SSSU 101
ISSN 0140 3818

UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF SHIP SCIENCE

FACULTY OF ENGINEERING AND APPLIED SCIENCE

THREE-DIMENSIONAL MULTIBLOCK GRID GENERATION: FLEXIMESH

N.C. Rycroft and S.R. Turnock

Ship Science Report 101

November 1997

3-D Multiblock Grid Generation : Fleximesh

N.C.Rycroft, S.R.Turnock
Dept. of Ship Science,
University of Southampton.

November 7, 1997

Abstract

The success of field methods in computational fluid dynamics is dependent upon the quality of the computational mesh used to represent the physical domain. This report details the theoretical basis and method of use of a powerful multiblock grid generator. The approach allows complex three-dimensional domains to be rapidly meshed whilst giving significant control over the local quality of the grid. The program 'Fleximesh' produces a multiblock mesh of hexahedral cells for input to a range of flow solvers including those using unstructured data. Written in ANSI C it utilises flexible data structures and dynamic memory allocation to maximize the potential problem size.

Ship Science Report No. 101

Contents

1 Introduction **5**

2 Mesh Construction **8**

2.1 Introduction 8

2.2 Node Definition 8

2.3 Edge Definition 8

2.4 Face Definition 10

2.5 Block Object Definition 10

2.6 Block Linking 10

3 Spline Definition **13**

3.1 Introduction 13

3.2 Cubic Spline Derivation 13

3.3 Parametric Cubic Splines 15

3.4 Line Point Distributions 15

4 Transfinite Interpolation **17**

4.1 Introduction 17

4.2 Theory 17

4.3 Interior Interpolations 18

5 Elliptical Refinement **21**

5.1 Introduction 21

5.2 Discretisation of Poisson Equation 21

5.3 Source Functions 22

5.4 Elliptical Refinement Solutions 23

6 Grid Singularities **27**

7 Use of Fleximesh **30**

7.1 Introduction 30

7.2 Input File Preparation 30

7.3 Output File Format 32

7.4 Using 'Fleximesh' 32

8 Case Study 1 : NACA 0012 Wing **34**

8.1 Introduction 34

8.2 Stage 1 : Simple Solid Wing Geometry 34

8.3 Downstream Region 34

8.4 Wing Tip 37

9 Future Developments **43**

10 Appendices **45**

10.1 Parametric Cubic Spline Algorithm 45

10.2 Transfinite Interpolation Algorithm 46

10.3 Elliptical Refinement Algorithm 48

10.4 Surface Definition 49

10.5 NACA 0012 Wing Input File 51

List of Figures

1 Grid Topologies . . . 7
2 Geometrical Object Hierarchy . . . 9
3 Edge to Face Mapping . . . 11
4 Edge to Face to Block Mappings . . . 11
5 Spline Definition . . . 14
6 Line Distributions . . . 16
7 Transfinite Interpolation Domain Definition . . . 18
8 Transfinite Interpolation of Simple Bound Regions a . . . 19
9 Transfinite Interpolation of Simple Bound Regions b . . . 19
10 Extension of Line Distributions into the Interior Using Transfinite Interpolation . . . 20
11 Elliptical Refinement of a Square with Two Line Sources . . . 24
12 Elliptical Refinement of Cube with a Point Source . . . 25
13 Elliptical Refinement of Cube With Point and Line Sources . . . 25
14 C-Grid Wrapped Around a NACA 0012 Aerofoil . . . 26
15 Elliptically Refined Solution of Grid Around NACA 0012 Aerofoil . . . 26
16 Computational Domains For Blocks Containing Various Degrees of Singularities . . . 28
17 Cubic Singularity Example a . . . 28
18 Cubic Singularity Example b . . . 29
19 Face to Block Mapping Input Definitions . . . 31
20 Grid and Block Structure for Blocks 0 and 1 . . . 35
21 Solid View of NACA 0012 Aerofoil without Wingtip . . . 36
22 External Faces of Resultant Grid in Block 0 of NACA 0012 Wing . . . 36
23 External Faces of Resultant Grid in Blocks 0 and 1 of NACA 0012 Wing . . . 38
24 Grid and Block Structure for Blocks 0,1,2, and 3 . . . 39
25 External Faces of Resultant Grid in Blocks 0,1,2, and 3 of NACA 0012 Wing . . . 40
26 Final Grid And Block Structure Including Wing Tip . . . 41
27 Wing Tip Detail for NACA 0012 Aerofoil . . . 42
28 External Faces of Final Grid Around NACA 0012 Wing . . . 42
29 Surface File Definition . . . 50

1 Introduction

Popular fields methods such as: finite difference, finite element, and finite volume; require the physical domain in which the problem is posed to be completely and wholly discretised. The accuracy of the numerical solution is partly determined by the nature of the mesh used to represent the physical domain. This is especially apparent within the Computational Fluid Dynamics (CFD) field, where numerical errors, dependent upon the quality of the mesh, become visible in the flow solution. As the sizes of the problems have increased, and the nature of the flows become more complex, the mesh generation challenge has had to adapt to accomodate these developments.

Structured mesh generation is a popular and successful method of discretising domains. The method is based upon a direct mapping from the physical domain to a computational domain. Boundary points prescribed within the physical domain are used to interpolate the interior points within the computational domain. The major advantage of structured meshes is that the implicit mapping inherently stores the connectivity of the mesh maintaining low memory overheads. The way in which one arranges the mapping is defined by the problem under investigation. Three strategies for generating a mesh around an aerofoil section are shown in Figure(1). The 'C-Grid' strategy generates a mesh which follows the topology of the aerofoil and the flow within the domain, and is the best solution for this geometry. The aerofoil surface is mapped onto a complete side of the computational domain using the 'O-Grid' strategy. This simplifies boundary condition treatment, but does not allow the mesh to conform to the natural flow. The 'H-Grid' can be exploited to model multiple body problems. By imposing identical flow on the top and bottom boundaries the domain can be used to model a stack of aerofoils.

This strategy has been utilized for turbine blade models. The method is restricted to cell topologies which obey the mapping. Thus only quadrilaterals and hexahedra can be used for two and three dimensional problems respectively. A restriction on the domain complexity which can be meshed accurately is also imposed using this method.

Unstructured grid generation techniques refer to methods which use no implicit mapping to store the connectivity of the mesh. The restrictions that structured generation techniques suffer from are thus alleviated and the process becomes more versatile. Domains are not restricted in their complexity and allow any cell topology to be used. Triangular or tetrahedral meshes are widely used for unstructured meshes. Using the Bower-Watson algorithm [3] [11] a complex domain can be triangulated retaining excellent control over the mesh quality and distribution. This extra versatility allows more efficient meshes to be produced reducing the effort required to solve a flow problem. The requirement to store the connectivity of the mesh along with the coordinate data substantially increases the storage requirements compared to those required for structured meshes.

Hybrid meshes are similar to unstructured meshes in that they use an identical method of data storage. The only difference, in fact, is that hybrid meshes are constructed using more than one cell topology. The advantage of this is that different cell topologies may be placed in regions of the flow regime where they are most suited.

The multiblock technique is an powerful extension to structured mesh generation allowing larger, more complex, and better quality meshes to be produced. Structured mesh techniques are applied to sub-grids or 'blocks' which are then linked together to produce a much larger mesh. The blocks remain structured, that is they map into a computational domain, however the resultant mesh does not necessarily have to adhere to the same mapping.

This document describes the grid generation and refinement techniques used to create blocks and the way in which the data is handled to provide an automatic block linking process. The possibility of constructing blocks using different mesh generation techniques to produce Hybrid meshes is investigated. This could be a major benefit for Reynolds Averaged Navier Stokes (RANS) calculations where a structured grid is preferred in regions where the viscous stresses are dominant and more efficient unstructured grids are preferred in areas where convective fluxes become dominant.

The program, 'Fleximesh' is written in ANSI C utilizing the ability to create complex data structures with great ease. Dynamic memory allocation is used to maximize the program versatility by not constricting the amount of memory used in each generation stage. This allows the user the versatility to maximize the memory use in regions where it is required, allowing very large meshes to be constructed on local workstations.

The program has been developed as part of a PhD project aimed towards producing a three- dimensional finite volume flow solver working with unstructured data types and able to cope with any cell topology. Meshes produced using 'Fleximesh' have been successfully used in a finite volume scheme solving the Euler equations.

Section 2 describes the block construction and introduces the geometrical objects used to describe and generate the grid; Section 3 introduces cubic splines as a method of curve definition, and includes the mathematical derivation of a cubic spline; Section 4 introduces and defines transfinite interpolation as an efficient and accurate method of constructing interior points given the location of boundary points in two and three dimensions; Section 5 describes elliptical refinement as a method of smoothing and controlling grid quality; Section 7 describes the use of 'Fleximesh' including input and output formats; Section 8 uses the NACA 0012 aerofoil geometry to demonstrate the abilities of 'Fleximesh'; and finally Section 9 concludes the report and introduces some future developments to 'Fleximesh'.

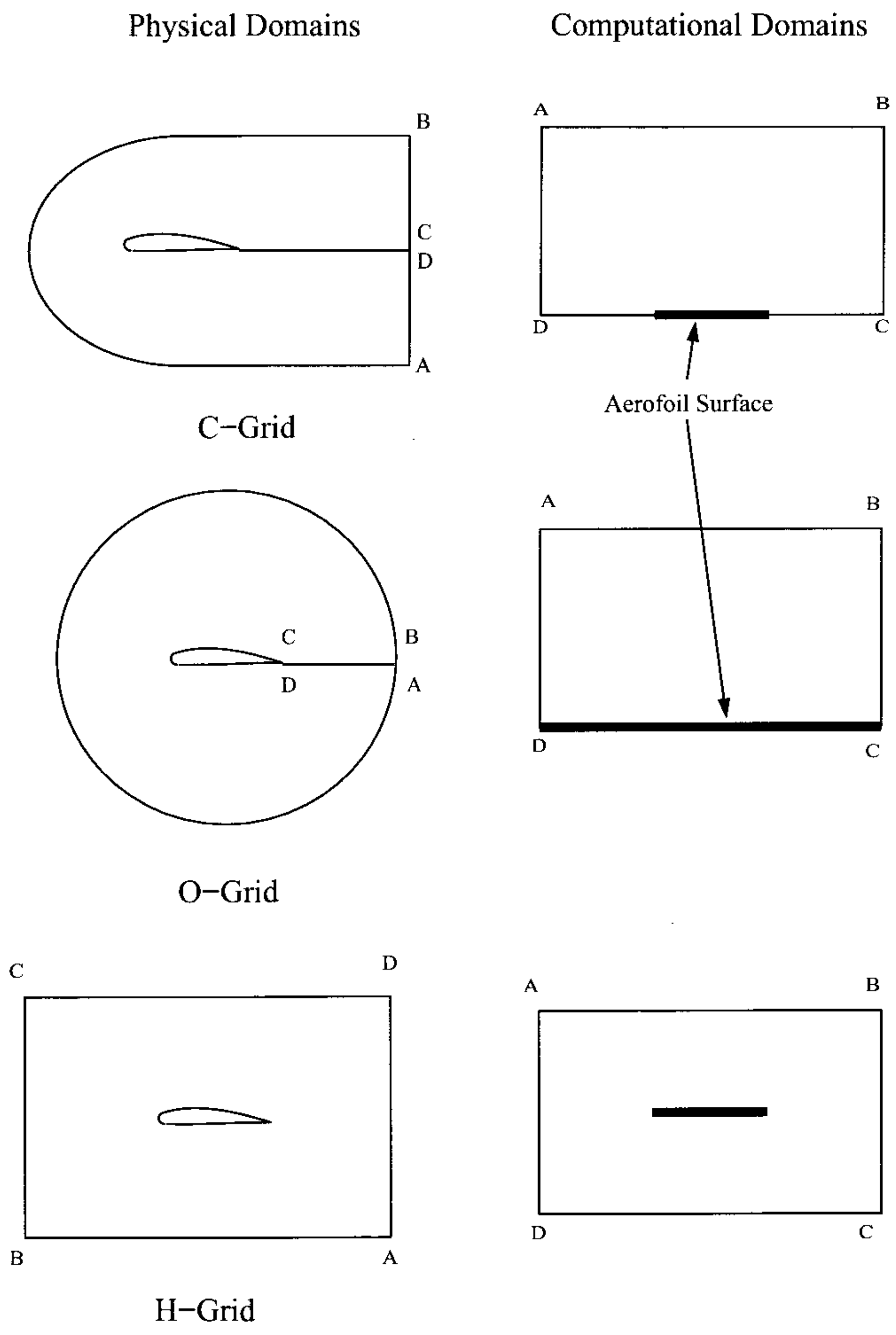


Figure 1: Physical and Computational Domains for Three Grid Topologies

2 Mesh Construction

2.1 Introduction

The *Grid* is produced by linking together a series of 'sub-grids' or '*Blocks*' which are generated separately. The key to the multiblock technique is the way in which the linking process is achieved such that no mesh discrepencies occur. This section describes the approach used to generate each *Block* separately whilst significantly simplifying the linking process.

Each *Block* is created using structured mesh generation techniques, using pre-defined boundary point positions to interpolate interior points. The definition of the boundary points is not simple. Thus the side points of a *Block* are generated using the same methods applied within a two-dimensional computational domain. The sides are subsequently mapped onto the *Block* side. However the generation of the sides requires the boundaries of the side to be defined. The generation of the boundary nodes is achieved in stages using a hierarchy of *Geometrical Objects*. *Nodes* are defined as points in space. Using these points or *Nodes*, *Edges* are defined by prescribing the *Nodes* which the *Edge* connects. *Edges* are used to define the sides of a *Face* and finally *Faces* are mapped onto the sides of a *Block*. The generation of each of the *Geometrical Objects* allows the boundary points on a higher order object to be defined. Thus the definition of points along an *Edge* allows the sides of a *Face* to be defined. These boundary points are then used to generate the interior points of the *Face* which are subsequently used to define boundary points on the sides of *Blocks*.

Nodes are the lowest ranking object and are defined by the user. All the subsequent objects can be created in their most simplest state automatically. Thus it is possible to generate a three dimensional *Block* by defining the eight *Nodes* which are the corner points of the *Block*. By defining how many points are required along the *Edges* the number of points on the *Faces* and in the *Blocks* are determined. The *Object* construction is achieved in two stages: the first being a mapping stage where a number of previously defined lower ranking *Objects* are mapped onto the boundaries of the higher ranking *Object*; and then a generation stage in which the boundary points are used in an interpolation routine to construct the interior points. This *Object* is then ready to be mapped onto a higher ranking *Object*. The only *Object* that this routine does not apply to is the *Node* which is the lowest ranking *Object* and has to be explicitly defined by the user. The *Geometrical Objects* form a hierachial structure shown in Figure(2).

2.2 Node Definition

The *Node* object is the lowest in the hierachial structure and contains coordinate data and a global identification. These are the basic points in space which describe key positions on the physical geometry and the *block* structure.

2.3 Edge Definition

The *Edge* object is defined by two end *Nodes*. However this would restrict the nature of the *Edge* to be a set of linear lines joining the end *Nodes*. The grid generator is a much more powerful tool if curves can be mapped onto the *Edge* object, essential for the description of basic engineering shapes such as aerofoils and ship hulls. Splines are used to provide an inexpensive yet accurate method for describing complex curves. In order to construct the spline a number of construction points are required along the curve. These points are not *Nodes*, rather a series of temporary coordinate locations. Once a spline has been constructed through these points they become redundant and the spline provides the means to generate the interior *Nodes* along the *Edge*. If no construction points are defined the new

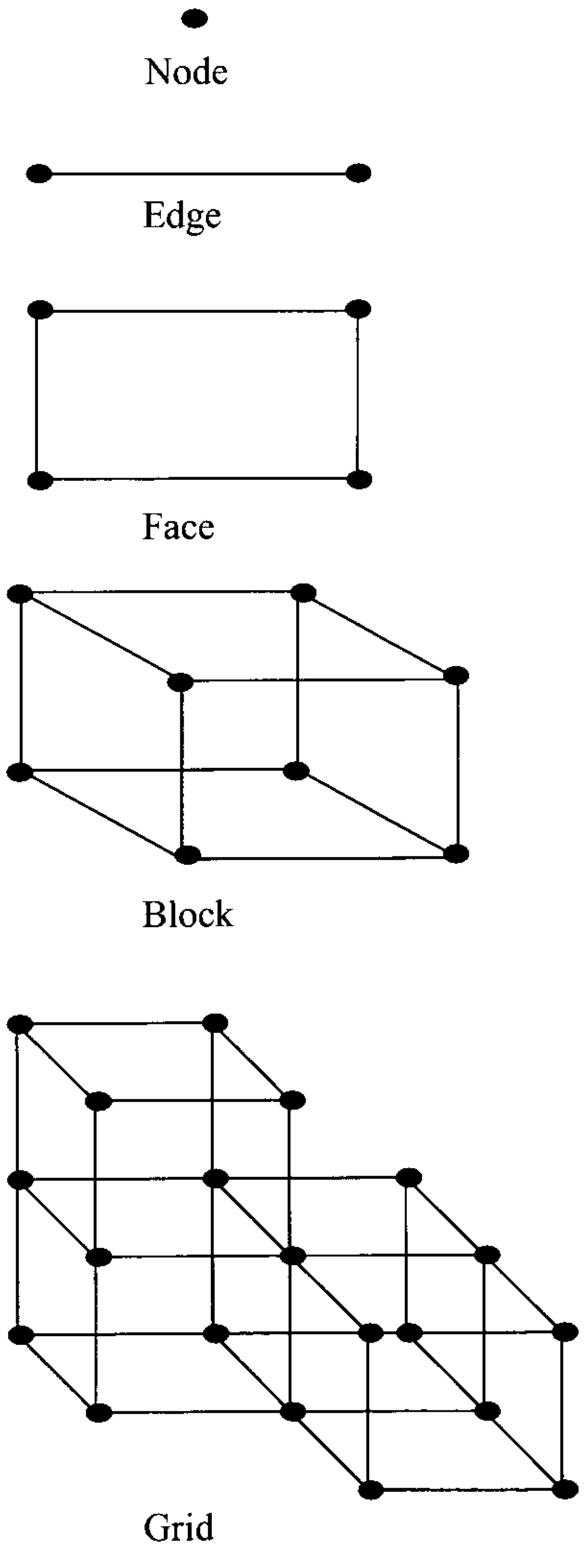


Figure 2: Geometrical Object Hierarchy

interior *Nodes* are constructed using linear interpolation between the two end *Nodes.* By allowing the interior points along an *Edge* to be constructed using splines, the complexity of shapes which can be represented increases. However, this is at the expense of increasing the amount of user defined information required by the process.

2.4 Face Definition

The first stage of *Face* construction is the *Edge* to *Face* mapping. The mapping procedure requires knowledge of how and where to place the *Edge* along the sides of the *Face.* The identity of the *Edge* and the direction in which it lies on top of a particular side has to be defined. However, the mapping does not restrict the number of *Edges* that map onto one side of the *Face.* Therefore any number of *Edges* can be mapped onto a *Face* side as long as the identities of the *Edges*, the directions of the mappings, and the order in which they occur along the *Face* side, are known. This increases the complexity of the overall *Block* construction but allows much more versatility in the *Block* definition. An *Edge* to *Face* mapping is described in Figure (3).

The second stage of the *Face* construction is the generation of the interior points. Once the defining *Edges* have been mapped onto the sides of the *Face* correctly, the interior *Nodes* can be found using two methods. In most cases Transfinite Interpolation, described in Section 4, is used to generate the interior *Nodes* based upon the boundary point distributions. However there are cases where the *Face* represents a complex surface which cannot be constructed through the transfinite interpolation of the bounding curves. In such circumstances a surface can be defined using a series of splines across the *Face.* As with the definition of *Edges* using splines, this process requires a list of construction points for each spline. Each spline must have enough construction points to adequately define the curve they are reconstructing, and there must be enough splines to accurately describe the curve of the surface in the opposite direction. For a complex surface such as a wing tip this requires a large number of construction points and thus this method of *Face* construction is only used when absolutely necessary. The construction points for such a complex surface can be derived ,for example, from a surface panel generator [9] or a ship lines code [1]. Usually the surface can be defined adequately using transfinite interpolation of bounding curves. Construction points are reallocated along the construction splines. The new curves are then used in an interpolation routine to construct the new interior *Nodes.* This process can be followed in more detail in Section(10.4).

2.5 Block Object Definition

The *Block* object is described by the six *Faces* that make up the hexahedral domain that a *Block* maps onto. It is possible to map more than one *Face* to a *Block* side using a system of 'Patches'. However the complexity of the initial input was deemed to be too complex without some sort of visual aid and the singular mapping was retained. The mapping of *Faces* to the sides of the *Block* requires three elements of information. As well as the direction of the *Face* on the *Block* side, each *Face* has a relative rotation to produce a total of eight different mappings. Thus the identity, direction, and position of the *Face* has to be defined for a correct mapping. Figure(4) shows the *Edge* to *Face* to *Block* mappings indicating the origins of the *Faces* and *Block.*

2.6 Block Linking

There is no need for a physical *Block* linking process since it is accomplished automatically through the unique identification system within the Geometrical Objects. Starting from the lowest ranking object, the *Node*, each of the new *Nodes* created has a unique identification number. When each object is used in a mapping procedure, the coordinates and the unique identification number are mapped

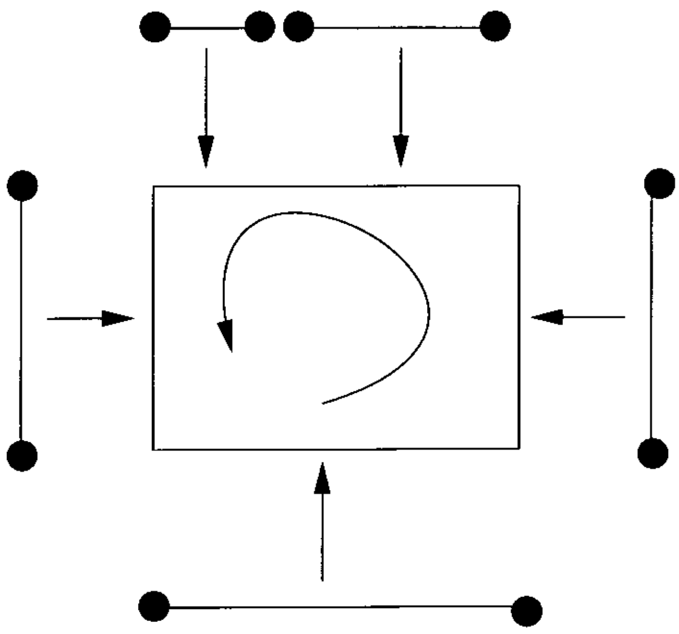


Figure 3: Edge to Face Mapping

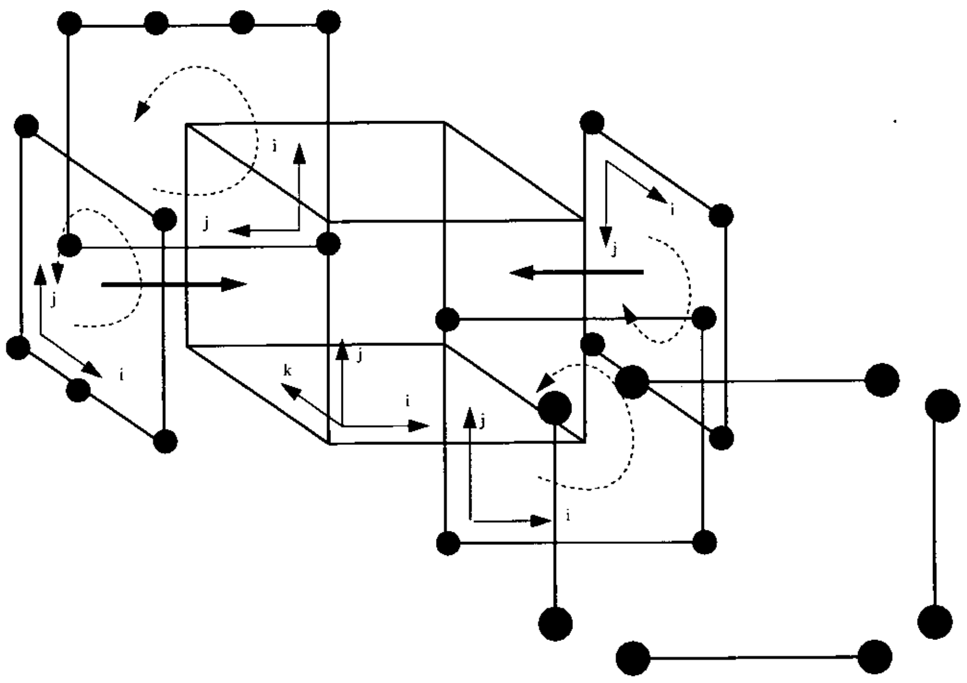


Figure 4: Edge to Face to Block Mappings

onto the higher ranking object. Thus the *Nodes* at the ends of an *Edge* retain the coordinates and identification of the *Nodes* used to define the *Edge*, and similarly the *Nodes* along the sides of a *Face* retain the coordinates and identifications of the *Edges* that were used to define the *Face.* The corner *Nodes* of the *Face* will in fact retain the coordinates and identities of the *Nodes* used to define the *Edges* that define the *Face.* This results in the construction of a global list of *Nodes* which are never replicated. *Blocks* can be defined using the same *Faces.* Thus the identities and coordinates of the *Nodes* on adjoining *Block* sides are automatically equivalent. This is the major motivation towards using this *Geometrical Object* hierarchy system within the grid generation process.

The output data format is a full unstructured data set. No inherent *Node* connectivity is assumed and thus has to be stored along with the coordinate data. A full unstructured data set refers to the inclusion of all geometrical objects such as nodes, edges, faces, and cells, in order to fully define the connectivity of the grid. Obvious similarities can be made between the full unstructured data set which is used as an output format and the way in which the *Grid* is initially defined. As a consequence of using this data format, the extra connectivity information has to be generated within the generation process. The *Node* construction has already been demonstrated and *Nodes* are obviously equivalent to the node structure in the unstructured data set. However during an *Edge* construction the edge structure has now to be correctly formed. This is a simple case of defining an edge by the *Nodes* that define it and allocate a unique identification to the structure. Both the edge and face structures have to be generated during the *Face* construction. These structures can easily be defined from the two dimensional array of *Nodes* on the *Face.* Similarly edge, face, and cell structures have to be defined during the *Block* construction using the three dimensional array of *Nodes.* Thus the unstructured data set is constructed as each of the geometrical objects is constructed.

3 Spline Definition

3.1 Introduction

Polynomials have been widely used to approximate other functions. However they suffer from the following serious drawbacks:

- High Degree polynomials may oscillate strongly.
- Polynomial interpolation is very sensitive to the choice of interpolation points.
- If the function to be approximated is badly behaved anywhere, the the approximation is poor everywhere.

The global dependence on local properties can be relieved by using a series of polynomial approximations. Splines combine a series of low order polynomials, to obtain a function which is as smooth as possible without it being a global polynomial. The cubic spline has been utilized within the grid generation procedure to accurately model curves that may be found in engineering situations. A example of such a curve is an aerofoil section which, using cubic splines, can be regenerated using relatively few data points. The cubic spline fits a cubic polynomial between each set of defining data points. The cubic splines are equal at the data points and the spline is thus continuous. If the gradient and the curvature are also assumed to be continuous then the spline can be derived. Other choices of curve representation such as Bezier curves are not as useful due to the non-direct mapping between the defining points and the resultant curve.

3.2 Cubic Spline Derivation

If a set of data points is defined a unique cubic polynomial can be defined between each set of points.

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ S_1(x) & x \in [x_1, x_2] \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ S_{n-1} & x \in [x_{n-1}, x_n] \end{cases} \tag{1}$$

where $S(x)$ is the cubic spline and S_i are the cubic polynomials between each set of points $[x_i, x_{i+1}]$. The cubic polynomials are equal at the defined end points and thus

$$S_{i-1}(x) = y_i = S_i(x), \quad i = 1, 2, \ldots, n-1 \tag{2}$$

Since S_i is a cubic polynomial over $[x_i, x_{i+1}]$, S_i'' is a linear function and is therefore given by a straight line between its end points.

$$S_i''(x) = \frac{S_i''(x_i)}{x_{i+1} - x_i}(x_{i+1} - x) + \frac{S_i''(x_{i+1})}{x_{i+1} - x_i}(x - x_i) \tag{3}$$

If this is integrated twice an expression for S_i is found.

$$S_i(x) = \frac{S_i''(x_i)}{6(x_{i+1} - x_i)}(x_{i+1} - x)^3 + \frac{S_i''(x_{i+1})}{6(x_{i+1} - x_i)}(x - x_i)^3 + C(x - x_i) + D(x_{i+1} - x) \tag{4}$$

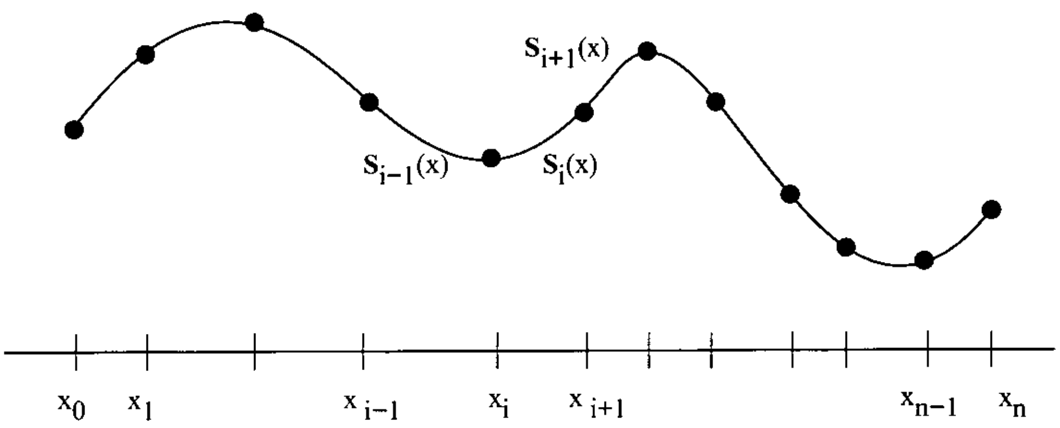


Figure 5: Spline Definition

where C and D are constants of integration. The known values of $S_i(x_i)$ and $S_i(x_{i+1})$ can be used to find the constants of integration.

$$\begin{aligned} C &= \left(\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{S_i''(x_{i+1})(x_{i+1}-x_i)}{6}\right) \\ D &= \left(\frac{y_i}{x_{i+1}-x_i} - \frac{S_i''(x_i)(x_{i+1}-x_i)}{6}\right) \end{aligned} \tag{5}$$

It is clear that if values of $S_i''(x_0)$, $S_i''(x_1)$, $S_i''(x_2)$, ..., $S_i''(x_{n-1})$ are known then $S(x)$ may be evaluated for any x within the interval $[x_0, x_n]$. In order to evaluate $S_i''(x_0)$, $S_i''(x_1)$, $S_i''(x_2)$, ..., $S_i''(x_{n-1})$ it is assumed that the gradient is continuous along the spline. The gradient $S_i'(x)$ can be found by differentiating Equation(4) and a discrete function $S_i'(x_i)$ found by substituting $x = x_i$.

$$S_i'(x_i) = -\frac{x_{i+1}-x_i}{3}S_i''(x_i) - \frac{x_{i+1}-x_i}{6}S_i''(x_{i+1}) - \frac{y_i}{x_{i+1}-x_i} + \frac{y_{i+1}}{x_{i+1}-x_i} \tag{6}$$

and by substituting $x = x_{i-1}$

$$S_{i-1}'(x_i) = \frac{x_i - x_{i-1}}{6}S_i''(x_{i-1}) + \frac{x_i - x_{i-1}}{3}S_i''(x_i) - \frac{y_{i-1}}{x_i - x_{i-1}} + \frac{y_i}{x_i - x_{i-1}} \tag{7}$$

The two equations can be equated to find

$$\begin{aligned} (x_i - x_{i-1}) \quad & S_i''(x_{i-1}) + 2((x_{i+1}-x_i) + (x_{i+2}-x_{i+1}))S_i''(x_i) + (x_i - x_{i-1})S_i''(x_{i+1}) = \\ & \frac{6}{x_{i+1}-x_i}(y_{i+1}-y_i) - \frac{6}{x_i - x_{i-1}}(y_i - y_{i-1}) \end{aligned} \tag{8}$$

This equation is used for $i = 1, 2, 3, \ldots, n-1$ and provides a system of $n-1$ linear equations for the $n+1$ unknowns $S_i''(x_0)$, $S_i''(x_1)$, $S_i''(x_3)$, ..., $S_i''(x_n)$. $S_i''(x_0)$ and $S_i''(x_n)$ are chosen arbitrarily to complete the system of linear equations. A 'Natural Cubic Spline' is found when $S_i''(x_0) = S_i''(x_n) = 0$. The resultant linear system of equations for $1 \le i \le n-1$ with $S_i''(x_0) = S_i''(x_n) = 0$ is symmetric, tridiagonal, diagonally dominant, and of the form

$$\begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & h_2 & u_3 & h_3 & & \\ & \cdot & \cdot & \cdot & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \cdot \\ \cdot \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ V_{n-2} \\ V_{n-1} \end{bmatrix} \tag{9}$$

where $h_i = x_{i+1} - x_i$, $u_i = 2(h_i + h_{i+1})$, $V_i = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$, and $M_i = S_i''(x_i)$.

3.3 Parametric Cubic Splines

The cubic spline theory presented in the previous section describes a cubic spline function S as a function of x. A more convenient method of constructing cubic splines is to use a parametric function $\eta(x, y, z)$ where η is a function of the distance along the curve. The function η is approximated by calculating the linear distance between each of the discrete defining points.

$$\eta_i = \sum_{j=0}^{i-1} \sqrt{(x_{j+1} - xj)^2 + (y_{j+1} - yj)^2 + (z_{j+1} - zj)^2} \tag{10}$$

The parametric function can be normalized for values between 0 and 1 which provides a simple method of defining new points along the spline. The parametric function is calculated for each of the defining points using Equation(10). The cubic spline theory is now defined in terms of the new parameter. A spline function is calculated such that x,y, and z values can be found for a value η with the spline equations now in terms of η and $\tilde{r} = (x, y, z)^T$.

$$S_i''(\eta) = \frac{S_i''(\eta_i)}{\eta_{i+1} - \eta_i}(\eta_{i+1} - \eta) + \frac{S_i''(\eta_{i+1})}{\eta_{i+1} - \eta_i}(\eta - \eta_i) \tag{11}$$

$$\widetilde{S_i}(\eta) = \frac{S_i''(\eta_i)}{6(\eta_{i+1} - \eta_i)}(\eta_{i+1} - \eta)^3 + \frac{S_i''(\eta_{i+1})}{6(\eta_{i+1} - \eta_i)}(\eta - \eta_i)^3 + \tilde{C}(\eta - \eta_i) + \tilde{D}(\eta_{i+1} - \eta) \tag{12}$$

$$\begin{aligned} \tilde{C} &= \left(\frac{\tilde{r}_{i+1}}{\eta_{i+1} - \eta_i} - \frac{S_i''(\eta_{i+1})(\eta_{i+1} - \eta_i)}{6} \right) \\ \tilde{D} &= \left(\frac{\tilde{r}_i}{\eta_{i+1} - \eta_i} - \frac{S_i''(\eta_i)(\eta_{i+1} - \eta_i)}{6} \right) \end{aligned} \tag{13}$$

and the coefficients in Equation(9) become $h_i = \eta_{i+1} - \eta_i$, $u_i = 2(h_i + h_{i+1})$, $V_i = \frac{6}{h_i}(\tilde{r}_{i+1} - \tilde{r}_i) - \frac{6}{h_i - 1}(\tilde{r}_i - \tilde{r}_{i-1})$, and $M_i = S_i''(\eta_i)$.

3.4 Line Point Distributions

The normalized parametric formulation of points distributed along a line enables a convenient method of defining new positions for points along the length of the line. By defining a value of η a new x, y, z position may be calculated either using the parametric cubic spline formulation or using simple linear interpolation. Distributions between the values of 0 and 1 can be defined using various methods. Sine and Cosine functions provide useful distributions due to their natural bounds. However these functions cannot always provide the desired distribution along a line and complicated combinations have to be applied. Alternative functions have been developed capable of providing versatile distributions using simple inputs. One such function, developed by Roberts[6] and modified by Eiseman[8], is:

$$s = p\eta^* + (1 - p)\left(1 - \frac{\tanh[q(1 - \eta^*)]}{\tanh q}\right) \tag{14}$$

where $\eta^* = \frac{\eta - \eta_1}{\eta_n - \eta_1}$, a normalized parametric function and p and q are parameters which control the distribution of points along a parametric series. p controls the slope of the distribution and q provides a damping effect by forcing the function to become less linear. This function can provide a wide range of versatile distributions by simply varying the values of p and q.

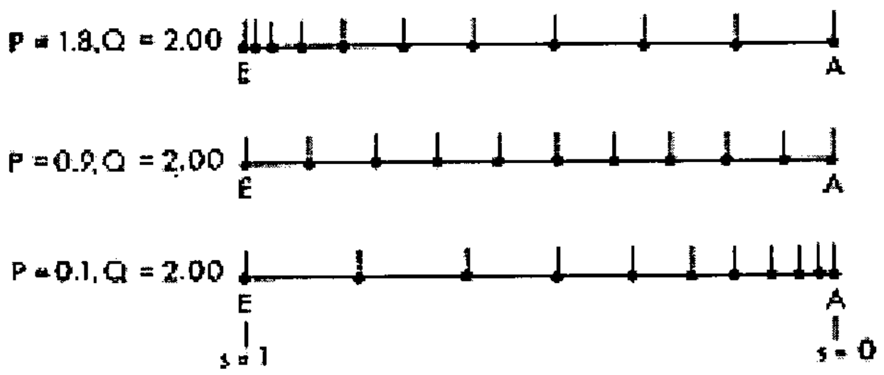


Figure 6: Line Distributions. [5]

4 Transfinite Interpolation

4.1 Introduction

Transfinite Interpolation[7] provides a method for linearly interpolating a region by specifying a continuous mapping on all of the boundaries with an interpolation in the ξ and η directions introduced in the interior. Thus unlike conventional interpolation methods, the mappings on all of the boundaries affect the interior interpolation.

4.2 Theory

Normalized parametric coordinates r and s are introduced in the ξ and η directions. Interpolation functions are then defined as :

$$\Phi_j(r) = \delta_{jr}, \quad j = 0, 1. \qquad \Psi_k(s) = \delta_{ks}, \quad k = 0, 1. \tag{15}$$

where $\delta_{jr} = 1$ if $j = r$ and 0 if $j \neq r$ and $\delta_{ks} = 1$ if $k = s$ and 0 if $k \neq s$.

Thus referring to Figure(7): $\Phi_0 = 1$, $\Phi_1 = 0$ on AD; $\Phi_0 = 0$, $\Phi_1 = 1$ on BC; $\Psi_0 = 0$, $\Psi_1 = 1$ on AB; and $\Psi_0 = 1$, $\Psi_1 = 0$ on CD. Interpolation in the r and s directions can be expressed as :

$$\begin{aligned} Z_r(r,s) &= \Phi_0(r)Z_{AD}(0,s) + \Phi_1(r)Z_{BC}(1,s) \\ Z_s(r,s) &= \Psi_0(s)Z_{AB}(r,0) + \Psi_1(s)Z_{CD}(r,1) \end{aligned} \tag{16}$$

where Z_{AD}, Z_{BC}, Z_{AB}, Z_{CD} are continuous mappings between the (ξ, η) and (x, y) planes on the four boundaries. $Z_r(r,s)$ and $Z_s(r,s)$ are the continuous mappings produced by interpolating between two opposite boundaries. These are equivalent mappings used in the simplest interpolation routines. In order to extend this method of to a two-dimensional interpolation a product interpolation is defined as :

$$Z_{rs}(r,s) = Z_r \cdot Z_s \tag{17}$$

This product interpolation is only compatible with the boundary functions at the four corners. To obtain an exact matching along all of the boundaries of a two-dimensional domain it is necessary to define a boolean sum interpolation:

$$Z(r,s) = Z_r(r,s) + Z_s(r,s) - Z_{rs}(r,s) \tag{18}$$

This boolean sum interpolation is central to transfinite interpolation.

The interpolation is in practice accomplished in two stages. The first stage simply calculates the one of the one-dimensional functions.

$$Z_r(r,s) = \sum_{j=0}^{1} \Phi_j(r) Z_b(j,s) \tag{19}$$

where b indicates the appropriate boundary. The second stage completes equation (18).

$$Z(r,s) = Z_r(r,s) + \sum_{k=0}^{1} \Psi_k(s)[Z_b(r,k) - Z_r(r,k)] \tag{20}$$

Transfinite Interpolation can easily be extended to three dimensions by adding a natural third stage.

$$Z(r,s,t) = Z_2(r,s,t) + \sum_{i=0}^{1} \Omega_i(t)[Z_b(r,s,i) - Z_2(r,s,i)] \tag{21}$$

where $Z_2(r,s,t)$ is equivalent to $Z(r,s)$.

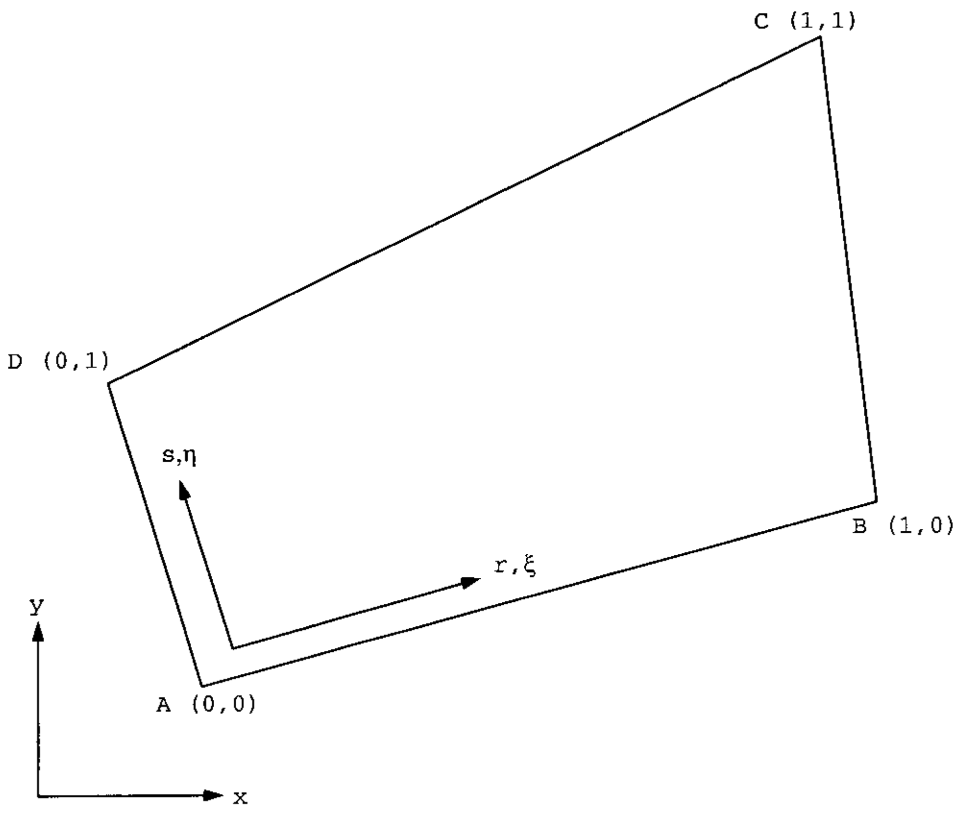
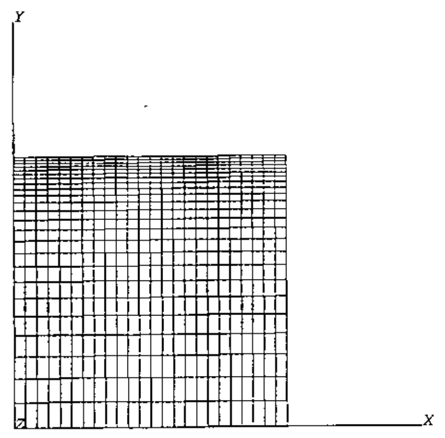


Figure 7: Transfinite Interpolation Domain Definition

4.3 Interior Interpolations

Transfinite interpolation uses continuous mappings specified along all boundaries in order to interpolate the interior mesh points. Thus the distributions along the *Edges* used to map onto the sides of a *Face* determine the interior distribution. Similarly the *Face* distributions determine the three-dimensional mesh distribution. It is clear that the line distributions used to generate *Edges*, determine the interior distributions on *Faces* and thus *Blocks*. The importance of the spline routine and line distribution functions, now become apparent.

Figures(8,9) demonstrate the generation of interior points using the transfinite interpolation method. Line distributions have been used to concentrate points along the boundaries. These distributions have been extended into the interior through the interpolation routine. Figure(10) shows the effect of using opposite line distributions along opposite boundaries. The distributions along the boundaries are extended into the interior to produce a highly skewed mesh.



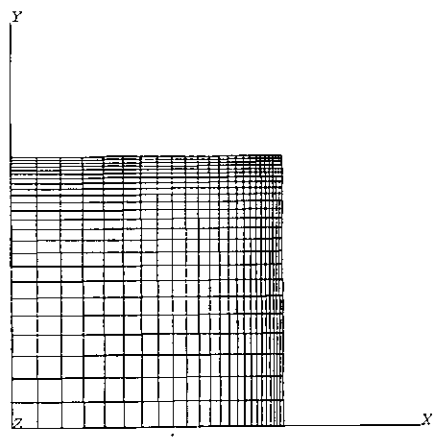
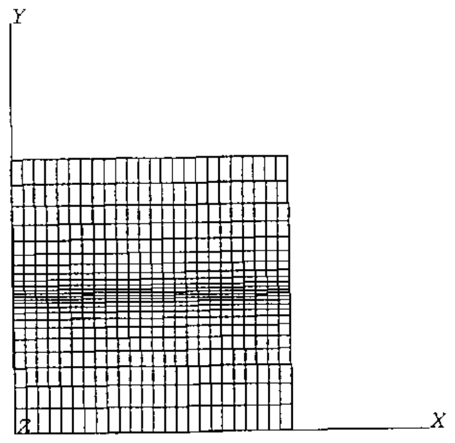


Figure 8: Transfinite Interpolation of Simple Bound Regions.



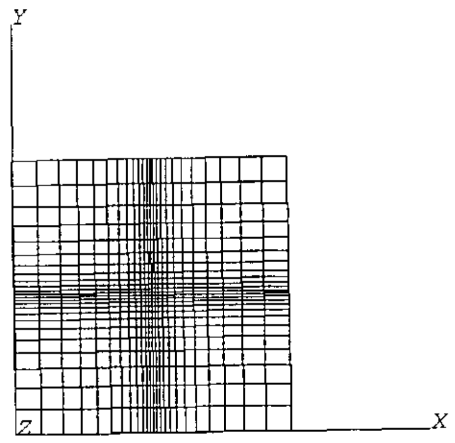


Figure 9: Transfinite Interpolation of Simple Bound Regions Using Split Spline Option

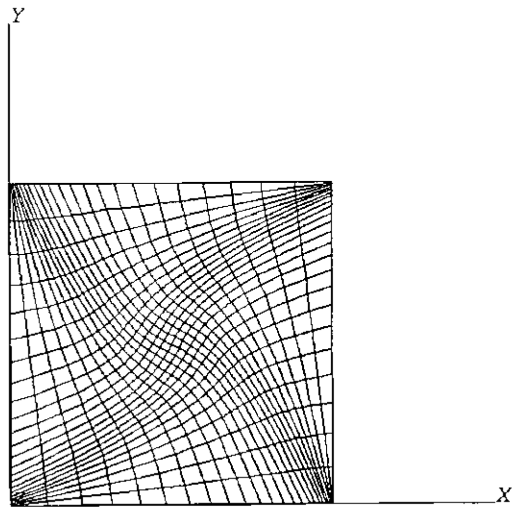


Figure 10: Extension of Line Distributions into the Interior Using Transfinite Interpolation.

5 Elliptical Refinement

5.1 Introduction

A linearly interpolated grid is often not of sufficient quality for some flow calculations where the grid quality is paramount in the flow solution process. It is therefore necessary to refine the grid in order to minimize errors in the flow code. This project includes an elliptical grid refinement module in an attempt to improve the grid quality. Elliptical grid refinement uses the Poisson equation to smooth the grid in the interior with Dirichlet fixed boundary points. The problem is solved in a generalized coordinate space, constructed such that a computational domain boundary coincides with a physical boundary. The distorted physical space is mapped onto either a rectangular or hexahedral space within the computational domain. The governing equations are transformed into and discretised in the computational domain, such that the generalized coordinates become the independent variables.

5.2 Discretisation of Poisson Equation

$$\begin{aligned}
\frac{\partial^2\xi}{\partial x^2}+\frac{\partial^2\xi}{\partial y^2}+\frac{\partial^2\xi}{\partial z^2} &= P(\xi,\eta,\zeta) \\
\frac{\partial^2\eta}{\partial x^2}+\frac{\partial^2\eta}{\partial y^2}+\frac{\partial^2\eta}{\partial z^2} &= Q(\xi,\eta,\zeta) \\
\frac{\partial^2\zeta}{\partial x^2}+\frac{\partial^2\zeta}{\partial y^2}+\frac{\partial^2\zeta}{\partial z^2} &= R(\xi,\eta,\zeta)
\end{aligned} \tag{22}$$

where the sources P, Q, and R are known source functions used to control the distribution of points within the grid. In the computational space Equation(22) transforms to

$$\alpha_{11}r_{\xi\xi}+\alpha_{22}r_{\eta\eta}+\alpha_{33}r_{\zeta\zeta}+2(\alpha_{12}r_{\xi\eta}+\alpha_{13}r_{\xi\zeta}+\alpha_{23}r_{\eta\zeta}) = -J^2(Pr_\xi+Qr_\eta+Rr_\zeta) \tag{23}$$

where $r=(x,y,z)^T$, $\alpha_{ij}=\sum\gamma_{mi}\gamma_{mj}$, and γ_{ij} is the ij^{th} cofactor of the matrix

$$M=\begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \tag{24}$$

and J is the determinate of the matrix M.

Equation(23) can be solved using an iterative method such as a Gauss-Siedel with a point 'Successive Over Relaxation', or SOR, for solution acceleration. This is demonstrated for the two dimensional equations below.

$$\alpha_{11}r_{\xi\xi}+\alpha_{22}r_{\eta\eta}+2(\alpha_{12}r_{\xi\eta}) = -J^2(Pr_\xi+Qr_\eta) \tag{25}$$

where $r=(x,y)^T$, and $\alpha_{ij}=\sum\gamma_{mi}\gamma_{mj}$, and γ_{ij} is the ij^{th} cofactor of the matrix

$$M=\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \tag{26}$$

and J is the determinate of the matrix M.

Thus using the finite difference approximations below Equation(25) can be discretised.

$$r_\xi = (r_{i+1,j}-r_{i-1,j})/2$$

$$\begin{aligned} r_\eta &= (r_{i,j+1} - r_{i,j-1})/2 \\ r_{\xi\xi} &= (r_{i-1,j} - 2r_{i,j} + r_{i+1,j}) \\ r_{\eta\eta} &= (r_{i,j-1} - 2r_{i,j} + r_{i,j+1}) \\ r_{\xi\eta} &= (r_{i+1,j+1} - r_{i-1,j+1}, -r_{i+1,j-1} + r_{i-1,j-1}) \end{aligned} \tag{27}$$

Using these approximations the α coefficients and determinate J can be expressed as

$$\begin{aligned} \alpha_{11} &= (x_\eta)^2 + (y_\eta)^2 = \left(\frac{(x_{i+1,j} - x_{i-1,j})^2}{4} + \frac{(y_{i+1,j} - y_{i-1,j})^2}{4}\right) \\ \alpha_{22} &= (x_\xi)^2 + (y_\xi)^2 = \left(\frac{(x_{i,j+1} - x_{i,j-1})^2}{4} + \frac{(y_{i,j+1} - y_{i,j-1})^2}{4}\right) \\ \alpha_{12} &= -(x_\xi x_\eta + y_\xi y_\eta) \\ &= -\left\{\left(\frac{(x_{i+1,j} - x_{i-1,j})}{2}\right)\left(\frac{(x_{i,j+1} - x_{i,j-1})}{2}\right) + \left(\frac{(y_{i+1,j} - y_{i-1,j})}{2}\right)\left(\frac{(y_{i,j+1} - y_{i,j-1})}{2}\right)\right\} \end{aligned}$$

and

$$\begin{aligned} J^2 &= (x_\xi y_\eta - x_\eta y_\xi)^2 \\ &= \left\{\left(\frac{x_{i+1,j} - x_{i-1,j}}{2}\right)\left(\frac{y_{i,j+1} - y_{i,j-1}}{2}\right) - \left(\frac{x_{i,j+1} - x_{i,j-1}}{2}\right)\left(\frac{y_{i+1,j} - y_{i-1,j}}{2}\right)\right\}^2 \end{aligned}$$

Using a Gauss Siedel iterative routine Equation(25) can be written in terms of a new value r^{n+1} where $(n+1)$ denotes a new pseudo time level.

$$\begin{aligned} r_{i,j}^{(n+1)} = \frac{1}{2(\alpha_{11} + \alpha_{22})}\Big\{ & \alpha_{11}(r_{i-1,j} + r_{i+1,j}) + \alpha_{22}(r_{i,j-1} + r_{i,j+1}) + \\ & \alpha_{12}(r_{i+1,j+1} - r_{i-1,j+1} - r_{i+1,j-1} + r_{i-1,j-1}) + \\ & J^2\left(P\frac{(r_{i+1,j} - r_{i-1,j})}{2} + Q\frac{(r_{i,j+1} - r_{i,j-1})}{2}\right)\Big\} \end{aligned} \tag{28}$$

Equation(28) can be accelerated using a point Successive Over Relaxation Scheme (SOR).

$$r_{i,j}^{(n+1)} = (1 - A)r_{i,j}^n + Ar_{i,j}^{pred} \tag{29}$$

where A is the acceleration factor and is typically $A \approx 1.4$. When a large number of sources are used to distort the natural elliptical solution an Successive Under Relaxation Scheme can be used to help convergence where typically $A \approx 0.5$.

5.3 Source Functions

The P, Q, and R functions can be any number of functions producing positive values within the domain. However the functions most commonly used are based on the sum of exponential functions from each source term. The following functions were found by Thompson et al.[10]:

$$\begin{aligned} P(\xi,\eta,\zeta) = &-\sum_{l=1}^{L} a_l sgn(\xi - \xi_l) exp(-c_l|\xi - \xi_l|) \\ &-\sum_{m=1}^{M} b_m sgn(\xi - \xi_m) exp[-d_m\{(\xi - \xi_m)^2 + (\eta - \eta_m)^2 + (\zeta - \zeta_m)^2\}^{1/2}] \end{aligned} \tag{30}$$

$$Q(\xi,\eta,\zeta) = -\sum_{l=1}^{L} a_l sgn(\eta-\eta_l)exp(-c_l|\eta-\eta_l|) \\ -\sum_{m=1}^{M} b_m sgn(\eta-\eta_m)exp[-d_m\{(\xi-\xi_m)^2+(\eta-\eta_m)^2+(\zeta-\zeta_m)^2\}^{1/2}] \quad (31)$$

$$R(\xi,\eta,\zeta) = -\sum_{l=1}^{L} a_l sgn(\zeta-\zeta_l)exp(-c_l|\zeta-\zeta_l|) \\ -\sum_{m=1}^{M} b_m sgn(\zeta-\zeta_m)exp[-d_m\{(\xi-\xi_m)^2+(\eta-\eta_m)^2+(\zeta-\zeta_m)^2\}^{1/2}] \quad (32)$$

where coefficients a_l, b_m, c_l and d_m are chosen such that the grid is clustered in the correct locations. In Equations(30,31,32) L represents the total number of line sources in the ξ, η, and ζ directions respectively and M represents the total number of point sources. A set of coefficients a, b, c and d are entered in order to set dominance of the particular source.

The sgn function has the property:

$$\begin{aligned} sgn(x) &= 1 & x > 0 \\ sgn(x) &= 0 & x = 0 \\ sgn(x) &= -1 & x < 0 \end{aligned} \quad (33)$$

and ensures that points on both sides of the source are under an attractive force. The first term in Equation(30) has the effect of moving ξ=constant lines towards the $\xi = \xi_l$ line, and likewise the first term in Equation(31) has the effect of moving η=constant lines towards the $\eta = \eta_l$ line. The second terms in both of the equations have the effect of attracting ξ=constant lines and η=constant lines towards the point (ξ, η). The a coefficients act as a magnitude for the attraction term whereas the c coefficient acts as a decay term determining the decay of the influence of the source. The effect of the b and d coefficients is much the same acting only on the point sources.

5.4 Elliptical Refinement Solutions

Elliptical refinement can be used to smooth the mesh to improve the quality of the geometrical forms, and also to help concentrate mesh points in regions of maximum flow gradients. In practice the former will often require some mesh point concentration in order to maintain the interior point distributions that result from transfinite interpolation. In fact, elliptical refinement is often a compensatory factor for poor boundary distributions and thus to produce a good quality mesh it is necessary to re-distribute *Nodes* along *Edges* and then smooth using elliptical refinement. The following Figures are designed to show how much influence elliptical refinement can have over the interior *Node* distribution and how it can be used to smooth the mesh over an aerofoil section, demonstrating some of the adverse effects that this can create.

Figure(11) shows a square region with a linear distribution of *Nodes* along the *Edges* after elliptical refinement, with two line sources and a point source. The distribution of the *Nodes* in the interior has been attracted to both the line sources and the point source and is clearly very different from the linear solution. This particular case is used to demonstrate the strong influence of the sources, and not to produce a high quality mesh. The source terms have produced undesirable, highly skewed *Cells* in the interior, and for this naturally orthogonal geometry the linear interpolation would produce the best solution.

Figures(12,13) are used for the same purpose in the three dimensional case. The first depicts a cubic region, elliptically refined with a source term at the centre of the cube. Again it clearly shows how

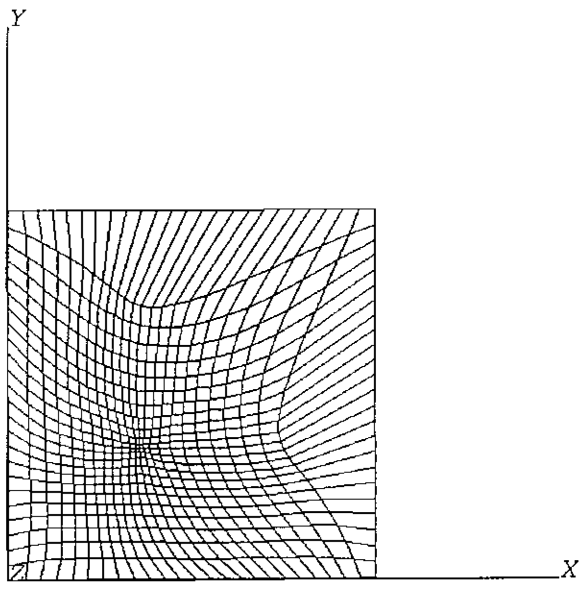


Figure 11: Elliptical Refinement of a Square with Two Line Sources at i=8 and j=8 of Strength=100, and Damping=0.002 and a Point Source at i=12,j=12 of Strength=1200 and Damping=0.0001

the interior mesh *Nodes* have been attracted to the centroid of the cube. If further source terms are added at the centres of all the *Faces* and additional line sources are added on all lines that intersect the centroid then the effect is to attract the mesh *Nodes* onto the three dimensional cross which these lines produce, with the attraction becoming more dominant as the distance from the centroid is reduced.

Figures(14) and (15) show the unrefined and refined solutions for the interior nodal positions on a two dimensional C-Grid *Face*. Transfinite interpolation is used to cluster grid points near to the aerofoil surface and near the trailing edge of the aerofoil which is where one would expect the highest flow gradients. The elliptically refined solution shows how the transfinite solution can be grossly distorted with often adverse effects without carefully choosing correct *Edge* distributions and using sources to control and maintain the desired grid point distributions. The refinement procedure has distorted all the clustering and although it has produced a smooth grid in the interior, the external boundary is obviously highly skewed and distorted. This kind of adverse affect can be compensated for with the combination of changing the *Edge* distributions to produce naturally orthogonal grids, and with the addition of sources to help maintain these *Edge* distributions.

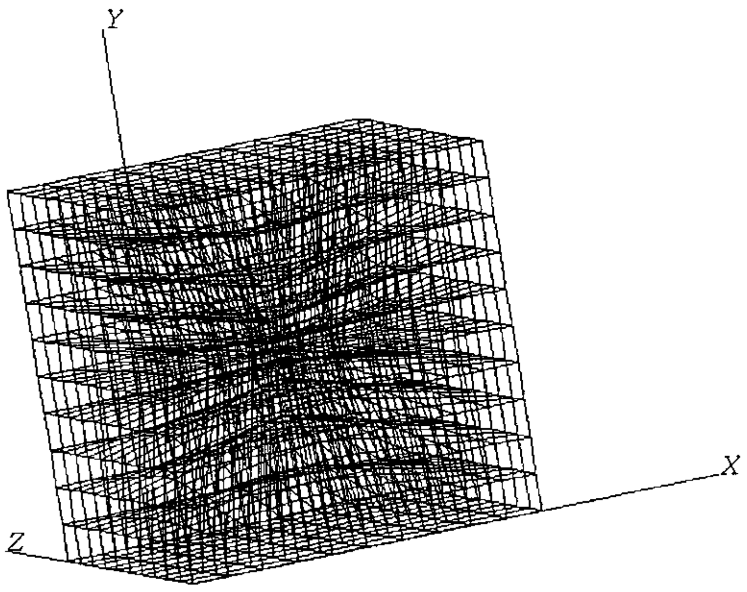


Figure 12: Elliptical Refinement of Cube with a Point Source at $i = 5, j = 5, k = 5$ of Strength=2000 and Damping=0.0001

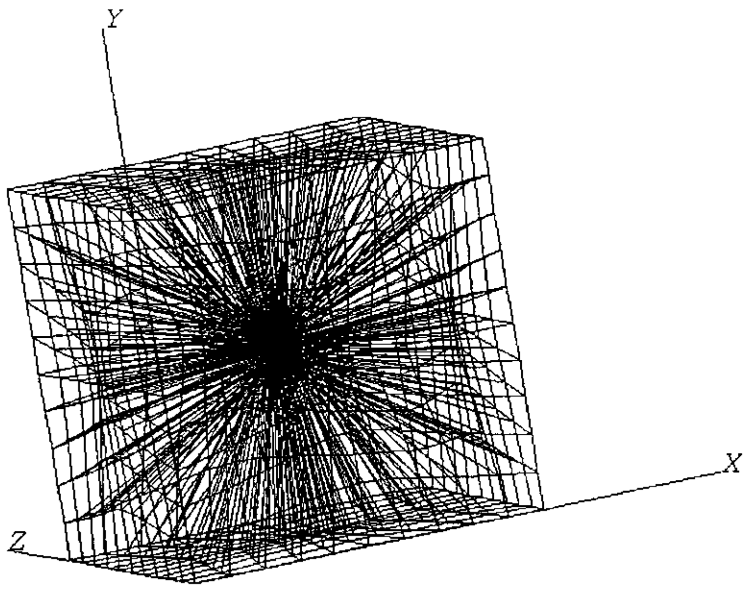


Figure 13: Elliptical Refinement of Cube with a Point Source at the Centre of Each Face and at the Centre of the Cube with Additional Line Sources on Lines that Intersect the Centroid.

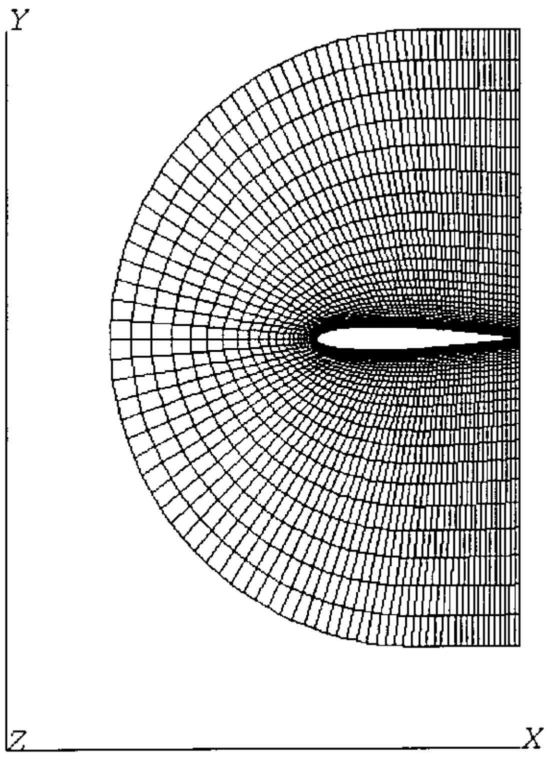


Figure 14: C-Grid Wrapped Around a NACA 0012 Aerofoil

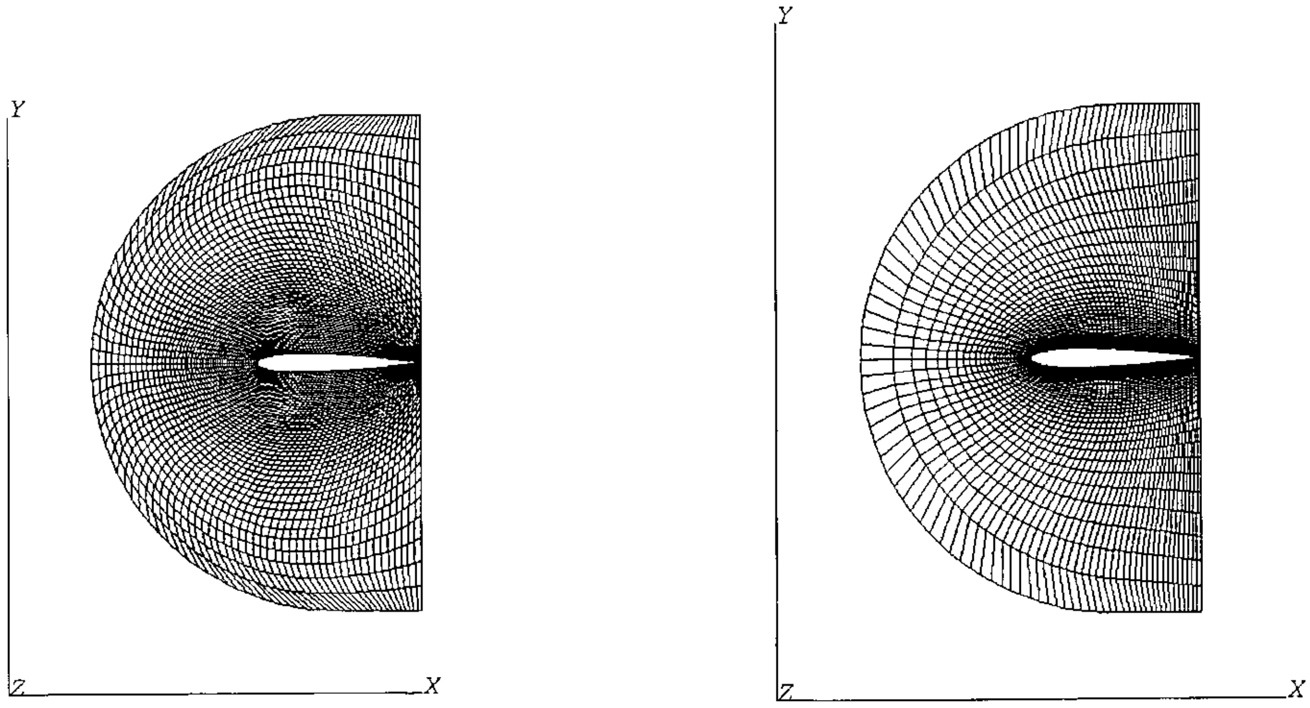


Figure 15: Elliptically Refined Solution With And Without Additional Sources

6 Grid Singularities

The multiblock method is a natural extension of the structured grid generation method enabling some of the restrictions that a structured grid imposes to be overcome. However some physical objects require a combination of grid topologies. The perfect example, and one which is demonstrated as a case study, is that of the wing tip which is typically described using a C-Grid wrapped around both the chord and the span of the wing. These topologies do not naturally join and produce a singularity, resulting in faces with three sides and cells with four or five sides. A simple method of dealing with these singularities within a flow solver is to accommodate edges with zero lengths. However some discussion of how this is generated within the grid generator is necessary. The geometry hierarchy enables us to define an *Edge* using the end *Nodes*. If an *Edge* is defined using only two identical end *Nodes* this indicates a zero length *Edge* is to be produced. The grid generator constructs a list of *Nodes* along the length of this *Edge* with the same coordinates and identities as the end *Nodes*. No increments are made to the global *Node* count. Thus these *Nodes* are invisible replications of the singular *Node* that the *Edge* is constructed from. This is necessary such that the interpolation routines work on the *Face*. However the unstructured data sets are still constructed such that a number of edges will be defined using the same *Node* as their defining *Node* but having unique identities. For a truly unstructured data set this is incorrect but for a structured data set this will enable a flow solver to identify singularities. A simple extension is planned to enable a complete unstructured data set to be constructed, with edges defined by identical *Nodes* to be deleted, faces defined by three edges only to de defined as triangles rather than quadrilaterals, and cells defined using less than six faces and or non-quadrilateral faces to be defined as tetrahedra, prisms, or pyramids. This would incorporate the singularity correctly within the unstructured data set.

To demonstrate the singularity feature Figures(16,17) depict the computational and physical domains of a cube. The closest face if the cube is collapsed by allowing the *Edges* to be defined as zero length *Edges*. The affect this has on the *Faces* of the cube are clearly visible with the final stage depicting the *Face* as a point in physical space.

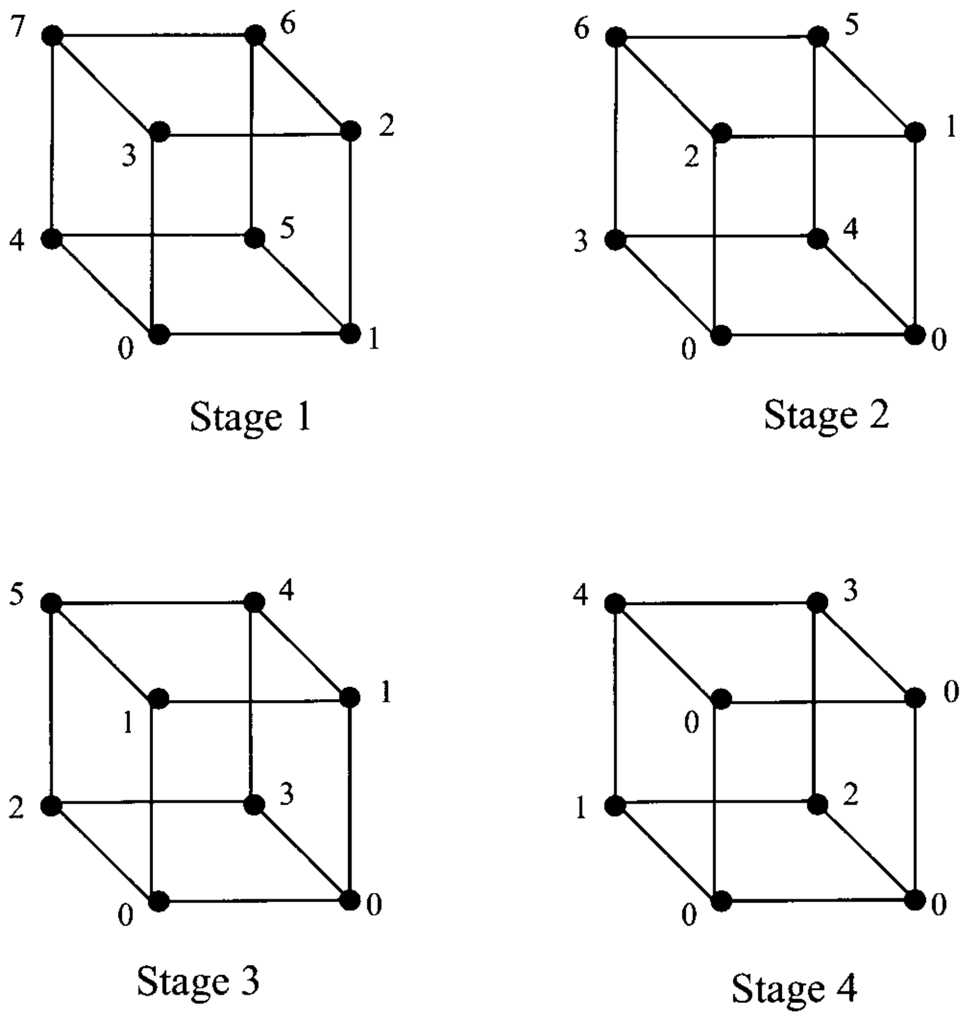


Figure 16: Computational Domains For Blocks Containing Various Degrees of Singularities

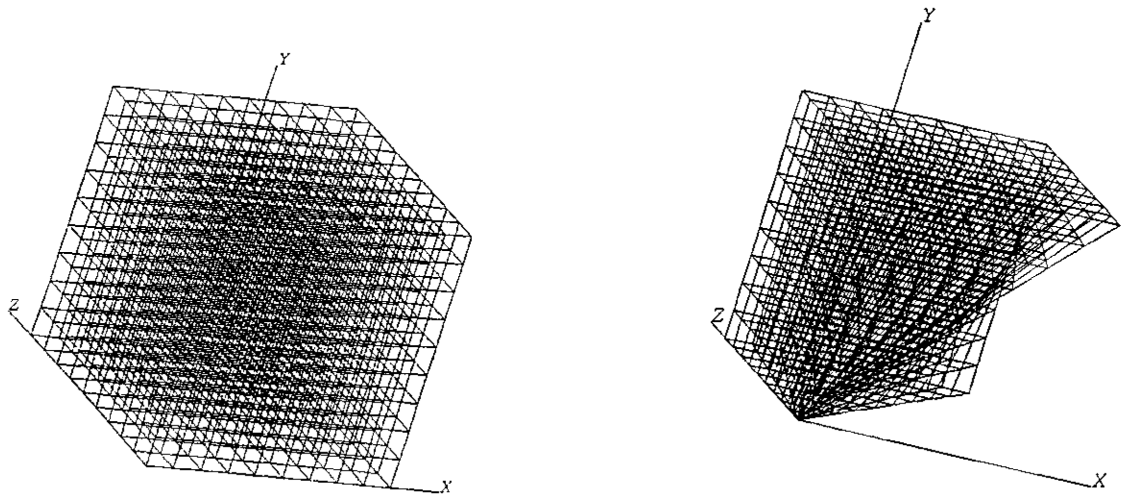


Figure 17: Stages 1 & 2 of Cubic Singularity Example

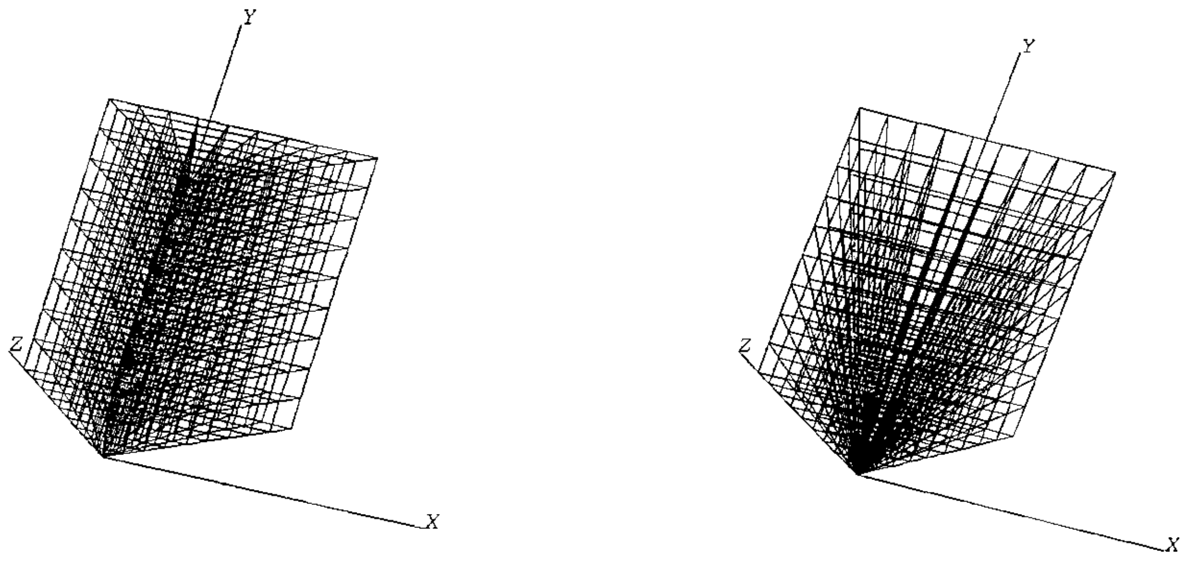


Figure 18: Stages 3 & 4 of Cubic Singularity Example

7 Use of Fleximesh

7.1 Introduction

'Fleximesh' has been developed on and is executed on the Unix operating system. It is an interactive command based program with no windows application. However at each stage of the grid generation process 'Fleximesh' produces '.avs' files which can be viewed using 'Application Visualisation System' (AVS). All the geometrical and connectivity information that's required is inputed from file. 'Fleximesh' constructs the basic linearly interpolated mesh which can then be refined or altered interactively during execution. Once the refining options have been set a script file can be written and 'Fleximesh' may be executed in the background for convenience and speed. Flags can be set in the command line to set the optimisation to memory or speed. The memory optimisation reduces the memory usage during execution by writing to temporary files during execution to reduce the overall memory usage.

7.2 Input File Preparation

The input file contains all the geometrical and connectivity information required to produce the basic linearly interpolated grid. The format of the file is shown below:

string [Description of Grid File]
int [No. of *Nodes*] int [No. of *Edges*] int [No. of *Faces*] int [No. of *Blocks*]

string ['NODE'] int [ID] float [x coord.] float [y coord.] float [z coord.]
...
...
string ['EDGE'] int [ID] int [*Node* 0 ID] int [*Node* 1 ID] int [No. Def. Pts.] int [No. Res. Pts.] int [P] int [Q]
int [Def. Pt. ID] float [Def. pt. x coord.] float [Def. pt. y coord.] float [Def. pt. z coord.]
...
...
string ['FACE'] int [ID] int [Boundary Condition ID]
string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···
string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···
string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···
string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···
string ['SOURCES'] int [No. i Line Sources] int [No. j Line Sources] int [No. Pt. Sources]
string ['I_SOURCES'] int [i location] float [strength of source] float [damping of source] ···
string ['J_SOURCES'] int [j location] float [strength of source] float [damping of source] ···
string ['P_SOURCES'] int [i location] int [j location] float [strength of source] float [damping of source] ···
...
...
string ['BLOCK'] int [ID]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['FACE'] int [Side ID] int [*Face* ID] int [*Face* Direction] int [*Face* Position]
string ['SOURCES'] int [No. i Line Sources] int [No. j Line Sources] int [No. k Line Sources] int [No. Pt. Sources]
string ['I_SOURCES'] int [i location] float [strength of source] float [damping of source] ···
string ['J_SOURCES'] int [j location] float [strength of source] float [damping of source] ···

string ['K_SOURCES'] int [k location] float [strength of source] float [damping of source] · · ·
string ['P_SOURCES'] int [i location] int [j location] int [k location] float [strength of source] float [damping of source]
. . .
. . .
. . .

The relative position and direction of the *Face* to *Block* mapping is defined using an integer code. Definitions of the codes are shown in Figure(19). The data file for the NACA0012 aerofoil is included in Section(10.5) as an example.

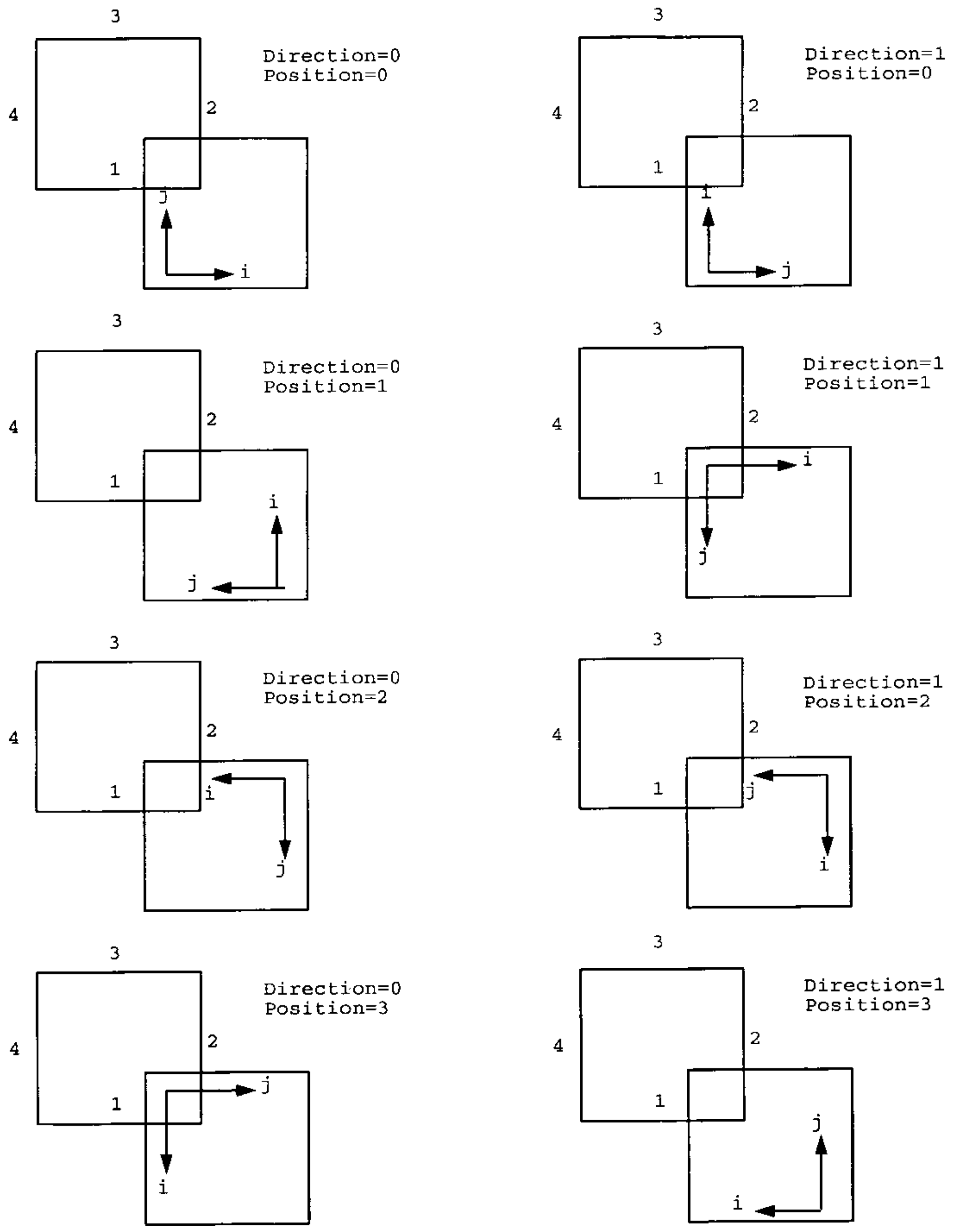


Figure 19: Face to Block Mapping Input Definitions

7.3 Output File Format

'Fleximesh' outputs two files of different formats. The first is an AVS '.imp' file or as labeled from 'Fleximesh', '.ucd'. This file format can be used to view the mesh or the external faces of the mesh using AVS. It can also be used as a neutral format for input into other applications. The format is well documented in the 'Developers Guide'[2]. The second of the output formats is a complete unstructured data set. The extension used for this format is '.flw' or for a decomposed mesh suitable for use in a distributed data application '.flwpp' although this decomposition facility is not built into 'Fleximesh' at present. The file format is given below :

```
string [Grid Description]

int [No. of Nodes] int [No. of Edges] int [No. of Faces] int [No. of Cells]
int [Node ID] float [Node x coord.] float [Node y coord.] float [Node z coord.]
...
...
int [Edge ID] int [Node 0 ID] int [Node 1 ID]
...
...
int [Face ID] int [Face Boundary ID] int [No. Edges] int [Edge ID] int [Edge Direction] ···
...
...
int [Cell ID] int [Cell Type] int [Face ID] int [Face Direction] ···
...
...
```

7.4 Using 'Fleximesh'

'Fleximesh' runs in a shell tool and is a prompt based program. The program is executed using the command 'fleximesh' from the command prompt. This command runs a script which executes the main program named 'Blockgen'. 'Blockgen' issues a set of prompts, the first of which is the name of the input file.

As of yet, there is no windows application to aid the construction of the input file and the task remains a non-trivial procedure. With the aid of this document, the case study, and examples included within the archive file, the user should become familiar with the input file definition before attempting complex grid constructions. It is recommended that a number of simple grids are constructed successfully before a user attempts to tackle larger problems. Complex geometries are often meshed more successfully when approached in stages of complexity. Only a limited number of bugs within the input file will be spotted by the program, and it is this file which is the most likely source of error in the event of a premature exit. It is also recommended that until a visual application has been developed that the user adopt a historical approach using pen and paper to visualize and debug the input file. Input file bugs reported as program bugs will not be appreciated.

'Blockgen' will automatically input the file and start the mesh construction. At the start of the procedure 'Blockgen' sets up the *Grid* structure using the global number of *Nodes*, *Edges*, *Faces*, and *Cells*. Each individual *Geometrical Object* is subsequently constructed starting with the lowest ranking objects. *Edges* are constructed automatically with no option to alter the distribution of points along the splines. *Faces* are initially constructed through an *Edge* to *Face* mapping, followed by the construction of the internal *Nodes* through transfinite interpolation. A temporary file named 'face< *face_id* >.tmp' is produced which can now be viewed using AVS. 'Blockgen' now prompts the user for either continuation onto the next *Face* or alteration and/or refinement of the present *Face*. If the *Face* is refined the maximum error for the iteration is displayed on the screen such that the refinement process can be monitored. If convergence fails the cycle will repeat using a Successive Under Relaxation parameter. In the event of divergence, the cycle will be terminated and the transfinite interpolated solution will be maintained. It is possible to change the input file at this stage to try and improve the solution. Following the completion of *Face* construction, 'Blockgen' continues onto the *Block* construction. A similar procedure as that used for the *Face* construction is used. *Faces* are initially mapped onto the sides of the *Block* followed by the interpolation of the interior *Nodes*. Similar options exist for the refinement and alteration of the *Block*. Each individual *Block* mesh may be viewed using files 'block< *block_id* >.tmp'. All temporary files are deleted at the completion of the execution.

Following the construction of all of the *Geometrical Objects* the *Grid* is written to the two output files described in Section 7.3.

'Fleximesh' does allow some default values to be set. These include both the SOR and SUR parameters, and convergence criteria. These can be stored in a file named 'elliptic.default', stored in the same directory as the executable. However, if this file does not exist 'Fleximesh' uses a set of default values.

'Fleximesh' is able to run in two modes. The default mode optimises the execution for speed whereas the second mode optimizes the execution for memory by placing some of the geometrical information into temporary files during execution, to reduce the memory usage. Experience has shown that the memory optimisation is rarely required due to the optimized data handling within the program.

A limited number of error messages are recorded in a log file name 'blockgen.log' which can be found in the executable directory.

8 Case Study 1 : NACA 0012 Wing

8.1 Introduction

A three dimensional zero sweep wing constructed using the NACA 0012 aerofoil section is used as a case study demonstrating the abilities of the grid generator to grid complex shapes. However this case study does not test the generator to the extreme. In fact as will be seen, only eight *Blocks* are used to generate the resultant grid. The only restriction on the number of geometrical objects that can be defined are the size of available memory of the workstation and the input file complexity. It should also be noted that the resultant grid is by no means a final solution for this geometry, rather a demonstration of some of the main features of the grid generator.

8.2 Stage 1 : Simple Solid Wing Geometry

The first stage of the process is to deal with the main body of the wing without the tip geometry. Figure(21) shows the an image of the geometry surface after the gridding process. However the grid lines show the morphology of the aerofoil surface and give the good visual impression of the geometry. The most common grid topology used for aerofoils is a C-Grid which wraps itself around the aerofoil surface, thus producing a grid which follow the contours of the surface well. In this case the aerofoil was split into an upper and lower surface to aid the construction process and two *Blocks* are produced for the two surfaces. A domain is generated for each of the surfaces extending a distance of one and a half chord lengths perpendicular to the chord and forming an arc around the leading edge, again one and a half chord lengths from the leading edge. The grid would be extended much further for a true solution and these lengths were chosen for easy grid visualisation. Figure(20) shows the defined *Nodes* required to produce this domain, and the computational domain that the two *Blocks* map to. It is clearly seen that more than one *Edge* has been used to define some *Face* sides and that although there are two *Edges* that use *Nodes* zero and one in their definition, they are still distinct *Edges* using different sets of construction points to define the spline which are used to generate the *Nodes* along them. The two *Edges* refer to the upper and lower curves of the aerofoil section which both begin at the trailing edge and end at the leading edge. This is true also for *Edges* defined using *Nodes* four and five. *Edge* and *Face* definitions have not been entered onto the diagrams as they confuse the image. However the data file should be referred to for a full understanding of these definitions.

Figure(22) shows the resultant grid created from *Block* zero. This covers the upper surface of the aerofoil and extends along the span of the wing. Grid lines have been concentrated near to the aerofoil surface as would be required in a flow solver. Concentrations have also been created near the trailing and leading edge regions. The resultant grid with the inclusion of *Block* one is shown in Figure(23).

8.3 Downstream Region

The downstream region is easily created using two more *Blocks*. In fact the mapping from the physical domain to the computational domain is much easier to visualize for the next two *Blocks*. The physical domain is simply extended downstream by five chord lengths. Again the magnitude of the extension has been reduced for ease of visualisation. The raw geometry and block structure are shown in Figure(24) for the C-Grid with the addition of the new downstream region. Notice that the two new *Blocks* share the *Face* defined by *Nodes* zero, four, fifteen, and fourteen. The directional and positional information required to map this *Face* onto the two separate *Blocks* is different. The resultant *Grid* is shown in Figure(25) which clearly shows the extension of the trailing edge concentration into the wake region of the flow.

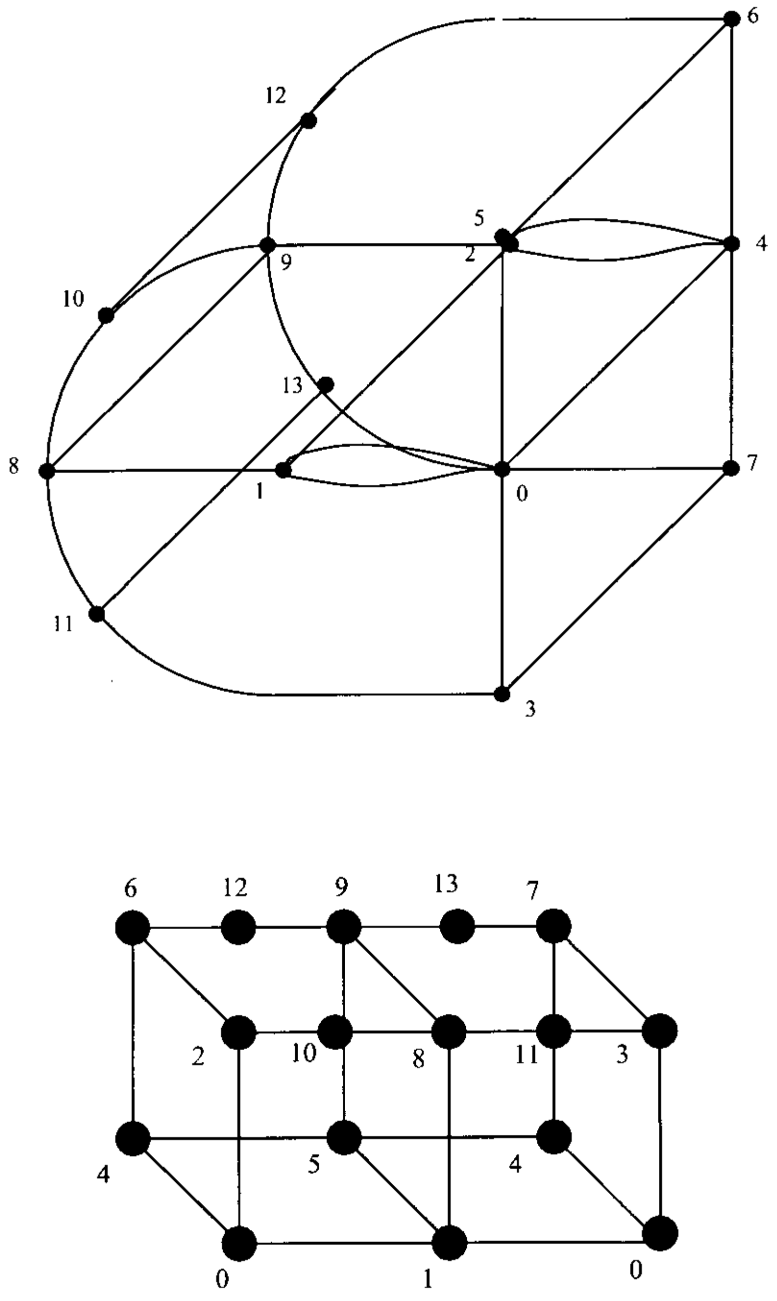


Figure 20: Block Structure in Physical and Computational Domains For Blocks 0 and 1.

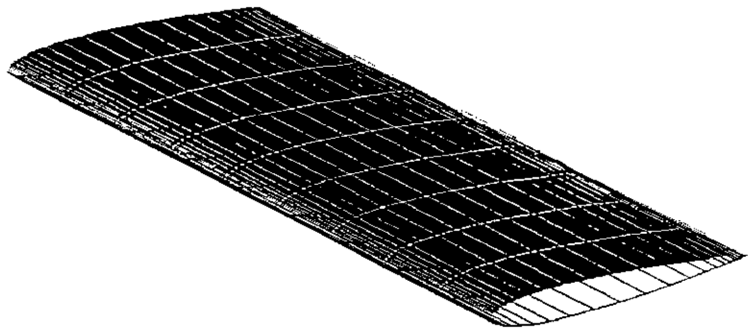


Figure 21: Solid View of NACA 0012 Aerofoil without Wingtip.

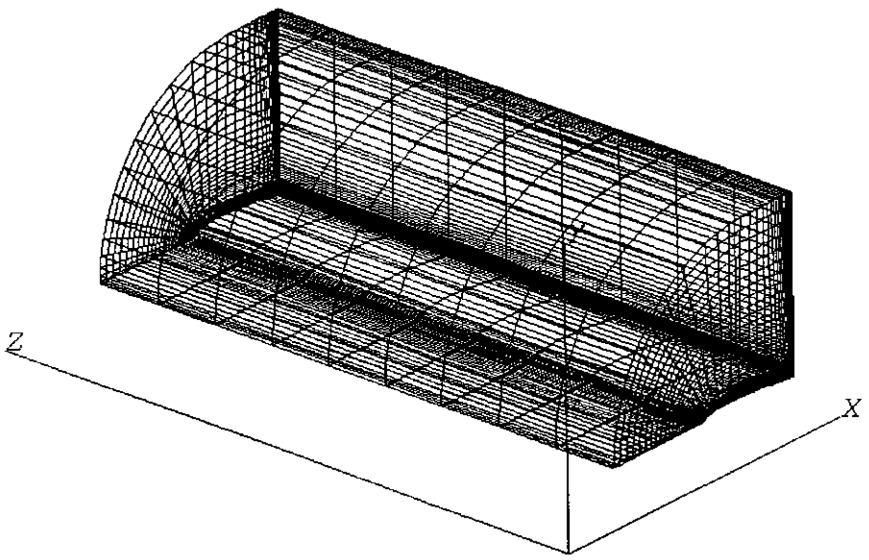


Figure 22: External Faces of Resultant Grid in Block 0.

8.4 Wing Tip

The geometry that has been defined would generally be surrounded by a pseudo two dimensional flow regime. A truly three dimensional flow is only achieved when the geometry becomes assymetric such as when a wing tip is included. A simple wing tip geometry is constructed by generating arcs between the upper and lower surfaces. However a C-Grid is now required to wrap around the span of the wing as well as around the chord. These two topologies do not naturally meet and *Grid* singularities are produced if this topology is followed. The advantages of including *Grid* singularities within the *Grid* are not expanded here since the case study is used as a demonstration of the capabilities of the generator and not an appraisal of various grid generation strategies. The singularities consist of *Edges* constructed from one *Node* thus producing *Faces* mapped onto a line.

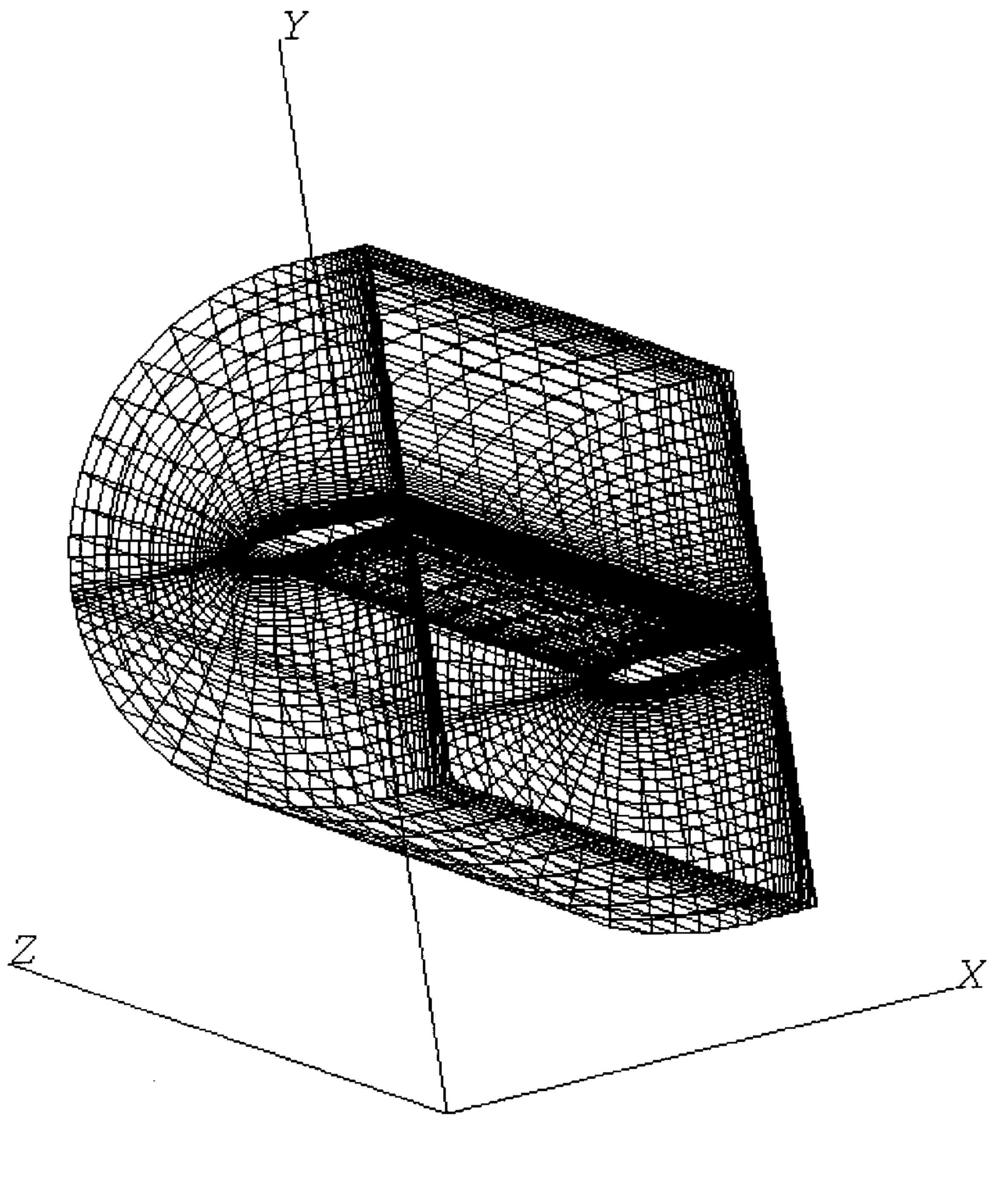
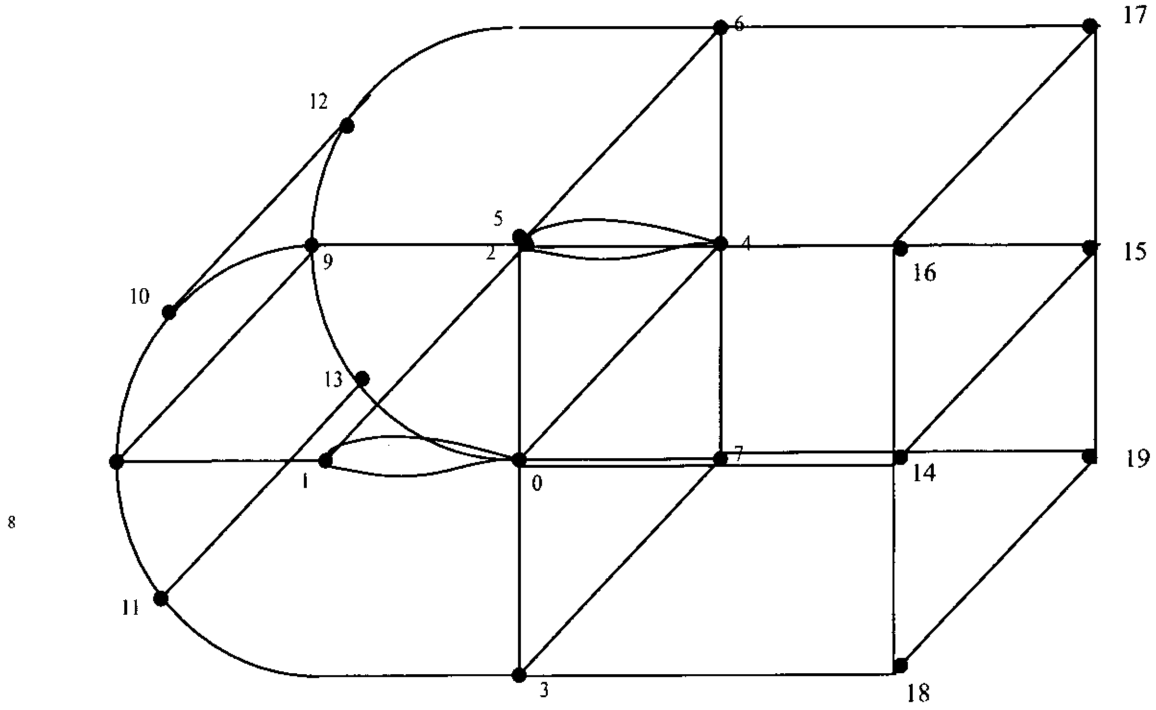


Figure 23: External Faces of Resultant Grid in Blocks 0 and 1.



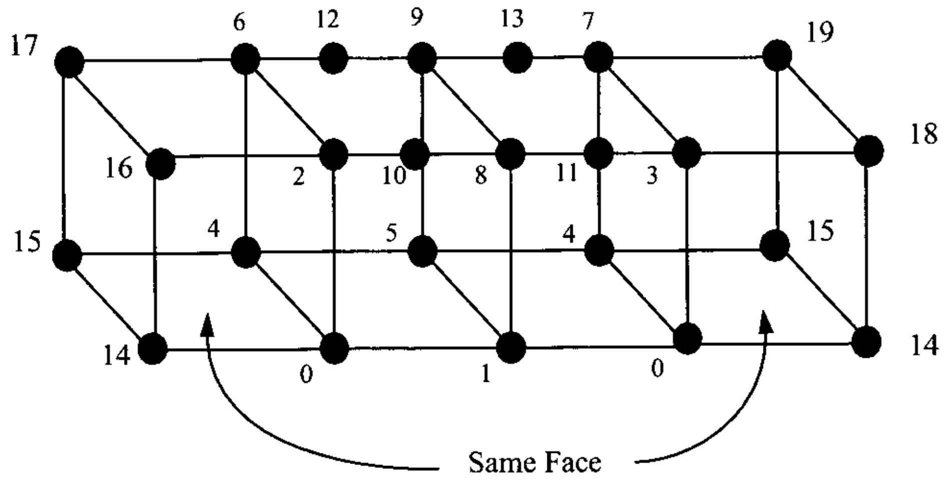


Figure 24: Block Structure in Physical and Computational Domains for Blocks 0,1,2, and 3

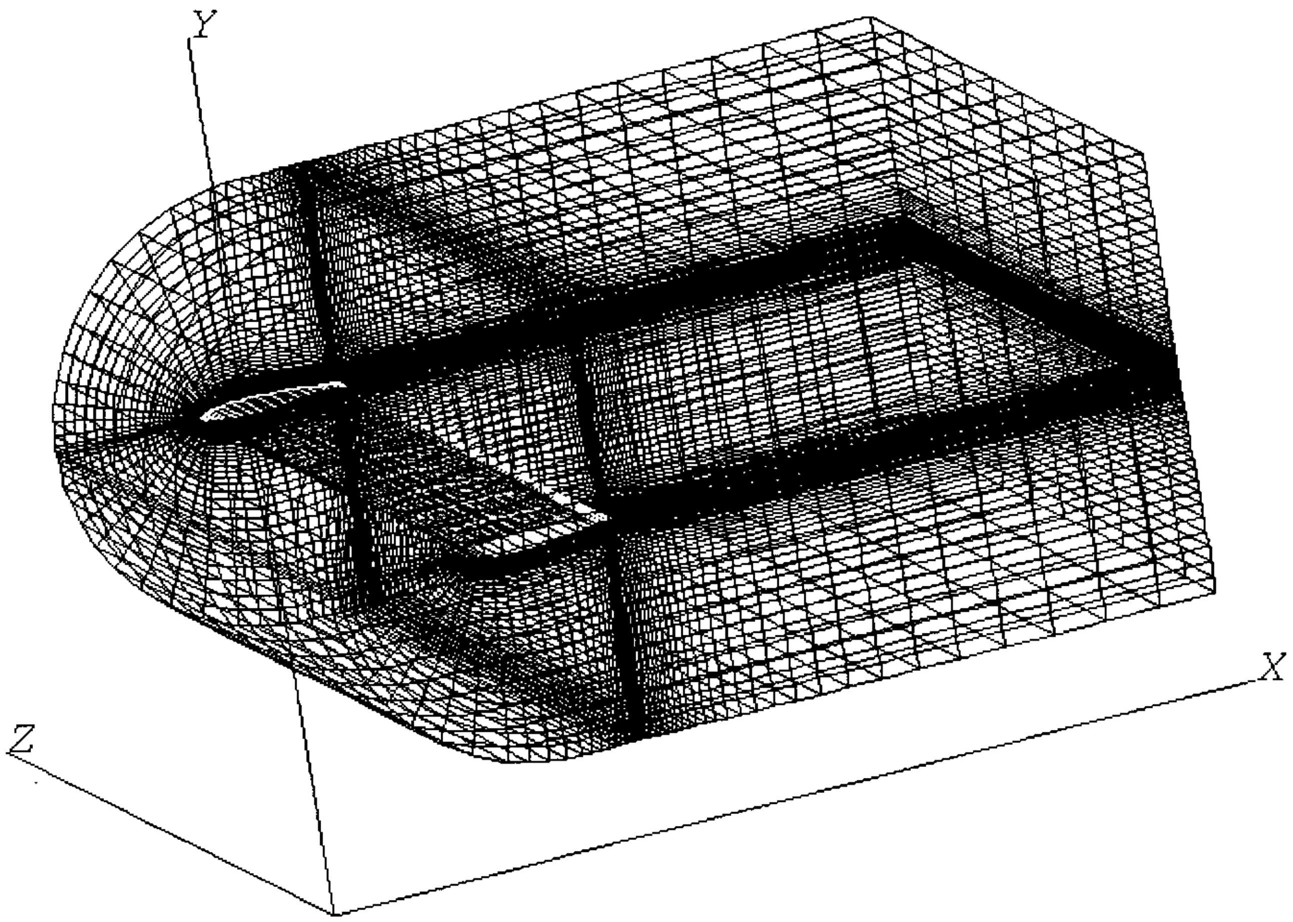


Figure 25: External Faces of Resultant Grid in Blocks 0,1,2, and 3

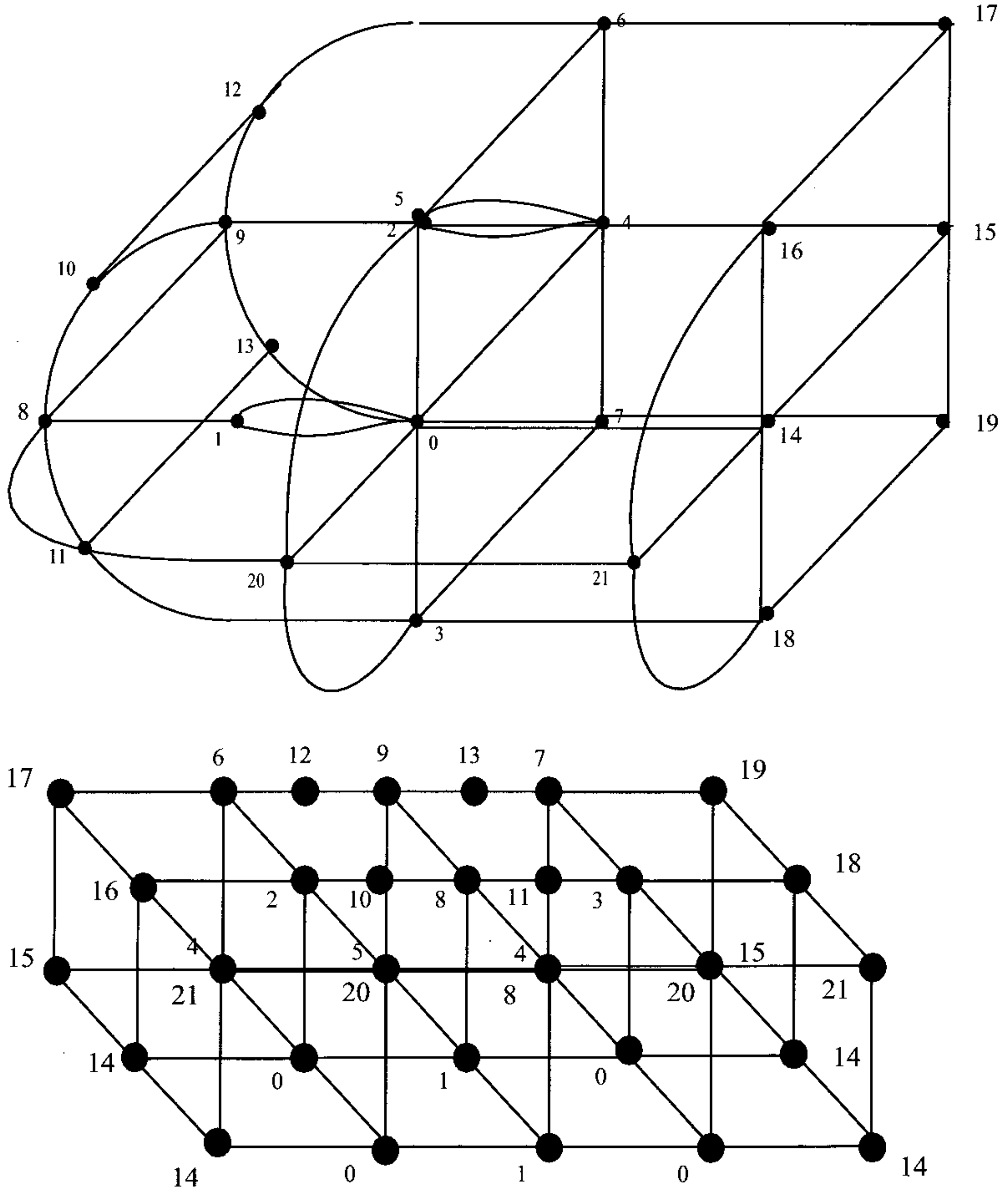


Figure 26: Block Structure in Physical and Computational Domains of Final Grid Including Wing Tip.

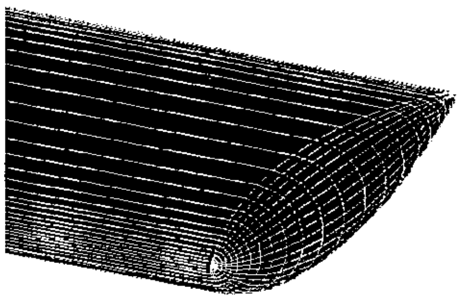


Figure 27: Wing Tip Detail for NACA 0012 Aerofoil

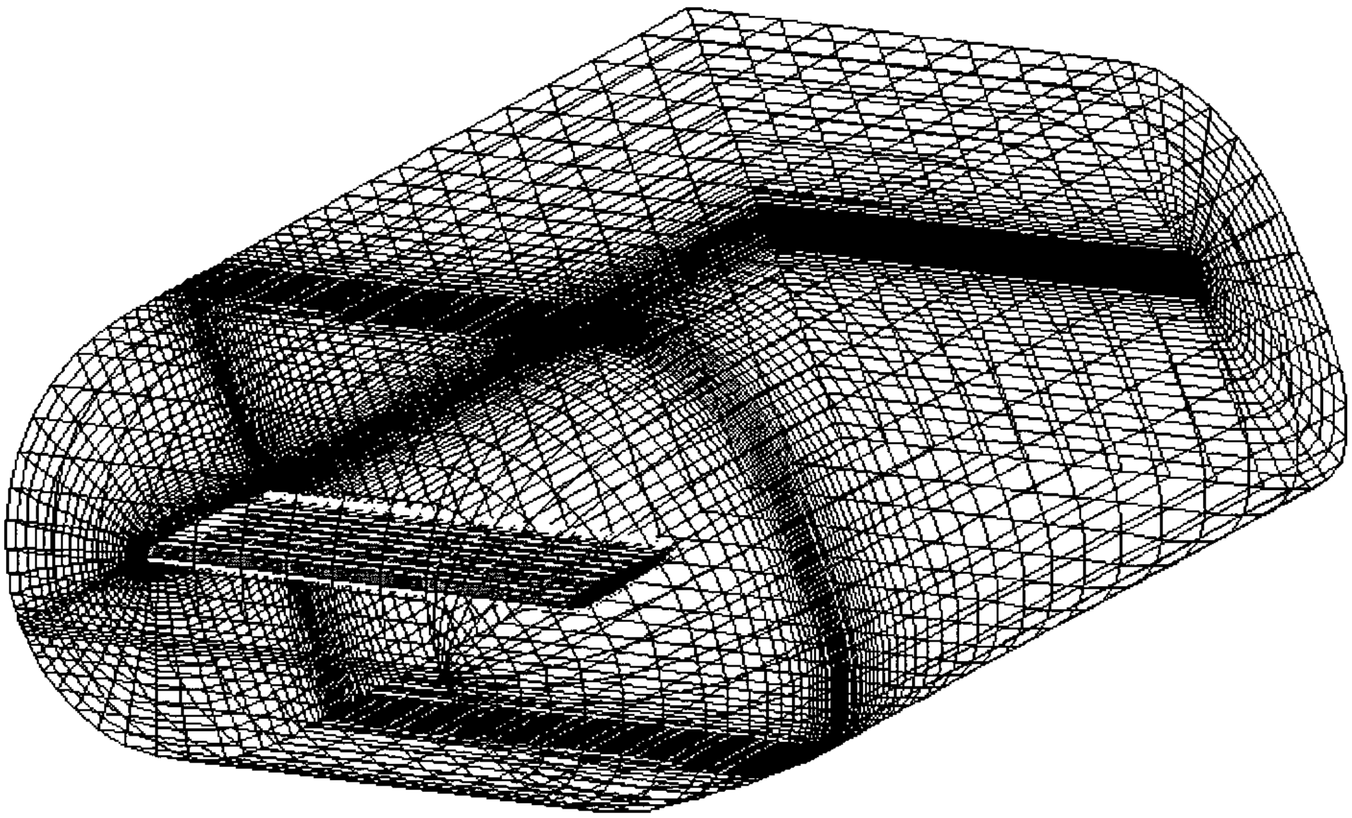


Figure 28: External Faces of Final Grid

9 Future Developments

'Fleximesh' has been developed as a versatile multiblock mesh generator. The use of customized data structures enables the program to concentrate upon the construction of individual *Geometrical Objects*. This allows the construction methods to be developed without the re-development of the whole program. This has already been proved to work with the incorporation of an advanced cubic spline algorithm into the main body of code with very little re-programming away from the spline library. Further advancements are hoped to follow this same philosophy.

There are four areas in which future advancements are hoped to follow.

- C++ :

  The development of the code used ANSCI C. However the type of data structures used lend themselves towards a more Object Orientated approach. The development is hoped to build on the code philosophy used so far and to extend and improve it such that the *Geometrical Object* construction techniques used within 'Fleximesh' could be linked to any grid generator built within the department.

- Visualisation :

  One of the main draw backs of 'Fleximesh' is the complex and time consuming input file preparation required to construct complex meshes. It is hoped that the development of at least a visualisation method will ease some of the difficulties that this stage presents.

- Surface Definition :

  In order to describe and insert a complex surface within a large mesh, a rather complex additional input file has to be constructed. The orientation of the *Face* has to accommodate the the way in which the surface has been defined. The development of this facility should incorporate the use of spline and bi-cubic patches in order to accurately and simply describe a complex surface. The surface definition should not impose any restrictions upon the *Face* definition. A system of 'patches' could be used to describe surfaces which could be mapped onto the surface of a *Face*. 'Patches' could also help prescribe the necessary boundary conditions required for the flow calculation.

- Variable Cell Types:

  The most advanced future development is the inclusion of modules which are able to generate meshes within *Blocks* using different cell types. This requires the use of different mesh generation algorithms to generate the interior points of a *Block*. Some of these methods are well documented and their implementation should not pose many difficulties. However, the real challenge is to provide a versatile method to join *Blocks* constructed from different cell types. Joining *Blocks* could be defined such that different cell topologies can become compatible. The automation of the 'Joining *Block*' process is necessary such that the user does not have to define more complex *Blocks*.

References

[1] Shipshape user manual, 1991.

[2] Avs developer's guide, 1993.

[3] Bowyer A. Computing dirichlet tessellations. *The Computer Journal*, 34(2):162–166, 1981.

[4] Press W.H. Teakolsky S.A. Vetterling W.T. Flannery B.P. *Numerical Recipies in C*. Press Syndicate of University of Cambridge, 2 edition, 1992.

[5] Fletcher C.A.J. *Computational Techniques for Fluid Mechanics*, volume 2. Springer-Verlag, 1991.

[6] Roberts G.O. *Lecture Notes in Physics*, volume 8. Springer, Berlin, 1971.

[7] Hall C.A. Gordon W.J. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7:461–477, 1973.

[8] Eiseman P.R. A multi-surface method of coordinate generation. *Journal of Computational Physics*, 33:118–150, 1979.

[9] Turnock S.R. *Prediction of Ship Rudder-Propeller Interaction Using Parallel Computations and Wind Tunnel Tests*. PhD thesis, University of Southampton, 1993.

[10] Mastin C.W. Thompson J.F., Thames F.C. Tomcat- a code for numerical generation of boundary-fitted curvilinear coordinate systems on fields containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 24:274–302, 1977.

[11] Watson W.F. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 34(2):167–173, 1981.

10 Appendices

10.1 Parametric Cubic Spline Algorithm

The algorithm used for the construction of cubic splines can be found in [4].

1. Calculate η_i for $i = 1, 2, 3, \ldots, n$ using equation (10).
2. Solve Equation(9) for the set $S_i''(\eta_i)$; $i = 1, 2, 3, \ldots, n-1$.
3. Find a new set of η_i for $i = 1, 2, 3, \ldots, m$ using an appropriate stretching function and where m is the new desired number of points.
4. Calculate a new set of discrete points $\widetilde{S}_i(\eta_i)$ which adequately describe the shape of the curve and give the desired point distribution along the line.
5. End.

10.2 Transfinite Interpolation Algorithm

2-D Algorithm

```
1. For $i = 2$ to $i = maxi - 1$ :
    (a) Calculate $r_{AB}$ and $r_{CD}$ using non-dimensional parameter function.
    (b) For $j = 2$ to $j = maxj - 1$ :
        i. Calculate $s_{AD}$ and $s_{BC}$ using non-dimensional parameter function.
        ii. Calculate :
            $\Phi_0 = \frac{(j-1)}{(maxj-1)} r_{CD} + \frac{(maxj-j)}{(maxj-1)} r_{AB}$
            $\Phi_1 = (1 - \Phi_0)$
            $\Psi_0 = \frac{(i-1)}{(maxi-1)} s_{BC} + \frac{(maxi-i)}{(maxi-1)} s_{AD}$
            $\Psi_1 = (1 - \Psi_0)$
        iii. Calculate
            $Z_{rs}(r,s) = \Phi_0 \Psi_0 Z_{(1,1)} + \Phi_1 \Psi_0 Z_{(maxi,1)} + \Phi_1 \Psi_1 Z_{(maxi,maxj)} + \Phi_0 \Psi_1 Z_{(1,maxj)}$
        iv. Calculate
            $Z_r(r,s) = \Phi_0 Z_{1,j} + \Phi_1 Z_{maxi,j}$
            $Z_s(r,s) = \Psi_0 Z_{i,1} + \Psi_1 Z_{i,maxj}$
        v. Sum to find
            $Z(r,s) = Z_r(r,s) + Z_s(r,s) + Z_{rs}(r,s)$
        vi. End.
    (c) End.
2. End.
```

3-D Algorithm

```
1. For $i = 2$ to $i = maxi - 1$ :
   (a) For $j = 2$ to $j = maxj - 1$ :
       i. For $k = 2$ to $k = maxk - 1$ :
          A. Calculate $r_{AE}$, $r_{CG}$, $s_{IL}$, $s_{JK}$, $t_{LK}$, and $t_{IJ}$ using non-dimensional parameter function.
             • A : Z(1,j,maxk)
             • C : Z(1,j,1)
             • E : Z(maxi,j,maxk)
             • G : Z(maxi,j,1)
             • I : Z(i,1,maxk)
             • J : Z(i,1,1)
             • K : Z(i,maxj,1)
             • L : Z(i,maxj,maxk)
          B. Calculate :
             $\Phi_0 = \frac{(k-1)}{(maxk-1)} r_{AE} + \frac{(maxk-k)}{(maxk-1)} r_{CG}$
             $\Phi_1 = (1-\Phi_0)$
             $\Psi_0 = \frac{(k-1)}{(maxk-1)} s_{IL} + \frac{(maxk-k)}{(maxk-1)} s_{JK}$
             $\Psi_1 = (1-\Psi_0)$
             $\Omega_0 = \frac{(j-1)}{(maxj-1)} t_{LK} + \frac{(maxj-j)}{(maxj-1)} t_{IJ}$
             $\Omega_1 = (1-\Omega_0)$
          C. Calculate corner terms :
             $Z_{rst}(r,s,t) = \Phi_0\Psi_0\Omega_0 Z_{(1,1,1)} + \Phi_0\Psi_0\Omega_1 Z_{(1,1,maxk)} + \Phi_0\Psi_1\Omega_0 Z_{(1,maxj,1)} + \Phi_0\Psi_1\Omega_1 Z_{(1,maxj,maxk)} + \Phi_1\Psi_0\Omega_0 Z_{(maxi,1,1)} + \Phi_1\Psi_0\Omega_1 Z_{(maxi,1,maxk)} + \Phi_1\Psi_1\Omega_0 Z_{(maxi,maxj,1)} + \Phi_1\Psi_1\Omega_1 Z_{(maxi,maxj,maxk)}$   (34)
          D. Calculate edge terms :
             • r,s terms
               $Z_{rs}(r,s,t) = \Phi_0\Psi_0 Z_{(1,1,k)} + \Phi_0\Psi_1 Z_{(1,maxj,k)} \Phi_1\Psi_0 Z_{(maxi,1,k)} + \Phi_1\Psi_1 Z_{(maxi,maxj,k)}$
             • r,t terms
               $Z_{rt}(r,s,t) = \Phi_0\Omega_0 Z_{(1,j,1)} + \Phi_0\Omega_1 Z_{(1,j,maxk)} \Phi_1\Omega_0 Z_{(maxi,j,1)} + \Phi_1\Omega_1 Z_{(maxi,j,maxk)}$
             • s,t terms
               $Z_{st}(r,s,t) = \Psi_0\Omega_0 Z_{(i,1,1)} + \Psi_0\Omega_1 Z_{(i,1,maxk)} \Psi_1\Omega_0 Z_{(i,maxj,1)} + \Psi_1\Omega_1 Z_{(i,maxj,maxk)}$
          E. Calculate
             $Z_r(r,s,t) = \Phi_0 Z_{1,j,k} + \Phi_1 Z_{maxi,j,k}$
             $Z_s(r,s,t) = \Psi_0 Z_{i,1,k} + \Psi_1 Z_{i,maxj,k}$
             $Z_t(r,s,t) = \Omega_0 Z_{i,j,1} + \Omega_1 Z_{i,j,maxk}$
          F. Sum to find
             $Z(r,s,t) = Z_r(r,s,t) + Z_s(r,s,t) + Z_t(r,s,t) + Z_{rs}(r,s,t) + Z_{rt}(r,s,t) + Z_{st}(r,s,t) + Z_{rst}(r,s,t)$
          G. End.
       ii. End.
   (b) End
2. End.
```

10.3 Elliptical Refinement Algorithm

```
1. Input Grid Boundaries.
2. Input Convergence Limit $\Delta\tilde{r}_{limit}$.
3. Set $count = 0$.
4. Set $\Delta\tilde{r} = 1.1 * \Delta\tilde{r}_{limit}$
5. Set $(Average)^n = \Delta\tilde{r}_{limit}$.
6. Set $(Average)^{n-1} = 1.1 * (Average)^n$.
7. Do :
   (a) $\Delta\tilde{r} = 0.0$
   (b) For $i = 2$ to $i = maxi - 1$ :
       i. For $j = 2$ to $j = maxj - 1$ :
          A. For $k = 2$ to $k = maxk - 1$ :
             • Calculate $\alpha_{i,j}$ Coefficients.
             • Calculate $J$.
             • Calculate $P$, $Q$, and $R$ Source Terms.
             • Calculate New Point $\tilde{r}^{n+1}_{i,j,k}$ from Equation (29) where $\tilde{r}^{pred}_{i,j,k}$ is found from the three dimensional
               equivalent of equation (27).
             • Calculate $\delta\tilde{r} = \tilde{r}^{n}_{i,j,k} - \tilde{r}^{n+1}_{i,j,k}$.
             • If $\delta\tilde{r} > \Delta\tilde{r}$   $\Delta\tilde{r} = \delta\tilde{r}$.
          B. End.
       ii. End.
   (c) End.
   (d) For $i = 2$ to $i = maxi - 1$ :
       i. For $j = 2$ to $j = maxj - 1$ :
          A. For $k = 2$ to $k = maxk - 1$ :
             • Re-initialize pseudo time level : $\tilde{r}^{n}_{i,j,k} = \tilde{r}^{n+1}_{i,j,k}$
          B. End.
       ii. End
   (e) End.
   (f) Update Error History. $History[count] = \Delta\tilde{r}$.
   (g) Update Count $count = count + 1$.
   (h) If $count = 10$
       i. Set $(Average)^{n-1} = (Average)^n$.
       ii. Set $(Average)^n = \frac{1}{10}\sum_{i=1}^{10} History[i]$.
       iii. Set $count = 0$
   (i) While $\Delta\tilde{r} > \Delta\tilde{r}_{limit}$ and $(Average)^n < (Average)^{n-1}$.
```

10.4 Surface Definition

'Fleximesh' includes a facility to allow complex surfaces to be approximated using a series of cubic splines. This facility becomes essential when the bounding curves of a *Face* are not adequate to construct the interior surface. The use of this module should be restricted to the representation of complex surfaces as the nature of the module at present complicates the initial grid description.

The module is recognized when an extra line is added to the *Face* description in the input file. The *Face* description format is altered such that it includes a link to a separate file which includes information required to construct the surface.

string ['FACE'] int [ID] int [Boundary Condition ID]

string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···

string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···

string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···

string ['SIDE'] int [Side ID] int [No. *Edges*] int [*Edge ID*] int [*Edge* Direction] ···

string ['FILE'] string [surface file name] string ['SOURCES'] int [No. i Line Sources] int [No. j Line Sources] int [No. Pt. Sources]

string ['I_SOURCES'] int [i location] float [strength of source] float [damping of source] ···

string ['J_SOURCES'] int [j location] float [strength of source] float [damping of source] ···

string ['P_SOURCES'] int [i location] int [j location] float [strength of source] float [damping of source] ···

...

...

The surface file contains the information required to construct splines across the *Face*. The splines are used to construct a new set of points which are then used to construct splines in the opposite direction. The second set of splines are used to interpolate a new set of points in the opposite direction resulting in a new set of points representing the surface. The number of points prescribed in the surface file must be adequate to describe the curvature of the surface in both directions. The format of the surface file is described below.

int [no. initial splines] int [no. required pts. in ξ dir.] int [no. required pts. in η dir.]
string ['DISTRIBUTION'] int [Side ID] float [P] float [Q]
string ['DISTRIBUTION'] int [Side ID] float [P] float [Q]
string ['DISTRIBUTION'] int [Side ID] float [P] float [Q]
string ['DISTRIBUTION'] int [Side ID] float [P] float [Q]
int [Spline ID] int [no. spline pts.]
int [Spline pt. ID] float [pt. x coord.] float [pt. y coord.] float [pt. z coord.]
int [Spline pt. ID] float [pt. x coord.] float [pt. y coord.] float [pt. z coord.]
int [Spline pt. ID] float [pt. x coord.] float [pt. y coord.] float [pt. z coord.]
...
...
int [Spline ID] int [no. spline pts.]
int [Spline pt. ID] float [pt. x coord.] float [pt. y coord.] float [pt. z coord.]
...
...
int [Spline ID] int [no. spline pts.]
int [Spline pt. ID] float [pt. x coord.] float [pt. y coord.] float [pt. z coord.]
...

...

Each initial spline must describe the curvature of the *Face* between sides 3 and 1. This requirement complicates the description of a *Face* as splines maybe simpler to describe in one direction rather than another. This implies that the description of the surface has to be taken into account before the *Face* is defined in order to find the simplest surface definition. The description of the splines in relation to the *Face* definition is shown in Figure(29).

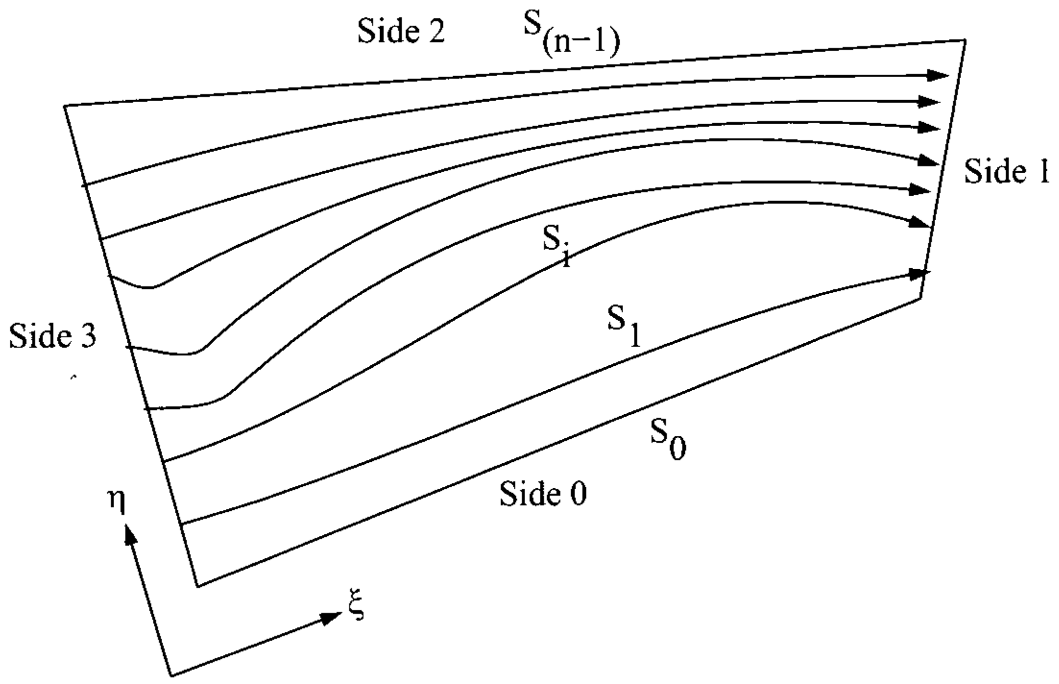


Figure 29: Surface File Definition

10.5 NACA 0012 Wing Input File

naca0012.dat :

```
Naca0012 Case Study Input File
23 50 34 8
NODE 0 1.0 0.0 0.0
NODE 1 0.0 0.0 0.0
NODE 2 1.0 1.5 0.0
NODE 3 1.0 -1.5 0.0
NODE 4 1.0 0.0 5.0
NODE 5 0.0 0.0 5.0
NODE 6 1.0 1.5 5.0
NODE 7 1.0 -1.5 5.0
NODE 8 -1.0 0.0 0.0
NODE 9 -1.0 0.0 5.0
NODE 10 0.036470 1.426580 0.00000
NODE 11 0.036470 -1.426580 0.00000
NODE 12 0.036470 1.426580 5.00000
NODE 13 0.036470 -1.426580 5.00000
NODE 14 5.0 0.0 0.0
NODE 15 5.0 0.0 5.0
NODE 16 5.0 1.5 0.0
NODE 17 5.0 1.5 5.0
NODE 18 5.0 -1.5 0.0
NODE 19 5.0 -1.5 5.0
NODE 20 1.0 0.0 -1.5
NODE 21 5.0 0.0 -1.5
NODE 22 0.036470 0.0 -1.426580
EDGE 0 0 1 26 33 -0.1 -3.5
0 1.0 0.0 0.0
1 0.95 -0.00807 0.0
2 0.9 -0.01448 0.0
3 0.8 -0.02623 0.0
4 0.7 -0.03664 0.0
5 0.6 -0.04563 0.0
6 0.5 -0.05294 0.0
7 0.4 -0.05803 0.0
8 0.3 -0.06002 0.0
9 0.25 -0.05941 0.0
10 0.2 -0.05737 0.0
11 0.18 -0.055945 0.0
12 0.16 -0.054286 0.0
13 0.15 -0.053450 0.0
14 0.14 -0.052238 0.0
15 0.12 -0.049734 0.0
16 0.1 -0.04683 0.0
17 0.08 -0.042924 0.0
18 0.075 -0.04200 0.0
19 0.06 -0.038235 0.0
20 0.05 -0.03555 0.0
21 0.04 -0.032152 0.0
22 0.025 -0.02615 0.0
23 0.02 -0.023501 0.0
24 0.0125 -0.01894 0.0
25 0.0 0.0 0.0
EDGE 1 0 1 26 33 -0.1 -3.5
0 1.0 0.0 0.0
1 0.95 0.00807 0.0
2 0.9 0.01448 0.0
3 0.8 0.02623 0.0
4 0.7 0.03664 0.0
5 0.6 0.04563 0.0
```

```
6 0.5 0.05294 0.0
7 0.4 0.05803 0.0
8 0.3 0.06002 0.0
9 0.25 0.05941 0.0
10 0.2 0.05737 0.0
11 0.18 0.055945 0.0
12 0.16 0.054286 0.0
13 0.15 0.053450 0.0
14 0.14 0.052238 0.0
15 0.12 0.049734 0.0
16 0.1 0.04683 0.0
17 0.08 0.042924 0.0
18 0.075 0.04200 0.0
19 0.06 0.038235 0.0
20 0.05 0.03555 0.0
21 0.04 0.032152 0.0
22 0.025 0.02615 0.0
23 0.02 0.023501 0.0
24 0.0125 0.01894 0.0
25 0.0 0.0 0.0
EDGE 2 0 2 2 40 0.1 3.0
0 1.0 0.0 0.0
1 1.0 1.5 0.0
EDGE 3 2 10 5 21 0.3 4.0
0 1.0 1.5 0.0
1 0.5 1.5 0.0
2 0.405810 1.497040 0.0
3 0.218930 1.473430 0.0
4 0.036470 1.426580 0.0
EDGE 4 0 3 2 40 0.1 3.0
0 1.0 0.0 0.0
1 1.0 -1.5 0.0
EDGE 5 4 5 26 33 -0.1 -3.5
0 1.0 0.0 5.0
1 0.95 -0.00807 5.0
2 0.9 -0.01448 5.0
3 0.8 -0.02623 5.0
4 0.7 -0.03664 5.0
5 0.6 -0.04563 5.0
6 0.5 -0.05294 5.0
7 0.4 -0.05803 5.0
8 0.3 -0.06002 5.0
9 0.25 -0.05941 5.0
10 0.2 -0.05737 5.0
11 0.18 -0.055945 5.0
12 0.16 -0.054286 5.0
13 0.15 -0.053450 5.0
14 0.14 -0.052238 5.0
15 0.12 -0.049734 5.0
16 0.1 -0.04683 5.0
17 0.08 -0.042924 5.0
18 0.075 -0.04200 5.0
19 0.06 -0.038235 5.0
20 0.05 -0.03555 5.0
21 0.04 -0.032152 5.0
22 0.025 -0.02615 5.0
23 0.02 -0.023501 5.0
24 0.0125 -0.01894 5.0
25 0.0 0.0 5.0
EDGE 6 4 5 26 33 -0.1 -3.5
0 1.0 0.0 5.0
1 0.95 0.00807 5.0
2 0.9 0.01448 5.0
3 0.8 0.02623 5.0
```

```
4 0.7 0.03664 5.0
5 0.6 0.04563 5.0
6 0.5 0.05294 5.0
7 0.4 0.05803 5.0
8 0.3 0.06002 5.0
9 0.25 0.05941 5.0
10 0.2 0.05737 5.0
11 0.18 0.055945 5.0
12 0.16 0.054286 5.0
13 0.15 0.053450 5.0
14 0.14 0.052238 5.0
15 0.12 0.049734 5.0
16 0.1 0.04683 5.0
17 0.08 0.042924 5.0
18 0.075 0.04200 5.0
19 0.06 0.038235 5.0
20 0.05 0.03555 5.0
21 0.04 0.032152 5.0
22 0.025 0.02615 5.0
23 0.02 0.023501 5.0
24 0.0125 0.01894 5.0
25 0.0 0.0 5.0
EDGE 7 4 6 2 40 0.1 3.0
0 1.0 0.0 5.0
1 1.0 1.5 5.0
EDGE 8 6 12 5 21 0.3 4.0
0 1.0 1.5 5.0
1 0.5 1.5 5.0
2 0.405810 1.497040 5.0
3 0.218930 1.473430 5.0
4 0.036470 1.426580 5.0
EDGE 9 4 7 2 40 0.1 3.0
0 1.0 0.0 5.0
1 1.0 -1.5 5.0
EDGE 10 0 4 2 10 1.0 0.1
0 1.0 0.0 0.0
1 1.0 0.0 5.0
EDGE 11 3 7 2 10 1.0 0.1
0 1.0 -1.5 0.0
1 1.0 1.5 5.0
EDGE 12 2 6 2 10 1.0 0.1
0 1.0 1.5 0.0
1 1.0 1.5 5.0
EDGE 13 3 11 5 21 0.3 4.0
0 1.0 -1.5 0.0
1 0.5 -1.5 0.0
2 0.405810 -1.497040 0.0
3 0.218930 -1.473430 0.0
4 0.036470 -1.426580 0.0
EDGE 14 7 13 5 21 0.3 4.0
0 1.0 -1.5 5.0
1 0.5 -1.5 5.0
2 0.405810 -1.497040 5.0
3 0.218930 -1.473430 5.0
4 0.036470 -1.426580 5.0
EDGE 15 1 8 2 40 0.1 3.0
0 0.0 0.0 0.0
1 -1.0 0.0 0.0
EDGE 16 5 9 2 40 0.1 3.0
0 0.0 0.0 5.0
1 -1.0 0.0 5.0
EDGE 17 8 9 2 10 1.0 0.1
0 -1.0 0.0 0.0
1 -1.0 0.0 5.0
```

```
EDGE 18 1 5 2 10 1.0 0.1
0 0.0 0.0 0.0
1 0.0 0.0 5.0
EDGE 19 10 8 11 13 1.8 2.0
0 0.036470 1.426580 0.0
1 -0.138670 1.357240 0.0
2 -0.303740 1.266490 0.0
3 -0.456140 1.155770 0.0
4 -0.593450 1.026820 0.0
5 -0.713530 0.881680 0.0
6 -0.814460 0.722630 0.0
7 -0.894660 0.552190 0.0
8 -0.952870 0.373030 0.0
9 -0.988170 0.188000 0.0
10 -1.0 0.0 0.0
EDGE 20 11 8 11 13 1.8 2.0
0 0.036470 -1.426580 0.0
1 -0.138670 -1.357240 0.0
2 -0.303740 -1.266490 0.0
3 -0.456140 -1.155770 0.0
4 -0.593450 -1.026820 0.0
5 -0.713530 -0.881680 0.0
6 -0.814460 -0.722630 0.0
7 -0.894660 -0.552190 0.0
8 -0.952870 -0.373030 0.0
9 -0.988170 -0.188000 0.0
10 -1.0 0.0 0.0
EDGE 21 12 9 11 13 1.8 2.0
0 0.036470 1.426580 5.0
1 -0.138670 1.357240 5.0
2 -0.303740 1.266490 5.0
3 -0.456140 1.155770 5.0
4 -0.593450 1.026820 5.0
5 -0.713530 0.881680 5.0
6 -0.814460 0.722630 5.0
7 -0.894660 0.552190 5.0
8 -0.952870 0.373030 5.0
9 -0.988170 0.188000 5.0
10 -1.0 0.0 5.0
EDGE 22 13 9 11 13 1.8 2.0
0 0.036470 -1.426580 5.0
1 -0.138670 -1.357240 5.0
2 -0.303740 -1.266490 5.0
3 -0.456140 -1.155770 5.0
4 -0.593450 -1.026820 5.0
5 -0.713530 -0.881680 5.0
6 -0.814460 -0.722630 5.0
7 -0.894660 -0.552190 5.0
8 -0.952870 -0.373030 5.0
9 -0.988170 -0.188000 5.0
10 -1.0 0.0 0.0
EDGE 23 3 18 2 30 0.1 3.0
0 1.0 -1.5 0.0
1 5.0 -1.5 0.0
EDGE 24 7 19 2 30 0.1 3.0
0 1.0 -1.5 5.0
1 5.0 -1.5 5.0
EDGE 25 0 14 2 30 0.1 3.0
0 1.0 0.0 0.0
1 5.0 0.0 0.0
EDGE 26 4 15 2 30 0.1 3.0
0 1.0 0.0 5.0
1 5.0 0.0 5.0
EDGE 27 2 16 2 30 0.1 3.0
```

```
0 1.0 1.5 0.0
1 5.0 1.5 0.0
EDGE 28 6 17 2 30 0.1 3.0
0 1.0 1.5 5.0
1 5.0 1.5 5.0
EDGE 29 14 18 2 40 0.1 3.0
0 5.0 0.0 0.0
1 5.0 -1.5 0.0
EDGE 30 14 16 2 40 0.1 3.0
0 5.0 0.0 0.0
1 5.0 -1.5 0.0
EDGE 31 15 19 2 40 0.1 3.0
0 5.0 0.0 5.0
1 5.0 -1.5 5.0
EDGE 32 15 17 2 40 0.1 3.0
0 5.0 0.0 5.0
1 5.0 1.5 5.0
EDGE 33 16 17 2 10 1.0 0.1
0 5.0 1.5 0.0
1 5.0 1.5 5.0
EDGE 34 14 15 2 10 1.0 0.1
0 5.0 0.0 0.0
1 5.0 0.0 5.0
EDGE 35 18 19 2 10 1.0 0.1
0 5.0 -1.5 0.0
1 5.0 -1.5 5.0
EDGE 36 22 8 11 13 1.8 2.0
0 0.036470 0.0 -1.426580
1 -0.138670 0.0 -1.357240
2 -0.303740 0.0 -1.266490
3 -0.456140 0.0 -1.155770
4 -0.593450 0.0 -1.026820
5 -0.713530 0.0 -0.881680
6 -0.814460 0.0 -0.722630
7 -0.894660 0.0 -0.552190
8 -0.952870 0.0 -0.373030
9 -0.988170 0.0 -0.188000
10 -1.0 0.0 0.0
EDGE 37 20 22 5 21 0.3 4.0
0 1.0 0.0 -1.5
1 0.5 0.0 -1.5
2 0.405810 0.0 -1.497040
3 0.218930 0.0 -1.473430
4 0.036470 0.0 -1.426580
EDGE 38 20 2 15 10 1.0 0.1
0 1.0 0.0 -1.5
1 1.0 0.188000 -1.5
2 1.0 0.373030 -1.497040
3 1.0 0.552190 -1.473430
4 1.0 0.722630 -1.426580
5 1.0 0.881680 -1.357240
6 1.0 1.026820 -1.266490
7 1.0 1.155770 -1.155770
8 1.0 1.266490 -1.026820
9 1.0 1.357240 -0.881690
10 1.0 1.426580 -0.722630
11 1.0 1.473430 -0.552190
12 1.0 1.497040 -0.373030
13 1.0 1.5 -0.188
14 1.0 1.5 0.0
EDGE 39 20 3 15 10 1.0 0.1
0 1.0 -0.0 -1.5
1 1.0 -0.188000 -1.5
2 1.0 -0.373030 -1.497040
```

```
3 1.0 -0.552190 -1.473430
4 1.0 -0.722630 -1.426580
5 1.0 -0.881680 -1.357240
6 1.0 -1.026820 -1.266490
7 1.0 -1.155770 -1.155770
8 1.0 -1.266490 -1.026820
9 1.0 -1.357240 -0.881690
10 1.0 -1.426580 -0.722630
11 1.0 -1.473430 -0.552190
12 1.0 -1.497040 -0.373030
13 1.0 -1.5 -0.188
14 1.0 -1.5 0.0
EDGE 40 0 20 2 40 0.1 3.0
0 1.0 0.0 0.0
1 1.0 0.0 -1.5
EDGE 41 14 21 2 40 0.1 3.0
0 5.0 0.0 0.0
1 5.0 0.0 -1.5
EDGE 42 21 16 15 10 1.0 0.1
0 5.0 0.0 -1.5
1 5.0 0.188000 -1.5
2 5.0 0.373030 -1.497040
3 5.0 0.552190 -1.473430
4 5.0 0.722630 -1.426580
5 5.0 0.881680 -1.357240
6 5.0 1.026820 -1.266490
7 5.0 1.155770 -1.155770
8 5.0 1.266490 -1.026820
9 5.0 1.357240 -0.881690
10 5.0 1.426580 -0.722630
11 5.0 1.473430 -0.552190
12 5.0 1.497040 -0.373030
13 5.0 1.5 -0.188
14 5.0 1.5 0.0
EDGE 43 21 18 15 10 1.0 0.1
0 5.0 0.0 -1.5
1 5.0 -0.188000 -1.5
2 5.0 -0.373030 -1.497040
3 5.0 -0.552190 -1.473430
4 5.0 -0.722630 -1.426580
5 5.0 -0.881680 -1.357240
6 5.0 -1.026820 -1.266490
7 5.0 -1.155770 -1.155770
8 5.0 -1.266490 -1.026820
9 5.0 -1.357240 -0.881690
10 5.0 -1.426580 -0.722630
11 5.0 -1.473430 -0.552190
12 5.0 -1.497040 -0.373030
13 5.0 -1.5 -0.188
14 5.0 -1.5 0.0
EDGE 44 20 21 2 30 0.1 3.0
0 1.0 0.0 -1.5
1 5.0 0.0 -1.5
EDGE 45 8 8 2 10 1.0 0.1
0 -1.0 0.0 0.0
1 -1.0 0.0 0.0
EDGE 46 1 1 2 10 1.0 0.1
0 0.0 0.0 0.0
1 0.0 0.0 0.0
EDGE 47 0 0 2 10 1.0 0.1
0 1.0 0.0 0.0
1 1.0 0.0 0.0
EDGE 48 14 14 2 10 1.0 0.1
0 5.0 0.0 0.0
```

```
1 5.0 0.0 0.0
EDGE 49 0 1 26 33 -0.1 -3.5
0 1.0 0.0 0.0
1 0.95 0.0 -0.00807
2 0.9 0.0 -0.01448
3 0.8 0.0 -0.02623
4 0.7 0.0 -0.03664
5 0.6 0.0 -0.04563
6 0.5 0.0 -0.05294
7 0.4 0.0 -0.05803
8 0.3 0.0 -0.06002
9 0.25 0.0 -0.05941
10 0.2 0.0 -0.05737
11 0.18 0.0 -0.055945
12 0.16 0.0 -0.054286
13 0.15 0.0 -0.053450
14 0.14 0.0 -0.052238
15 0.12 0.0 -0.049734
16 0.1 0.0 -0.04683
17 0.08 0.0 -0.042924
18 0.075 0.0 -0.04200
19 0.06 0.0 -0.038235
20 0.05 0.0 -0.03555
21 0.04 0.0 -0.032152
22 0.025 0.0 -0.02615
23 0.02 0.0 -0.023501
24 0.0125 0.0 -0.01894
25 0.0 0.0 0.0
FACE 0 0
SIDE 0 1 1 0
SIDE 1 1 15 0
SIDE 2 2 19 1 3 1
SIDE 3 1 2 1
SOURCES 2 1 0
I_SOURCES 0 200 0.15 32 900 0.2
J_SOURCES 0 800 0.18
FACE 1 0
SIDE 0 1 18 0
SIDE 1 1 16 0
SIDE 2 1 17 1
SIDE 3 1 15 1
FACE 2 0
SIDE 0 1 6 1
SIDE 1 1 7 0
SIDE 2 2 8 0 21 0
SIDE 3 1 16 1
SOURCES 0 0 0
FACE 3 0
SIDE 0 1 10 1
SIDE 1 1 2 0
SIDE 2 1 12 0
SIDE 3 1 7 1
SOURCES 0 0 0
FACE 4 0
SIDE 0 1 10 0
SIDE 1 1 6 0
SIDE 2 1 18 1
SIDE 3 1 1 1
SOURCES 0 0 0
FACE 5 0
SIDE 0 2 3 0 19 0
SIDE 1 1 17 0
SIDE 2 2 21 1 8 1
SIDE 3 1 12 1
```

```
SOURCES 0 0 0
FACE 6 0
SIDE 0 1 0 1
SIDE 1 1 4 0
SIDE 2 2 13 0 20 0
SIDE 3 1 15 1
SOURCES 0 0 0
FACE 7 0
SIDE 0 1 10 0
SIDE 1 1 9 0
SIDE 2 1 11 1
SIDE 3 1 4 1
SOURCES 0 0 0
FACE 8 0
SIDE 0 1 5 0
SIDE 1 1 16 0
SIDE 2 1 11 1
SIDE 3 1 4 1
SOURCES 0 0 0
FACE 9 0
SIDE 0 1 18 0
SIDE 1 1 5 1
SIDE 2 1 10 1
SIDE 3 1 0 0
SOURCES 0 0 0
FACE 10 0
SIDE 0 2 20 1 13 1
SIDE 1 1 11 0
SIDE 2 2 14 0 22 0
SIDE 3 1 17 1
SOURCES 0 0 0
FACE 11 0
SIDE 0 1 25 0
SIDE 1 1 29 0
SIDE 2 1 23 1
SIDE 3 1 4 1
SOURCES 0 0 0
FACE 12 0
SIDE 0 1 34 0
SIDE 1 1 31 0
SIDE 2 1 35 1
SIDE 3 1 29 1
SOURCES 0 0 0
FACE 13 0
SIDE 0 1 26 1
SIDE 1 1 9 0
SIDE 2 1 24 0
SIDE 3 1 31 1
SOURCES 0 0 0
FACE 14 0
SIDE 0 1 10 0
SIDE 1 1 26 0
SIDE 2 1 34 1
SIDE 3 1 25 1
SOURCES 0 0 0
FACE 15 0
SIDE 0 1 23 0
SIDE 1 1 35 0
SIDE 2 1 24 1
SIDE 3 1 11 1
SOURCES 0 0 0
FACE 16 0
SIDE 0 1 25 1
SIDE 1 1 2 0
SIDE 2 1 27 0
SIDE 3 1 30 1
SOURCES 0 0 0
FACE 17 0
SIDE 0 1 26 0
SIDE 1 1 32 0
SIDE 2 1 28 1
SIDE 3 1 7 1
SOURCES 0 0 0
FACE 18 0
SIDE 0 1 34 1
SIDE 1 1 30 0
SIDE 2 1 33 0
SIDE 3 1 7 1
SOURCES 0 0 0
FACE 19 0
SIDE 0 1 27 1
SIDE 1 1 12 0
SIDE 2 1 28 0
SIDE 3 1 33 1
SOURCES 0 0 0
FACE 20 0
SIDE 0 1 25 0
SIDE 1 1 41 0
SIDE 2 1 44 1
SIDE 3 1 40 1
SOURCES 0 0 0
FACE 21 0
SIDE 0 1 48 1
SIDE 1 1 29 0
SIDE 2 1 43 1
SIDE 3 1 41 1
SOURCES 0 0 0
FACE 22 0
SIDE 0 1 47 0
SIDE 1 1 40 0
SIDE 2 1 39 0
SIDE 3 1 41 1
SOURCES 0 0 0
FACE 23 0
SIDE 0 1 47 1
SIDE 1 1 25 0
SIDE 2 1 48 0
SIDE 3 1 25 1
SOURCES 0 0 0
FACE 24 0
SIDE 0 1 44 0
SIDE 1 1 43 0
SIDE 2 1 23 1
SIDE 3 1 39 1
SOURCES 0 0 0
FACE 25 0
SIDE 0 1 49 1
SIDE 1 1 40 0
SIDE 2 2 37 0 36 0
SIDE 3 1 15 1
SOURCES 0 0 0
FACE 26 0
SIDE 0 1 46 0
SIDE 1 1 15 0
SIDE 2 1 45 0
SIDE 3 1 15 1
SOURCES 0 0 0
FACE 27 0
```

```
SIDE 0 1 46 1
SIDE 1 1 0 1
SIDE 2 1 47 0
SIDE 3 1 49 0
FILE face27surface.dat
SOURCES 0 0 0
FACE 28 0
SIDE 0 2 36 1 37 1
SIDE 1 1 39 0
SIDE 2 2 13 0 20 0
SIDE 3 1 45 1
SOURCES 0 0 0
FACE 29 0
SIDE 0 1 47 0
SIDE 1 1 40 0
SIDE 2 1 38 0
SIDE 3 1 2 1
SOURCES 0 0 0
FACE 30 0
SIDE 0 1 47 1
SIDE 1 1 1 0
SIDE 2 1 46 0
SIDE 3 1 49 1
FILE face30surface.dat
SOURCES 0 0 0
FACE 31 0
SIDE 0 2 37 0 36 0
SIDE 1 1 45 0
SIDE 2 2 19 1 3 1
SIDE 3 1 38 1
SOURCES 0 0 0
FACE 32 0
SIDE 0 1 48 0
SIDE 1 1 41 0
SIDE 2 1 42 0
SIDE 3 1 30 1
SOURCES 0 0 0
FACE 33 0
SIDE 0 1 44 1
SIDE 1 1 38 0
SIDE 2 1 27 0
SIDE 3 1 42 1
SOURCES 0 0 0
BLOCK 0
FACE 0 6 0 0
FACE 1 7 0 0
FACE 2 8 0 0
FACE 3 1 1 3
FACE 4 9 0 0
FACE 5 10 0 0
SOURCES 0 0 0 0
BLOCK 1
FACE 0 0 0 0
FACE 1 1 0 0
FACE 2 2 0 0
FACE 3 3 0 0
FACE 4 4 0 0
FACE 5 5 0 0
SOURCES 0 0 0 0
BLOCK 2
FACE 0 11 0 0
FACE 1 12 0 0
FACE 2 13 0 0
FACE 3 7 1 3
FACE 4 14 0 0
FACE 5 15 0 0
SOURCES 0 0 0 0
BLOCK 3
FACE 0 16 0 0
FACE 1 3 1 3
FACE 2 17 0 0
FACE 3 18 0 0
FACE 4 14 1 1
FACE 5 19 0 0
SOURCES 0 0 0 0
BLOCK 4
FACE 0 20 0 0
FACE 1 21 0 0
FACE 2 11 1 3
FACE 3 22 0 0
FACE 4 23 0 0
FACE 5 24 0 0
SOURCES 0 0 0 0
BLOCK 5
FACE 0 25 0 0
FACE 1 22 1 3
FACE 2 6 1 3
FACE 3 26 0 0
FACE 4 27 0 0
FACE 5 28 0 0
SOURCE 0 0 0 0
BLOCK 6
FACE 0 25 1 3
FACE 1 26 1 3
FACE 2 0 1 3
FACE 3 29 0 0
FACE 4 30 0 0
FACE 5 31 0 0
SOURCES 0 0 0 0
BLOCK 7
FACE 0 20 1 3
FACE 1 29 1 3
FACE 2 16 1 3
FACE 3 32 0 0
FACE 4 23 1 1
FACE 5 33 0 0
SOURCES 0 0 0 0
```

face27surface.dat :

```
26 10 33
DISTRIBUTION 0 1.0 0.1
DISTRIBUTION 1 -0.1 -3.5
DISTRIBUTION 2 1.0 0.1
DISTRIBUTION 3 -0.1 -3.5
0 3
0 1.0 0.0 0.0
1 1.0 0.0 0.0
2 1.0 0.0 0.0
1 3
0 0.95 0.0 -0.00807
1 0.95 -0.00570635 -0.00570635
2 0.95 -0.00807 0.0
2 3
0 0.9 0.0 -0.01448
1 0.9 -0.0102389 -0.0102389
```

```
2 0.9 -0.01448 0.0
3 3
0 0.8 0.0 -0.02623
1 0.8 -0.0185474 -0.0185474
2 0.8 -0.02623 0.0
4 3
0 0.7 0.0 -0.03664
1 0.7 -0.0259083 -0.0259083
2 0.7 -0.03664 0.0
5 3
0 0.6 0.0 -0.04563
1 0.6 -0.0322652 -0.0322652
2 0.6 -0.04563 0.0
6 3
0 0.5 0.0 -0.05294
1 0.5 -0.0374342 -0.0374342
2 0.5 -0.05294 0.0
7 3
0 0.4 0.0 -0.05803
1 0.4 -0.0410334 -0.0410334
2 0.4 -0.05803 0.0
8 3
0 0.3 0.0 -0.06002
1 0.3 -0.0425505 -0.0425505
2 0.3 -0.06002 0.0
9 3
0 0.25 0.0 -0.05941
1 0.25 -0.0420092 -0.0420092
2 0.25 -0.05941 0.0
10 3
0 0.2 0.0 -0.05737
1 0.2 -0.0405667 -0.0405667
2 0.2 -0.05737 0.0
11 3
0 0.18 0.0 -0.055945
1 0.18 -0.039559 -0.039559
2 0.18 -0.055945 0.0
12 3
0 0.16 0.0 -0.054286
1 0.16 -0.0383859 -0.0383859
2 0.16 -0.054286 0.0
13 3
0 0.15 0.0 -0.05345
1 0.15 -0.0377948 -0.0377948
2 0.15 -0.05345 0.0
14 3
0 0.14 0.0 -0.052238
1 0.14 -0.0369378 -0.0369378
2 0.14 -0.052238 0.0
15 3
0 0.12 0.0 -0.049734
1 0.12 -0.0351672 -0.0351672
2 0.12 -0.049734 0.0
16 3
0 0.1 0.0 -0.04683
1 0.1 -0.0331138 -0.0331138
2 0.1 -0.04683 0.0
17 3
0 0.08 0.0 -0.042924
1 0.08 -0.0303518 -0.0303518
2 0.08 -0.042924 0.0
18 3
0 0.075 0.0 -0.04200
1 0.075 -0.0296984 -0.0296984
```

```
2 0.075 -0.04200 0.0
19 3
0 0.06 0.0 -0.038235
1 0.06 -0.0270362 -0.0270362
2 0.06 -0.038235 0.0
20 3
0 0.050 0.0 -0.03555
1 0.050 -0.0251376 -0.0251376
2 0.050 -0.03555 0.0
21 3
0 0.04 0.0 -0.032152
1 0.04 -0.0227348 -0.0227348
2 0.04 -0.032152 0.0
22 3
0 0.025 0.0 -0.02615
1 0.025 -0.0184908 -0.0184908
2 0.025 -0.02615 0.0
23 3
0 0.02 0.0 -0.023501
1 0.02 -0.0166177 -0.0166177
2 0.02 -0.023501 0.0
24 3
0 0.0125 0.0 -0.01894
1 0.0125 -0.0133926 -0.0133926
2 0.0125 -0.01894 0.0
25 3
0 0.0 0.0 0.0
1 0.0 0.0 0.0
2 0.0 0.0 0.0
```

face30surface.dat :

```
26 10 33
DISTRIBUTION 0 1.0 0.1
DISTRIBUTION 1 -0.1 -3.5
DISTRIBUTION 2 1.0 0.1
DISTRIBUTION 3 -0.1 -3.5
0 3
0 1.0 0.0 0.0
1 1.0 0.0 0.0
2 1.0 0.0 0.0
1 3
0 0.95 0.0 -0.00807
1 0.95 0.00570635 -0.00570635
2 0.95 0.00807 0.0
2 3
0 0.9 0.0 -0.01448
1 0.9 0.0102389 -0.0102389
2 0.9 0.01448 0.0
3 3
0 0.8 0.0 -0.02623
1 0.8 0.0185474 -0.0185474
2 0.8 0.02623 0.0
4 3
0 0.7 0.0 -0.03664
1 0.7 0.0259083 -0.0259083
2 0.7 0.03664 0.0
5 3
0 0.6 0.0 -0.04563
1 0.6 0.0322652 -0.0322652
2 0.6 0.04563 0.0
6 3
0 0.5 0.0 -0.05294
```

```
1 0.5 0.0374342 -0.0374342
2 0.5 0.05294 0.0
7 3
0 0.4 0.0 -0.05803
1 0.4 0.0410334 -0.0410334
2 0.4 0.05803 0.0
8 3
0 0.3 0.0 -0.06002
1 0.3 0.0425505 -0.0425505
2 0.3 0.06002 0.0
9 3
0 0.25 0.0 -0.05941
1 0.25 0.0420092 -0.0420092
2 0.25 0.05941 0.0
10 3
0 0.2 0.0 -0.05737
1 0.2 0.0405667 -0.0405667
2 0.2 0.05737 0.0
11 3
0 0.18 0.0 -0.055945
1 0.18 0.039559 -0.039559
2 0.18 0.055945 0.0
12 3
0 0.16 0.0 -0.054286
1 0.16 0.0383859 -0.0383859
2 0.16 0.054286 0.0
13 3
0 0.15 0.0 -0.05345
1 0.15 0.0377948 -0.0377948
2 0.15 0.05345 0.0
14 3
0 0.14 0.0 -0.052238
1 0.14 0.0369378 -0.0369378
2 0.14 0.052238 0.0
15 3
0 0.12 0.0 -0.049734
1 0.12 0.0351672 -0.0351672
2 0.12 0.049734 0.0
16 3
0 0.1 0.0 -0.04683
1 0.1 0.0331138 -0.0331138
2 0.1 0.04683 0.0
17 3
0 0.08 0.0 -0.042924
1 0.08 0.0303518 -0.0303518
2 0.08 0.042924 0.0
18 3
0 0.075 0.0 -0.04200
1 0.075 0.0296984 -0.0296984
2 0.075 0.04200 0.0
19 3
0 0.06 0.0 -0.038235
1 0.06 0.0270362 -0.0270362
2 0.06 0.038235 0.0
20 3
0 0.050 0.0 -0.03555
1 0.050 0.0251376 -0.0251376
2 0.050 0.03555 0.0
21 3
0 0.04 0.0 -0.032152
1 0.04 0.0227348 -0.0227348
2 0.04 0.032152 0.0
22 3
0 0.025 0.0 -0.02615
```

```
1 0.025 0.0184908 -0.0184908
2 0.025 0.02615 0.0
23 3
0 0.02 0.0 -0.023501
1 0.02 0.0166177 -0.0166177
2 0.02 0.023501 0.0
24 3
0 0.0125 0.0 -0.01894
1 0.0125 0.0133926 -0.0133926
2 0.0125 0.01894 0.0
25 3
0 0.0 0.0 0.0
1 0.0 0.0 0.0
2 0.0 0.0 0.0
```