AEROELASTIC COMPUTATIONS IN THE TIME DOMAIN USING UNSTRUCTURED MESHES

by Karen E. Willcox
B.E. Engineering Science, University of Auckland (1994)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1996

Copyright ©Massachusetts Institute of Technology, 1996. All rights reserved.

Signature of Author _____
Department of Aeronautics and Astronautics
January 19, 1996

Certified by _____
Professor Jaime Peraire
Thesis Advisor

Accepted by _____
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

FEB 21 1996

1

LIBRARIES

ARCHIVES

AERCELASTIC COMPUTATIONS IN THE TIME DOMAIN USING
UNSTRUCTURED MESHES

by

Karen E. Willcox

Submitted to the Department of Aeronautics and Astronautics in January 1996 in partial
fulfillment of the requirements for the Degree of Master of Science in Aeronautics and
Astronautics

Abstract

A strategy for solving the time dependent Euler equations on unstructured triangular meshes is proposed. An implicit LU factorisation scheme for unstructured meshes is presented and coupled with a typical section aeroelastic wing model. The fluid and structural models are integrated simultaneously in time at every timestep. An efficient algorithm is constructed by coupling the LU factorisation scheme with a GMRES algorithm. In this case the LU scheme plays the role of a preconditioner. The response of structural sections in different flow regimes is determined and flutter boundaries are computed. In the transonic regime beyond points of linear stability the section is found to exhibit limit cycle behaviour.

Acknowledgements

First and foremost, I would like to thank my advisor Professor Jaime Peraire for his constant support and encouragement throughout the year. Never too busy to help with my problems he was a constant source of ideas.

Thanks must also go to the other students in CASL and SSL. To those who shared the stress of writing a thesis and taking the qualifiers, and especially to Tolu who I constantly plagued with Unix problems. Thanks especially to Graeme, Ray, Mike, Jim, Carmen and Guy who kept me sane in what could have otherwise been a miserable existence.

Of course I must also acknowledge the support of my friends and family back home, who even though 12,000 miles away provide the encouragement to keep me here in the merciless weather of Boston. Life just would not have been the same without those food parcels, even though I do have the world market cornered on vegemite for the next few years.

Contents

1	Introduction	12
2	Finite Volume Discretisation of the Euler Equations	16
2.1	Governing Equations	17
2.2	Unstructured Mesh Algorithm	19
2.2.1	Finite Volume Formulation	19
2.2.2	Geometric Conservation Law	24
2.2.3	Artificial Dissipation	25
3	Time Integration	30
3.1	LU Implicit Scheme	31
3.2	LU-Gauss-Seidel Relaxation	38
3.3	GMRES Algorithm	39

4	Fluid-Aeroelastic Coupling	42
4.1	Structural Aeroelastic Model	43
4.2	Dynamic Mesh Algorithm	45
5	Results	47
5.1	Steady State Flow	47
5.2	Unsteady Forced Oscillation	51
5.3	Fully Coupled Fluid-Structural Problem	54
6	Conclusions	60

List of Figures

2.1	Control volume V_j associated to a generic node j of an unstructured grid (a)	
	Interior node, (b) Boundary node.	20
3.1	Splitting of sides according to chosen sweep direction.	35
3.2	Structure of vector used to order sides and nodes.	37
4.1	Typical section wing model geometry	45
5.1	Pressure contours for steady inviscid transonic flow.	50
5.2	Residuals for steady inviscid transonic flow.	50
5.3	Lift coefficient and moment coefficient for unsteady inviscid transonic flow.	53
5.4	Residuals for unsteady inviscid transonic flow.	53
5.5	Damped Response. $M = 0.6$, $V_f = 1.1$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega l}{\omega_\alpha} = 1$	57
5.6	Unstable Response. $M = 0.6$, $V_f = 1.15$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega l}{\omega_\alpha} = 1$	57
5.7	Limit Cycle Response. $M = 0.82$, $V_f = 0.71$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega l}{\omega_\alpha} = 1$	58

5.8	Flutter Boundary. $V_f = U_\infty / b\omega_\alpha \mu^{0.5}$	58
5.9	Flutter mode phase difference. $\mu = 60$	59
5.10	Second mode response. $M_\infty = 0.875$, $V_f = 2.6$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega_f}{\omega_\alpha} = 1$.	59

Nomenclature

a	Non-dimensional location of elastic axis behind midchord
$\mathcal{A}, \mathcal{B}$	Jacobian matrices
$A, \bar{A}$	Edge matrices
b	Airfoil semichord
B_j	Set of boundary edges meeting at node j
c	Speed of sound
c	Airfoil chord
C	Constant depending on CFL number
C_l	Lift coefficient
C_m	Moment coefficient
d	Artificial dissipation
e	Total specific energy
E_j	Set of all edges meeting at node j
F	Load vector
F, G	Inviscid flux vectors
h	Plunging displacement of elastic axis
$H, \bar{H}$	Weighted fluxes
$\mathcal{I}$	Identity matrix

I_α	Section moment of inertia about elastic axis
k	Edge stiffness parameter
k_c	Reduced frequency
K_h	Plunging spring constant
K_α	Pitching spring constant
$[K]$	Stiffness matrix
l	Edge length
L	Limiter function
$\mathcal{L}, \mathcal{U}, \mathcal{D}$	Upper, lower and diagonal matrices
$[M]$	Mass matrix
M	Freestream Mach number
n_x, n_y	Unit normal vector cartesian components
p	Pressure
p_k	Search direction
q	Displacement vector
$R, \tilde{R}$	Residual
$\mathbf{s}$	Sweep direction
S	Characteristic variables
S	Safety factor

S_α	Static unbalance
t	Physical time
u, v	Cartesian velocity components
$\tilde{u}, \tilde{v}$	Contravariant velocity components
$\bar{U}$	State vector
U_∞	Freestream velocity
V	Volume
V_f	Flutter speed index
$W_x, W_y, \bar{W}_x, \bar{W}_y$	Weights
x, y	Cartesian coordinates
x_t, y_t	Cartesian components of mesh velocity
x_{pv}	Pivotal point of flutter motion
X	Matrix containing right eigenvectors of A
α	Angle of attack
γ	Ratio of specific heats
Γ	Boundary of domain
δ_x, δ_y	Grid displacement components
Λ	Matrix containing eigenvalues of A
λ	Damping factor for GMRES

μ	Airfoil mass ratio
ρ	Density
τ	Pseudo-time
ω_f	Frequency of forced oscillations
ω_h, ω_α	Uncoupled natural frequencies in plunge and pitch
Ω	Computational domain

Chapter 1

Introduction

In recent years, Computational Fluid Dynamics has reached a considerable level of maturity and has become an essential tool for aerodynamic analysis. With the latest developments in automatic unstructured mesh generation and efficient multigrid flow solution algorithms, it is now possible to routinely obtain steady Euler solutions about complete aircraft configurations [1] [2] and quickly assess the effect of design modifications.

Unsteady flows are of importance for a variety of practical problems such as flutter, rapid manoeuvres or control deflections. In particular, aeroelastic modelling becomes essential in the transonic regime where nonlinear effects are important. A fully coupled solution of the nonlinear problem must be considered to predict phenomena such as transonic dip [3]. The movements of shocks on the surface of an airfoil interact dynamically with the motion of

the airfoil, a factor closely related to the process of limit cycle formation [4]. Obtaining an accurate time response of these shock motions is crucial in obtaining design criteria in the transonic regime, since small disturbance codes predict only the onset of growing oscillations and cannot model limit cycle behaviour.

The computation of time dependent flows requires significantly more resources than those required for steady state problems. If explicit timestep methods are used, the allowable timestep is determined by stability considerations and is a function of the smallest grid size used to discretise the spatial domain. In some cases this timestep can be several orders of magnitude smaller than that required to obtain accurate solutions. When using unstructured meshes it is in general very difficult to carefully control the size and shape of the elements generated, especially in three dimensions. The appearance of distorted elements, even if they occur in very small numbers, may severely limit the timestep size.

The alternative is to use an implicit formulation so that the timestep can be based on accuracy considerations only. In this case, it is necessary to devise a procedure for solving the resulting sparse equation system at each timestep. A possible approach is to discretise the equations in time using backward differencing formulae and extend the multigrid techniques developed for the solution of steady flows to solve the resulting non-linear system of equations. This approach was proposed in [5] and has been extended to the solution of the two dimensional time dependent Navier-Stokes equations [6]. This method can also be used in the context of unstructured meshes and preliminary computations in

three dimensions were reported in [7]. For unstructured meshes however, the automatic generation of coarse grids may prove difficult, especially if complex geometries are involved. This may present a limitation, since the ability to create very coarse grids is critical to the performance of the multigrid approach. An alternative strategy is to use a recursive algorithm for automatically generating coarser grids by agglomeration of fine grid cells [8] [9].

The objective of this work was to investigate a different approach which utilises a single grid and which is effective for both steady and unsteady computations. The idea presented is an extension of the LU factorisation schemes proposed in [10] to unstructured meshes. The algorithm presented is non-vectorisable but can be parallelised efficiently. A description of the finite volume discretisation of the unsteady Euler equations is presented in Chapter 2. Chapter 3 contains the details of the time integration schemes developed for these equations.

The typical wing section model is a two-dimensional analog of a three-dimensional wing [11] for which the equations of motion can be solved to find the motion of the airfoil. These equations must be coupled to the fluid solver in such a way that the two sets of equations are advanced in time simultaneously. Additionally, the computational domain must be reshaped at every timestep in order to account for the deformation of the airfoil. This can be achieved by modelling each edge in the grid as a spring [12]. The structural model and dynamic mesh algorithm used to couple the fluid and structural equations are discussed in Chapter 4.

Finally, a number of test cases are considered to determine the relative merits of the algorithms developed. A steady flow case is first presented which enables the performance of the flow solvers to be determined. Two unsteady problems are discussed in which the structural coupling and dynamic mesh algorithm are also included in the solution procedure. The results for these flow cases are presented in Chapter 5, with conclusions and recommendations for future work in Chapter 6.

Chapter 2

Finite Volume Discretisation of the Euler Equations

In this chapter, the governing equations for unsteady inviscid flow in a moving domain are presented, along with their two-dimensional finite volume discretised form. The equations are discretised on an unstructured triangular grid which can deform in time. An edge-based data structure is introduced which allows the equations to be written in compact form as a weighted summation of flux differences. Artificial dissipation must be added to the scheme in such a manner as to ensure that the algorithm remains stable and can handle shock waves and discontinuities but is accurate in regions of smooth flow. In order to avoid errors induced by the moving mesh, the scheme must also satisfy a geometric con-

servation law. This is necessary to ensure that a uniform flow remains an exact solution of the numerical algorithm.

2.1 Governing Equations

Consider a two-dimensional control volume $\Omega(t)$, with boundary $\Gamma(t)$ both of which are a function of time. The Euler equations governing the unsteady two-dimensional flow of an inviscid compressible fluid can be written in integral form as

$$\frac{\partial}{\partial t} \int_{\Omega} U dx dy + \oint_{\Gamma} (F n_x + G n_y) d\Gamma = 0 \quad (2.1)$$

where n_x and n_y are unit vectors pointing out of Ω , U is the unknown vector of conserved variables and F and G are the inviscid flux vectors given by

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix} \quad F = \begin{pmatrix} \rho(u - x_t) \\ p + \rho u(u - x_t) \\ \rho v(u - x_t) \\ \rho u + e(u - x_t) \end{pmatrix} \quad G = \begin{pmatrix} \rho(v - y_t) \\ \rho u(v - y_t) \\ p + \rho v(v - y_t) \\ \rho v + e(v - y_t) \end{pmatrix} \quad (2.2)$$

Here ρ, u, v, p and e denote density, cartesian velocity components, pressure and total energy respectively. x_t and y_t are the speeds in the x and y directions with which the boundary

$\Gamma(t)$ moves. Also, for an ideal gas the equation of state becomes

$$e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2) \quad (2.3)$$

These equations will be discretised on an unstructured triangular grid covering the computational domain and approximations to the unknown vector U will be sought at the vertices of that grid. At the surface of the airfoil, the flow tangency boundary condition must be satisfied. This is done by setting the normal component of flow velocity at the surface to be equal to the normal component of the airfoil's motion. In the farfield Riemann invariants are used to determine the flow at the outer boundary of the domain. We define the four Riemann invariants as follows

$$\begin{aligned} R_1 &= u_n + \frac{2c}{\gamma - 1} \\ R_2 &= u_n - \frac{2c}{\gamma - 1} \\ R_3 &= u_t \\ R_4 &= \frac{p}{\rho^\gamma} \end{aligned} \quad (2.4)$$

Here, u_n and u_t are the normal and tangential components of velocity, and c is the speed of sound. The four quantities given in (2.4) are propagated with the velocities $u_n + c$, $u_n - c$, u_n

and u_n respectively. If the propagation velocity is into the domain, the Riemann invariant is prescribed using farfield conditions. For propagation out of the domain, the information is extrapolated from the calculated flow variables.

2.2 Unstructured Mesh Algorithm

2.2.1 Finite Volume Formulation

By considering equation (2.1) for each volume V_j consisting of all the triangles having vertex j as shown in figure 2.1, the following set of ordinary differential equations may be written

$$\frac{d}{dt}(V_j U_j) + \int_{\Gamma_j} (F n_x + G n_y) d\Gamma = 0 \quad (2.5)$$

where Γ_j is the boundary of V_j and U_j represents the average value of U over volume V_j .

The above equation may be approximated as

$$\frac{d}{dt}(V_j U_j) + \sum_{i' \in \Gamma_j} \left[\frac{1}{2}(F_i + F_{i+1}) n_x^{i'} + \frac{1}{2}(G_i + G_{i+1}) n_y^{i'} \right] l^{i'} = 0 \quad (2.6)$$

where the summation extends over all the edges i' which comprise the boundary Γ_j . In the above expression $n_x^{i'}$, $n_y^{i'}$ and $l^{i'}$ are the cartesian components of the outward unit normal and length of a generic edge i' connecting vertices i and $i + 1$.

An edge based data structure can be used to express (2.6) in terms of a weighted

Figure 2.1: Control volume V_j associated to a generic node j of an unstructured grid (a) Interior node, (b) Boundary node.

summation of flux differences. Grouping together all the contributions from node i to node j we can define weights as

$$W_x^{ij} = \frac{1}{2}(n_x^{i'-1}l^{i'-1} + n_x^{i'}l^{i'}) , \quad W_y^{ij} = \frac{1}{2}(n_y^{i'-1}l^{i'-1} + n_y^{i'}l^{i'}) \quad (2.7)$$

when ij is an interior edge and

$$W_x^{ij} = \frac{1}{4}n_x^{ij}l^{ij} + \frac{1}{2}n_x^{i'}l^{i'} , \quad W_y^{ij} = \frac{1}{4}n_y^{ij}l^{ij} + \frac{1}{2}n_y^{i'}l^{i'}$$

$$\overline{W}_x^{ij} = \frac{1}{4}n_x^{ij}l^{ij} , \quad \overline{W}_y^{ij} = \frac{1}{4}n_y^{ij}l^{ij} \quad (2.8)$$

when ij is a boundary edge. Noting that

$$\sum_{ijeE_j} W_x^{ij} + 3 \sum_{ijeB_j} \bar{W}_x^{ij} = 0 \quad , \quad \sum_{ijeE_j} W_y^{ij} + 3 \sum_{ijeB_j} \bar{W}_y^{ij} = 0 \quad (2.9)$$

where E_j denotes the set of edges meeting at point j and B_j the subset of these edges which also lie on the boundary on the domain Ω , the following equation can be written

$$\begin{aligned} \frac{d}{dt} (V_j U_j) + \sum_{ijeE_j} \left[W_x^{ij} (F_i - F_j) + W_y^{ij} (G_i - G_j) \right] \\ + \sum_{ijeB_j} \left[\bar{W}_x^{ij} (F_i - F_j) + \bar{W}_y^{ij} (G_i - G_j) \right] = 0 \end{aligned} \quad (2.10)$$

Note that summation is now over the edges meeting at node j rather than over those edges forming the boundary of the volume V_j . It is easily verified that the weights defined in (2.7) and (2.8) satisfy the following symmetry relations

$$\begin{aligned} W_x^{ij} = -W_x^{ji} \quad , \quad W_y^{ij} = -W_y^{ji} \\ \bar{W}_x^{ij} = \bar{W}_x^{ji} \quad , \quad \bar{W}_y^{ij} = \bar{W}_y^{ji} \end{aligned} \quad (2.11)$$

Thus for interior nodes, the contribution from node i to node j has the same magnitude as that from j to i but is opposite in sign. For boundary nodes, the contributions are identical. The scheme can be efficiently implemented by considering each edge in the domain in turn,

calculating the fluxes associated to that edge, and sending the contribution to each node with the appropriate sign.

The next step is to incorporate the fluxes F and G into a single term for each edge. We define the fluxes associated to an edge ij

$$\begin{aligned} H_i^{ij} &= W_x^{ij} F_i + W_y^{ij} G_i = -H_i^{ji} \\ \bar{H}_i^{ij} &= \bar{W}_x^{ij} F_i + \bar{W}_y^{ij} G_i = \bar{H}_i^{ji} \end{aligned} \quad (2.12)$$

The semi-discrete scheme for a generic point j may now be written as

$$\frac{d}{dt}(V_j U_j) + \sum_{ij \in E_j} (H_i^{ij} - H_j^{ij}) + \sum_{ij \in B_j} (\bar{H}_i^{ij} - \bar{H}_j^{ij}) = 0 \quad (2.13)$$

The spatial integral in equation (2.5) has been replaced by a summation of weighted flux differences. It is more convenient to consider the flux differences as the product of a matrix times the state vector

$$H_i - H_j = A_{ij}(U_i - U_j) \quad (2.14)$$

The matrix A is given by

$$A_{ij} = W_x^{ij} A(U_i, U_j) + W_y^{ij} B(U_i, U_j)$$

$$\bar{A}_{ij} = \bar{W}_x^{ij} \mathcal{A}(U_i, U_j) + \bar{W}_y^{ij} \mathcal{B}(U_i, U_j) \quad (2.15)$$

where $\mathcal{A}$ and $\mathcal{B}$ are the Jacobian matrices $\frac{\partial F}{\partial U}$ and $\frac{\partial G}{\partial U}$. Equation (2.13) is now written as

$$\frac{d}{dt} (V_j U_j) + \sum_{ij \in E_j} A_{ij} (U_i - U_j) + \sum_{ij \in B_j} \bar{A}_{ij} (U_i - U_j) = 0 \quad (2.16)$$

The A_{ij} and $\bar{A}_{ij}$ are determined by differentiating the flux vectors F and G with respect to the state vector U . These matrices are evaluated at the Roe average between U_i and U_j [13]. At this point, the flow variables are calculated as functions of values at nodes i and j . The density at the Roe average point is given by

$$\rho = \sqrt{\rho_i \rho_j} \quad (2.17)$$

while the velocity u is calculated as

$$u = \frac{\sqrt{\rho_i} u_i + \sqrt{\rho_j} u_j}{\sqrt{\rho_i} + \sqrt{\rho_j}} \quad (2.18)$$

The velocity v and enthalpy h are evaluated in an analogous manner to u .

2.2.2 Geometric Conservation Law

The scheme must also satisfy a geometric conservation law to avoid errors induced by the moving mesh and ensure that a uniform flow remains an exact solution of the numerical algorithm. We write the semi-discrete scheme (2.13) as

$$\frac{d}{dt} (V_j U_j) + \mathcal{F}_j(U, \mathbf{x}, \mathbf{x}_t) = 0 \quad (2.19)$$

where $\mathcal{F}_j$ represents all the flux summations applying to node j and is a function of the flow solution U , the mesh position $\mathbf{x}$ and the mesh velocity $\mathbf{x}_t$. Integrating from time t^n to time t^{n+1} we obtain

$$V_j^{n+1} U_j^{n+1} - V_j^n U_j^n + \int_{t^n}^{t^{n+1}} \mathcal{F}_j dt = 0 \quad (2.20)$$

A conservative finite volume scheme must have the property that a uniform flow is an exact solution of the scheme. If we consider the uniform solution $U^n = U^{n+1} = U$, (2.20) reduces to

$$V_j^{n+1} - V_j^n = \int_{t^n}^{t^{n+1}} \int_{\Gamma_j} \mathbf{x}_t d\Gamma dt \quad (2.21)$$

where spatial integration is now over Γ_j , the boundary of the volume V_j . Equation (2.21) defines the geometric conservation law which states that the change in area of each control volume over a given time interval must be equal to the area swept by the cell boundary during that interval. The left hand side of this equation can be computed exactly from the mesh

updating scheme, so for the flow solver to satisfy the GCL, the integral $\int_{t^n}^{t^{n+1}} \int_{\Gamma_j} \mathbf{x}_t d\Gamma dt$ must also be exactly computed. For a two dimensional scheme, it can be shown [14] that the GCL will be satisfied if the fluxes are evaluated at the midpoint of the time interval where the positions and velocities of the grid are given by

$$\mathbf{x}^{n+\frac{1}{2}} = \frac{\mathbf{x}^n + \mathbf{x}^{n+1}}{2} \quad \mathbf{x}_t^{n+\frac{1}{2}} = \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} \quad (2.22)$$

The implication of this result is that at each timestep, the weights must be reevaluated according to geometry evaluated at time level $n + \frac{1}{2}$.

2.2.3 Artificial Dissipation

In order to maintain a stable algorithm which can handle shock waves and discontinuities, the above finite volume scheme requires the addition of some numerical dissipation. This can be achieved by the addition of a diffusive term to the edge fluxes. For edge ij the diffusive flux is given by

$$d^{ij} = |A_{ij}| (U_i - U_j) \quad (2.23)$$

where

$$|A| = X|\Lambda|X^{-1} \quad (2.24)$$

X is the matrix whose columns are the right eigenvectors of A and $|\Lambda|$ is a diagonal matrix whose entries are the absolute values of the corresponding eigenvalues. For the two-dimensional inviscid fluxes considered here, the eigenvalues are

$$u_n - x_{t_n} \quad u_n + c - x_{t_n} \quad u_n - c - x_{t_n}$$

where u_n is the velocity normal to the edge and c is the speed of sound. x_{t_n} is the normal component of the local mesh velocity and is given by

$$x_{t_n} = x_t n_x + y_t n_y \quad (2.25)$$

It is noted that the A matrices here are simply related to the Roe matrices which are more commonly used. They have the same eigenvectors, while the relationship between eigenvalues is

$$\Lambda = \Lambda_{Roe} - x_{t_n} \mathcal{I} \quad (2.26)$$

where in this case $\mathcal{I}$ is the 3x3 identity matrix. With the addition of artificial dissipation, the scheme (2.16) thus becomes

$$\frac{d}{dt} (V_j U_j) + \sum_{ij \in E_j} [A_{ij} - |A_{ij}|] (U_i - U_j) + \sum_{ij \in B_j} [\bar{A}_{ij} - |\bar{A}_{ij}|] (U_i - U_j) = 0 \quad (2.27)$$

This scheme is stable and can handle shock waves and discontinuities but introduces excessive dissipation for practical applications which destroys accuracy in smooth regions of flow. A more accurate scheme can be obtained by introducing anti-diffusive terms in a controlled manner. This can be done by writing the diffusive fluxes as

$$d^{ij} = |A_{ij}| \left[\Delta U - \frac{1}{2} [L(\Delta U^+, \Delta U) + L(\Delta U, \Delta U^-)] \right] \quad (2.28)$$

where $\Delta U = U_i - U_j$ and ΔU^+ and ΔU^- are upstream and downstream differences associated to edge ij . These differences are given by

$$\Delta U^+ = \left[\left(\frac{\partial U}{\partial x} \right)_j (x_i - x_j) + \left(\frac{\partial U}{\partial y} \right)_j (y_i - y_j) \right] \quad (2.29)$$

$$\Delta U^- = \left[\left(\frac{\partial U}{\partial x} \right)_i (x_i - x_j) + \left(\frac{\partial U}{\partial y} \right)_i (y_i - y_j) \right] \quad (2.30)$$

The function $L(u, v)$ in the diffusive flux expression (2.28) represents a limited average of u and v . Several options are possible but the expression adopted for the results reported here is a differentiable function given in [15]

$$L(u, v) = \frac{1}{2}(u + v) \frac{2uv + \epsilon}{u^2 + v^2 + \epsilon} \quad (2.31)$$

where ϵ is a small number proportional to the cube of the length of the edge ij . This function

acts as a switch which controls the amount of anti-diffusive fluxes added depending on the smoothness of the solution at that point. Clearly the dissipative flux in (2.28) reduces to that in (2.23) when the limiter function is zero. It is noted that if U is locally a linear function of x and y and the computed gradients are exact for linear data then the dissipative terms become zero and the scheme reduces to that of expression (2.16).

In practice best results are obtained if the limiting process is performed on the characteristic variables $\Delta S = X^{-1} \Delta U$. Then (2.28) is replaced by

$$d^{ij} = X |\Lambda| \left[\Delta S - \frac{1}{2} [L(\Delta S^+, \Delta S) + L(\Delta S^-, \Delta S)] \right] \quad (2.32)$$

with $\Delta S^+ = X^{-1} \Delta U^+$ and $\Delta S^- = X^{-1} \Delta U^-$.

The gradients at the nodes can be calculated by employing an analogous algorithm to that derived for the Euler fluxes. Using Gauss' theorem the governing equations (2.1) can be written in differential form

$$\int_{\Omega} \left(\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} \right) d\Omega = 0 \quad (2.33)$$

From equation (2.10) we can see that in the finite volume formulation developed, the spatial

differentials have been approximated as a weighted summation of differences

$$\int_{\Omega} \frac{\partial F}{\partial \mathbf{x}} d\Omega = \sum_{ij \in E_j} [W_{\mathbf{x}}^{ij}(F_i - F_j)] + \sum_{ij \in B_j} [\bar{W}_{\mathbf{x}}^{ij}(F_i - F_j)] \quad (2.34)$$

$$\int_{\Omega} \frac{\partial G}{\partial \mathbf{y}} d\Omega = \sum_{ij \in E_j} [W_{\mathbf{y}}^{ij}(G_i - G_j)] + \sum_{ij \in B_j} [\bar{W}_{\mathbf{y}}^{ij}(G_i - G_j)] \quad (2.35)$$

By letting $\left(\frac{\partial U}{\partial \mathbf{x}}\right)_j$ represent the average value of $\frac{\partial U}{\partial \mathbf{x}}$ over volume V_j , the following analogous expression can be obtained for the derivatives of U

$$V_j \left(\frac{\partial U}{\partial \mathbf{x}}\right)_j = \sum_{ij \in E_j} W_{\mathbf{x}}^{ij}(U_i - U_j) + \sum_{ij \in B_j} \bar{W}_{\mathbf{x}}^{ij}(U_i - U_j) \quad (2.36)$$

Similarly for $\frac{\partial U}{\partial \mathbf{y}}$

$$V_j \left(\frac{\partial U}{\partial \mathbf{y}}\right)_j = \sum_{ij \in E_j} W_{\mathbf{y}}^{ij}(U_i - U_j) + \sum_{ij \in B_j} \bar{W}_{\mathbf{y}}^{ij}(U_i - U_j) \quad (2.37)$$

Chapter 3

Time Integration

In this chapter, the time discretisation schemes developed for solution of the finite volume formulation of Chapter 2 are described. It is desirable to obtain an implicit method so that the size of the timestep can be based on accuracy considerations only. However, a fully implicit scheme would require the inversion of a large global matrix at each timestep which is computationally infeasible. To avoid this, we seek an approximate factorisation of the implicit scheme which allows the equations to be advanced in time without the inversion of a global matrix. Two schemes are described in this chapter. The first is an approximate LU factorisation which is implemented by sweeping back and forth over the computational domain. The second is an LU-Gauss-Seidel relaxation scheme and is equivalent to a modified form of the LU implicit scheme. These algorithms can be improved through use of the

generalised minimal residual algorithm (GMRES) which is described in the final part of the chapter.

3.1 LU Implicit Scheme

The scheme of equation (2.16) can be discretised in time using a backward Euler formula to give an unconditionally stable implicit algorithm. For second order accuracy in time, the fully discrete system is

$$\frac{3(V_j U_j)^{n+1} - 4(V_j U_j)^n + (V_j U_j)^{n-1}}{2\Delta t} + \tilde{R}_j^{n+1} = 0 \quad (3.1)$$

where the residual $\tilde{R}$ is given by

$$\tilde{R}_j^n = \sum_{ij \in E_j} [A_{ij}(U_i - U_j) - d^{ij}]^n + \sum_{ij \in B_j} [\bar{A}_{ij}(U_i - U_j) - \bar{d}^{ij}]^n \quad (3.2)$$

To solve (3.1) the following system is solved iteratively until a steady state solution is obtained

$$V_j \frac{dU_j^*}{dt} + R(U_j^*) = 0 \quad (3.3)$$

where the modified residual R is given by

$$R_j^p = \tilde{R}_j^p + \frac{3(V_j U_j)^p - 4(V_j U_j)^n + (V_j U_j)^{n-1}}{2\Delta t} \quad (3.4)$$

Equation (3.3) is discretised in time using a backward Euler formula to give

$$V_j^p \Delta U_j + \Delta \tau R_j^{p+1} = 0 \quad (3.5)$$

where $\Delta U_j = U_j^{p+1} - U_j^p$. τ now represents pseudo-time whereas the residual contains Δt , the physical timestep. By driving (3.5) to steady state, we are in fact driving the modified residual R to zero which in turns solves the original system (3.1). The steady state solution is most efficiently attained by using local timestepping. The timestep is chosen independently for each node according to

$$\Delta \tau_j = \frac{2SV_j}{\sum_{ij \in E_j} (|\bar{u}W_x^{ij} + \bar{v}W_y^{ij}| + c \ l^{ij}) + \sum_{ij \in B_j} (3|\bar{u}\bar{W}_x^{ij} + \bar{v}\bar{W}_y^{ij}| + c \ \bar{l}^{ij})} \quad (3.6)$$

where S is a safety factor which for an unconditionally stable scheme can be taken to be as large as desired, c is the speed of sound and l^{ij} is a modulus given by

$$l^{ij} = \sqrt{(W_x^{ij})^2 + (W_y^{ij})^2} \quad (3.7)$$

The contravariant velocities $\bar{u}$ and $\bar{v}$ include the velocity of the moving mesh and are related to the local flow velocity by

$$\bar{u} = u - x_t \quad \bar{v} = v - y_t$$

The nodal velocities and volume are evaluated at time level p .

Using a Taylor series expansion to linearise about time level n , R^{p+1} can be written as

$$R_j^{p+1} = R_j^n + \left. \frac{\partial R}{\partial U_j} \right|^n (U^{p+1} - U^n) \quad (3.8)$$

where higher order terms have been dropped. A diagonally dominant linearised version of the scheme (3.5) can thus be written as

$$\Delta U_j + \frac{\Delta \tau_j}{V_j^p} \sum_{ij \in E_j} \tilde{A}_{ij} (\Delta U_i - \Delta U_j) + \frac{3\Delta \tau_j}{2\Delta t} \Delta U_j = -\frac{\Delta \tau_j}{V_j^p} R_j^p \quad (3.9)$$

where

$$\tilde{A} = A_{ij}^p - |A_{ij}^p|$$

when ij is an interior edge and

$$\tilde{A} = A_{ij}^p - |A_{ij}^p| + \overline{A}_{ij}^p - |\overline{A}_{ij}^p|$$

when ij is a boundary edge.

Equation (3.9) gives an implicit unconditionally stable scheme, which when considered for each node in the grid results in a sparse equation system. Direct solution of this system would involve the inversion of a large global matrix. To avoid this costly computation, an

approximate factorisation can be derived. Consider a sweep direction defined by a vector (s_x, s_y) , and renumber the nodes in the grid according to their s coordinate defined as

$$s_j = x_j s_x + y_j s_y \quad (3.10)$$

Further, the edges meeting at node j are subdivided into two sets E_j^+ and E_j^- according to

$$\begin{aligned} ij \in E_j^+ & \quad \text{if } (x_i - x_j)s_x + (y_i - y_j)s_y \geq 0 \\ ij \in E_j^- & \quad \text{if } (x_i - x_j)s_x + (y_i - y_j)s_y < 0 \end{aligned}$$

as shown in figure 3.1. Thus, (3.9) can be rewritten

$$\left[\mathcal{I} + C_j \frac{\Delta\tau}{V_j^p} \sum_{ij \in E_j^+} \bar{A}_{ij} \delta_{ij} + C_j \frac{\Delta\tau}{V_j^p} \sum_{ij \in E_j^-} \bar{A}_{ij} \delta_{ij} \right] \Delta U_j = -C_j \frac{\Delta\tau}{V_j^p} R_j^p \quad (3.11)$$

where $\mathcal{I}$ is the 4x4 identity matrix and δ_{ij} is a spatial operator which acts on U_j in the following manner

$$\delta_{ij} U_j = U_i - U_j \quad (3.12)$$

C_j is a constant for each node which depends on the CFL number S used in (3.6) to choose the timestep and is given by

$$C_j = \frac{1}{1 + \frac{3\Delta\tau_l}{2\Delta t}} \quad (3.13)$$

An approximate LU factorisation analogous to that proposed in [10] for structured

Figure 3.1: Splitting of sides according to chosen sweep direction.

meshes is obtained by replacing (3.11) by

$$\left[\mathcal{I} + C_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} \tilde{A}_{ij} \delta_{ij} \right] \left[\mathcal{I} + C_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} \tilde{A}_{ij} \delta_{ij} \right] \Delta U_j = -C_j \frac{\Delta \tau}{V_j^p} R_j^p \quad (3.14)$$

It is easily verified that when the above equation is applied to every grid node, the first factor in the left hand side leads to a block lower triangular matrix, whereas the second factor leads to a block upper triangular matrix. Thus, (3.14) results in an equation system of the form

$$[\mathcal{L} + \mathcal{D}^+] [\mathcal{U} + \mathcal{D}^-] \Delta U = R \quad (3.15)$$

where $\mathcal{L}$ and $\mathcal{U}$ are strictly block lower and upper triangular matrices, the $\mathcal{D}^+$ and $\mathcal{D}^-$ matrices contain the block diagonal terms, and ΔU and R are global vectors containing the increments of nodal unknowns and the right hand side terms of equation (3.14) respectively. The block entries of the $\mathcal{L}$ and $\mathcal{U}$ matrices can be obtained by extracting the terms multiplying ΔU_i in equation (3.14). For the lower triangular matrix the entry $\mathcal{L}_{ji}$ where $j > i$ is given by

$$\mathcal{L}_{ji} = \left[c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} \bar{A}_{ij} \right] \quad (3.16)$$

while for $\mathcal{U}_{ji}$

$$\mathcal{U}_{ji} = \left[c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} \bar{A}_{ij} \right] \quad (3.17)$$

where now $j < i$. The block diagonal entries $\mathcal{D}_j^+$ and $\mathcal{D}_j^-$ are the remaining terms in (3.14) which multiply ΔU_j and are given by

$$\mathcal{D}_j^+ = \left[\mathcal{I} - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} \bar{A}_{ij} \right] \quad \mathcal{D}_j^- = \left[\mathcal{I} - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} \bar{A}_{ij} \right] \quad (3.18)$$

The scheme (3.15) can be solved in two stages

$$[\mathcal{L} + \mathcal{D}^+] \Delta U^* = R$$

$$[\mathcal{U} + \mathcal{D}^-] \Delta U = \Delta U^* \quad (3.19)$$

each stage requiring the inversion of a 4x4 block matrix for each grid point. In order to efficiently implement this scheme in an unstructured mesh, it is convenient to create a sorted list containing all points and edges. This list is not unique but must satisfy two conditions:

- i) The nodes in the list appear in the order implied by their s coordinate
- ii) The edge connecting nodes i and j where $s_i < s_j$ is listed after node i and before node j as illustrated in figure 3.2.

Figure 3.2: Structure of vector used to order sides and nodes.

The forward and back substitutions required to solve (3.19) can then be performed by looping over the elements of this vector in forward and reverse order. When a node is encountered a 4x4 matrix is inverted to obtain the increment of nodal unknowns. When an edge is found, the matrix $\tilde{A}_{ij}$ associated to that edge is computed and used to modify the equation for the corresponding downstream node.

3.2 LU-Gauss-Seidel Relaxation

The linearised implicit scheme (3.9) can be simulated with forward and backward Gauss-Seidel relaxation sweeps as discussed in [10] [16].

$$\Delta U_j^* + c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} A_{ij} (\Delta U_i^* - \Delta U_j^*) - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} A_{ij} \Delta U_j^* + c_j \frac{\Delta \tau}{V_j^p} R_j^p = 0 \quad (3.20)$$

$$\Delta U_j + c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} A_{ij} (\Delta U_i - \Delta U_j) + c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} A_{ij} (\Delta U_i^* - \Delta U_j) + c_j \frac{\Delta \tau}{V_j^p} R_j^p = 0 \quad (3.21)$$

In the forward sweep, the terms involving U_i in the E_j^- summation are unknown, and thus set to zero. On the backward sweep, the same terms but now in the E_j^+ summation are unknown, so are replaced by their most recent value, in this case U_i^* . Rearranging equation (3.20), the forward sweep can be written

$$\left[\mathcal{I} - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^\pm} A_{ij} \right] \Delta U_j^* + c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^+} A_{ij} \Delta U_i^* = -c_j \frac{\Delta \tau}{V_j^p} R_j^p$$

and the backward sweep can be rewritten, after subtracting (3.20) from (3.21), as

$$\left[\mathcal{I} - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^\pm} A_{ij} \right] \Delta U_j + c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^-} A_{ij} \Delta U_i = \left[\mathcal{I} - c_j \frac{\Delta \tau}{V_j^p} \sum_{ij \in E_j^\pm} A_{ij} \right] \Delta U_j^*$$

Using the notation introduced in the last section, the combined scheme can be expressed as

$$(\mathcal{D} + \mathcal{L})\mathcal{D}^{-1}(\mathcal{D} + \mathcal{U})\Delta U = R \quad (3.22)$$

where $\mathcal{D} + \mathcal{I} = \mathcal{D}^+ + \mathcal{D}^-$. It is noted that (3.22) can be regarded as an LU implicit scheme and therefore can be implemented using the same data structure described in the previous section. In practice the scheme (3.22) performs better than the LU factorisation and furthermore has the advantage that only one block diagonal matrix needs to be inverted for each pair of forward and backward sweeps.

3.3 GMRES Algorithm

The convergence of the LU schemes can be further improved through use of a non-linear generalised minimal residual algorithm (GMRES) [17].

The algorithm is best described by considering a differentiable system expressed in vector form as

$$F(U) = 0 \quad (3.23)$$

Given an approximate solution U^n we construct k orthonormal search directions, p_j , as follows

$$p_1 = \frac{F(U^n)}{\|F(U^n)\|} \quad (3.24)$$

For $j = 1, 2, \dots, k - 1$ take

$$\tilde{p}_{j+1} = \overline{F}(U^n; p_j) - \sum_{i=1}^j b_{ij} p_i \quad (3.25)$$

and set

$$p_{j+1} = \frac{\tilde{p}_{j+1}}{\|\tilde{p}_{j+1}\|}$$

where

$$b_{ij} = \overline{F}(U^n; p_j) \cdot p_i$$

and $\overline{F}(U^n; p_j)$ denotes the directional derivative of F evaluated at U^n in the direction of p_j and is approximated as

$$\overline{F}(U^n; p_j) \sim \frac{F(U^n + \epsilon p_j) - F(U^n)}{\epsilon} \quad (3.26)$$

where ϵ is taken to be a small number proportional to $\|U^n\|$.

Once the k search directions are known, U^n is updated according to

$$U^{n+1} = U^n + \sum_{j=1}^k a_j p_j \quad (3.27)$$

where the coefficients a_j are evaluated by minimising $\|F(U^{n+1})\|^2$.

The performance of this algorithm is very dependent on the use of a suitable preconditioner. The LU factorisation algorithm described in the previous section can be used very

effectively for that purpose. After performing a number of relaxation sweeps with a method M , an update to U^n will be obtained. This can be written symbolically as

$$U^{n+1} = M(U^n) \quad (3.28)$$

Convergence is obtained when $U^{n+1} = U^n$, and hence we can use GMRES to obtain the solution of the preconditioned system

$$U - M(U) = 0 \quad (3.29)$$

The nonlinear version of GMRES approximates Newton's method, and thus can diverge unless started sufficiently close to the solution. To avoid this problem, we follow [17] and introduce a damping factor λ into equation (3.29) and apply GMRES to

$$U^{n+1} - (1 - \lambda)M(U^{n-1}) - \lambda M(U^n) = 0 \quad (3.30)$$

The value of λ is initially taken to be close to 1 and is decreased to 0 as convergence is approached.

Chapter 4

Fluid-Aeroelastic Coupling

In this chapter, the coupling of the Euler equations with a typical section wing model is described. The flow equations are solved implicitly using the scheme outlined in the previous chapter and the equations of motion of the structure are decomposed into a set of first-order ordinary differential equations. This first order system is then solved implicitly so that the flow equations and structural model are fully coupled and are advanced simultaneously in time. Once the solution of this system has been obtained and the new position of the wing section determined, a dynamic mesh algorithm is used to deform the mesh accordingly.

4.1 Structural Aeroelastic Model

The wing section is modelled with two degrees of freedom as shown in figure 4.1. h is the plunging coordinate, which is positive down, while α is the angle of attack. The equations of motion can be written in non-dimensional form

$$[M]\{\ddot{q}\} + [K]\{q\} = \{F\} \quad (4.1)$$

where $[M]$ and $[K]$ are the non-dimensional mass and stiffness matrices given by

$$[M] = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix} \quad [K] = \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix} \quad (4.2)$$

Here ω_h and ω_α are the uncoupled natural frequencies of the section in plunge and pitch respectively, $S_\alpha = mbx_\alpha$ is the static unbalance and $I = mb^2r_\alpha^2$ is the section moment of inertia about the elastic axis. The load and displacement vectors are

$$\{F\} = \frac{1}{\pi\mu k_c^2} \begin{Bmatrix} -C_l \\ 2C_m \end{Bmatrix} \quad \{q\} = \begin{Bmatrix} \frac{h}{b} \\ \alpha \end{Bmatrix} \quad (4.3)$$

where k_c is the reduced frequency defined as

$$k_c = \frac{\omega b}{U_\infty} \quad (4.4)$$

C_l is the section lift coefficient and C_m is the moment coefficient about the elastic axis. b is the airfoil semichord, and μ is the airfoil mass ratio given by

$$\mu = \frac{m}{\pi \rho b^2} \quad (4.5)$$

(4.1) can be written as a system of two first order linear equations by letting $v = \dot{q}$. This leads to the following system of equations

$$\begin{Bmatrix} \dot{v} \\ \dot{q} \end{Bmatrix} + \begin{bmatrix} 0 & M^{-1}K \\ -I & 0 \end{bmatrix} \begin{Bmatrix} v \\ q \end{Bmatrix} = \begin{Bmatrix} M^{-1}F \\ 0 \end{Bmatrix} \quad (4.6)$$

The 4x4 matrix here is constant and needs only to be inverted once. At every timestep we can therefore solve (4.6) to find the new position of the airfoil. The fluid and structural equations are fully coupled and are advanced simultaneously in time. In order for the structural equations to be solved implicitly, the forcing vector must be evaluated at time level $n + 1$. However the updated values of C_l and C_m are not available until the fluid equations have been resolved, which in turn can not be achieved until the position of the airfoil is determined. An iterative procedure similar to that described in [6] is used. The values of C_l and C_m from the previous time level are used initially to solve the structural equations. The flow is then recomputed using the new airfoil position and an updated estimate of C_l and C_m is obtained which is used to recalculate the airfoil displacement. This

procedure is repeated until convergence is obtained and the system can then be integrated forward to the next time level.

Figure 4.1: Typical section wing model geometry

4.2 Dynamic Mesh Algorithm

Once the new position of the airfoil has been determined from the structural equations, the grid must be deformed so that the fluid equations can be solved to find the resulting flow. This is done by modelling each edge in the grid as a spring with stiffness inversely proportional to the length of the edge. For an edge connecting nodes i and j

$$k_{ij} = \frac{1}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \quad (4.7)$$

The position of points on the airfoil surface are prescribed by the calculated surface motion, while points on the outer farfield boundary are held fixed. At each timestep we must solve

the static equilibrium equations which result from force summation for every node. Jacobi iterations of these equations are performed to obtain the displacements in the x and y directions.

$$\delta_{x_i}^{n+1} = \frac{\sum_j k_{ij} \delta_{x_j}^n}{\sum_j k_{ij}} \quad \delta_{y_i}^{n+1} = \frac{\sum_j k_{ij} \delta_{y_j}^n}{\sum_j k_{ij}} \quad (4.8)$$

The new position of each node can then be determined

$$x_i^{n+1} = x_i^n + \delta_{x_i}^{n+1} \quad y_i^{n+1} = y_i^n + \delta_{y_i}^{n+1} \quad (4.9)$$

Generally, of the order of ten Jacobi iterations are required to obtain convergence.

Chapter 5

Results

In this chapter the results obtained using the schemes outlined previously are described. Firstly, a steady flow is considered in order to evaluate the relative performances of the fluid solvers developed. An unsteady problem is then considered in which the motion of the airfoil is prescribed. Finally, a fully coupled aeroelastic flow problem is considered in which all elements of the algorithm are brought together.

5.1 Steady State Flow

The first test case considered is that of inviscid transonic flow past a NACA 0012 airfoil. The freestream Mach number was 0.8 with an angle of attack of 1.25° . The farfield boundary was placed at a distance of 25 chord lengths from the airfoil, and a compressible

vortex perturbation was used to improve the farfield boundary conditions. The solution was obtained using an explicit predictor/corrector scheme, the LU-Gauss-Seidel factorisation and the GMRES algorithm preconditioned with the LU-Gauss-Seidel scheme on an unstructured mesh containing 16634 nodes and 32908 elements. For the LU schemes, the sweep vector $\mathbf{s}$ was taken to be (1,1) and the timestep was chosen according to (3.6) with a safety factor of 10000. The value of ϵ used for the finite difference computation of the directional derivative was not found to affect the performance of the algorithm significantly, however varying the number of search directions used in each cycle was found to alter the convergence properties.

The pressure contours of the steady state solution are plotted in figure 4. Convergence histories for the three schemes are shown in figure 5.2. The residual is plotted as a function of the number of calls made to the solution routine. For the LU schemes, these calls require the inversion of 4×4 block matrices for each grid point, so the computational cost is greater than for the equivalent number of calls in the explicit predictor-corrector scheme. Two sets of results are presented for the GMRES scheme with LU-Gauss-Seidel as a preconditioner. In the first, the number of search directions is increased from 5 to a maximum of 30 in steps of 5 per cycle, while for the second we increase from 2 to 60 in steps of 2. The number of cycles required to reach convergence for these cases was 46 and 34 respectively. The implicit schemes show a big improvement in convergence rate over the explicit predictor corrector scheme. GMRES can be seen to further improve the performance of the LU-Gauss-Seidel

scheme.

Figure 5.1: Pressure contours for steady inviscid transonic flow.

Figure 5.2: Residuals for steady inviscid transonic flow.

5.2 Unsteady Forced Oscillation

The unsteady problem considered was the AGARD test case number 5 for the NACA 0012 airfoil [18]. A steady state solution is first computed then the airfoil is set in motion for unsteady calculations. The airfoil was caused to oscillate about its quarter chord point. The angle of attack is given by

$$\alpha = \alpha_m + \alpha_0 \sin(\omega t) \quad (5.1)$$

where ω is defined in terms of a reduced frequency

$$k_c = \frac{\omega c}{2U_\infty} \quad (5.2)$$

where c is the chord of the airfoil and U_∞ is the freestream velocity. For the case considered, $M_\infty = 0.755$, $\alpha_m = 0.016^\circ$, $\alpha_0 = 2.51^\circ$, $k_c = 0.0814$.

The GMRES algorithm with the LU-Gauss-Seidel scheme as a preconditioner was used to compute the solution. The grid used had 4258 nodes and 8320 triangular elements. First and second order accurate timestep schemes were considered. The number of timesteps per oscillation cycle was taken to be 36. Because the transient solution computed at each timestep is close to the previous solution, it is not necessary to use any damping in the GMRES algorithm. The resulting lift and moment coefficients for the second order time

integration scheme are plotted in figure 5.3, along with the experimental data [18] and minor deviations are observed. The residuals are plotted in figure 5.4. The number of search directions was chosen to be 20, requiring approximately 12-15 GMRES cycles per timestep.

Figure 5.3: Lift coefficient and moment coefficient for unsteady inviscid transonic flow.

Figure 5.4: Residuals for unsteady inviscid transonic flow.

5.3 Fully Coupled Fluid-Structural Problem

A fully coupled structural-fluid problem was then considered. This is the two-dimensional swept wing model (Case A of Isogai [3]). The structural parameters are $x_\alpha = 1.8$, $r_\alpha^2 = 3.48$, $\frac{\omega_h}{\omega_\alpha} = 1$ and $\mu = 60$ with the elastic axis positioned one chord length ahead of the leading edge. The steady state solution for flow about a NACA 64-A010 airfoil is first computed for a given Mach number. The airfoil is then forced sinusoidally in pitch about the elastic axis for three complete cycles at a frequency ω_f . At the end of these forced cycles the system is allowed to respond to its self-induced loads. The response is computed for a number of different Mach numbers and different values of the speed index V_f .

A noted difference in the characteristic behaviour of the airfoil response is observed between subsonic and transonic Mach numbers. The pitching and plunging responses of the airfoil are shown in figures 5.5 and 5.6 for a Mach number of 0.6. For this subsonic case, we see a clear transition from stable response at a speed index of 1.1 to unstable response at the only slightly higher value of 1.15. In the transonic regime however, nonlinear effects become important and the transition from stable to unstable response is not so sharp. Figure 5.7 shows the response obtained at a Mach number of 0.82 and a speed index of 0.71. For speed indices less than 0.7 the response was found to be purely damped, however there is a range of values of V_f above this where limit cycle behaviour is observed such as that shown in figure 5.7. When V_f becomes high enough, the response will consist of

growing oscillations such as those shown in figure 5.6, however in the region where limit cycle behaviour occurs, the response cannot be labelled as purely damped or undamped. In many cases it is necessary to obtain the solution for an extended time period in order to determine whether the oscillations will decay.

The flutter boundary for the airfoil can be determined by finding the speed index at which flutter occurs for several different Mach numbers. Figure 5.8 shows results from the present code and from references [6] and [19]. The results demonstrate the transonic dip phenomenon as discussed in [3].

The results also demonstrate that for certain Mach numbers there is a phase difference between the pitching and plunging modes of response. Figure 5.9 shows these phase differences at the flutter point for a several different Mach numbers. The phase difference is largest for the subsonic case $M = 0.6$ and reduces to almost zero at a Mach number of 0.875. The natural frequencies of the section can be calculated by determining the eigenvalues of the structural system (4.1). For the structural parameters considered, the natural frequencies of the section were calculated to be 0.71 and 5.34. The eigenvectors of the system determine the shape of the structural response. In this case there is one symmetric and one anti-symmetric mode. For the first mode, the pitch and plunge of the airfoil are in phase which implies the existence of a pivotal point ahead of the leading edge. This point was calculated to be at $x_{pv} = -3.87$. Because the phase difference at $M = 0.875$ is almost

zero, the pivotal point for this flutter motion can also be calculated using the expression

$$x_{pv} = a - \frac{h/b}{\alpha} \quad (5.3)$$

which relates the relative amplitudes of the plunging and pitching motion. For the $M = 0.875$ case x_{pv} was calculated to be -3.96. This flutter mode is thus very close to the first natural mode which is pure pitching motion about the pivotal point $x_{pv} = -3.87$. In the range of this particular Mach number, as the speed index is varied multiple flutter points are observed. Increase in V_f past the second flutter point once again results in a stable response. As figure 5.10 shows, flutter at the higher point occurs in the second mode of vibration with a higher frequency and in the antisymmetric mode. Because the airfoil is now responding with a much higher frequency, it is necessary to decrease the size of the timestep in order to maintain accuracy in the calculations.

Figure 5.5: Damped Response. $M = 0.6$, $V_f = 1.1$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega_f}{\omega_a} = 1$

Figure 5.6: Unstable Response. $M = 0.6$, $V_f = 1.15$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega_f}{\omega_a} = 1$

Figure 5.7: Limit Cycle Response. $M = 0.82$, $V_f = 0.71$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega_f}{\omega_\alpha} = 1$

Figure 5.8: Flutter Boundary. $V_f = U_\infty / b\omega_\alpha \mu^{0.5}$

Figure 5.9: Flutter mode phase difference. $\mu = 60$

Figure 5.10: Second mode response. $M_\infty = 0.875$, $V_f = 2.6$, $\alpha_m = 0^\circ$, $\alpha_o = 1^\circ$, $\frac{\omega_f}{\omega_\alpha} = 1$

Chapter 6

Conclusions

An implicit finite volume algorithm has been developed for solution of the unsteady Euler equations on unstructured grids. An approximate LU factorisation was derived to avoid the solution of a full matrix system at every timestep. The performance of the solver was improved through use of a nonlinear GMRES algorithm. This fluid solver has been successfully coupled with a typical section aeroelastic model and full convergence of the two sets of equations, fluid flow and structural, is achieved at every timestep.

The fully coupled system was used to investigate the behaviour of a NACA 64A010 section in unsteady flows. Nonlinear effects were found to be important in the transonic regime, emphasizing the need for time accurate solutions rather than just an analysis of the linearised equations as is done in small disturbance codes. For subsonic flows, there

is a clear transition between damped response and oscillations which grow in time. For transonic cases however, there is a range of values of the speed index where limit cycle behaviour is observed. Multiple flutter points were observed for some Mach numbers, with the response at the higher flutter point occurring in the second mode. Varying amounts of phase difference between the pitching and plunging responses were also observed across the range of Mach numbers considered.

In order to predict other unsteady phenomena such as buffeting, viscous effects must be considered. Work is underway to include these contributions in the fluid model. Also, the algorithm must be extended to three dimensions in order to model practical configurations.

Bibliography

- [1] A. Jameson. Computational Algorithms for Aeodynamic Analysis and Design. *Applied Numerical Mathematics*, 13:383–422, 1992.
- [2] J. Peraire, J. Peiro, and K. Morgan. Multigrid Solutions of the 3D Compressible Euler Equations on Unstructured Tetrahedral Grids. *International Journal for Numerical Methods in Engineering*, 36:1029–1044, 1993.
- [3] K. Isogai. Transonic Dip Mechanism of Flutter of a Sweptback Wing: Part II. *AIAA*, 19:1240–1242, September 1981.
- [4] K.A. Kousen and O.O. Bendiksen. Nonlinear Aspects of the Transonic Aeroelastic Stability Problem. AIAA-88-2306, 1988.
- [5] A. Jameson. Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings. AIAA 91-1596, 1991.

- [6] J.J. Alonso and A. Jameson. Fully-Implicit Time-Marching Aeroelastic Solutions. AIAA-94-0056, 1994.
- [7] K. Morgan, J. Peraire, and J. Peiro. Unstructured Grid Method for High Speed Compressible Flows. Int. Conference on Mathematics of Finite Elements and Applications, Academic Press, 1993.
- [8] M.H. Lallemand, H. Steve, and A. Dervieux. Unstructured Multigriding by Volume Agglomeration: Current Status. *Computers and Fluids*, 21:397–433, 1992.
- [9] V. Venkatakrishnan and D. Mavriplis. Agglomeration Multigrid for the 3D Euler Equations. AIAA-94-0069, 1994.
- [10] A. Jameson and S. Yoon. LU Implicit Schemes with Multiple Grids for the Euler Equations. AIAA-86-0105, 1986.
- [11] R.L. Bisplinghoff and H. Ashley. *Principles of Aeroelasticity*. Dover Publications, New York, NY, 1975.
- [12] J.T. Batina. Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex-Aircraft Aeroelastic Analysis. AIAA-89-1189, 1989.
- [13] P.L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, 43:357–372, 1981.

- [14] M. Lesoinne and C. Farhat. Geometric Conservaytion Laws for Aeroelastic Computations using Unstructured Dynamic Meshes. AIAA-95-1709, 1995.
- [15] G.D. Van Albada, B. Van Leer, and W.W. Roberts. A Comparative Study of Computational Methods in Cosmic Gas Dynamics. *Astronomy and Astrophysics*, 108:76–85, 1982.
- [16] S. Yoon and A. Jameson. An LU-SSOR Scheme for the Euler and Navier-Stokes Equations. AIAA-87-0600, 1987.
- [17] L.B. Wigton, N.J. Yu, and D.P. Young. GMRES Acceleration of Computational Fluid Dynamics Codes. AIAA-85-1494, 1985.
- [18] AGARD. Compendium of Unsteady Aerodynamic Measurements. Report No. 702, 1982.
- [19] J. W. Edwards, R.M. Bennett, W. Whitlow Jr., and D.A. Seidel. Time-Marching Transonic Flutter Solutions Including Angle-of-Attack Effects. *Journal of Aircraft*, 20:899–906, November 1983.