



# *ECERTA*

## Enabling Certification by Analysis

——

**Marie Curie Excellence Team**

**Start: 01 January 2007**
**Duration: 48 months**

www.cfd4aircraft.com

# *Full Potential Code for Aeroelastic Computations*

Prepared by: **Simão Marques**

***Document control data***

| | | | |
|---|---|---|---|
| Deliverable No.: | D **1.2** | Due date: | **31 January 2008** |
| Version: | **Version 1** | Team Leader: | **Prof. Ken Badcock** |
| Date delivered: | 31 January 2008 | **Host Organisation : University of Liverpool** | |

| **Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | **x** |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |

# Contents

Summary . . . . . 1

**1 Introduction** **2**

**2 Theory** **8**

2.1 Flow Equation . . . . . 8

2.2 Boundary Conditions . . . . . 10

2.3 Kutta Condition and Circulation . . . . . 10

2.4 Numeric Stabilising Schemes . . . . . 11

**3 Numerical Algorithm** **15**

3.1 Spatial Discritisation . . . . . 15

3.2 Flux Calculation . . . . . 16

3.3 Boundary and Kutta Condition . . . . . 19

3.3.1 Linear System . . . . . 26

**4 Results** **27**

4.1 2D NACA0012 - Subsonic Case . . . . . 27

**A Grid Generation** **31**

**B Grid Converter** **47**

## Summary

A review of Full-Potential (FP) Methods is given. FP methods are then analysed in terms of their place in the hierarchy of models available to compute transonic flows. Subsequent sections detail a particular approach to solving the FP equation, making use of unstructured grids and modern numerical methods. Results of this approach are compared with Euler solutions for aerofoils.

# Chapter 1

# Introduction

The advent of modern CFD came with the possibility of solving non-linear transonic flow equations. The availability of digital computers in the 1960's, made it increasingly possible to tackle non-linear flow problems. The particular characteristics of transonic flow physics made obtaining insight into transonic flow aerodynamics extremely difficult by analytical methods. The Euler equations describe the most important aspects of transonic flows, however the computational resources in the early 1970's were not available yet to solve the Euler equations. Potential methods can describe the non-linearities of transonic flows, by solving one single equation. For similar computational grids, it is expected that a FP code is an order of magnitude faster than solving the Euler equations. This simplicity comes at a modelling cost, that will be explained in the next sections.

Solving transonic flow problems became the focus of many researchers during the 1970's. The numerical difficulties associated with the Full-Potential and Transonic Small Disturbances (TSD) equations stem from the changing nature of the partial differential equations (PDE). When the flow is subsonic, the PDE's are elliptic in nature; however, in supersonic regions the equations are hyperbolic. The breakthrough numerical methods was made by Murman and Cole [33]. Murman and Cole applied their algorithm to the TSD equation, that switched from a central differencing scheme within the subsonic regions to an upwind scheme in regions of supersonic flow, according to the local Mach number. From this point onwards, a variety of methods were proposed to improve efficiency and the range of applicability of FP methods. Notably, the successive overrelaxation (SOR)

introduced by Steger and Lomax [41], was applied to 2D aerofoils. An improvement over SOR methods was introduced by the use of approximation factorisation (AF) methods [4]. Ballhaus and Bailey [5], and Bailey and Steger [3] solved the TSD equation over wings, using AF algorithms.

AF algorithms with multi-grid formulations, were substantially improved, over the following decades and are still the formulation used in current codes for aeroelastic analysis, e.g. CAP-TSD [6, 13] and the ASP3D [7].

Another successful method was introduced by Caughey and Jameson [26]. Unlike the previous methods, that used finite difference schemes, the authors applied a finite-volume formulation. To stabilise the scheme when in the presence of supersonic flow, artificial viscosity was introduced, to give the scheme an upwind bias. This methodology was based of the widely used and successful codes of the FLO series, developed by Jameson and co-workers [27].

Although improvements over the last 30 years, in both algorithms and computational power have been continuous and impressive, FP methods still have a place in the toolbox of the aerodynamicist. Solution of the Euler and Reynolds Average Navier-Stokes (RANS) equations are obtained routinely nowadays. Nevertheless, 3D complex geometries, unsteady aerodynamic problems, fluid/structures (CFD/CSD) interaction, design optimisation problems still require significant computational resources. For these types of situations FP methods are very competitive, as long as the problem is within the region of validity of the physical model. The limitations of FP methods result from assumptions when formulating the FP or TSD equation: an inviscid, isentropic and irrotational flow is assumed. This limits the applicability of the method up to low supersonic Mach numbers, typical freestream Mach numbers of 1.3 [24]. Since most aircraft (military and civil) have high subsonic cruising velocities, FP methods can provide very accurate resultsat low computational cost, for most cruise conditions.

Therefore, it is logical to apply TSD/FP methods to computationally demanding problems, such as the ones found in aeroelasticity. Good examples of continuous development in the capability of this approach in the field of aeroelasticity is illustrated by the evolution of several computer codes: TRANAIR [29], CAP-TSD [23], ASP3D [7]. The CAP-TSD code has been used extensively for predicting aeroelastic phenomenon such as

flutter and has been applied to several complex configurations, including full aircraft configurations, [9, 11, 28]. The ASP3D is a recent example of a new code being developed based on the TSD equation. This latest example improves on several shortcomings on the previous generation CAP-TSD, namely entropy, vorticity effects and viscous-inviscid capability. Initial validation studies have been performed for several transonic flows, ranging from 2D aerofoils to typical fighter configurations.

**Hierarchy of Models for Transonic Aerodynamics**

Figure 1.1 illustrates some of the most common methods available in Computational Aerodynamics. Several of the methods mentioned, such as DNS, are still well beyond the required resources in order for them to be use in engineering problems. URANS calculations have been used in several problems relevant to aeroelasticity, e.g. rotors, flutter, LCO, [10, 15]. Applying URANS to complete aircraft configurations still requires significant resources, as shown in table 1.1; Euler simulations are still the main tool to model the aeroelastics of complex configurations at transonic conditions. A lower level of approximation for compressible aerodynamics is to use the full potential and transonic small-disturbance equations. These equations, respectively, are the lowest level of approximation to non-linear compressible flows. In the interest of exploring the computational cost, each level of approximation has been further developed and its range of validity extended. A good example is given by Mavriplis [30] for the DLR-F4 model, where coupling *Euler* and *Boundary Layer* equations greatly improved the results for the wing-body configuration at transonic conditions; the coupling yields signifficant improvement in shock location and strength, as shown in figures 1.2 - 1.3, at a marginal increased in computational costs.

An interesting comparison between types of models was performed by Bennet and Edwards [10] and by Eastep et al [15]. Both groups of authors investigated an aeroelastic problem with the CAP-TSD code and the Euler/Navier-Stokes codes, ENS3DAE and the CFL3D-Euler code. The results of this work show a close relation in the majority of cases between all models; a few results indicated strong influence of viscous effects, mainly at supersonic speeds, while the second test wing was more susceptible to

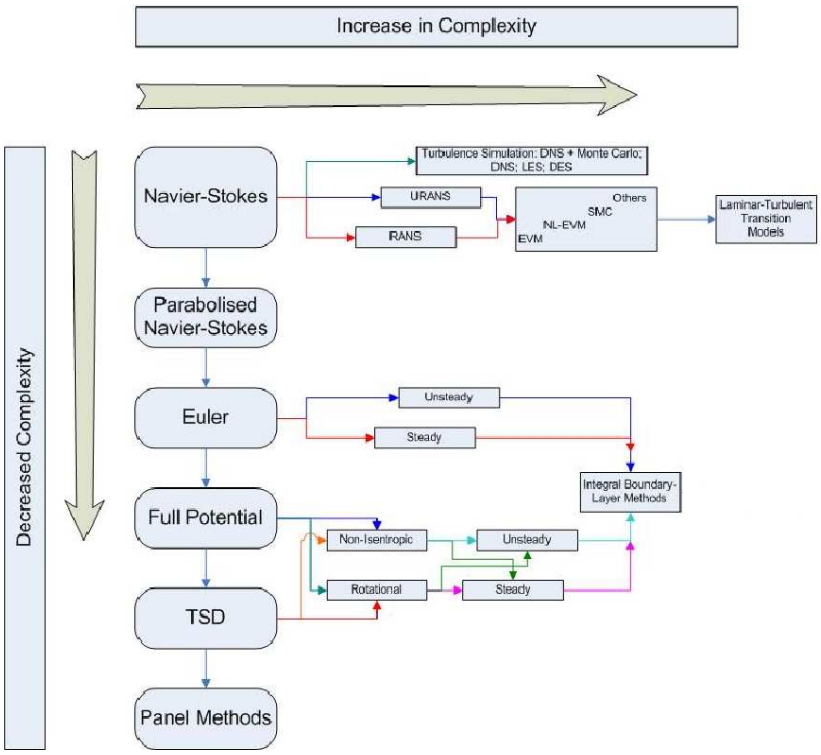
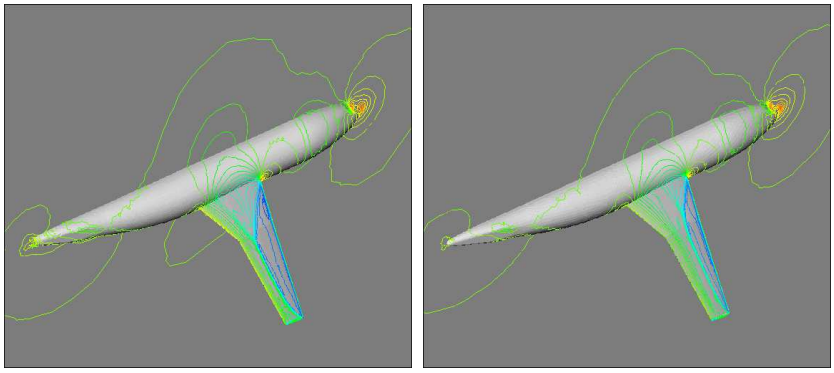



Figure 1.1: Hierarchy of Models for Transonic Aerodynamics

viscous effects before the flutter dip; laminar-turbulent transition location could increase the flutter speed by up to 10%. Eastep et al investigated control surface reversal at transonic conditions, here the model tested showed larger differences between the solutions computed using TSD and Euler methods, in fact the inclusion of viscous effects into the TSD scheme did not yield any improvement to the solution in this particular case. The authors also compared the computation times required for several methods, shown here in table 1. It was concluded that the results indicate that, for

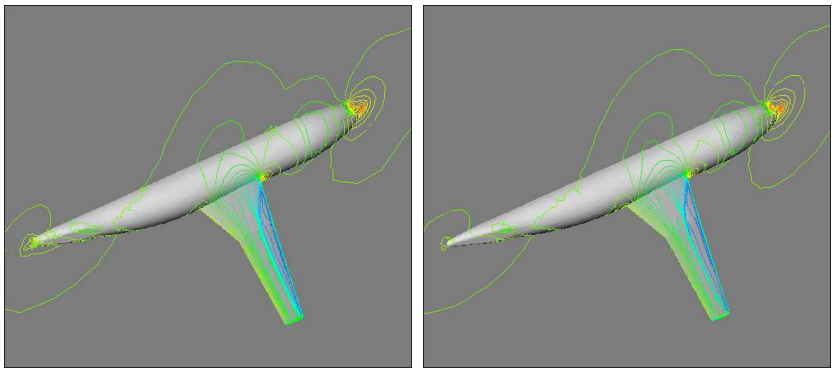
| Model | Grid Size | MG Cycles to Convergence | Normalised Run Time |
|---|---|---|---|
| Euler | $10^5 - 10^6$ | $50 - 100$ | 1.0 |
| Euler + IBL | $10^5 - 10^6$ | $100 - 300$ | $2 - 3$ |
| RANS | $10^6 - 10^7$ | $250 - 1000$ | $50 - 100$ |
| DES | $10^7 - 10^8$ | $5000 - 10000$ | $5000 - 10000$ |
| LES | $10^9 - 10^{10}$ | $O(10^6)$ | $O(10^8)$ |

Table 1.1: Physical Model Requirements (Unstructured grids), [30]



(a) (b)

Figure 1.2: a) Euler Simulation; b) RANS Simulation, [30]



(a) (b)

Figure 1.3: a) Euler+IBL Simulation; b) RANS Simulations, [30]

preliminary design, inviscid calculations provide suitable, although conservative, solutions.

| Aerodynamic Method | Time |
|---|---|
| Linear (surface panels) | 0.1 |
| Inviscid linear (CAP-TSD) | 1.0 |
| Inviscid non-linear (CAP-TSD) | 1.2 |
| Viscous non-linear (CAP-TSDV) | 20 |
| Inviscid non-linear (Euler) | 20 |
| Viscous non-linear (N.S.) | 50 |

Table 1.2: CPU time based on a single workstation, from ref. [15]

# Chapter 2

# Theory

The derivation of the Full-Potential equation is well documented in the literature, [24, 31], and can have several different formulations. The main differences are concerned with the use of a conservative or non-conservative scheme. In this work a conservative formulation is adopted, the reader is pointed towards reference [24] for details of alternative formulations. A brief overview of the derivation of the FP equation is given next.

## 2.1 Flow Equation

The development of any FP method, starts with the velocity potential. This potential exists if it is assumed the flowfield is irrotational [24]:

$$\nabla \times \mathbf{q} = 0 \tag{2.1}$$

An exact velocity potential function can then be defined as:

$$\nabla \phi = \mathbf{q} \tag{2.2}$$

In Cartesian coordinates this is equivalent to:

$$\nabla \phi = \mathbf{q} = u\mathbf{i} + v\mathbf{j} + w\mathbf{k} => \begin{cases} \phi_x = u \\ \phi_y = v \\ \phi_z = w \end{cases} \tag{2.3}$$

The Full Potential equation can be derived from the continuity equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}\big(\rho \mathbf{q}\big) = 0 \tag{2.4}$$

From eq.2.3, the continuity equation, eq.2.4, can be written as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \nabla \phi = 0 \tag{2.5}$$

Eq.2.5 still has two unknowns, $\phi$ and $\rho$, therefore to close the system it is necessary to obtain a relationship between the density and potential function. One way of doing this difficulty is by using Crocco's unsteady equation:

$$T\nabla s + \mathbf{q} \times \mathbf{\Omega} = \nabla h_0 + \frac{\partial \mathbf{q}}{\partial t} \tag{2.6}$$

where:

- $s$ - entropy
- $\mathbf{\Omega}$ - vorticity
- $h$ - enthalpy
- $h_0$ - stagnation enthalpy

Assuming irrotational ($\mathbf{\Omega} = 0$) and isentropic ($s$=constant) flow, and applying eq. 2.2 and eq.2.3 the potential function can be defined as:

$$\frac{\partial \phi}{\partial t} = -h_0 = -\left[h + \frac{1}{2}(u^2 + v^2 + w^2)\right] \tag{2.7}$$

As well as isentropic, the working fluid in this work is air and it is assumed that it behaves as a perfect gas. Hence the following relations for pressure and density are also valid:

$$\frac{p}{p_\infty} = \left(\frac{\rho}{\rho_\infty}\right)^\gamma \tag{2.8}$$

$$p = \rho R T \tag{2.9}$$

By combining eq.2.3, 2.7-2.9 a relationship between the density and potential function derivatives (i.e. velocity components) is obtained:

$$\frac{\rho}{\rho_\infty} = \left[1 + \frac{\gamma - 1}{2} M_\infty^2 \Big(1 - \phi_t - \phi_x - \phi_y - \phi_z\Big)\right]^{\frac{1}{\gamma - 1}} \tag{2.10}$$

## 2.2 Boundary Conditions

For aerodynamic applications, it is necessary to specify the flow conditions at the outer boundaries and solid surfaces. For the outer boundaries, freestream conditions are assumed. This makes the task of calculating the potential function at the farfield straightforward:

$$\phi_\infty = \mathbf{q}_\infty \cdot \overrightarrow{r} \tag{2.11}$$

where:

- $\mathbf{q}_\infty$ - is the freestream velocity
- $\overrightarrow{r}$ - is the position vector of the outer boundary point

For the solid surfaces a flow tangency condition is imposed, i.e. the normal velocity to the surface is 0. This condition is given by:

$$\rho \mathbf{q}_\infty \cdot \mathbf{n} = \rho \frac{\partial \phi}{\partial \mathbf{n}} = 0 \tag{2.12}$$

## 2.3 Kutta Condition and Circulation

In order for the FP methods to be applied in aerodynamics, they should be able to predict aerodynamic loads, i.e. lift, drag, moments. Following the *Kutta-Joukowski* theorem, lift is proportional to circulation:

$$L = \rho_\infty q_\infty \Gamma \tag{2.13}$$

where $\Gamma$ represents the circulation. For a typical aerofoil section, the circulation is defined as:

$$\Gamma = \oint_l \mathbf{q}\, dl \tag{2.14}$$

According to Stokes theorem, the circulation is related to the vorticity by:

$$\Gamma = \oint_l \mathbf{q}\ dl = \int_S \nabla \times \mathbf{q} \cdot \vec{n}\ dS \tag{2.15}$$

From assumptions given by eq.2.1, i.e. vorticity is zero, potential flows can not produce circulation, and therefore lift. To overcome this, a point vortex is added and its contribution is added to the non linear potential function.The outer boundary freestream conditions are modified according to:

$$\phi_{ob} = \phi_\infty + \phi_{vo} \tag{2.16}$$

where:

$$\phi_{vo} = \frac{\Gamma}{2\pi}\theta \tag{2.17}$$

The vortex potential function, $\phi_{vo}$, is a function of the circulation and $\theta$ is the angle formed by the outer boundary location and the wake cut (positive in the anti-clockwise direction), as illustrated by figure 2.1 . Note that $\theta$ is double-valued at 0 and $2\pi$. This corresponds to a jump in the velocity potential from the value at $\theta = 0$ to when $\theta = 2\pi$, across the wake cut. The magnitude of the velocity jump is determined by the Kutta condition. The Kutta condition calculates $\Gamma$ and the components of velocity normal to the wake in such a way that forces the airfoil trailing edge upper and lower pressures to match. This correction is then applied along the cut, until the outer boundary. To calculate the $\Gamma$ at the trailing edge, it is necessary to calculate the potential function contributions from the upper and lower sides, $\Gamma$ is then given by:

$$\Gamma = \phi_{u,te} - \phi_{l,te} \tag{2.18}$$

## 2.4 Numeric Stabilising Schemes

### Type Depending Differencing

As mentioned before, transonic flow equations require an adaptive algorithm to account for the changing nature of the flow equations, when the flow is

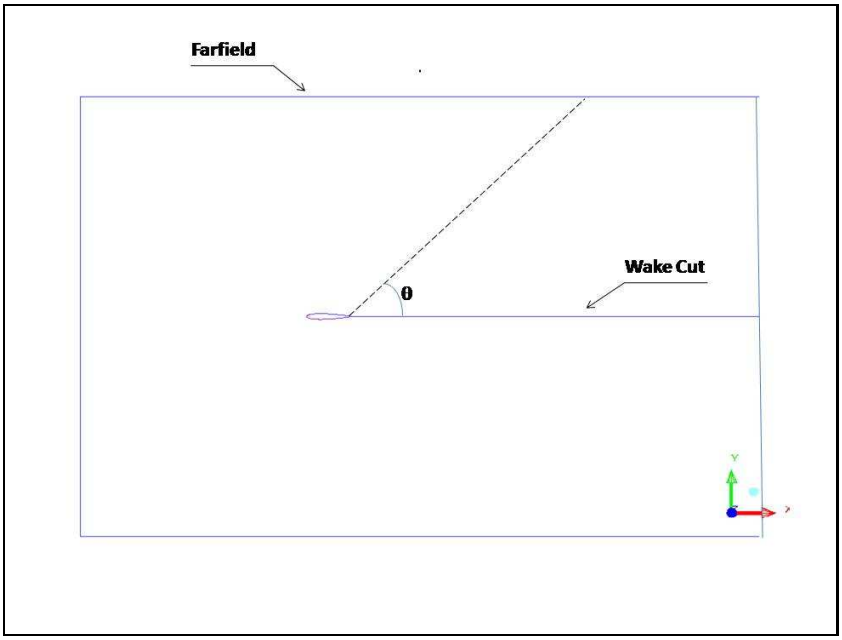



Figure 2.1: Vortex Potential Function

subsonic or supersonic. The first scheme to incorporate such ability [33], makes use of a simple switch to change between a central differencing scheme and an upwind one. Murman and Cole modified the fluxes according to the formula:

$$\overline{f}_{i+1/2,j} = \mu_i f_{i+1/2,j} + (1-\mu_i) f_{i-1/2,j} \tag{2.19}$$

and $\mu$ is defined according to:

$$\mu_i = \begin{cases} 0, & M_{i,j} > 1 \\ 1, & M_{i,j} \leq 1 \end{cases} \tag{2.20}$$

### Artificial Viscosity

An alternative to this method was proposed by Caughey and Jameson [26]. Here, the authors introduced viscosity explicitly into the system. This artificial viscosity modifies the potential equation by introducing extra terms. Eq. 2.5 can be expressed as:

$$\frac{\partial}{\partial x}(\rho\phi_x) + \frac{\partial}{\partial y}(\rho\phi_y) + \frac{\partial}{\partial z}(\rho\phi_z) = 0 \tag{2.21}$$

this is modified by adding the artificial viscosity terms, $P$, $Q$ and $R$:

$$\frac{\partial}{\partial x}(\rho\phi_x + P) + \frac{\partial}{\partial y}(\rho\phi_y + Q) + \frac{\partial}{\partial z}(\rho\phi_z + R) = 0 \tag{2.22}$$

$$\hat{P} = \mu\frac{\rho}{a^2}\Big(u^2\delta_{xx} + uv\mu_{xy}\delta_{xy} + uw\mu_{xz}\delta_{xz}\Big)\phi \tag{2.23}$$

$$\hat{Q} = \mu\frac{\rho}{a^2}\Big(uv\mu_{xy}\delta_{xy} + v^2\delta_{yy} + vw\mu_{yz}\delta_{yz}\Big)\phi \tag{2.24}$$

$$\hat{R} = \mu\frac{\rho}{a^2}\Big(uw\mu_{xz}\delta_{xz} + vw\mu_{yz}\delta_{yz} + w^2\delta_{zz}\Big)\phi \tag{2.25}$$

where $a$ is the speed of sound and $\mu$ is the switching function, given by:

$$\mu = max\left[0, \left(1 - \frac{a^2}{q^2}\right)\right] \tag{2.26}$$

The final component is given by, for example:

$$P = \begin{cases} \hat{P}_{i,j,k}, & u > 0 \\ \hat{P}_{i+1,j,k}, & u < 0 \end{cases} \tag{2.27}$$

**Artificial Density**

A similar, widely used approach, consists of modifying the density formula, [14, 19, 22]. The FP equation, eq.2.28 is modified according to:

$$\frac{\partial}{\partial x}(\tilde{\rho}\phi_x) + \frac{\partial}{\partial y}(\tilde{\rho}\phi_y) + \frac{\partial}{\partial z}(\tilde{\rho}\phi_z) = 0 \tag{2.28}$$

$$\tilde{\rho} = \rho - \frac{\mu}{q}\big[u\rho_x\Delta x + v\rho_y\Delta y + w\rho_z\Delta z\big] \tag{2.29}$$

In this technique the density is calculated at the cell interface; the density gradient $(\rho_x, \rho_y, \rho_z)$ is calculated at the upwind cell centre and the values $\Delta x$, $\Delta y$, $\Delta z$, are twice the distance from the cell interface to the upwind cell centre. The switch, $\mu$ is defined by:

$$\mu = max\left[0, \left(1 - \frac{M_c^2}{M^2}\right)\right] CM^2 \tag{2.30}$$

where $M$ is the local Mach number, $M_c$ is a cut off Mach number (typical 0.95), $C$ is an adjustable constant between the values of 1 and 2.

### Flux Upwind Schemes

One drawback of the Murman and Cole schemes, is that it allows entropy violating expansion shocks. This led researchers to look for procedures that would eliminate non-physical solutions shocks. Engquist and Osher [37] developed such a scheme, their work was further generalised by Osher et al. [38] who compared their scheme with Godunov flux upwinding methods. Following ref. [24], the two flux upwinding schemes, for a 1D equation, can be described as:

$$(\rho\phi_x)_x \cong \frac{1}{\Delta_x}\Big[(\overline{\rho u})_{i+1/2} - (\overline{\rho u})_{i-1/2}\Big] \tag{2.31}$$

where

$$(\overline{\rho u})_{i+1/2} = \rho^* u^* - max\big[\Delta^+_{i-1/2}, \Delta^-_{i+1/2}\big] \text{ , Godunov} \tag{2.32}$$

$$(\overline{\rho u})_{i+1/2} = \rho^* u^* - \Delta^+_{i-1/2} - \Delta^-_{i+1/2} \text{ , Engquist-Osher} \tag{2.33}$$

$$\Delta^+_{i-1/2} = \begin{cases} \rho^* u^* - (\rho u)_{i-1/2},\ if\ u_{i-1/2} > u^* \\ 0,\ if\ u_{i-1/2} < u^* \end{cases} \tag{2.34}$$

$$\Delta^-_{i+1/2} = \begin{cases} 0,\ if\ u_{i+1/2} > u^* \\ \rho^* u^* - (\rho u)_{i+1/2},\ if\ u_{i+1/2} < u^* \end{cases} \tag{2.35}$$

In this procedure the variables $\rho^*$ and $u^*$ represent the values at sonic conditions. The overbar refers to upwind terms. Comparisons between flux upwind schemes and the previous schemes mentioned can be found in refs. [18] and [42]. Overall the authors conclude that the differences between the algorithms are minimal.

# Chapter 3

# Numerical Algorithm

## 3.1 Spatial Discritisation

Several methods were developed over the years to numerically solve equations like eq.2.4 and eq.2.5. The approaches found in the CFD literature, [1, 12, 17, 21], are finite-differences, finite-volumes and finite-element methods. The aim of this particular work, is to develop a fast analysis method for complex configurations. A key task is the necessity to generate computational grids to discretise the flow domain. Taken into consideration the time required to generate computational grids for complex geometries, an unstructured finite volume approach was preferred.

It is now necessary to discretise eq.2.5 in space. This is followed by a second step, where the equations are integrated in time. A finite-volume spatial discretisation is applied to the domain $\mathbf{\Omega}$, which is divided into a finite number of non-overlapping sub-domains or control volumes. The finite-volume method allows certain choices for the control-volumes of a given computational grid. If the grid cells coincide with the control volumes and the variables are stored at the cell centres, the scheme is called cell-centred (as is the case here); another possibility is to use a dual grid and store the variables at the grid nodes, this is known as cell-vertex scheme. The full potential equation, eq.2.5, is applied to each individual control volume $k$. For steady state cases the following equation for mass conservation can be written:

$$\oint_{S_k} \rho \nabla \phi \cdot \vec{n_k} \; ds = 0 \tag{3.1}$$

where $\mathbf{\Omega_k}$ refers to control-volume $k$, $S_k$ is the sub-domain boundary and $\vec{n_k}$ is the unit outward normal at the boundary. For each control-volume $k$, eq.3.1 can be approximated by:

$$\sum_{j=1}^{m-faces} \left(\rho \nabla \phi \cdot \vec{n_j}\right) \Delta A_j = \mathcal{R}_k = 0 \tag{3.2}$$

where $\mathcal{R}_k$ represents the residual of a particular cell.

## 3.2 Flux Calculation

In order to compute the residual, eq.3.2, the flux on each cell face has to be calculated. For the potential function the flux consists of the density and velocity; both these terms are a function of $\phi$ only. Eq. 2.10 defines the density, therefore this can be calculated at any face, once the velocity vector is obtained. This leaves the calculation of the velocity vector, therefore the potential gradient. A typical method used in unstructured grids to compute gradients is the least squares method. In this work we follow an approach proposed by Neel [34].

The starting point to calculate the velocity vector at the cell faces is the potential function, which is stored at the cell centres of the original mesh. For each face $i$, the stencil $S_i$ is defined, as shown in figure 3.1. For the stencil $S_i$, it is possible to fit a $k$ degree polynomial that reconstructs $\phi$ and its gradients:

$$\phi = C_0^i + C_1^i \overline{x} + C_2^i \overline{y} \tag{3.3}$$

where

- $C_1^i$ - $u_i$
- $C_2^i$ - $v_i$
- $\overline{x}$ - $x$ component of distance from face centre to cell centre
- $\overline{y}$ - $y$ component of distance from face centre to cell centre

The objective of this formulation is to calculate $C_1^i$ and $C_2^i$. A weighted least-squares formulation can be used. One advantage of performing a weighted

least squares fit, is that the system reduces to a set of three equations and three unknowns (four in 3D), where the coefficients depend only on geometric values and the weights. The matrix form of the system is:

$$\begin{bmatrix} \sum w_j & \sum w_j\overline{x}_j & \sum w_i\overline{y}_j \\ \sum w_j\overline{x}_j & \sum w_j\overline{x}_j^2 & \sum w_j\overline{x}_j\overline{y}_j \\ \sum w_j\overline{y}_j & \sum w_j\overline{x}_j\overline{y}_j & \sum w_j\overline{y}_j^2 \end{bmatrix} \begin{bmatrix} C_{0,0} \\ C_{1,0} \\ C_{0,1} \end{bmatrix} = \begin{bmatrix} \sum w_j\phi_j \\ \sum w_j\overline{x}_j\phi_j \\ \sum w_j\overline{y}_j\phi_j \end{bmatrix} \tag{3.4}$$

where $w_j$ are the weights of each cell, the index $j$ varies from 1 to m, the total number of cells in the stencil. Since the matrix consists only of geometric values and the weights, it can be computed as a pre-processing step. Instead of storing the matrix itself, it is more efficient to store the inverse. If the matrix is denoted by $\mathbf{B}$, then its inverse is given by:

$$\mathbf{B}^{-1} = \begin{bmatrix} b'_{11} & b'_{12} & b'_{13} \\ b'_{21} & b'_{22} & b'_{23} \\ b'_{31} & b'_{32} & b'_{33} \end{bmatrix} \tag{3.5}$$

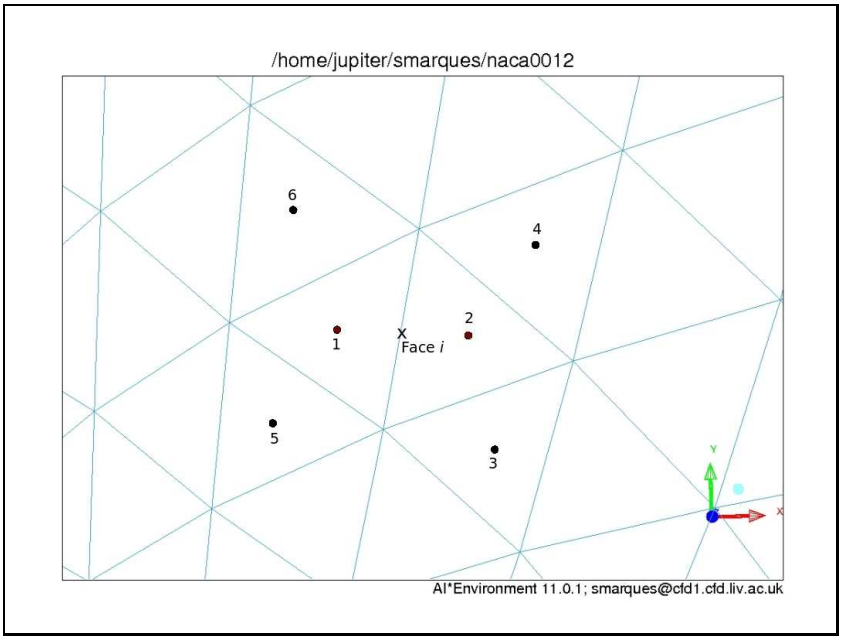



Figure 3.1: Stencil to compute gradient at face $i$

Hence, the solution is simply given by:

$$C_{0,0} = b'_{11}\sum_{j=1}^{m} w_j\overline{\phi}_j + b'_{12}\sum_{j=1}^{m} w_j\overline{x}_j\overline{\phi}_j + b'_{13}\sum_{j=1}^{m} w_j\overline{y}_j\overline{\phi}_j \tag{3.6}$$

$$C_{1,0} = b'_{21}\sum_{j=1}^{m} w_j\overline{\phi}_j + b'_{22}\sum_{j=1}^{m} w_j\overline{x}_j\overline{\phi}_j + b'_{23}\sum_{j=1}^{m} w_j\overline{y}_j\overline{\phi}_j \tag{3.7}$$

$$C_{0,1} = b'_{31}\sum_{j=1}^{m} w_j\overline{\phi}_j + b'_{32}\sum_{j=1}^{m} w_j\overline{x}_j\overline{\phi}_j + b'_{33}\sum_{j=1}^{m} w_j\overline{y}_j\overline{\phi}_j \tag{3.8}$$

It is also necessary to calculate the derivatives of the velocity components with respect to all the values of $\phi_j$ within the stencil. The derivatives are calculated from the least squares reconstruction; with respect to face $i$ and the stencil $S_i$, the derivatives are given by:

$$\frac{\partial u_i}{\partial \phi_j} = b'_{21}w_j + b'_{22}w_j\overline{x}_j + b'_{23}w_j\overline{y}_j \tag{3.9}$$

$$\frac{\partial v_i}{\partial \phi_j} = b'_{31}w_j + b'_{32}w_j\overline{x}_j + b'_{33}w_j\overline{y}_j \tag{3.10}$$

The density at the face centre is calculated using the face centre values for the velocity:

$$\rho_i = \left[1 + \frac{\gamma-1}{2}M_\infty^2\left(1 - u_i^2 - v_i^2\right)\right]^{\frac{1}{\gamma-1}} \tag{3.11}$$

The derivative of the density with respect to $\phi$ is given by:

$$\frac{\partial \rho}{\partial \phi_j} = \frac{\partial \rho}{\partial u_i}\frac{\partial u_i}{\partial \phi_j} + \frac{\partial \rho}{\partial v_i}\frac{\partial v_i}{\partial \phi_j} \tag{3.12}$$

with

$$\frac{\partial \rho}{\partial u_i} = -(M_\infty^2 u_i)\left[1 + \frac{\gamma-1}{2}M_\infty^2\left(1 - u_i^2 - v_i^2\right)\right]^{\frac{2-\gamma}{1-\gamma}} \tag{3.13}$$

$$\frac{\partial \rho}{\partial v_i} = -(M_\infty^2 v_i)\left[1 + \frac{\gamma-1}{2}M_\infty^2\left(1 - u_i^2 - v_i^2\right)\right]^{\frac{2-\gamma}{1-\gamma}} \tag{3.14}$$

For the purely subsonic case, the flux can be computed by:

$$\mathbf{F}_i = A_i\rho_i\left(u_i\vec{n}_x^i + v_i\vec{n}_y^i\right) \tag{3.15}$$

where $A_i$ is the face area and the derivatives are given by:

$$\frac{\partial \mathbf{F}_i}{\partial \phi_j} = A_i \rho_i \left( \frac{\partial u_i}{\partial \phi_j} \vec{n}_x^i + \frac{\partial v_i}{\partial \phi_j} \vec{n}_y^i \right) + A_i \left( u_i \vec{n}_x^i + v_i \vec{n}_y^i \right) \frac{\partial \rho_i}{\partial \phi_j} \tag{3.16}$$

The residual for each control volume is formed by adding the contributions of all its faces. If face $i$ is part of cell $k$, then:

$$\mathcal{R}_k = \mathcal{R}_k + \sigma_{ki} \mathbf{F}_i \tag{3.17}$$

where

$$\sigma_{ki} = \begin{cases} -1, \text{ if } n_x^i \vec{i} + n_y^i \vec{j} \text{ is the outer normal} \\ 1, \text{ if } n_x^i \vec{i} + n_y^i \vec{j} \text{ is the inner normal} \end{cases} \tag{3.18}$$

$$\tag{3.19}$$

Finally, the derivative, $\sigma_{ki} \dfrac{\partial F_i}{\partial \phi_j}$ is added into the $k$th row and $j$th column of the Jacobian matrix.

## 3.3 Boundary and Kutta Condition

### Boundary Conditions

In section 2.2, the formulation for the boundary conditions was introduced. These conditions are enforced by using a layer of *halo cells* adjacent to the grid boundaries (both solid surfaces and farfield cells). No calculation is actually performed on the halo cells; these are just used in order to set the right values on the boundary edges, given by eq. 2.11-2.12, therefore it is only necessary to store the halo cell centres. The position of the halo cells is obtained by mirroring the cell centre of the boundary cell about the boundary edge. This is illustrated in figure 3.2.

(a) Far Field

The potential function in the halo cells is given by:

$$\phi_{hi} = U_\infty x_{hi} + V_\infty y_{hi} + \Gamma_i \theta_i \tag{3.20}$$

(b) Solid Surfaces

From the flow tangency condition, eq. 2.12, the following relationship can be written as:

$$u_i n_x^i + v_i n_y^i = 0 \tag{3.21}$$

The least squares reconstruction can be re-written to separate out the contribution of the halo cell, $h_i$:

$$\begin{aligned} u_i = b'_{21} \sum_{\substack{j \in S_i \\ j \neq h_i}} w_j \phi_j + b'_{22} \sum_{\substack{j \in S_i \\ j \neq h_i}} w_j \phi_j x_j + b'_{23} \sum_{\substack{j \in S_i \\ j \neq h_i}} w_j \phi_j y_j + \\ + b'_{21} w_{hi} \phi_{hi} + b'_{22} w_{hi} \phi_{hi} x_{hi} + b'_{23} w_{hi} \phi_{hi} y_{hi} = \\ = m_0^i + m_1^i \phi_{hi} \end{aligned} \tag{3.22}$$

Similarly

$$v_i = m_2^i + m_3^i \phi_{hi} \tag{3.23}$$

By applying eq.3.22-3.23 to the boundary condition, eq.3.21, it is possible to calculate the conditions for the halo cell:

$$n_x^i \left( m_0^i + m_1^i \phi_{hi} \right) + n_y^i \left( m_2^i + m_3^i \phi_{hi} \right) = 0 \tag{3.24}$$

Eq.3.24 can be rearranged to give:

$$\phi_{hi} = \frac{-n_x^i m_0^i - n_y^i m_2^i}{n_x^i m_2^1 + n_y^i m_3^i}. \tag{3.25}$$

(c) Wake Cut

Another type of boundary condition must be enforced across the wake cells. Figure 3.3 illustrates the wake cut. Cells marked with a $k$ are actually on different faces, i.e. the top cells correspond to the wake upper cut and the lower cells correspond to the faces in the lower cut. Let's denote faces $i_u$ and $i_l$ as being corresponding faces, with the same normal: $n_x^i \vec{i} + n_y^i \vec{j}$.

Then there are two conditions which must be satisfied across $i_u$ and $i_l$:

- continuity of normal velocity component:

$$n_x^i u_{i_u} + n_y^i v_{i_u} = n_x^i u_{i_l} + n_y^i v_{i_l} \tag{3.26}$$

- jump in potential according to the required circulation, $\Gamma$:

$$\phi_u - \phi_l = \Gamma \tag{3.27}$$

where $\phi_u$ is the value reconstructed from the stencil $S_{i_u}$ and $\phi_l$ is the value of $\phi$ reconstructed from $S_{i_l}$. The stencil $S_{i_u}$ features cells above the cut plus a halo cell below the wake cut. On the other hand, the stencil $S_{i_l}$ contains only cells below the cut plus a halo cell above the cut. Hence $\phi_u$ and $\phi_l$ are given by:

$$\phi_u = b_{11}^{'i_u} \sum_{j \in S_{i_u}} w_j \phi_j + b_{12}^{'i_u} \sum_{j \in S_{i_u}} w_j \phi_j x_j + b_{13}^{'i_u} \sum_{j \in S_{i_u}} w_j \phi_j y_j \tag{3.28}$$

$$\phi_l = b_{11}^{'i_l} \sum_{j \in S_{i_l}} w_j \phi_j + b_{12}^{'i_l} \sum_{j \in S_{i_l}} w_j \phi_j x_j + b_{13}^{'i_l} \sum_{j \in S_{i_l}} w_j \phi_j y_j \tag{3.29}$$

The two equations that must be satisfied across the faces $i_u$ and $i_l$ allow the calculation of the halo values for $h_{i_u}$ and $h_{i_l}$. As before, extracting out the halo cells contributions to $\phi_i,\ \phi_l,\ u_{i_u},\ u_{i_l},\ v_{i_u}$ and $v_{i_l}$, it is possible to write the following equations for $\phi_{h_u}$ and $\phi_{h_l}$ as:

$$\begin{aligned} n_x^i \left(m_0^{i_u} + m_1^{i_u}\phi_{h_u}\right) + n_y^i \left(m_2^{i_u} + m_3^{i_u}\phi_{h_u}\right) = \\ = n_x^i \left(m_0^{i_l} + m_1^{i_l}\phi_{h_l}\right) + n_y^i \left(m_2^{i_l} + m_3^{i_l}\phi_{h_l}\right) \end{aligned} \tag{3.30}$$

and

$$\left(m_4^{i_u} + m_5^{i_u}\phi_{h_u}\right) - \left(m_4^{i_l} + m_5^{i_l}\phi_{h_l}\right) = -\Gamma \tag{3.31}$$

Eq.3.30-3.31 can be written in matrix form as:

$$H \begin{bmatrix} \phi_{h_u} \\ \phi_{h_l} \end{bmatrix} = \vec{g} \tag{3.32}$$

then:

$$\begin{bmatrix} \phi_{h_u} \\ \phi_{h_l} \end{bmatrix} = H^{-1}\vec{g} \tag{3.33}$$

where:

$$H = \begin{bmatrix} n_x^i m_1^{i_u} + n_y^i m_3^{i_u} & -\left(n_x^i m_1^{i_l} + n_y^i m_3^{i_l}\right) \\ m_5^{i_u} & -m_5^{i_l} \end{bmatrix} \tag{3.34}$$

$$\vec{g} = \begin{bmatrix} -\left(n_x^i m_0^{i_u} + n_y^i m_2^{i_u}\right) + n_x^i m_0^{i_l} + n_y^i m_2^{i_l} \\ -m_4^{i_u} + m_4^{i_l} - \Gamma \end{bmatrix} \tag{3.35}$$

Finally, it is possible to obtain:

$$\begin{aligned} \phi_{h_u} &= H_{11}^{-1}\left[-\left(n_x^i m_0^{i_u} + n_y^i m_2^{i_u}\right) + \left(n_x^i m_0^{i_l} + n_y^i m_2^{i_l}\right)\right] \\ &\quad + H_{12}^{-1}\left(-m_4^{i_u} + m_4^{i_l} - \Gamma\right) \end{aligned} \tag{3.36}$$

$$\begin{aligned} \phi_{h_l} &= H_{21}^{-1}\left[-\left(n_x^i m_0^{i_u} + n_y^i m_2^{i_u}\right) + \left(n_x^i m_0^{i_l} + n_y^i m_2^{i_l}\right)\right] \\ &\quad + H_{22}^{-1}\left(-m_4^{i_u} + m_4^{i_l} - \Gamma\right) \end{aligned} \tag{3.37}$$

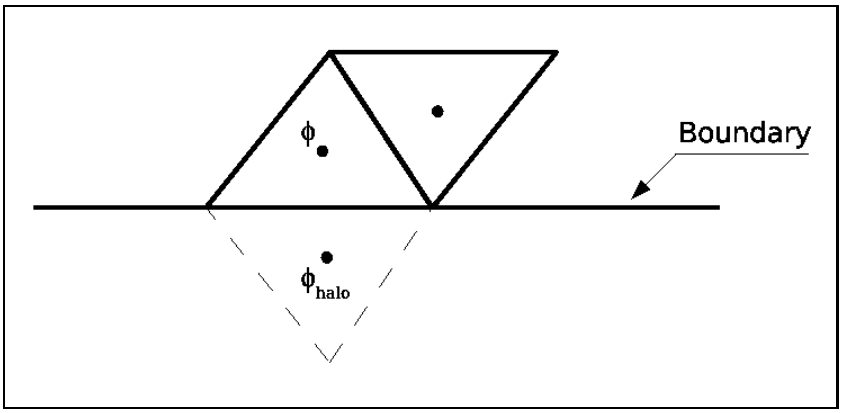



Figure 3.2: Halo Cell

### Kutta Condition

As explained in section 2.3, in order to compute aerodynamic loads, it is necessary to enforce the Kutta condition at the trailing edge of the aerofoil, up to the farfield. In this work, this process is linked with the grid generated to solve a particular problem. Since unstructured grids are being used, it is possible to generate or find, a sequence of edges that link the aerofoil

trailing edge to the farfield. This sequence represents the cut due to the double value of $\theta$ and it is referred to as *Kutta Line*. The only care taken in constructing this artifice, is to ensure that each cell is only updated once by this process. Figure 3.3 illustrates the *Kutta Line* and the cells tagged for updating. The circulation is calculated at the trailing edge, by splitting the calculation in contributions from the upper and lower surfaces. Hence, to reconstruct $\phi$ at the trailing edge, two stencils are required, one for the upper surface and another for the lower surface, $S^{teu}$ and $S^{tel}$. These stencils should be upstream of the trailing edge, see figure 3.3.

The values for $\phi^u_{te}$ and $\phi^l_{te}$ can be reconstructed by using the least-squares formulation and the corresponding stencils, $S^{teu}$ and $S^{tel}$:

$$\phi^u_{te} = b'^{teu}_{11} \sum_{j \in S^{teu}} w_j \phi_j + b'^{teu}_{12} \sum_{j \in S^{teu}} w_j \phi_j x_j + b'^{teu}_{13} \sum_{j \in S^{teu}} w_j \phi_j y_j \tag{3.38}$$

$$\phi^l_{te} = b'^{tel}_{11} \sum_{j \in S^{tel}} w_j \phi_j + b'^{teu}_{12} \sum_{j \in S^{tel}} w_j \phi_j x_j + b'^{tel}_{13} \sum_{j \in S^{tel}} w_j \phi_j y_j \tag{3.39}$$

Then, the circulation is given by:

$$\Gamma = -\left(\phi^u_{te} - \phi^l_{te}\right) \tag{3.40}$$

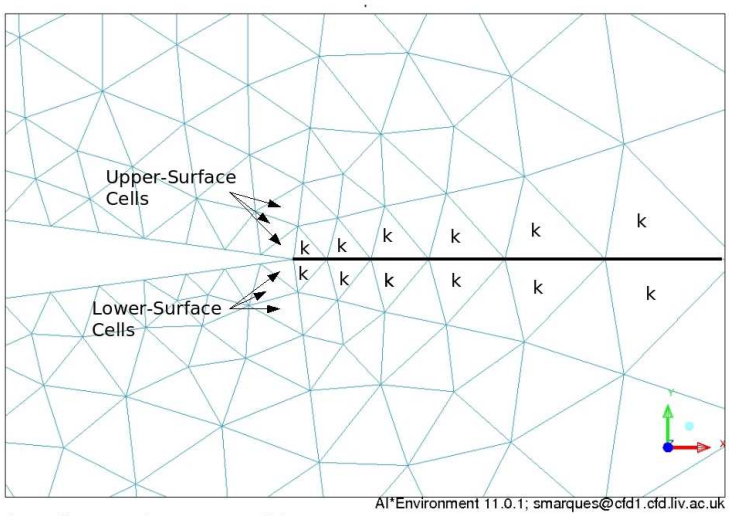



Figure 3.3: Kutta Line

Note that $\Gamma$ depends on the values calculated both faces forming the trailing edge, as showed in figure 3.3, i.e. in either of the stencils $S^{teu}$ and $S^{tel}$.

The far field and wake halo cells depend on the value of $\Gamma$, and hence on the cells in $S^{teu}$ and $S^{tel}$.

### Modified Jacobians for Boundary Conditions

Now, the derivatives added into the Jacobian matrix include those with respect to the halo cells values of the potential. The boundary conditions have related these halo cells to internal cell values and to the circulation. In general it is possible to write:

$$\phi_{h_i} = \phi_{h_i}(\phi_k, \phi_p) \text{ for } k \in S_i^h,\ p \in \left(S^{teu} \cup S^{tel}\right) \tag{3.41}$$

where $S_i^h$ is the stencil for halo cell $h_i$.

Hence:

$$\frac{\partial \mathbf{F}_i}{\partial \phi_j} = \frac{\partial \mathbf{F}_i}{\partial \phi_{h_i}} \frac{\partial \phi_{h_i}}{\partial \phi_j} + \frac{\partial \mathbf{F}_i}{\partial \phi_{h_i}} \frac{\partial \phi_{h_i}}{\partial \phi_k},\ \text{for } j \in S_i^h,\ \text{for } k \in \left(S^{teu} \cup S^{tel}\right) \tag{3.42}$$

with:

$$\frac{\partial \mathbf{F}_i}{\partial \phi_{h_i}} \frac{\partial \phi_{h_i}}{\partial \phi_k} = \frac{\partial \mathbf{F}_i}{\partial \phi_{h_i}} \frac{\partial \phi_{h_i}}{\partial \Gamma} \frac{\partial \Gamma}{\partial \phi_k},\ \text{for } k \in \left(S^{teu} \cup S^{tel}\right) \tag{3.43}$$

To complete the formulation, it is still necessary to define $\dfrac{\partial \phi_{h_i}}{\partial \phi_j}$ for the 3 types of boundary conditions, and $\dfrac{\partial \phi_{h_i}}{\partial \Gamma}$, $\dfrac{\partial \Gamma}{\partial \phi_k}$.

(a) Far field:

$$\frac{\partial \phi_{h_i}}{\partial \Gamma} = \theta_{h_i} \tag{3.44}$$

where $\theta_{h_i}$ is defined in section 2.3.

(b) Solid Surface:

$$\frac{\partial \phi_{h_i}}{\partial \phi_j} = -\frac{n_x^i}{n_x^i m_1^i + n_y^i m_3^i} \frac{\partial m_0^i}{\partial \phi_j} - \frac{n_y^i}{n_x^i m_1^i + n_y^i m_3^i} \frac{\partial m_2^i}{\partial \phi_j} \tag{3.45}$$

where:

$$\frac{\partial m_0^i}{\partial \phi_j} = b'_{21} w_j + b'_{22} w_j x_j + b'_{23} w_j y_j \tag{3.46}$$

$$\frac{\partial m_2^i}{\partial \phi_j} = b'_{31} w_j + b'_{32} w_j x_j + b'_{33} w_j y_j \tag{3.47}$$

$S_i^h$ is obtained by excluding $h_i$ from $S_i$.

Finally:

$$\frac{\partial \phi_{h_i}}{\partial \Gamma} = 0 \tag{3.48}$$

(c) Wake Cut:

$$\frac{\partial \phi_{hi_u}}{\partial \phi_j} = H_{11}^{-1} \left[ -n_x^i \frac{m_0^{i_u}}{\partial \phi_j} - -n_y^i \frac{m_2^{i_u}}{\partial \phi_j} + n_x^i \frac{m_0^{i_l}}{\partial \phi_j} + n_y^i \frac{m_2^{i_l}}{\partial \phi_j} \right] \tag{3.49}$$

$$\frac{\partial \phi_{hi_l}}{\partial \phi_j} = H_{21}^{-1} \left[ -n_x^i \frac{m_0^{i_u}}{\partial \phi_j} - -n_y^i \frac{m_2^{i_u}}{\partial \phi_j} + n_x^i \frac{m_0^{i_l}}{\partial \phi_j} + n_y^i \frac{m_2^{i_l}}{\partial \phi_j} \right] \tag{3.50}$$

Here:

$$\left.\begin{aligned} \frac{\partial m_0^{i_u}}{\partial \phi_j} &= b'^{i_u}_{21} w_j + b'^{i_u}_{22} w_j x_j + b'^{i_u}_{23} w_j y_j \\ \frac{\partial m_2^{i_u}}{\partial \phi_j} &= b'^{i_u}_{31} w_j + b'^{i_u}_{32} w_j x_j + b'^{i_u}_{33} w_j y_j \end{aligned}\right\}, \text{ for } j \in S_{i_u},\ j \neq g_{i_u} \tag{3.51}$$

similarly for $\frac{\partial m_0^{i_l}}{\partial \phi_j}$, $\frac{\partial m_2^{i_l}}{\partial \phi_j}$

also:

$$\frac{\partial \phi_{h_{i_u}}}{\partial \Gamma} = -H_{12}^{-1} \tag{3.52}$$

$$\frac{\partial \phi_{h_{i_l}}}{\partial \Gamma} = -H_{23}^{-1} \tag{3.53}$$

(d) Circulation

$$\frac{\partial \Gamma}{\partial \phi_j} = b'^{teu}_{11} w_j + b'^{teu}_{12} w_j x_j + b'^{teu}_{13} w_j y_j, \text{ for } j \in S^{teu} \tag{3.54}$$

$$\frac{\partial \Gamma}{\partial \phi_j} = b'^{tel}_{11} w_j + b'^{tel}_{12} w_j x_j + b'^{tel}_{13} w_j y_j, \text{ for } j \in S^{tel} \tag{3.55}$$

### 3.3.1 Linear System

The method described above is used to set up the following linear system:

$$\frac{\partial \mathcal{R}}{\partial \Phi}\Delta\Phi = -\mathcal{R}^n \tag{3.56}$$

$$\Delta\Phi = \Phi^{n+1} - \Phi^n \tag{3.57}$$

$\Phi$ and $\mathcal{R}$ are vectors containing the potential function and the residual, respectively, while $n$ represents the current time level. These vectors are formed by using the method described above. The system is solved using GMRES with an ILU$\Phi$ preconditioner. The package PETSc is used to perform these operations, [36].

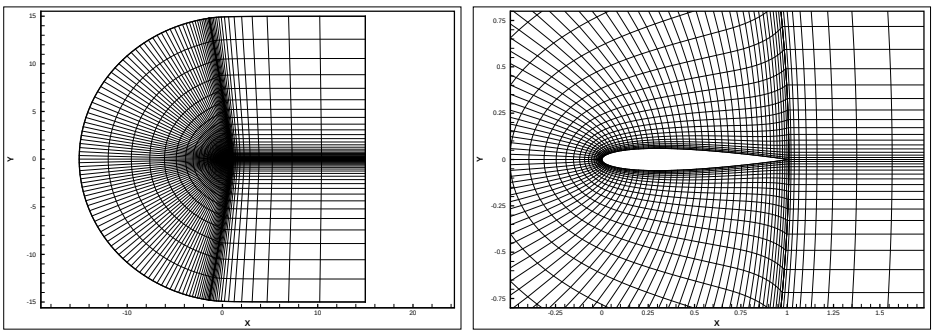
# Chapter 4

# Results

## 4.1 2D NACA0012 - Subsonic Case

As mentioned before, the validity of full-potential methods is limited to Mach number less than 1.3. The first test case used to validate the method presented in this work, was the NACA 0012 aerofoil in subsonic flow, M=0.2.

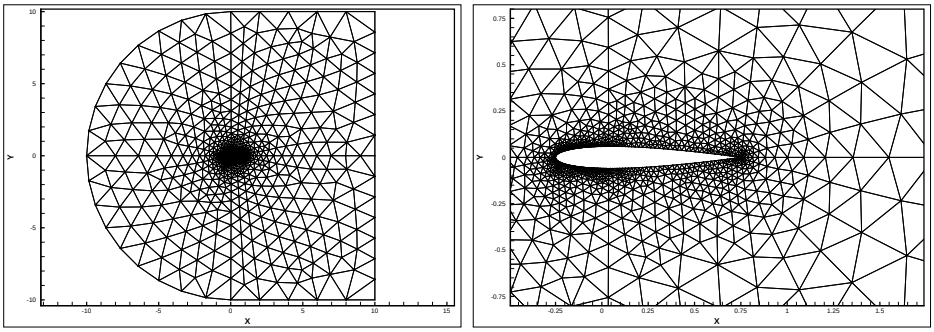
The Euler solutions were obtained in a structured grid, finite-volume code. The grid used is showed in Figure 4.1and consists of 4096 cells and 4323 points. The Full Potential solutions were obtained in an unstructured grid, as shown in Figure 4.2, and the grid consists of 1510 points and 2792. The surface resolution in both cases is identical.



(a) Overview (b) Aerofoil Detail

Figure 4.1: Euler Structured Grid

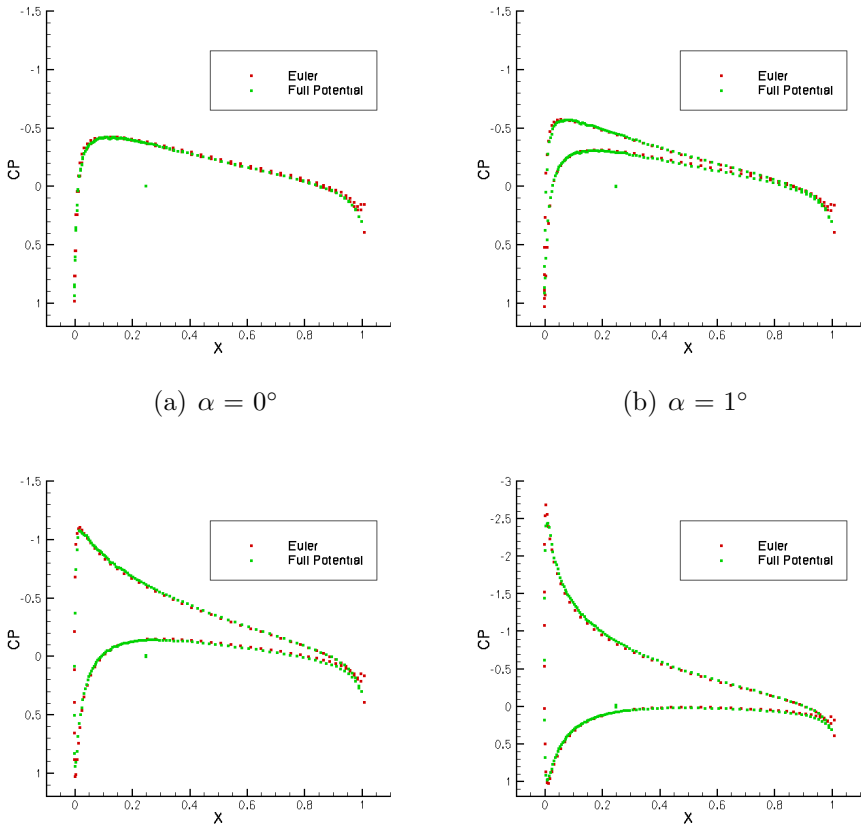
Figure 4.3 shows the $C_P$ distribution at four different incidence angles: 0°, 1°, 3°, 6°, while Figure 4.4 show the iso-Mach contours at 0° and 6° incidence, respectively.



(a) Overview (b) Aerofoil Detail

Figure 4.2: Full-Potential Unstructured Grid

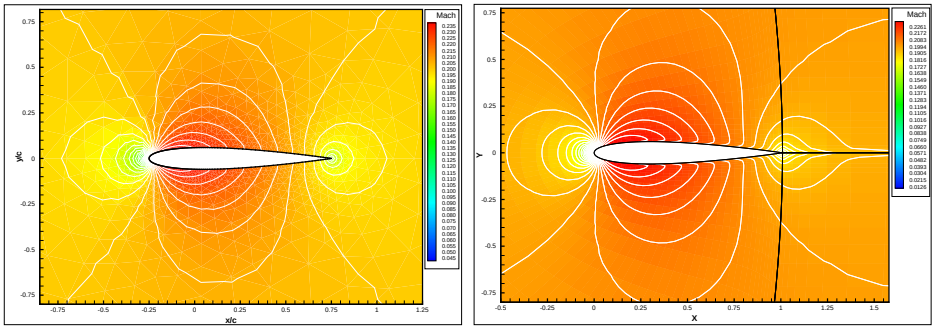
Inspection of both types of plots, shows excellent agreement with the establish Euler solver. The pressure coefficient plots show a good agreement in predicting both stagnation points and suction peaks. The iso-Mach contours also reveal identical flow features between both solver methodologies. The efficiency of this method is illustrated by the convergence plots, shown in figure 4.5. The Full-Potential method reaches a converged solution in 38 implicit steps. The initial results are encouraging and work is being carried out to extend the Full Potential method to deal with compressibility features, such as shock-waves.




(a) $\alpha = 0°$ (b) $\alpha = 1°$

(c) $\alpha = 3°$ (d) $\alpha = 6°$

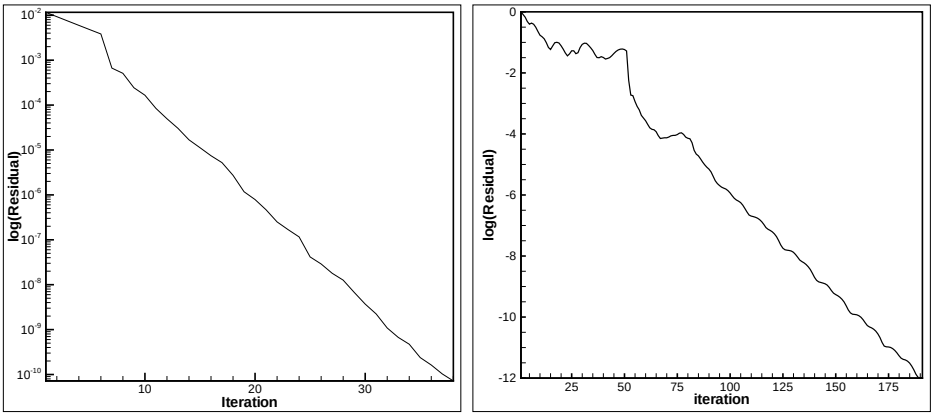
Figure 4.3: NACA 0012 $C_P$ Distribution - M=0.2




(a) Full Potential - $\alpha = 0°$ (b) Euler - $\alpha = 0°$

(c) Full Potential - $\alpha = 6°$ (d) Euler - $\alpha = 6°$

Figure 4.4: NACA 0012 Iso-Mach Contours - M=0.2




(a) Full Potential (b) Euler

Figure 4.5: NACA 0012 Convergence History

# Appendix A

# Grid Generation

In this chapter a step by step grid generation procedure is described. The process for an aerofoil is described first followed by the corresponding approach in 3D.

## 2D Grid Generation

The first step to start generating grids using ICEM is to start the software and starting a *new project*. To launch ICEM open a terminal window and setup the correct ICEM version by typing:

```
> source /software/ro/ICEMCFD11SP1/setup
> icemcfd
```

With the ICEM GUI initiated, a new project can initiated by making:

```
File > New Project
```

A suitable name for the project is requested and this will be the reference used to define several output files created by ICEM.

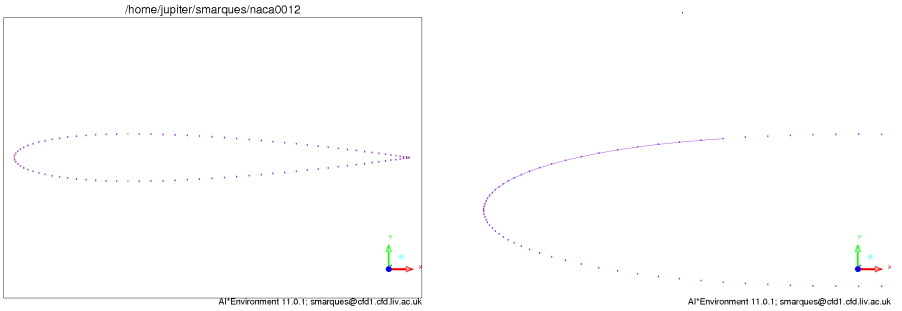
Usually, for a 2D section, the geometry is in the form of a list of points. The first step in generating the grid is inputing all the points that define the geometry. If the points are defined in a file, it is required that the first line includes 2 parameters, i.e. a line with two integers, that correspond to the number of points for each curve and the number of curves for each surfaces. To load the file use the main menu:

```
File > Import Geometry > Formatted Point Data
```

Once the points are loaded, they should appear similar to the example in Figure A.1. To create the aerofoil shape, it is necessary to form a close surface by creating a B-Spline by joining all the points; these commands can be found in *Geometry* toolbar:

```
Geometry > Create/Modify Curve > From points
```

For a typical aerofoil, this will consist of two splines, corresponding to the upper and lower surfaces. With the geometry created, it is necessary



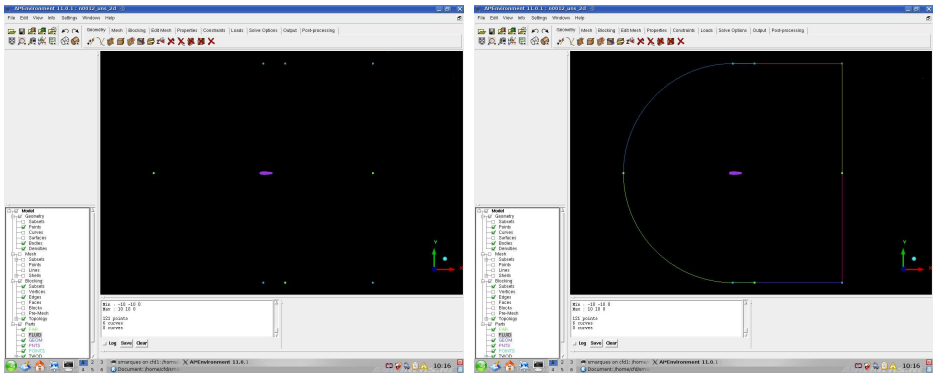

(a) (b)

Figure A.1: Grid Overview

to define the outer boundaries of the flowfield. In this example, the upstream boundary will be formed by a semi-circle and the remainder of the outer boundaries will be defined by a rectangle. This shape is not a requirement and can/should be adapted to the characteristics of the problem. The starting point is to create 8 new points. The coordinates are shown in Table A.1. The actual boundary is formed by connecting each pair of points with the exception of the semi-circle, which is generated by the specific tool for creating arc with 3 points. The ICEM GUI output is illustrated in Figure A.2.

| | X | Y |
|---|---|---|
| Point 1 | 40.0 | 40.0 |
| Point 2 | 40.0 | 0.0 |
| Point 3 | 40.0 | -40.0 |
| Point 4 | 5.0 | -40.0 |
| Point 5 | -5.0 | -40.0 |
| Point 6 | -45.0 | 0.0 |
| Point 7 | -5.0 | 40.0 |
| Point 8 | 5.0 | 40.0 |

Table A.1: Outer Boundary Points

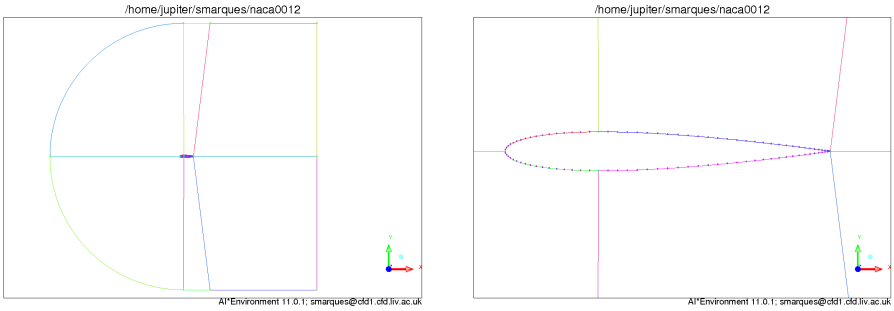


(a) (b)

Figure A.2: Farfield Definition

In order to generate a pure 2D grid in ICEM, what is actually done is generating a surface mesh. Hence the next step is to define the surfaces where the grid will actually be constructed. In general, geometry and topology definition follow a bottom-up approach, i.e. first points are generated, then curves, then surfaces and from the surfaces, volumes can be defined. In this example, the aim is to generate a smooth unstructured grid. To facilitate this task, extra surfaces and lines are used of what would be absolutely necessary to generate a 2D grid. In the end 6 surfaces are generated. The final surfaces and lines used are illustrated in Figure A.3 In this example, surfaces are defined by sets of 4 connected curves. When selecting this option from the *Geometry* toolbar, the smallest tolerance aloud should be used.

`Geometry > Create/Modify Surface > Simple Surface`




(a) (b)

Figure A.3: Surface Lines

Repeat the process for each set of 4 curves, to generate the 6 surfaces. Figure A.4 shows the first surface created, where the selected curves are highlighted in yellow. Figure A.5shows the representation of the surfaces in ICEM. To ensure an adequate cell size distribution across the grid, it is possible to assign node distributions along selected curves. The aerofoil surface boundary is divided into four curves. The two leading edge curves are assigned 55 nodes, with initial and final spacings of 0.005 and 0.01 respectively. The two downstream parts of the aerofoil are discretise with 40 nodes with initial and final spacings of 0.01. Hyperbolic laws are used to distribute the nodes and achieving the correct spacings. The result is presented in Figure A.6 and Figure A.7. The same process is applied to the remainder of the curves. Particular attention is given to the regions close to the aerofoil to ensure the node spacing is consistent between adjacent curves. At this stage, is possible to generate the grid. ICEM offers several alternatives to generate the final grid. The most robust generator is referred to as *Octree* method. To apply this method, in the *Global Mesh Paramenter* menu, select *Patch Independent* as the mesh method. On the other hand if instead, the user selects *Patch Dependent* and applies the option to keep the option *Respect line elements* **ON**, ICEM will respect the curve spacings and in general produce a more smooth grid. The differences are illustrated in Figure A.8 and Figure A.9. Once all parameters are set, the command to generate the grid is found in the *Compute Mesh* menu, for 2D cases the *Surface Mesh Only* option must be selected.

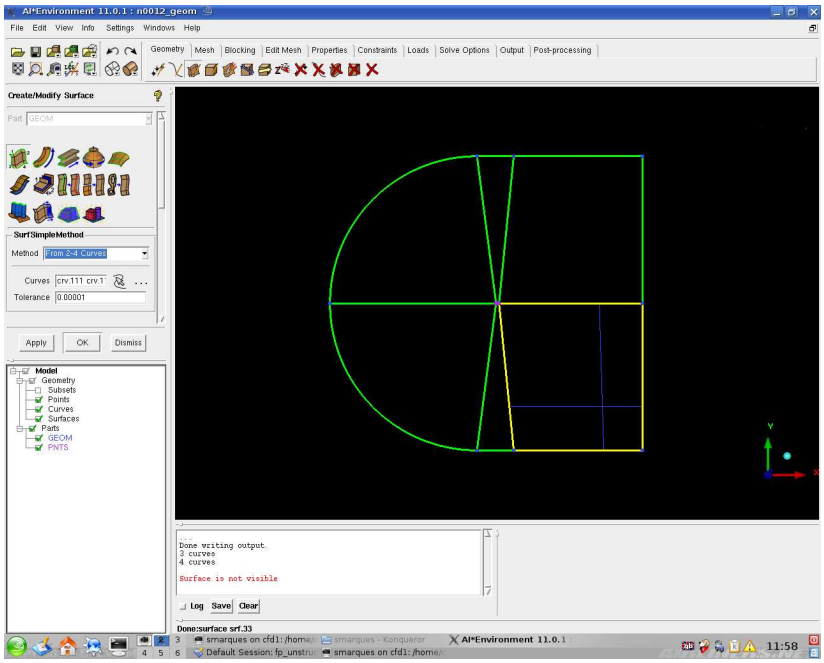
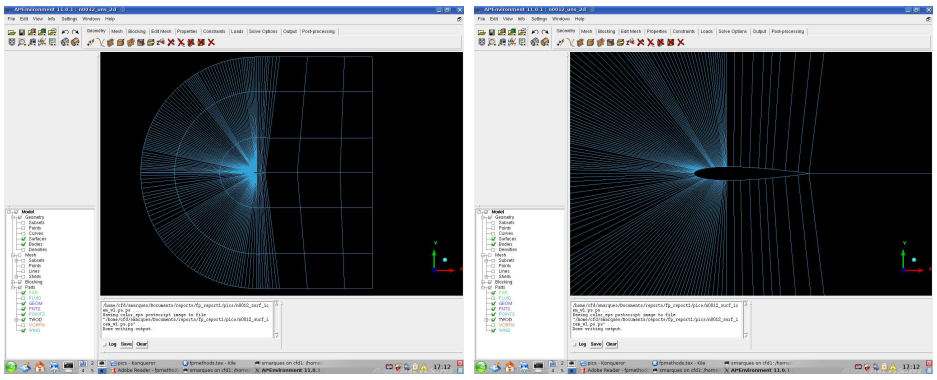



Figure A.4: Creating Surfaces



(a) (b)

Figure A.5: Grid Surfaces

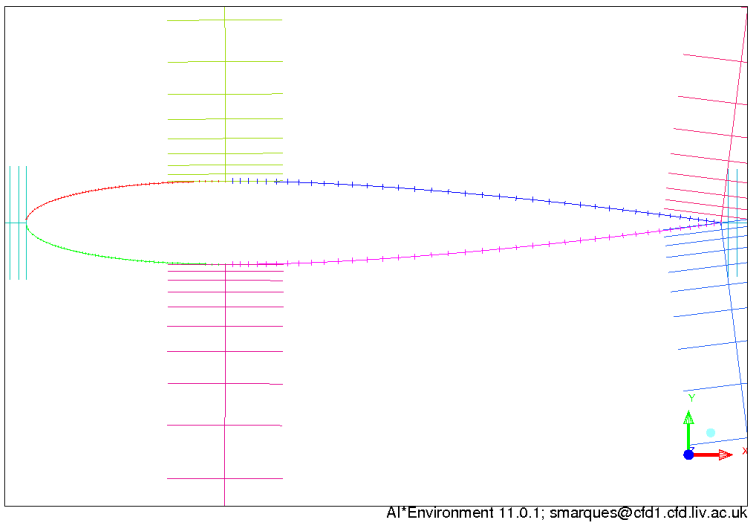



Figure A.6: Aerofoil Node Distribution

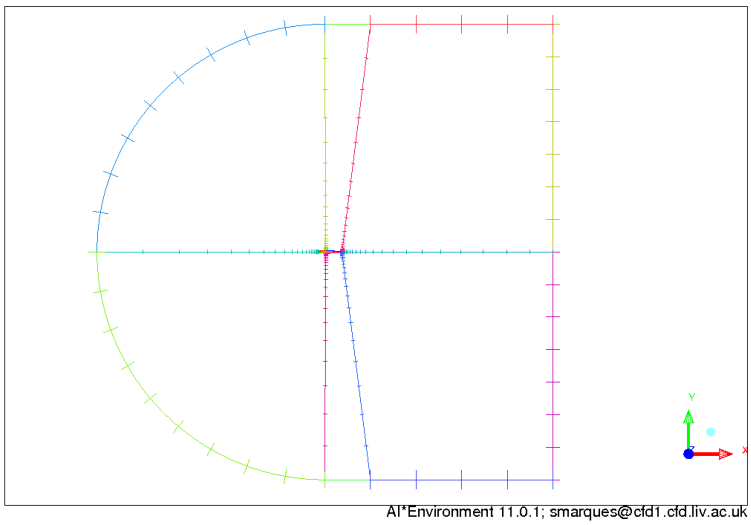
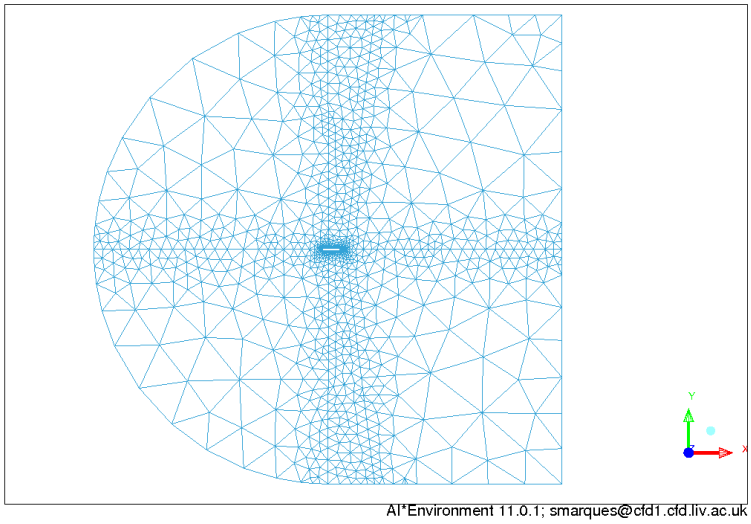
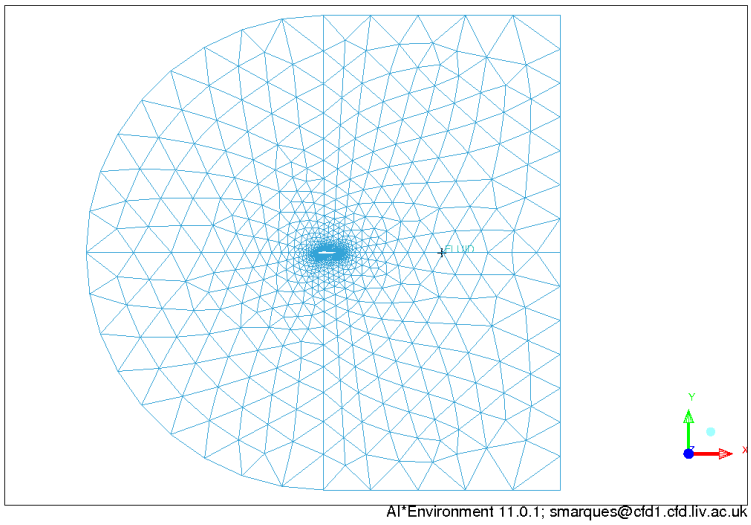



Figure A.7: Curve Node Distribution

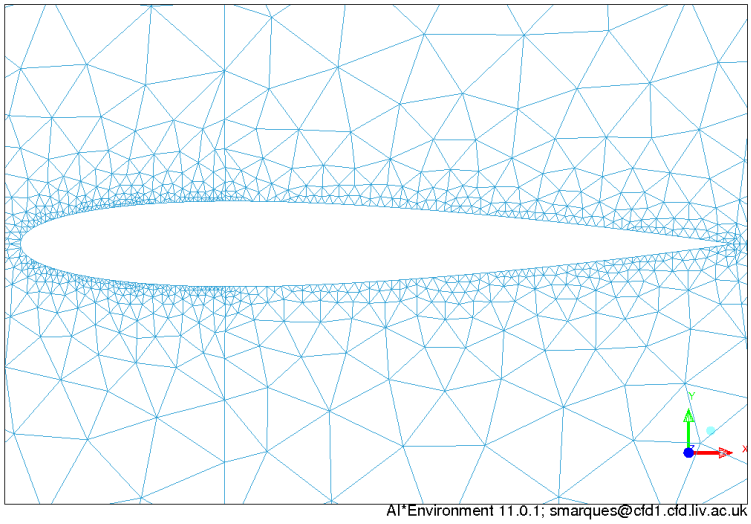



(a)

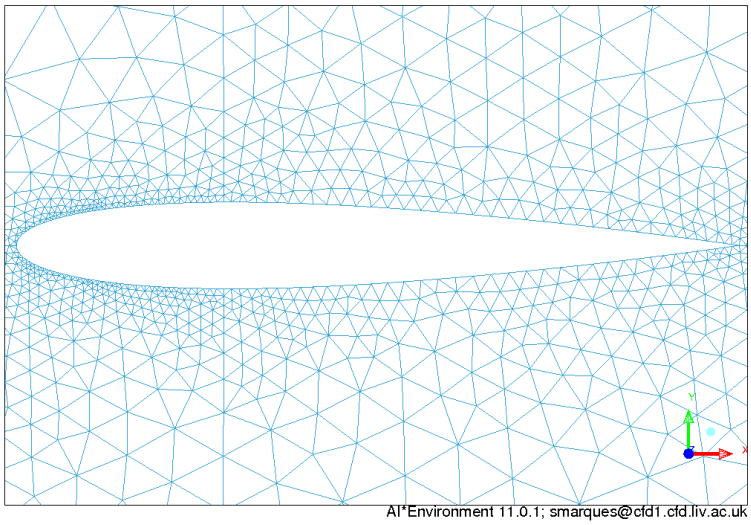



(b)

Figure A.8: Grid Method Comparison - Grid General View




(a)



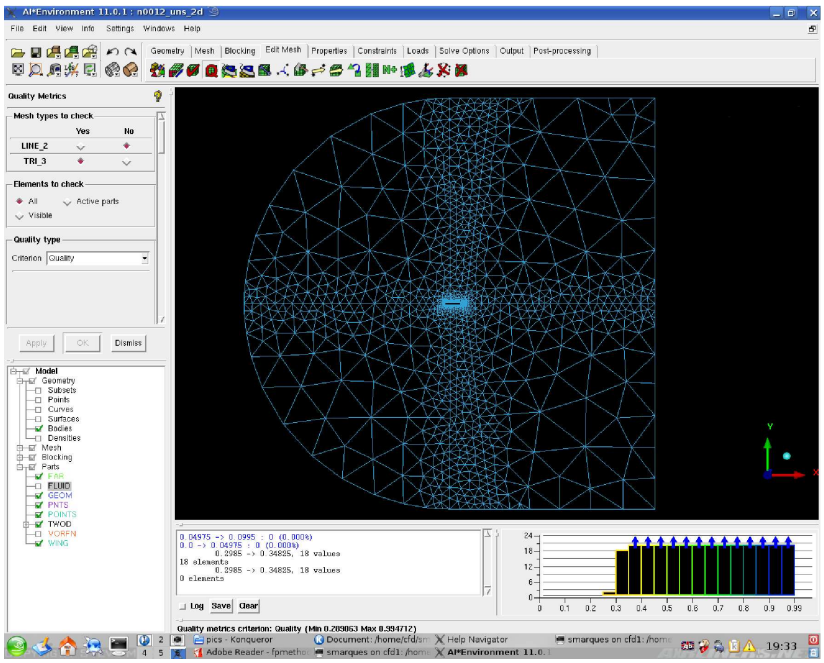

(b)

Figure A.9: Grid Method Comparison - Aerofoil Detail View

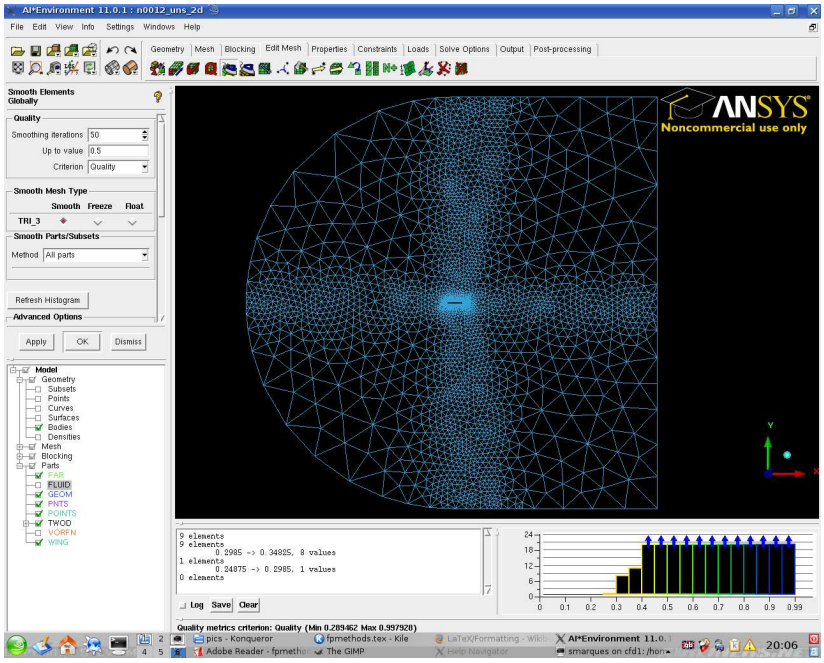
Once the grid is generated it is possible to check and improve the quality of the elements. Several other operations are also available such as refine/coarsen the whole or parts of the grid. There are several measures to improve the quality of a given grid. The main options are located in the *Edit Mesh* menu.

- Repair Mesh
  - $->$ Remesh Bad Elements
  - $->$ Smooth Surface
- Smooth Elements
- Repair Mesh
  - $->$ Edge Swap
  - $->$ (Split Tree Elements)

The results of applying these techniques are shown in Figure A.10. The quality improvement is marginal, but this is mainly due to the simplicity of the problem. Nevertheless, as the histograms in both figures indicate, there have been clear improvements. For more complex geometries and 3D grids this techniques are fundamental to obtain good results.




(a)




(b)

Figure A.10: Grid Quality Improvement: a) Initial Grid; b) Final Grid

## 3D Grid Generation

As expected, the situation is more complex for 3D grids. The example explained here follows one of the tutorials from ICEM. The tutorials included in the software are very useful for the user to become familiar with the several options available. However, in this particular example, the results from the tutorial are not suitable for the FP method previously described. Furthermore, the resultant grid would not consist of tetrahedral exclusively.

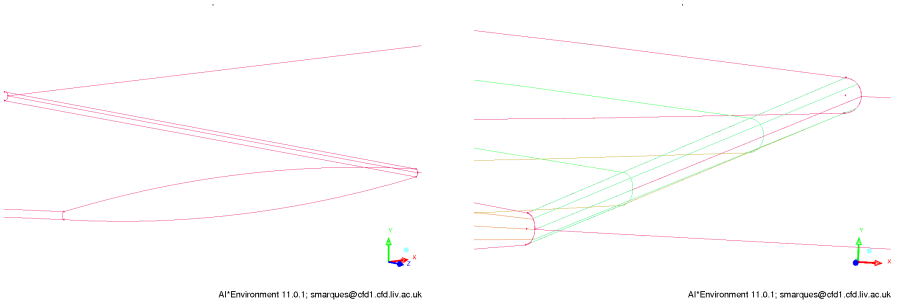
In general it is expected for a 3D geometry to be created by some CAD package and then imported into ICEM. In this case the geometry is available from the tutorial directories installed with the software. The geometry is loaded by using the icon corresponding to *open geometry file* and follow the path to the tutorial directory:

```
.../v110/docu/Tutorials/CFD_Tutorial_Files>FinConfig
```

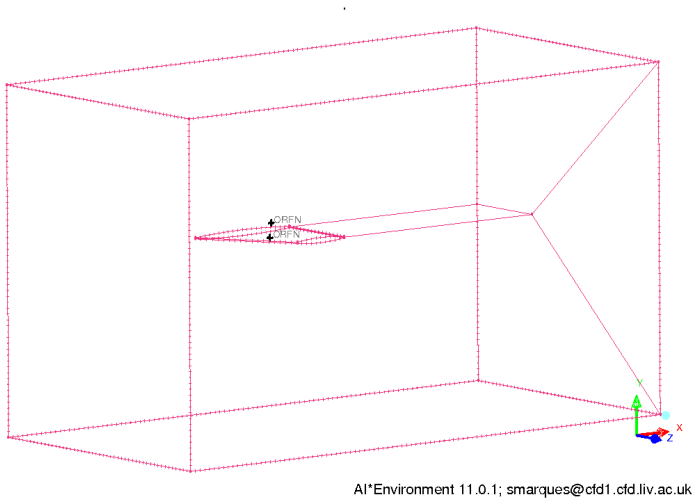
The first task when importing geometry components into ICEM, is to check for any errors from the geometry native file. ICEM provides several options under the *Geometry* > *Repair Geometry* menu

The geometry file already includes most of the surfaces that are necessary, however since the objective is to obtain a grid for the particular FP method described in previous sections, it is require to add the surface that is going to be used for the enforcement of the Kutta condition. To achieve this, the trailing edge curves and surface are split in two, the new points dividing the trailing edge curves are projected into the rear surface to allow setting up new for curves connecting each pair of new points. Figure A.11 illustrates these steps.

In order to achieve a well balanced and as regular as possible grid, at this stage it is possible to define the grid spacing for the curves that make up the geometry. When constructing the final grid ICEM will try to match these spacings wherever possible. By turning on the node spacing it is possible to visualise the several distributions. In Figure A.12 6-10 uniformely distributed points were used in the farfield edges. For the curves created for the *kutta surface*, an hyperbolic function is used to distribute 30 points with a bias towards the fin. The user should ensure the clustering towards




(a) (b)




(c)

Figure A.11: Generating the Kutta Plane

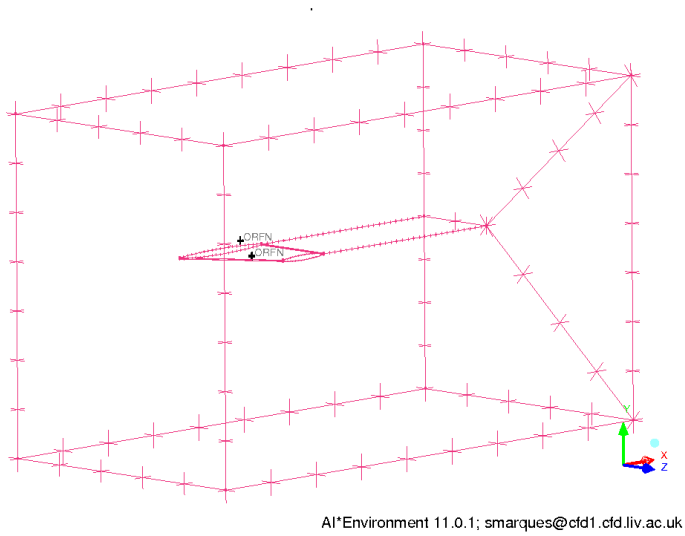



Figure A.12: Curve Spacings

the wing is adequate and smooth. For this case, the surface were already defined by the CAD file. At this stage, it is necessary to discriminate them by creating corresponding parts. In this example, the different surfaces will correspond to the different parts. New parts are created from the *Display Tree* by right-click on *Parts* > *Create Part*. In this case the following parts were created:

- fin upper surface → suction
- fin lower surface → pressure
- leading edge surface → le
- upper trailing edge surface → te-upper
- lower trailing edge surface → te-lower
- fin tip → tip
- kutta surface → kutta
- wall surface → symm
- outer boundaries → farfield

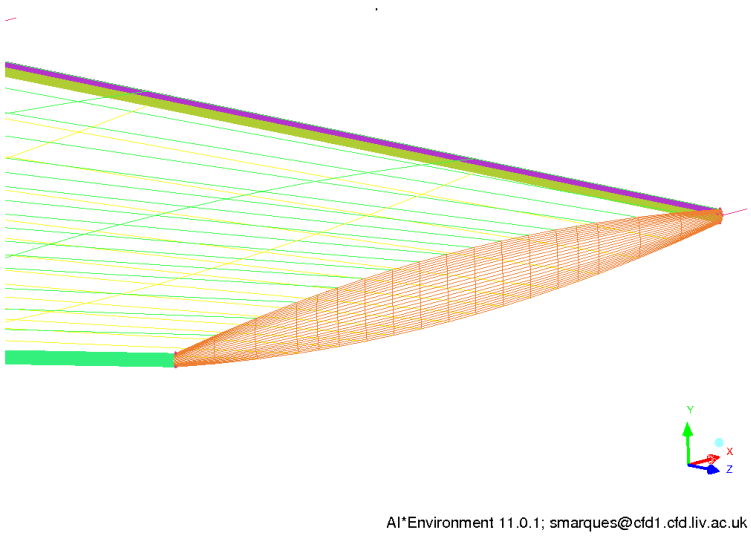



Figure A.13: Parts Creation

Figure A.13 shows a detail view of the fin tip and leading edge. One key difference between 2D and 3D grid in ICEM, is the requirement to define the *material point* of the grid. This correspond to the actual fluid domain of the problem and differentiates this region from any interior spaces created by the geometry. The *material point* must be place within the volume to be meshed, that is achieve by selecting two points, e.g.: fin tip and any corner from the farfield, in the *Create Body* menu, i.e.:

```
Geometry > Create Body > Material Point
```

The actual mesh generation procedure, starts by defining the mesh global sizes. For this case the value of 32 is used for maximum tetrahedral size. Note Figure A.14 and how the actual size is overlayed on the geometry. This parameter is selected from the *Mesh* toolbar:

```
Mesh > Set Global Mesh Size > Global Mesh Size
```

Following the definition of the global sizes, it is possible to define the element sizes for each surface/part, with the exception of the *kutta plane*. The maximum size for each surface much ensure the grid resolution is sufficiently true to the original geometry. Hence, regions of high curvature will require smaller elements than larger flat areas. As for the previous parameter, this parameter is edited from the *Mesh* toolbar:

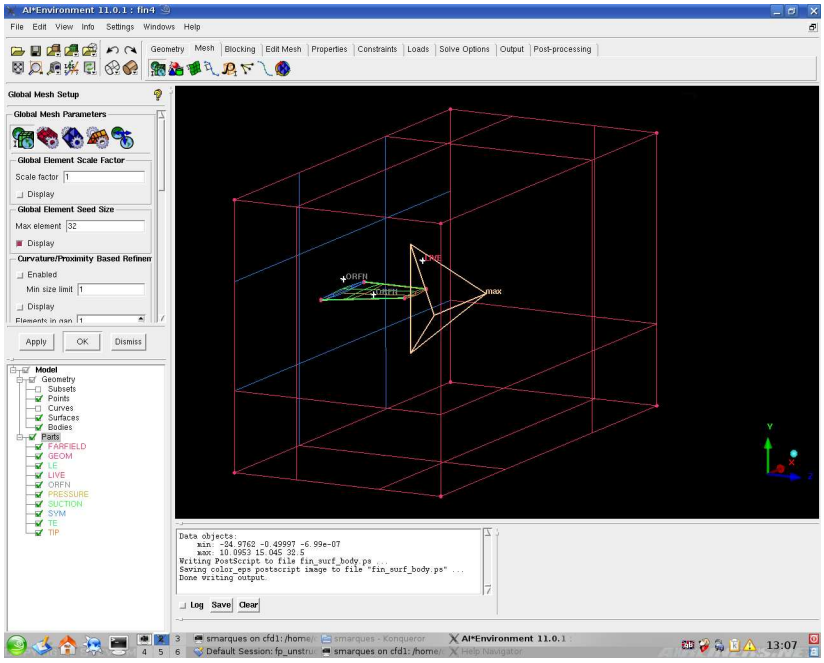



Figure A.14: Setting Elements Global Maximum Size

`Mesh > Surface Mesh Setup > Select surface`

With the surface element sizes assigned, it is possible to compute the mesh. In this case it is necessary to select *Volume Mesh* to mesh the complete domain and not just the surfaces:

`Mesh > Compute Mesh > Volume Mesh > Tetra/Mixed > Robust (Octree)`

Figure illustrates the surface grid at this stage. To make the volume grid consistent with the *kutta plane*, it is necessary to return to the *Surface Mesh* menu. By selecting the *kutta plane* surface and toggle on the *Remesh Selected Surfaces* button **ON**, when the new surface grid, ICEM will ask if the user wishes to make the surface grid consistent with the existing volume grid. If the *kutta plane* surface grid scales are consistent with the volume mesh, this should result in a valid grid. Figure A.15 shows the ICEM output during this procedure, while Figure A.16 shows the final grid:

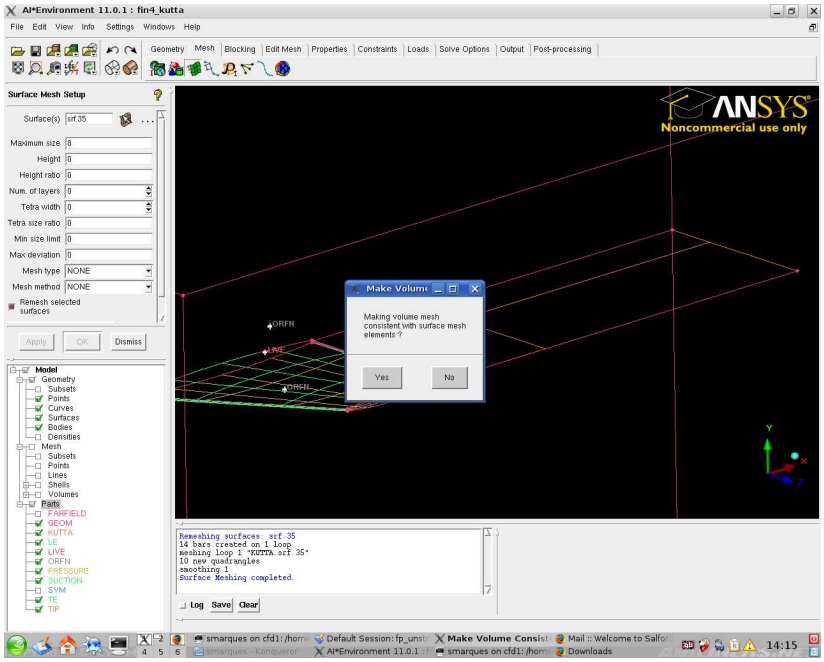



Figure A.15: Meshing the *kutta plane*

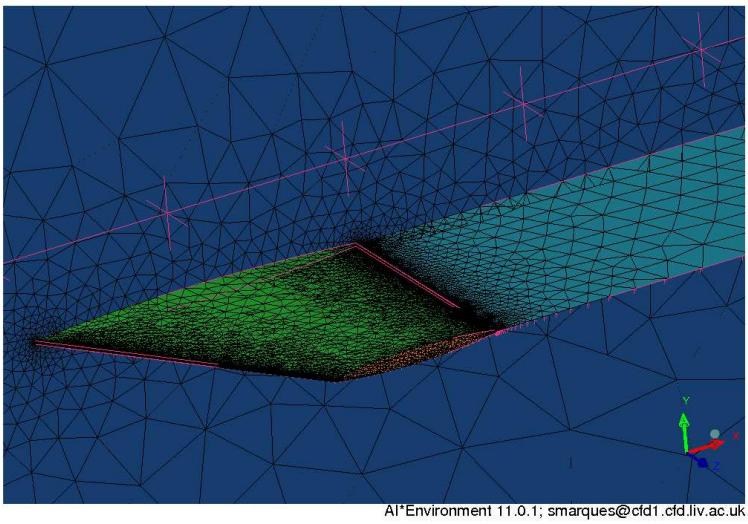



Figure A.16: Final Surface Grid including *kutta plane*

# Appendix B

# Grid Converter

Unstructured grids require to carry the mesh connectivity information explicitly. This can be done in many different ways. ICEM offers several types of output for the connectivity data structure. The FP solver described in the previous chapters, uses a face base structure. It assumes two lists: one containing the grid nodes cartesian coordinates; the second list contains the connectivity information for each face - adjacent cells and forming points. Table B.1 and Figure B.1 describe the face connectivity.

| Face | Left Cell | Right Cell | Point 1 | Point 2 | Point 3 |
|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... |
| $i$ | Cell 1 | Cell 2 | A | B | C |
| ... | ... | ... | ... | ... | ... |

Table B.1: Face data list

Besides interpreting the grid connectivity, the flow solver requires identification of the several boundary conditions: surface faces, farfield and *kutta plane*. ICEM offers several types of output, from general formats such as *STL*, to specific solvers such as *Fluent*. After analysing several output formats, it was decided to use the *FIDAP* solver output structure. This ASCII format, outputs the grid node list, followed by a list of pointers for the nodes of each cell and a list of pointers for the nodes of each surface of each part. This is why it is vital and required for user to create specific parts in ICEM. The grid converter sequence of operations is illustrated in Figure B.2: In

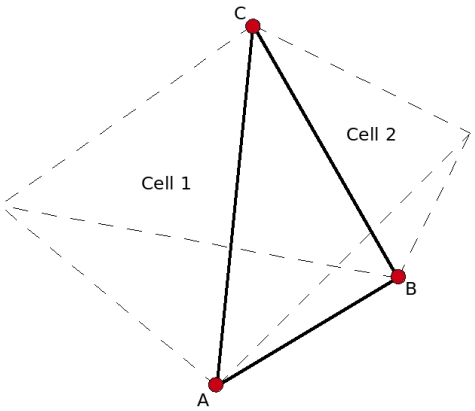



Figure B.1: Face Base Data Structure

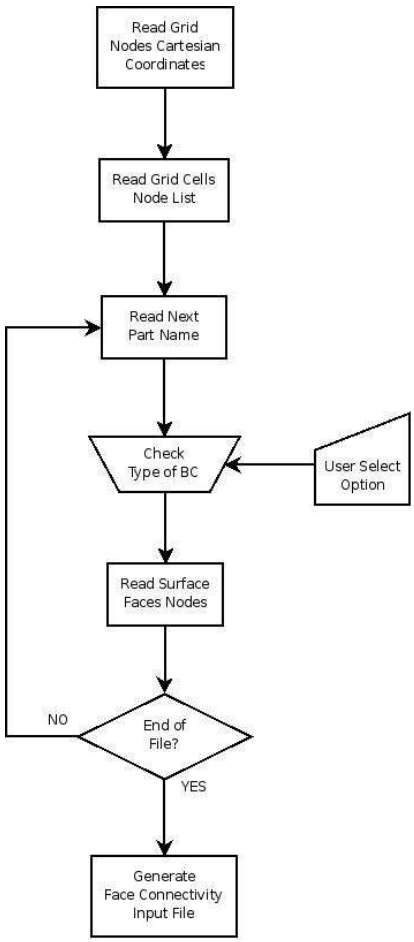



Figure B.2: Grid Converter Fluxo-Gram

task 4 of the fluxogram, the program shows the part name and requires the user to input the correspondent boundary type, at this stage the following conditions are available:

1. Surface
2. Farfield
3. Kutta Plane
4. Symmetry
5. Other

The solver input file is generated at the last step. The grid faces are ordered starting with the surface faces, this is followed by the farfield faces, Kutta Plane, Symmetry and finally all interior faces.

# Bibliography

[1] J. D. Anderson. *Computational Fluid Dynamics - The Basics with Applications*, New York, McGraw-Hill, 1995.

[2] J. D. Anderson, *Modern Compressible Flow*, McGraw Hill, $3^{rd}$ Ed., 2003

[3] J. Bailey and J. L. Steger, Relaxation techniques for three-dimensional transonic flow about wings, *Proc.* $10^th$ *AIAA Aerospace Sciences Meeting*, San Diego, Calif., 1972

[4] W. F. Ballhaus and J. L. Steger, Implicit approximate factorization schemes for the low-frequency transonic equation, *NASA TM X-73082*, 1975.

[5] W. Ballhaus, J. Bailey, Numerical calculation of transonic flow about swept wings, *Proc.* $5^th$ *AIAA Fluid and Plasma Dynamics Conference*, Boston, Mass., 1972

[6] J. Batina, An Efficient Algorithm for Solution of the Unsteady Transonic Small-Disturbance Equation, *NASA TM 89014*, 1986

[7] J. Batina, Advanced Small Perturbation Potential Flow Theory for Unsteady Aerodynamic and Aeroelastic Analyses, *NASA TM 2005-213908*, 2005

[8] J. Batina, Unsteady Transonic Small Disturbance including entropy and vorticity effects, *J. of Aircraft*,vol. 26, pp. 531-8, 1989

[9] J. Batina, D. Seidel, S. Bland, R. Bennet, Unsteady Transonic Small Disturbance including entropy and vorticity effects, *J. of Aircraft*,vol. 26, pp. 21-8, 1989

[10] R. Bennet, J. Edwards, An Overview of Recent Developments in Computational Aeroelasticity, $29^{th}$ *AIAA Fluid Dynamics Conference*, AIAA paper 98-2421, 1998

[11] P. Beran, N. Khot, F. Eastep,R. Snyder, J. Zweber, Numerical Analysis of Store-Induced Limit-Cycle Oscillation, *Journal of Aircraft*, Vol. 41, No. 6, NovemberDecember, 2004

[12] T. J. Chung, *Computational Fluid Dynamics*, Cambridge Press, 2002.

[13] H. Cunningham, J. Herbert, J. Batina, R. Bennett, Modern wing flutter analysis by computational fluid dynamics methods, *Journal of Aircraft*, vol.25 no.10, pp.962-968, 1988

[14] A. Eberle, A Finite Volume Method for Calculating Transonic Potential Flow around Wings from the Pressure Minimum Integral, *NASA TM-75324*, 1978.

[15] F. Eastep, G. Andersen, P. Beran, R. Kolonay, Control Surface Reversal in the Transonic Regime Including Viscous Effects, *Journal of Aircraft*, vol.38, no.4, 2001

[16] B. Eussen, M. Hounjet, J. Meijer, B. Prananta and I W. Tjatra, Perspectives of NLR aeroelastic methods to predict wing/store flutter and dynamic loads of fighter-type aircraft, *NLR-TP-2000-447*, 2000

[17] C. A. Fletcher, *Computational Techniques for Fluid Dynamics*, Berlin, Springer-Verlag, 1988.

[18] W. Habashi, M. Hafez, Finite Element Solutions of Transonic Flow Problems, *AIAA Journal*, vol.(20), pp.1368-1376, 1982

[19] M. Hafez, L. Wellford, C. Merkle, E. Murman, Numerical Computation of Transonic Flows by Finite-Element and Finite-Difference Methods, *NASA CR 3070*, 1978

[20] M. Hafez, E. Murman, J. South, Artificial Compressibility Methods for Numerical Solution of Transonic Full Potential Equation, *AIAA Journal*, vol. 17, 838-44, 1979

[21] C. Hirsh, *Numerical Computation of Internal and External Flows*, Computational Methods for Inviscid and Viscous Flows, John Wiley, 1990

[22] T. Holst, W. Ballhaus, Fast Conservative Schemes for the Full Potential Equation applied to Transonic Flows, *AIAA Journal*, vol.17,145-52, 1979

[23] T. Holst, On Approximate Factorization Schemes for Solving the Full Potential Equation, *NASATM-110435*, 1997

[24] T. Holst, Transonic flow computations using nonlinear potential methods, *Progress in Aerospace Sciences*, vol. 36, 2000

[25] J. Howlett, Efficient Self-Consistent Viscous-Inviscid Solutions for the Unsteady Transonic Flow, *Journal of Aircraft*, vol.24, no.11, 2001

[26] A. Jameson, D. Caughey, A Finite Volume Method for Transonic Potential Flow Calculations, *Proc.* $3^rd$ *AIAA Computational Fluid Dynamics Conference* Albuquerque, N. Mex., 1977

[27] A. Jameson, Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple Grid Method, *Proc.* $4^th$ *AIAA Computational Fluid Dynamics Conference*, Williamsburg, July 1979

[28] S. Janardhan, R. Grandhi, F. Eastep, Brian Sanders, Parametric Studies of Transonic Aeroelastic Effects of an AircraftWing/Tip Store, *Journal of Aircraft*, Vol. 42, No. 1, JanuaryFebruary 2005

[29] F. T. Johnson, S. S. Samant, M. B. Bieterman, R. G. Melvin, D. P. Young, J. E. Bussoletti, C. L. Hilmes, TranAir: A Full-Potential, Solution-Adaptive, Rectangular Grid Code for Predicting Subsoni, Transonic and Supersonic Flows About Arbitrary Configurations, *NASA CR 4348*, 1992

[30] D. Mavriplis, Drag Reduction, *Presentation prepared for VKI*, 2003

[31] J. Moran, *An Introduction to Theoretical and Computational Aerodynamics*, New-York, John Wiley & Sons, 1984.

[32] J. Steger, H. Lomax, Transonic Flow about Two-Dimensional Airfoils by Relaxation Procedures, *AIAA Journal*, v.10, pp49

[33] E. Murman, J. Cole, Calculation of Plane, Steady, Transonic Flows, *AIAA Journal*, Vol.9, Jan. 1971, pp. 114-121.

[34] R. Neel, Advances in Computational Fluid Dynamics: Turbulent Separated Flows and Transonic Potential Flows, PhD Thesis, Virginia Polytechnic Institute and State University, 1997

[35] W. Press, S. A. Teukolsky, W. Vetterling, B. Flannery, *Numerical Recipes: The Art of Scientific Computing*,$3^{rd}$ Edition, Cambridge University Press, 2007

[36] S. Balay, W. Gropp, L. McInnes, F. Smith, *PETSc Users Manual*, ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2003

[37] B. Engquist, S. Osher, Stable and Entropy Satisfying Approximations for Transonic Flow Calculations, *Mathematics of Computation*, vol. 34, No. 149, pp. 45-75, 1980

[38] S. Osher, M. Hafez; W. Whitlow, Entropy Condition Satisfying Approximations for the Full Potential Equation of Transonic Flow, *Mathematics of Computation*, vol. 44, No. 169, pp. 1-29, 1985

[39] D. Schuster, D. Liu, L. Huttsell, Computational Aeroelasticity: Success, Progress, Challenge, *Journal of Aircraft*, vol.40, n. 5, 2003

[40] S. Stahara, Operational Manual for 2D Transonic Code TSFOIL, *NASA CR 3064*, 1978

[41] J. L. Steger and H. Lomax, Transonic Flow About Two-Dimensional Airfoils by Relaxation Procedures, *AIAA Journal*, vol. 10, pp. 49-54, 1972

[42] G. Volpe, A. Jameson, Transonic potential Flow Calculations by two Artificial Density Methods, *AIAA Journal*, vol.(26), pp.425-429, 1988

[43] A. Wissink, A. Lyrintzis, A. Chronopoulos, Efficient Iterative Methods Applied to the Solution of Transonic Flows, *Journal of Computational Physics*, vol.123, 1996