure, a quad-tree search algorithm can be used to find an element close to a given point. This algorithm will not find the closest point, merely one in the correct neighborhood. With this cell as a seed to a more specific search algorithm, the cell actually surrounding the element may be found.

Finding a point in a quad-tree structure close to a specified point is a simple task. One must step down into a quad tree staying in subquadrants which surround the specified point until a quadrant with no further subdivision is reached. Any points within this quadrant will be close to the specified point relative to all other points.

An additional algorithm to be considered is the search for all points within a certain n-dimensional volume (a rectangle in two dimensions). This routine is useful for finding all points on the generation front within some radius of a calculated ideal point. If all of the points on the current front are placed in a quad-tree structure, the points in a given rectangle algorithm can be used to efficiently find all close points surrounding a particular point. These points may then be individually checked to see if they are within the prescribed radius.

Finding all points within a given volume becomes a more efficient task when a tree structure is employed. If a quadrant does not intersect the given volume, no points within that quadrant are within that volume, and none of the subquadrants within that quadrant need to be checked. If a quadrant does intersect with a given volume, only the subquadrants within that quadrant which intersect the volume need to be checked. When a quadrant intersecting the volume is reached which contains only points, each point must be individually checked to see if it lies within the volume.

When properly implemented, the use of a quad-tree structure for these search routines results in algorithms which are order $N\ log(N)$.

VITA

The author was born in Oak Ridge, Tennessee on September 4, 1964. He entered the Aerospace and Ocean Engineering program at Virginia Polytechnic Institute and State University in Blacksburg, Va in 1982. While at VPI&SU he participated in the cooperative education program as a student engineer at the David Taylor Naval Research and Development Center. He received his Batchelor of Science degree, Magna Cum Laude in June, 1987. He immediately entered the graduate program in Aerospace Engineering at VPI&SU. The author is currently employed as a research scientist at AeroSoft, Inc. in Blacksurg, Va.