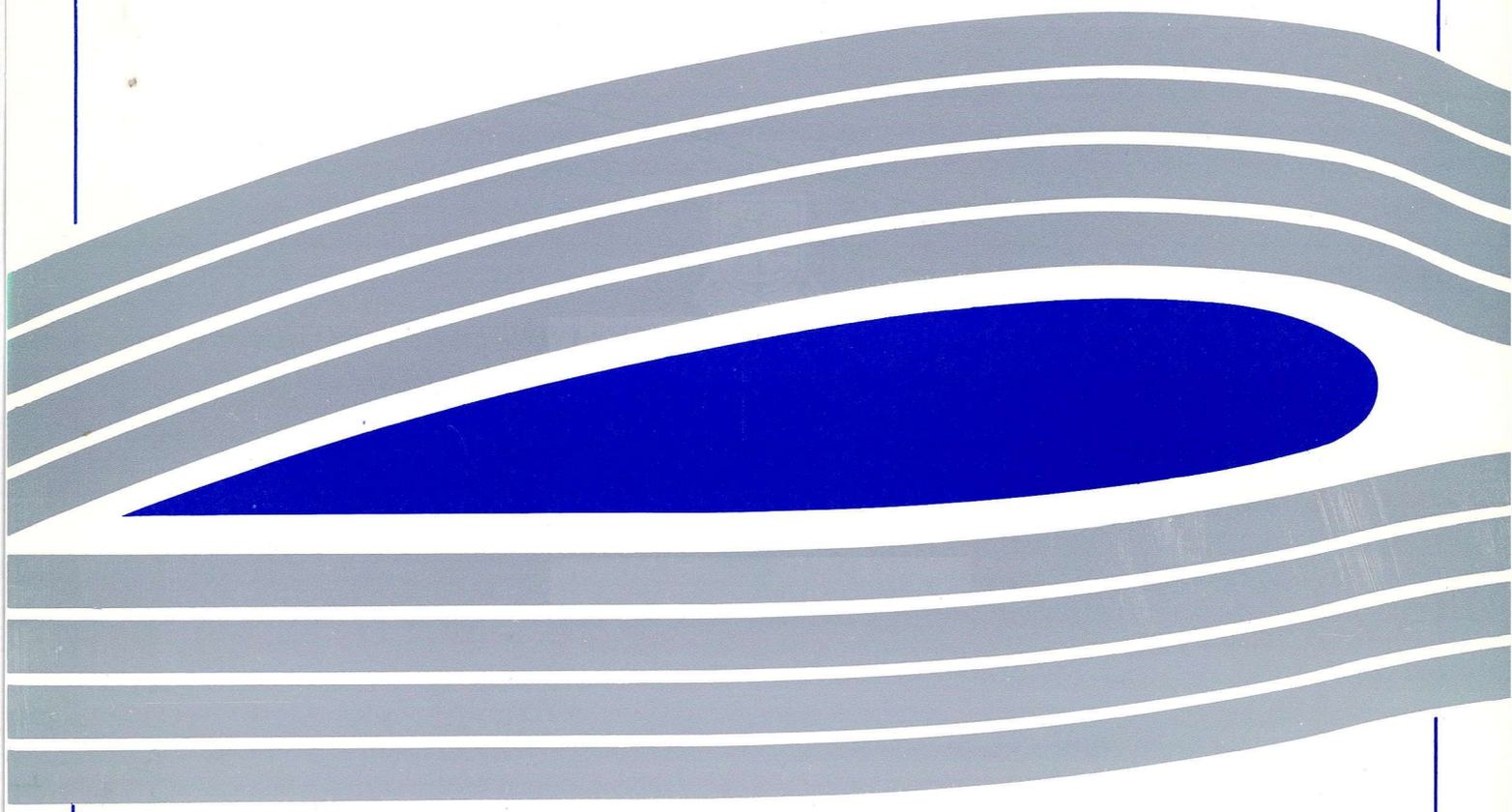
University of Glasgow
DEPARTMENT OF
AEROSPACE
ENGINEERING

.F-CGS-P:An Unfactored Method in Parallel
›r Turbulent Pitching Aerofoil Flows

K.J. Badcock

Aero Report 9323



AF-CGS-P:An Unfactored Method in Parallel for Turbulent Pitching Aerofoil Flows

K.J. Badcock

Aero Report 9323

September 14, 1993

AF-CGS-P:An Unfactored Method in Parallel for Turbulent Pitching Aerofoil Flows

K.J. Badcock *
Aerospace Engineering Department
University of Glasgow,
Glasgow, G12 8QQ, U.K.
EMAIL:gnaa36@uk.ac.glasgow.udcf

September 14, 1993

Abstract

An unfactored implicit method for the unsteady, turbulent Navier-Stokes equations is implemented in parallel. The algorithm uses the alternating direction implicit (ADI) method as a preconditioner for a conjugate gradient solution of the linear system at each time step. The sequential nature of the ADI process is tackled by using a transposition method to solve complete ADI sweeps on single processors. Results are presented for a standard pitching aerofoil case solved on the iPSC.

*This work was carried out under SERC contract.

Contents

1 Introduction

The solution of the Navier-Stokes equations is becoming increasingly common for aeroelastic problems (see [1] for a review). The major computational work in such an analysis is the solution of the unsteady aerodynamic model. CFD research at Glasgow has centred on using upwind schemes and fully implicit methods for hypersonic and transonic viscous flows. This work is being extended to develop an unsteady 3-D Navier-Stokes solver for incorporation into an aeroelastic simulation code.

Two main algorithms have been under development for aerofoil flows. Parallel implementation has been a major consideration due to the extensive memory requirements of 3-D implicit schemes in general and the intensive CPU needs of a 3-D unsteady Navier-Stokes solution.

The first algorithm is partially implicit. Normal-direction terms are treated implicitly to help to remove the stability limits arising from the fine mesh spacing needed to resolve boundary layers. The terms in the streamwise direction are differenced explicitly. This yields an algorithm which has very efficient parallel properties in that the implicit part can proceed indepently on separate nodes without the need for any communication during this phase of the calculation. Parallel efficiencies of around ninety percent have been achieved on 16 nodes of the iPSC. However, the stability restrictions arising from the streamwise direction can prove restrictive and for aerofoil problems the algorithm is too slow unless a large number of nodes are available to take advantage of its good parallel properties. The method has been tested for aerofoil and viscous shocktube flows and full details can be found in [2].

To achieve a more efficient method it is necessary to examine ways of increasing the allowable time step. The conventional way of achieving this is to include more implicitness in the time-stepping method. By treating all of the spatial terms implicitly the stability limits can be removed entirely and the time step can then be chosen from accuracy considerations.

However, the large sparse linear system that needs to be solved at each step represents a significant computational obstacle. Traditionally for Navier-Stokes codes this system is solved by an approximate factorisation method. There is an inherently sequential nature to this process which limits the utility of parallel implementations. This was tackled in [3] by using a special communication of the data known as transposition.

The factorisation error introduces a practical stability limit and also degrades the accuracy of the solution. An unfactored method based on an approximate factorisation as a preconditioner for a conjugate gradient method was discussed in [4] and [1]. The method proved to be successful in

serial and in this paper a parallel version of the AF-CGS code is developed using the transposition idea.

The AF-CGS method is briefly restated in the following two sections. The reader is referred to previous publications for a more detailed description of the method and a more thorough demonstration of its capabilities. Then the parallel version of the algorithm is given and results for standard pitching aerofoil flows are given.

2 AF-CGS Method

The thin-layer Navier-Stokes equations in generalised co-ordinates are given by

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial \mathbf{f}}{\partial \xi} + \frac{\partial \mathbf{g}}{\partial \eta} = (Re)^{-1}\frac{\partial \mathbf{s}}{\partial \xi} \tag{1}$$

where

$$\mathbf{w} = J^{-1}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \mathbf{f} = J^{-1}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ U(e+p) - \xi_t p \end{bmatrix}, \mathbf{g} = J^{-1}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ V(e+p) - \eta_t p \end{bmatrix}$$

$$\mathbf{s} = J^{-1}\begin{bmatrix} 0 \\ \mu m_1 u_\xi + (\mu/3) m_2 \xi_x \\ \mu m_1 v_\xi + (\mu/3) m_2 \xi_y \\ \mu m_1 m_3 + (\mu/3) m_2 (\xi_x u + \xi_y v) \end{bmatrix}.$$

Here,

$$U = \xi_t + \xi_x u + \xi_x v$$

$$V = \eta_t + \eta_x u + \eta_x v$$

$$m_1 = \xi_x^2 + \xi_y^2$$

$$m_2 = \eta_x^2 + \eta_y^2$$

$$m_3 = (u^2 + v^2)/2 + Pr^{-1}(\gamma - 1)^{-1}(c^2)_\xi$$

and J is the determinant of the Jacobian of the transformation $x = x(\xi, \eta, t)$ and $y = y(\xi, \eta, t)$. Here ρ, u, v, e, p, Pr, Re, c, γ denote density, the two components of velocity, energy, pressure, the Prandtl number, the Reynolds number, the speed of sound and the constant ratio of the specific heats respectively. The viscosity is composed of a part due to the natural viscosity of the fluid and a term to account for turbulence. Sutherland's law is used to describe the variation of the fluid viscosity with temperature. The Baldwin-Lomax model is used to provide a value for the turbulent viscosity. Since none of the flows examined herein involve massive separation no modification of the turbulence model is used.

To solve this system of partial differential equations a finite volume scheme is used which has various features. For the spatial terms Osher's method is used. For general geometries the details of Osher's method are described in [5]. A MUSCL interpolation is used to provide second or third order accuracy and the Von Albada limiter prevents spurious oscillations from occuring around shock waves. Central differencing is employed for the viscous terms. Far-field boundary conditions are imposed by Riemann-invariants and no vortex correction is applied due to the unsteadyness of the flow.

The temporal discretisation is based on the backward Euler method. An efficient mixed analytic and finite difference procedure is used to generate the required Jacobian of the spatial discretisation. The linear system obtained is solved by the conjugate gradient squared (CGS) method with the alternating direction implicit approximate factorisation providing a preconditioner. The reader is refered to [4] [1] for full details. A summary is given in the next section.

The problems we consider herein relate to pitching aerofoils. Since no deformation of the aerofoils is considered the mesh can be rotated rigidly with the aerofoil. Defining the starting mesh points by $\{x = x_0\}_{i,j}$, $\{y = y_0\}_{i,j}$ the transformed mesh at time t_1 is given by

$$\{x = r_0 cos(\alpha(t_1) - \alpha_0) + l(1 - cos(\alpha_0))\}_{i,j}$$

$$\{y = r_0 sin(\alpha(t_1) - \alpha_0) + l sin(\alpha_0)\}_{i,j}$$

where the aerofoil pitches about the point (1,0), the initial incidence is α_0 and $r^2_{0_{i,j}} = x^2_{0_{i,j}} + y^2_{0_{i,j}}$. The mesh velocities can be easily evaluated given an analytical form for α. No-slip boundary conditions on the aerofoil can then be imposed.

3 The Solution of the linear system

One time step of the method can be denoted by

$$(I + \Delta t \frac{\partial R_\xi}{\partial w} + \Delta t \frac{\partial R_\eta}{\partial w})\delta w = -\Delta t(R_\xi + R_\eta) \qquad (2)$$

where R_ξ and R_η are terms arising from the spatial discretisation in the ξ and η directions respectively. The derivatives required on the left-hand side of 2 are calculated by an efficient mixed analytic/finite difference procedure [6]. The alternating direction implicit (ADI) factorisation of 2 is

$$(I + \Delta t \frac{\partial R_\xi}{\partial w} + \Delta t \frac{\partial R_\eta}{\partial w})\delta w \approx (I + \Delta t \frac{\partial R_\xi}{\partial w})(I + \Delta t \frac{\partial R_\eta}{\partial w})\delta w = -\Delta t(R_\xi + R_\eta). \qquad (3)$$

Equation 3 is much easier to solve than 2 because each of the factors is typically block tridiagonal or block pentadiagonal. However, the solution to the ADI factored system is not an exact one for equation 2 and in practice the factorisation error leads to a stability limit on the time step and introduces another source of error to the calculation.

The approach adopted in the present work involves solving equation 2 to a required tolerance by using a conjugate gradient method. Denoting the unfactored linear system 2 to be solved at each time step by

$$A\mathbf{x} = \mathbf{b} \tag{4}$$

we seek an approximation to $A^{-1} \approx C^{-1}$ which yields a system

$$C^{-1}A\mathbf{x} = C^{-1}\mathbf{b} \tag{5}$$

more amenable to conjugate gradient methods. The ADI method gives a fast way of calculating an approximate solution to 4 or, restating this , of forming the matrix vector product

$$C^{-1}\mathbf{b} = \mathbf{x}. \tag{6}$$

Hence, if we use the inverse of the ADI factorisation as the preconditioner then multiplying a vector by the preconditioner can be achieved simply by solving a linear system with the right-hand side given by the multiplicand and the left hand side given the approximate factorisation. The factors in C can be diagonalised once at each time step with the row operations being stored for use at each multiplication by the preconditioner.

The exact form of the algorithm for one step of the Navier-Stokes solution is

- calculate matrices and diagonalise ADI factors
- calculate updated solution by ADI
- use this solution as starting solution for CGS
- perform CGS iterations until 4 has been solved to required tolerance

4 Parallel Implementation

The major obstacle to an efficient parallel implementation of the AF-CGS method is the inherently sequential nature of the ADI procedure. This was overcome in [3] by using a transposition of the data to allow complete ADI sweeps to proceed independently on each processor. We use this approach here although extra communication is required for the present method because of the matrix-vector products required in the CGS algorithm.

The computational space in generalised coordinates is mapped onto the nodes by grouping several mesh lines in both the ξ and the η directions onto one node. Care has to be taken to make sure that ξ lines separated only by the wake cut are mapped to the same processor. The computation then falls into three phases. First, the matrix is generated and the factors are diagonalised. The next phase is the multiplication of a vector by the matrix and finally we have multiplication of a vector by the preconditioner which reduces to back substititution on the diagonalised factors of the ADI factorisation. For each phase data is held on a node for complete lines in one direction and the entire computation relating to that direction is completed. The data is then communicated so that information for lines in the other direction is held on each node and the computation for that direction proceeds.

For the calculation of the matrix, the solution from the previous time step is initially held on nodes by lines in the η direction. The fluxes and derivatives are calculated for each line and the contributions to the residual are formed. At this stage the factor

$$I + \Delta t \frac{\partial R_\eta}{\partial w} \tag{7}$$

is diagonalised. The solution and the residual are then transposed so that each node holds information on complete lines in the ξ direction. No communication of the components of the matrix $\partial R_\eta / \partial w$ is made. They are stored on the nodes on which they are calculated. The calculation proceeds so that we have the left hand side of 2 stored in complete ξ and η lines on a node and the solution and residual stored on complete ξ lines only.

Next, one step of ADI is performed to obtain a starting solution for CGS. This is achieved by performing ADI sweeps in the ξ direction first, then transposing the solution obatined and performing sweeps in the η direction to yield the updated ADI solution. A multiplication by the preconditioner is performed in the same way. To ensure that the CGS starting solution and the right hand side of the linear system are stored in the same direction the residual is transposed.

The CGS solution proceeds with inner products and matrix multiplications being required. Contributions to an inner product are calculated locally on each node and are then transmitted to a chosen node for summing. The results is then transmitted back to all the nodes for use. A matrix multiplication is achieved by finding the product in the current direction (η), transposing the multiplicand and the partial product and then finding the product in the other direction. This has the effect of changing the direction of the storage. The matrix multiplication is always followed

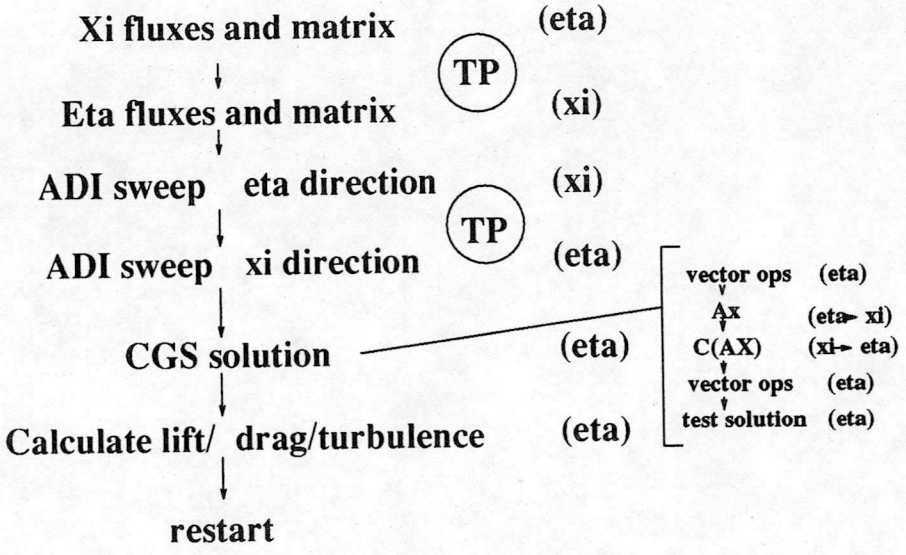


Figure 1: *Schematic diagram of the parallel AF-CGS method. Here the operation is denoted on the left hand side, the direction of storage is indicated in parenthesis and TP denotes when a transposition of the data takes place. The box inset shows the flow of the main parts of the CGS algorithm computation.*

by a multiplication by the preconditioner as described above and this reverses the direction back again. As a result of this all of the CGS vectors lie in the direction which is current on entry to the CGS subroutine. The complete sequence of transpositions is shown in figure 1.

Finally, the turbulent viscosity and the lift and drag coeffients are calculated in parallel before the next step proceeds.

The parallel code was tested on the intel Hypercube at the SERC Daresbury Laboratory. The test case considered was to calculate three non-dimensional time units of the flow for the pitching aerofoil case 2 defined below. The CPU times are shown in table 1 and indicate that the transpositions degrade the parallel efficiency. However, this is the case for most parallel preconditioned conjugate gradient solvers and the efficiencies are good compared to the incomplete ILU-conjugate gradient solver of [7]. It is anticipated that the efficiency will increase on finer meshes as the amount of computation increases on each node. A significant speed up is still achieved in the present case. In addition, the large storage requirements of the algorithm have been divided amongst the processors. The breakdown of the time taken by each main phase of the calculation is

Machine	CPU time	efficiency
SPARC 10	338	–
4 nodes	187	1.00
8 nodes	106	0.88
16 nodes	68	0.69

Table 1: *CPU times in seconds on various machines. The parallel efficiency is based on the CPU time for 4 nodes since the problem is too large in terms of memory to fit onto one node.*

Machine	total	Jacobian calculation and ADI diagonalisation	ADI step	CGS solution
SPARC 10	338	8.00	0.31	1.82
iPSC-8 nodes	106	2.27	0.21	0.58
speed up	3.2	3.5	1.48	3.14

Table 2: *Breakdown of calculation CPU times in seconds on a parallel and a serial machine.*

shown in table 2. The net speed up achieved by using the parallel machine is 3.2. The main computational work is involved with calculating the jacobian and diagonalising the ADI factors It is clear that the parallelisation is most effective for this phase of the calculation which takes up about eighty percent of the total work. The second most intensive part of the work is the CGS solution and this too is effectively parallelised even though the ADI part of the CGS solution slightly degrades the efficiency. Finally, an ADI step is relatively inefficient due to the large communication time relative to the computational time involved.

5 Test Cases

The AGARD sub-committee on aeroelasticity defined test cases to act as standard flows for computer code evaluation and verification. In this paper we present results for two of these test cases for pitching aerofoils. The motion is defined by the angle of attack as a function of time and the centre of rotation x_c which is given herein as a distance along the chord as a percentage of the chord length. The angle of attack is defined as

$$\alpha(\tau) = \alpha_m + \alpha_0 sin(k\tau) \tag{8}$$

where $\tau = tl/V_{\text{inf}}$ is the non-dimensional time.

number	aerofoil	M_{inf}	$Re \times 10^6$	α_m	α_0	k	x_c
1	NACA0012	0.60	4.8	4.86	2.44	0.1620	0.25
2	NACA64A010	0.796	12.56	0.	1.0	0.204	0.248

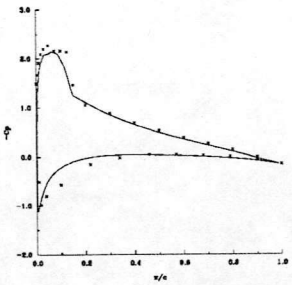
Table 3: *Test cases examined in this paper.*

The cases considered, which are listed in table 3, were selected because detailed pressure distributions are available at a number of points during the cycle. All of the results were obtained on a 71 by 33 C-mesh which was generated by the Eagle grid generation package. The far field was set at 10 chords. The AFCGS method was used throughout with the CGS tolerance set at 10^{-1}. The number of time steps per cycle for each case is 150. The unsteady computations are started impulsively from the converged steady solution at the mean angle of attack and the results are shown after two cycles of the motion. Detailed comparisons for all the cases are shown for the pressure distributions at eight separate times during one cycle. Data for case 2 was only available for the upper surface. For a more detailed discussion of the results the reader is referred to [1]. The detailed pressure distribution comparisons are shown in figures 2-3. Excellent agreement with the experimental data of [8] is noted.

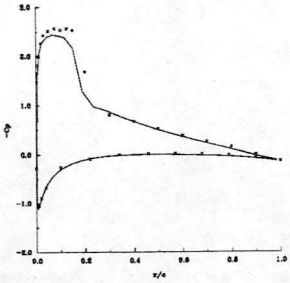
6 Conclusions

It was demonstrated in [4] and [1] that the AF-CGS method can produce efficient and accurate results to attached aerofoil flows. A parallel version of the algorithm was presented in this paper and an efficiency comparable with other preconditioned conjugate gradient algorithms was achieved. Accurate results for standard pitching aerofoil flows can be achieved in less than 15 minutes per period using eight nodes of the iPSC. A drawback of implicit methods is the substantial of memory required. The parallel version tackles this problem by splitting the memory needs between nodes without duplication.

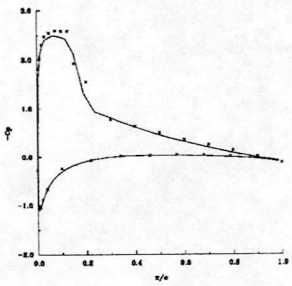
Tackling three-dimensional unsteady aerodynamic problems on the iPSC with the AF-CGS method now seems feasible and will for the next major task of this work. It is also intended to examine some two-dimensional aeroelastic applications in collaboration with British Aerospace.



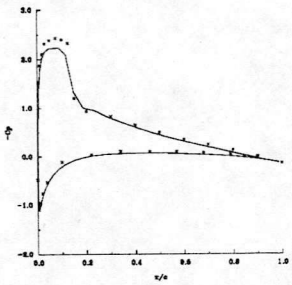
$\alpha = 5.94^o$



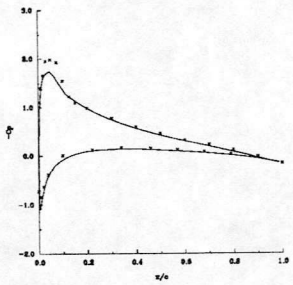
$\alpha = 6.96^o$



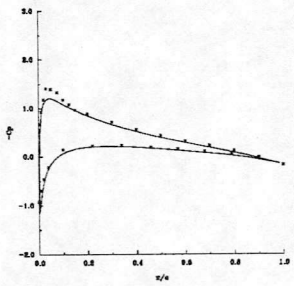
$\alpha = 6.59^o$



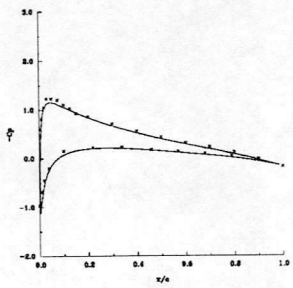
$\alpha = 5.12^o$



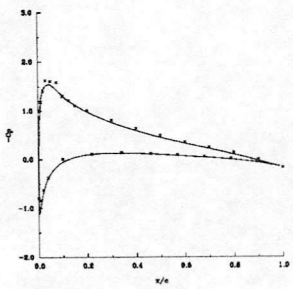
$\alpha = 3.51^o$



$\alpha = 2.44^o$

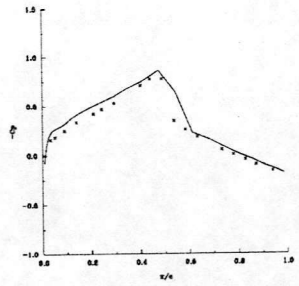


$\alpha = 2.67^o$

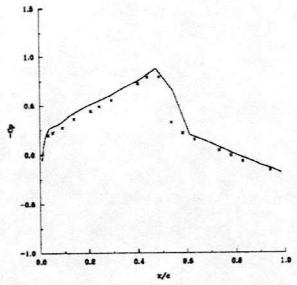


$\alpha = 4.27^o$

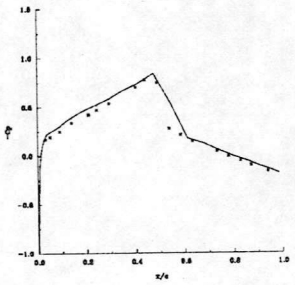
Figure 2: $-C_p$ vs x/c for case 1. x-experiment, line - computed



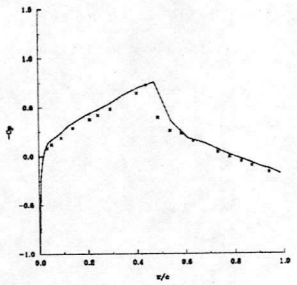
$\alpha = 1.00^o$



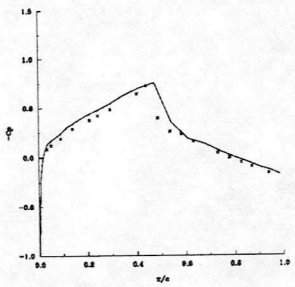
$\alpha = 0.74^o$



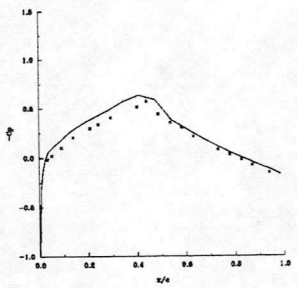
$\alpha = -1.06^o$



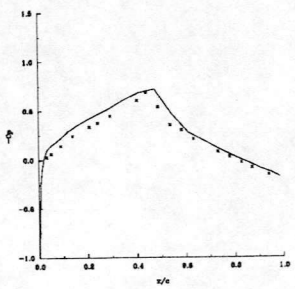
$\alpha = -0.73^o$



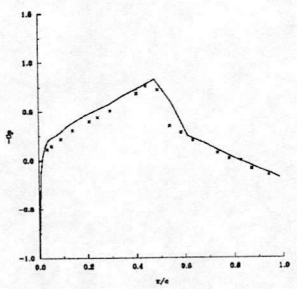
$\alpha = -1.01^o$



$\alpha = -0.59^o$



$\alpha = 0.21^o$



$\alpha = 0.87^o$

Figure 3: $-C_p$ vs x/c for case 2. x-experiment, line - computed

References

[1] K.J.Badcock. Computation of turbulent pitching aerofoil flows. Technical report, G.U. Aero report 9322, 1993.

[2] K.J.Badcock. A parallelisable partially implicit method for unsteady viscous aerofoil flows. Technical report, G.U. Aero report 9312, 1993.

[3] R. Pelz T. Chyczewski, F. Marconi and E. Churchitser. Solution of the Euler and Navier-Stokes equations on a parallel processor using a transposed/Thomas ADIalgorithm. In *11th AIAA Computational Fluid Dynamics Conference*. AIAA, 1993.

[4] K.J.Badcock. An efficient unfactored implicit method for unsteady aerofoil flows. Technical report, G.U. Aero report 9313, 1993.

[5] S.Osher and S.R.Chakravarthy. Upwind schemes and boundary conditions with applications to Euler equations in general coordinates. *J. Comp. Phys.*, 50:447–481, 1983.

[6] K.J.Badcock. Newton's method for laminar aerofoil flows. Technical report, G.U. Aero report 9310, 1993.

[7] P.J. Wesson. *Parallel Algorithms for Systems of Equations*. PhD thesis, Oxford University Computing Laboratory, 1992.

[8] J.J. Olsen. Compendium of unsteady aerodynamic measurements. Technical Report 702, AGARD, 1982.