Implementation of Multigrid for Aerodynamic Computations on Multi-Block Grids

by

Luis Manzano

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Implementation of Multigrid for
Aerodynamic Computations on
Multi-Block Grids

Luis Manzano
Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto
1999

A multigrid algorithm to accelerate the convergence of the computation of compressible aerodynamic flows on multi-block grids is presented. The diagonal form of an approximately-factored time-marching method is employed in each block to advance the solution to steady state. Block interfaces are treated by overlapping the blocks by one column of points and using halo points. In the multigrid cycle, there is communication between blocks at all levels. Turbulent viscosity is calculated using the Spalart-Allmaras turbulence model. A wide variety of tests are studied, including inviscid flows around single airfoils for C-mesh and H-mesh topologies. Two multi-element test cases are investigated, the NLR-7301 airfoil and flap, and case A-2 from the AGARD Advisory Report No. 303. Good performance is shown in all test cases. Compared to the single-grid solver, multigrid converges to steady state in one-third to one-sixth the time.

Acknowledgements

I would like to thank the following individuals who made significant contributions to this research:

- Professor Zingg for his excellent guidance and supervision throughout my research.
- Doctor Nelson for his help with the TORNADO solver, patience and excellent teaching on CFD.
- Todd Chisholm for his multigrid explanations.
- Stan De Rango for his help with SC1, TORNADO ...
- Jason Lassaline for proof-reading my thesis and help with multigrid and Latex.
- The rest of the CFD group for their friendship, good humour and support.
- My girlfriend and my parents for being so supportive, understanding and motivating.
- Mike Sullivan for putting up with all the network technical problems.

LUIS MANZANO

University of Toronto

January 20, 1999

Contents

List of Figures

List of Symbols

$\hat{A}, \hat{B}$	flux Jacobian matrices
$\hat{E}, \hat{F}$	curvilinear convective flux vectors
$\hat{M}$	viscous flux Jacobian
J	metric Jacobian
M	Mach number
Pr	Prandtl number
Pr_t	turbulent Prandtl number
$\hat{Q}$	curvilinear state vector
Re	Reynolds number
$\hat{S}$	viscous flux vector
T	eigenvector matrix
U, V	contravariant velocities
μ	molecular viscosity
μ_t	eddy viscosity

ρ	density
a	speed of sound
e	total energy
p	pressure
u, v	Cartesian velocities
h	time step
H_0	total enthalpy

ξ, η	curvilinear coordinates

Q^0	multigrid initial solution vector
Q	multigrid working solution vector
Q^+	multigrid corrected solution vector
P	forcing function

RESTRCT	restrict
SMOOTH	perform one iteration
PLNG	prolong
RHS	right hand side

Chapter 1

Introduction

1.1 Background

In computational fluid dynamics there are many aspects that must be considered when solving the Navier-Stokes equations. It is important to consider the robustness, accuracy and computational expense of the algorithm employed, grid generation and physical nature of the flow. One important aspect which is considered here is the convergence rate for steady flows.

Early efforts on the development of efficient solvers concentrated on approximate factorization methods and methods of exploiting multigrid in some manner. Beam and Warming [1] introduced the approximate factorization method for the Euler and Navier-Stokes equations in 1976. Steger [2] used this algorithm in the well known flow solver ARC2D, which was further developed by Pulliam [3]. ARC2D is an approximately-factored implicit algorithm that uses centered spatial differences with artificial dissipation to solve the Navier-Stokes equations.

Multigrid acceleration techniques were extended to the Euler equations by Jameson [4] [5], and to the Navier-Stokes equations by Martinelli et al. [6]. Jameson's approach includes an explicit multi-stage iterative method, local time-stepping, and implicit residual smoothing [7]. Caughey et al. [8] [9] [10] extended multigrid to an approximately-factored implicit multi-block algorithm to solve the Euler equations. Chisholm [11] applied a multigrid technique to the ARC2D solver in 1997. Chisholm and Caughey found that multigrid accelerated the convergence to steady state approximately by factors of 4 and 3.2, respectively. Jespersen, et al. [12] added multigrid to OVERFLOW, a three-dimensional

multiblock Navier-Stokes solver. Their results show speed-ups by a factor of 4 when using multigrid.

When solving the Navier-Stokes equations over complex geometries, there are difficulties in generating smooth meshes. In order to overcome this obstacle, two basic approaches have been proposed. The first is to employ unstructured meshes as in methods described by Jameson [4]. The second approach involves the use of multiblock structured grids as in methods presented by Thompson [13]. This approach divides the flow-domain into sub-domains or blocks. In each of these blocks, a structured mesh can be generated independently. Nelson [14] [15] [16] developed a Navier-Stokes multiblock solver, known as TORNADO (University of TOronto multi-block NAvier-stokes two DimensiOnal solver). This code is an extension of ARC2D to multiblock grids. TORNADO is used to analyse high-lift flows over multi-element airfoils, which are necessary for take-off and landing (see [17]).

1.2 Project Definition and Objectives

The objective of this project is to extend Chisholm's multigrid methodology to the Spalart-Allmaras turbulence model (previously the Baldwin-Lomax model was used) and to the TORNADO multi-block flow solver to speed-up its convergence rate.

Chapter 2

Governing Equations and Numerical Method

The multigrid technique will be introduced in Chapter 3. First, the Navier-Stokes equations and a description of the numerical algorithm used in TORNADO are presented.

2.1 Thin-Layer Navier-Stokes Equations

The two-dimensional thin-layer Navier-Stokes equations written in generalized curvilinear coordinates are

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = \mathrm{Re}^{-1} \frac{\partial \hat{S}}{\partial \eta} \tag{2.1}$$

where

$$\hat{Q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \tag{2.2}$$

The convective flux vectors are

$$\hat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U - \xi_t p \end{bmatrix} \tag{2.3}$$

$$\hat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V - \eta_t p \end{bmatrix} \tag{2.4}$$

with

$$U = \xi_t + \xi_x u + \xi_y v, \quad V = \eta_t + \eta_x u + \eta_y v \tag{2.5}$$

the contravariant velocities.

The viscous flux vector is

$$\hat{S} = J^{-1} \begin{bmatrix} 0 \\ \eta_x m_1 + \eta_y m_2 \\ \eta_x m_2 + \eta_y m_3 \\ \eta_x(u m_1 + v m_3 + m_4) \\ +\eta_y(u m_2 + v m_3 + m_5) \end{bmatrix} \tag{2.6}$$

with

$$\begin{aligned} m_1 &= (\mu + \mu_t)(4\eta_x u_\eta - 2\eta_y v_\eta)/3 \\ m_2 &= (\mu + \mu_t)(\eta_y u_\eta + \eta_x v_\eta) \\ m_3 &= (\mu + \mu_t)(-2\eta_x u_\eta + 4\eta_y v_\eta)/3 \\ m_4 &= (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\eta_x \partial_\eta(a^2) \\ m_5 &= (\mu \mathcal{P}r^{-1} + \mu_t \mathcal{P}r_t^{-1})(\gamma - 1)^{-1}\eta_y \partial_\eta(a^2) \end{aligned} \tag{2.7}$$

Pressure is related to the conservative flow variables, $\hat{Q}$, by the equation of state for a perfect gas, as follows

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{2.8}$$

The variable J represents the metric Jacobian of the transformation:

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi) \tag{2.9}$$

2.2 Numerical Algorithm

The first step in solving the set of equations 2.1-2.9 is the spatial discretization of the first derivative operators ∂_ξ and ∂_η, which are approximated using second-order central

differences. The second and fourth-difference dissipation model described in [3] is added to maintain stability and to prevent oscillations at shocks. The first-order implicit Euler time-marching method is employed to advance to steady-state. Consider Eq. 2.1, and apply the first order implicit Euler method to obtain

$$\hat{Q}^{n+1} - \hat{Q}^n + h\left(\partial_\xi \hat{E}^{n+1} + \partial_\eta \hat{F}^{n+1} - \mathrm{Re}^{-1}\partial_\eta \hat{S}^{n+1}\right) = 0 \tag{2.10}$$

where h is the time-step. The flux vectors $\hat{E}$, $\hat{F}$, $\hat{S}$ are linearized in time by a Taylor series such that

$$\begin{aligned} \hat{E}^{n+1} &= \hat{E}^n + \hat{A}^n \Delta\hat{Q}^n + O(h^2) \\ \hat{F}^{n+1} &= \hat{F}^n + \hat{B}^n \Delta\hat{Q}^n + O(h^2) \\ Re^{-1}\hat{S}^{n+1} &= Re^{-1}\left[\hat{S}^n + M^n \Delta\hat{Q}^n\right] + O(h^2) \end{aligned} \tag{2.11}$$

where $\Delta\hat{Q} = \hat{Q}^{n+1} - \hat{Q}^n$, and the matrices $\hat{A}$, $\hat{B}$, and $\hat{M}$ are the flux Jacobians, defined by

$$\hat{A} = \frac{\partial \hat{E}}{\partial \hat{Q}}, \; \hat{B} = \frac{\partial \hat{F}}{\partial \hat{Q}}, \text{ and } \hat{M} = \frac{\partial \hat{S}}{\partial \hat{Q}}.$$

The Delta form of the algorithm is obtained by applying Eqs. 2.11 to Eq. 2.10 and combining $\Delta\hat{Q}^n$ terms.

$$\begin{aligned} &\left[I + h\partial_\xi \hat{A}^n + h\partial_\eta \hat{B}^n - Re^{-1}h\partial_\eta \hat{M}\right]\Delta\hat{Q}^n = \\ &\quad -h\left(\partial_\xi \hat{E}^n + \partial_\eta \hat{F}^n - Re^{-1}\partial_\eta \hat{S}^n\right) \end{aligned} \tag{2.12}$$

The left and right hand sides of Eq. 2.12 are known as the "implicit" and "explicit" sides of the algorithm respectively.

In order to make the integration less expensive, the process is simplified by introducing an approximate factorization into the flux Jacobians. This reduces the "implicit" side to two block pentadiagonal implicit operators. The "implicit" side of Eq. 2.12, ignoring terms of second order, can be written as

$$\begin{aligned} &\left[I + h\partial_\xi \hat{A}^n + h\partial_\eta \hat{B}^n - Re^{-1}h\partial_\eta \hat{M}\right]\Delta\hat{Q}^n = \\ &\quad \left[I + h\partial_\xi \hat{A}^n\right]\left[I + h\partial_\eta \hat{B}^n - Re^{-1}h\partial_\eta \hat{M}\right]\Delta\hat{Q}^n + O(h^2) \end{aligned} \tag{2.13}$$

The resulting factored form of Eq. 2.12 is

$$\left[I + h\partial_\xi \hat{A}^n\right]\left[I + h\partial_\eta \hat{B}^n - Re^{-1}h\partial_\eta \hat{M}\right]\Delta\hat{Q}^n = \hat{R}^n \tag{2.14}$$

where

$$\hat{R}^n = -h\left(\partial_\xi \hat{E}^n + \partial_\eta \hat{F}^n - Re^{-1}\partial_\eta \hat{S}^n\right) \tag{2.15}$$

The computational work required for the inversion of the "implicit" side can be further reduced by introducing a diagonalization of the implicit blocks. The flux Jacobians $\hat{A}$ and $\hat{B}$ can be diagonalized as

$$\begin{aligned} \Lambda_\xi &= T_\xi^{-1}\hat{A}^n T_\xi \\ \Lambda_\eta &= T_\eta^{-1}\hat{B}^n T_\eta \end{aligned} \tag{2.16}$$

where T_ξ and T_η are matrices whose columns are the eigenvectors of the flux Jacobians $\hat{A}$ and $\hat{B}$ respectively. Applying Eqs. 2.16 to the factored form Eq. 2.14 gives

$$\left[T_\xi T_\xi^{-1} + h\partial_\xi\left(T_\xi \Lambda_\xi T_\xi^{-1}\right)\right]\left[T_\xi T_\xi^{-1} + h\partial_\xi\left(T_\xi \Lambda_\xi T_\xi^{-1}\right)\right] = \hat{R}^n \tag{2.17}$$

where $\hat{R}^n$ is the right hand side of Eq. 2.14. The following equation results by factoring the T_ξ and T_η eigenvector matrices outside the spatial derivatives ∂_ξ and ∂_η.

$$T_\xi\left[I + h\partial_\xi \Lambda_\xi\right] N \left[I + h\partial_\eta \Lambda_\eta\right] T_\eta^{-1}\Delta\hat{Q}^n = \hat{R}^n \tag{2.18}$$

where $N = T_\xi^{-1}T_\eta$. The viscous flux Jacobian $\hat{M}$ cannot be diagonalized with $\hat{B}$. An approximation to the viscous Jacobian eigenvalues is introduced to the left hand side of Eq. 2.18 (see [3]).

The rate of convergence can also be increased by using local time stepping. This is obtained by scaling the time step with the Jacobian of the coordinate transformation.

$$h = \frac{h_{ref}}{1 + \sqrt[2]{J}} \tag{2.19}$$

2.3 Boundary Conditions

In a multiblock solver, several types of boundary conditions are considered: far-field boundaries, wall boundaries and interface boundaries between blocks. Figure. 2.1 shows the transformation of a 3-block C mesh from the physical domain to the computational domain, where block interfaces are treated in a special manner, using a column of points from the neighbouring block as explicit boundaries.

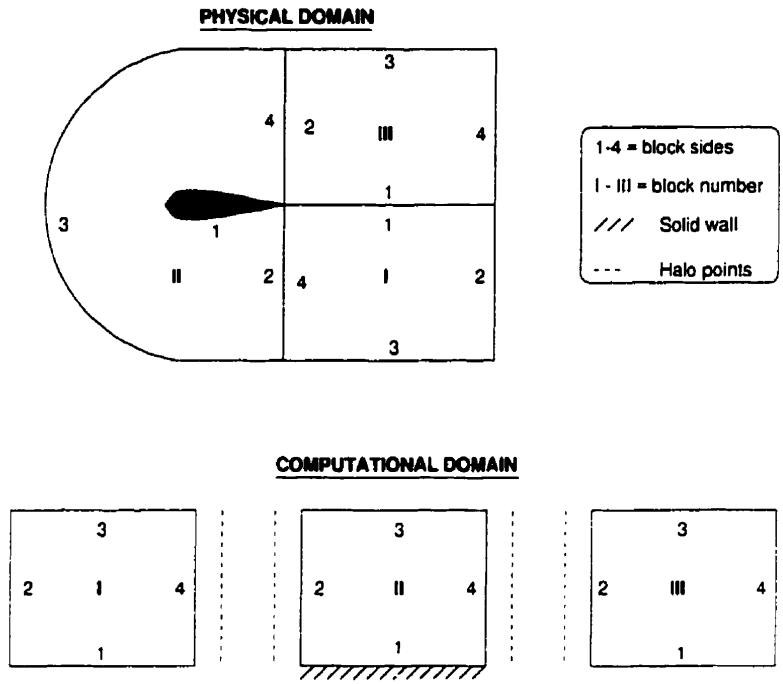


Figure 2.1: 3-Block C-Mesh

2.3.1 Far-field Boundary Conditions

For lifting airfoils, the far-field boundary may affect the solution, unless it is placed far away, which would require more nodes. In ARC2D and TORNADO, the far-field boundary conditions are solved explicitly with Riemann invariants. A circulation correction, as described by Pulliam [3], is applied in order to minimize the effect of the far-field boundary.

2.3.2 Wall Boundary Conditions

For inviscid flows, tangency at the body is required ($V_n = 0$). The tangential component is extrapolated linearly. Pressure at the surface is obtained through a first-order extrapolation, whereas density is determined from the equation of stagnation enthalpy, which is held constant at the surface.

$$H_0 = \frac{e+p}{\rho} \tag{2.20}$$

For viscous flows, the tangential and normal components of velocity are set to zero. Pressure and density are obtained by a zeroth-order extrapolation.

2.3.3 Interface Conditions

In TORNADO, the interface conditions are updated depending on the nature of the block side. If the block side is in the η direction, they are obtained by overlapping neighbouring blocks at the interface, and an extra column of points taken from the neighbouring block (known as the "halo" column) provides explicit boundary conditions. If the block side is in the ξ direction, the interface is updated through a linear extrapolation of the interior nodes.

In order to explain the sequence of events when updating the interface conditions using "halo" columns, consider a rectangular 2-block grid (see Figure 2.2). The first interior column of block 2 is stored in the "halo" column of block 1, and the last interior column of block 1 is stored in the "halo" column of block 2. To advance a time-step, blocks 1 and 2 are updated independently of one another, which causes two solutions at their interface. The average of these two solutions is stored at the interface and the process is repeated until the residual is driven to machine zero. The interface is completely transparent at steady state.

Consider now a case in which no "halo" points are used to update the boundaries. Once the extrapolation from the interior points of both blocks has been performed, the interface will store the average of the solutions obtained from both blocks. With this approach the interface is not transparent, even at steady state, and some degradation of accuracy will occur.

2.4 Treatment of Singular Points

In a multiblock solver, nodes which are shared by more than two blocks require special consideration. These nodes, such as the leading edge or the trailing edge of an airfoil can greatly affect the convergence of the solver, depending on how they are treated.

2.4.1 Treatment of the Leading Edge

The leading edge of the airfoil, as shown in Fig. 2.3 is shared by three or four blocks, depending on the type of mesh (3 blocks for a C mesh, 4 blocks for an H mesh). From Fig. 2.3, it can be seen that $(J3, K3)$ and $(J4, K4)$ are copied to points $(J2, K2)$ and $(J1, K1)$ respectively. For viscous cases, the average of leading edge point on blocks 1 and 2 is stored in $(J1, K1)$ and $(J2, K2)$

2.4.2 Treatment of the Trailing Edge

The trailing edge of the body is handled in a manner similar to that of the leading edge. As presented in Fig. 2.4, the trailing edge points of the block having the body surface in one of its sides are copied to the neighbouring block. The average of points $(J3, K3)$ and $(J4, K4)$ is copied to $(J1, K1)$, $(J2, K2)$, $(J3, K3)$ and $(J4, K4)$.

In the case of a blunt trailing edge, two copies are made to the blocks behind the airfoil as shown in Fig. 2.5

2.5 Spalart-Allmaras Turbulence Model and the Multigrid Method

The Spalart-Allmaras turbulence model [18] is a widely used one-equation model (approximately-factored implicit algorithm). The turbulence model is local (the equation at one point does not depend on the solution at other points), and yields fairly rapid convergence to steady-state. It is well suited for the phenomena that occur in the multi-element case, such as flow separation and confluent boundary layers.

The turbulence model and the flow solver are loosely coupled: take one step of the turbulence model, take one step of the flow solver. The same approach is used with the multigrid algorithm: one step of the turbulence model, one step of the multigrid algorithm.

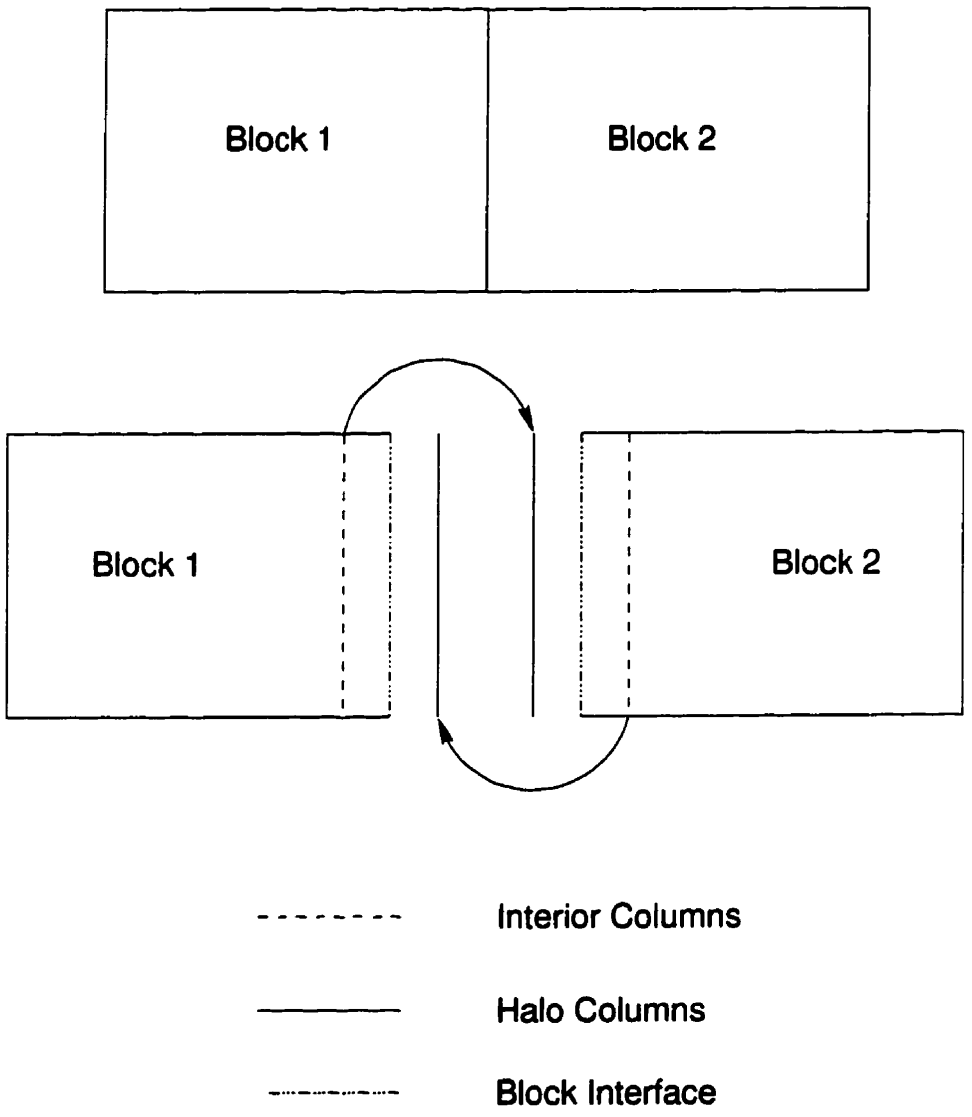


Figure 2.2: 2-Block Grid with Halo Data

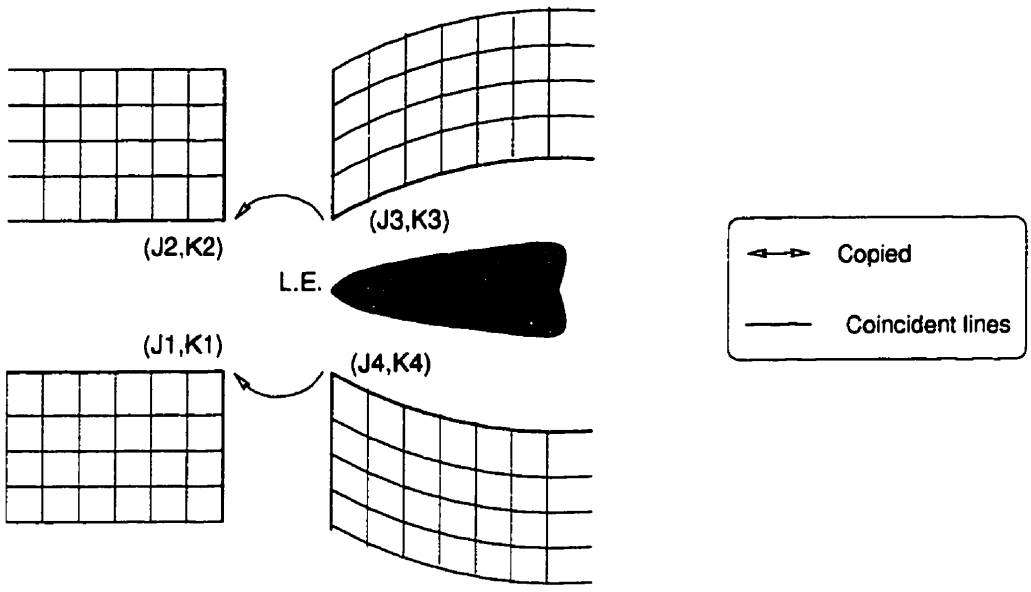


Figure 2.3: Leading Edge Treatment

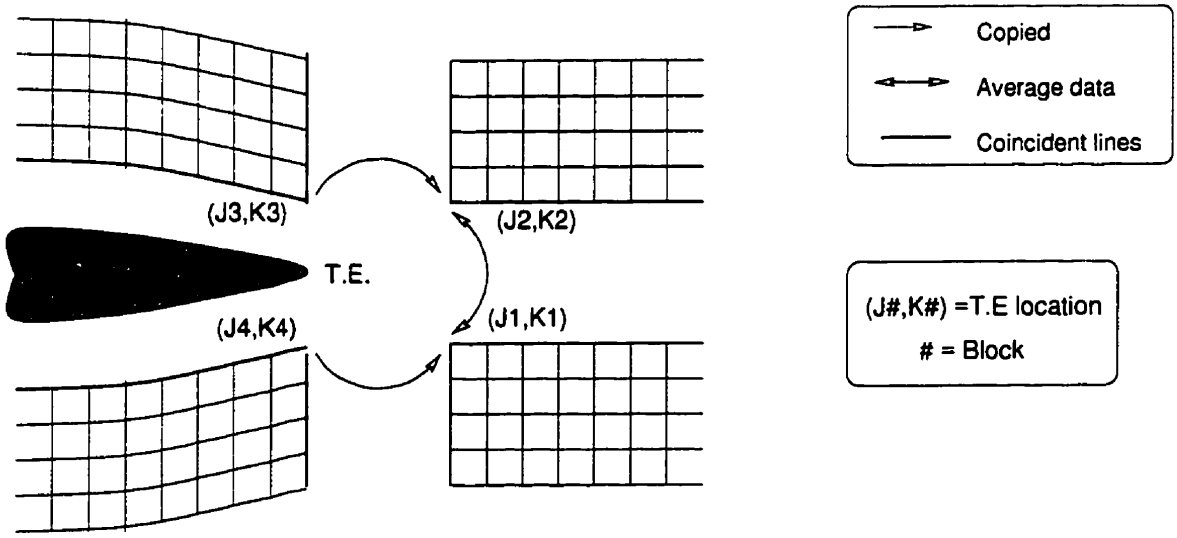


Figure 2.4: Trailing Edge Treatment

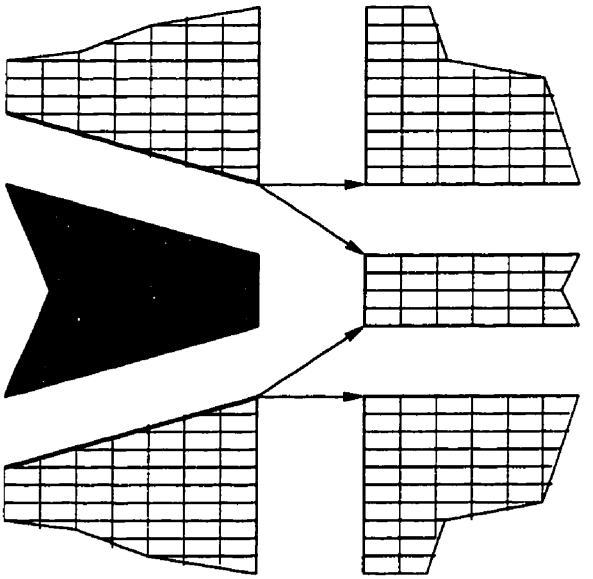


Figure 2.5: Blunt Trailing Edge Treatment

The turbulence model is only solved in the fine grid of the multigrid cycle, no steps are taken in the coarse grids. Necessary turbulent quantities, such as turbulent viscosity, are passed to the coarse grids to complete the multigrid cycle.

Chapter 3

Multigrid

3.1 The Multigrid Concept

Multigrid methods are used to accelerate the numerical solution of partial differential equations, such as the compressible Navier-Stokes equations. The essence of multigrid methods is to use a family of discrete problems, defined in several grids, all approximating the same continuous problem, to help in the solution of the large systems of equations [19] [20]. The main purpose of the multigrid technique is to accelerate the solution by removing low frequency errors. It was originally developed for elliptic differential equations. Since then it has been applied to other type of systems, such as the Navier-Stokes equations.

When an iterative solver is applied to the spatially discretized equations, the residual is reduced until the steady solution is obtained. Solvers tend to reduce the error differently depending on its frequency. They are normally much better at reducing high frequency errors. Consider an error of one frequency of a one-dimensional grid with uniform grid spacing. If the error is passed to a coarser grid formed by the removal of every other node of the finer grid, its frequency is doubled. Hence, the iterative solver will remove this error more quickly since it is of higher frequency. This is illustrated in Fig. 3.1, where a wave with 16 points per wavelength on the finest grid has a higher frequency (4 points per wavelength) on the coarsest grid. This coarse grids are formed by the removal of every other node. Hence, the frequency has effectively been raised. The objective of the multigrid technique is to reduce the low frequency error by using several grids to remove this type of error faster than the conventional iterative solvers do by the use of one grid only. Thus, the effectiveness of the multigrid technique depends largely on the capacity of the smoother to remove the

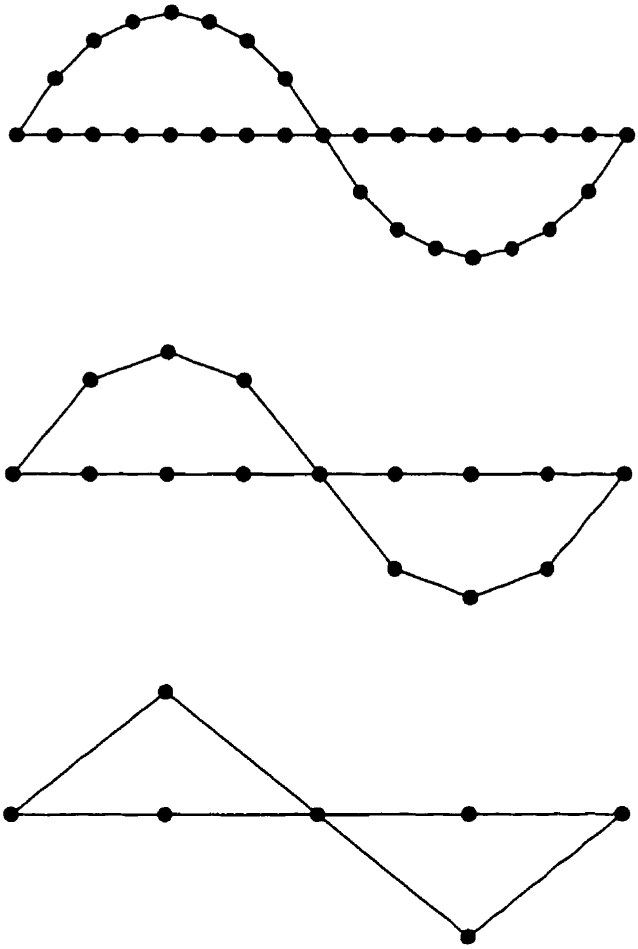


Figure 3.1: 1D Illustration of Multigrid

high frequency error. For further explanation on multigrid techniques and equations, please refer to Jameson [7]. See [11, 12] for multigrid applied to ARC2D.

3.2 Multigrid Cycles

The basis of the multigrid process applied to the Navier-Stokes equations can be found in [21]. The basic idea of the multigrid method is to use coarser grids to accelerate the convergence rate.

The multigrid method used in TORNADO is based on Chisholm's work [11] on ARC2D. It is important to explain a multigrid cycle before describing how the transfer

between grids is performed. The diagonalized equation 2.18 can be written in a simple form as

$$A\Delta Q = RHS(Q) \tag{3.1}$$

A and RHS represent the left hand side matrix and residual vector of eq. 2.18 respectively. (For the remainder of this explanation, the " ^ " will be dropped for simplicity.)

A full approximation storage multigrid is required when solving non-linear systems of equations, such as the Navier-Stokes equations. Two vectors, Q and the RHS must be transferred to a coarse grid to complete a multigrid cycle. The transfer of vectors Q or RHS is known as restriction, whereas the transfer from a coarse level to a fine level is called prolongation. Consider a 2-level cycle as shown in figure 3.2. This is the simplest type of multigrid cycle. Note that vector Q represents the conservative variables, vectors S and R are the residual vectors (these variables are used to store the RHS). S is used as the driving residual, whereas R stores the residual before and after the restriction to the coarse grid occurs. Superscipts denote the current state in the cycle. Subscripts indicate the level of the cycle. Restriction and prolongation transfers will be explained in detail later. To complete a cycle, the following steps are required:

1. Store initial fine-grid Q vector in Q_1^0

2. Form the right hand side from Q_1^0 and store it in S_1^0

$$S_1^0 = RHS\left(Q_1^0\right)$$

3. Take a time-step on the fine-grid and store Q in Q_1^i

$$Q_1^i = SMOOTH\left(Q_1^0, S_1^0\right)$$

4. Restrict Q_1^i to coarse grid and store it in Q_2^0

$$Q_2^0 = RESTRCT\left(Q_1^i\right)$$

5. Form the fine-grid residual from Q_1^i and store it in R_1^i

$$R_1^i = RHS\left(Q_1^i\right)$$

6. Restrict R_1^i to the coarse grid and store it in R_2^r

$$R_2^r = RESTRCT\left(R_1^i\right)$$

7. Calculate the initial residual on the coarse grid

$$R_2^0 = RHS\left(Q_2^0\right)$$

8. Form the forcing function (fine-grid correction)

$$P = R_2^r - R_2^0$$

9. Form the residual on the coarse grid. Note that is composed of the RHS and the forcing function

$$S_2^0 = R_2^0 + P$$

10. Take a time-step in the coarse grid and store solution in Q_2^+

$$Q_2^+ = SMOOTH\left(Q_2^0, S_2^0\right)$$

11. Make the correction of the fine level Q and store it in Q_1^+. First, the difference between the initial and final Q vectors on the coarse level must be prolonged. This step completes the multigrid cycle.

$$Q_1^+ = Q_1^i + PLNG\left(Q_2^+ - Q_2^0\right)$$

On the coarse level, the "forcing function" (P) is added to the right hand side in order to impose the fine-grid approximation. Note that if the number of smoothing passes or time-steps taken in the coarse level is only one, the coarse driving residual S_2^0 is equal to R_2^r, the restricted residual from the fine grid. This can be noticed by combining steps 5-9.

Several types of cycles are used in multigrid algorithms: Sawtooth cycles, V cycles and W cycles. These are illustrated in Figure 3.3. When performing a sawtooth cycle, a time-step is taken in the finest grid and after the restriction occurs. No time-steps are taken after correcting the solution from a coarse grid to a fine one. The V cycle differs from the sawtooth cycle in that time-steps are taken after each correction. The theoretically

best cycle is the W-cycle, where more time is spent on the coarse levels. The coarse grid corrections are smoothed before they are transferred back to the finest level.

Grid sequencing is also widely used to accelerate the convergence of the solver. In aerodynamic flows, the conservative variables are initialized to the free stream conditions. Grid sequencing provides a better initial guess. Several iterations are taken in a coarse level and then the Q vector is transferred to the desired grid. The combination of this technique with multigrid is known as full-multigrid. Figure 3.4 shows a 3-level full-multigrid, in which some iterations are taken in the coarse grids before the solution is transferred to the finest grid. In levels 2 and 1, the multigrid algorithm is applied.

A large part of the time required to perform a multigrid cycle is spent calculating the RHS (it is calculated twice to form the forcing function), and performing the restriction and prolongation transfers (see [11]). One way to reduce the number of RHS calculations and transfers is to perform several iterations in each multigrid level. As a result, the multigrid cycle becomes more efficient, and the CPU time required to converge to machine zero is reduced significantly. Some caution should be taken: performing too many smooths per level can reduce the effectiveness of the multigrid cycle. In TORNADO, the optimal number of smooths per level is 4.

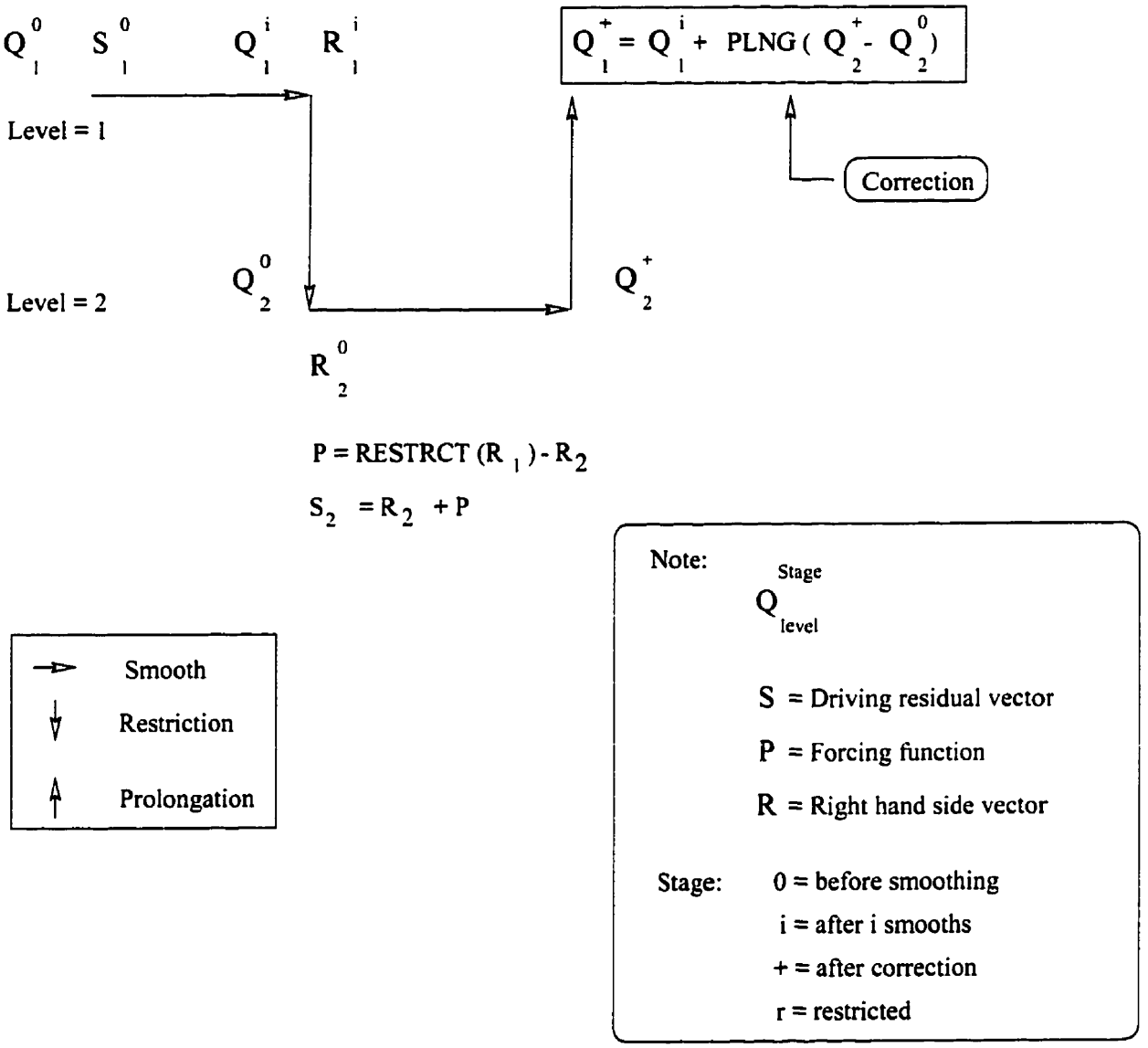
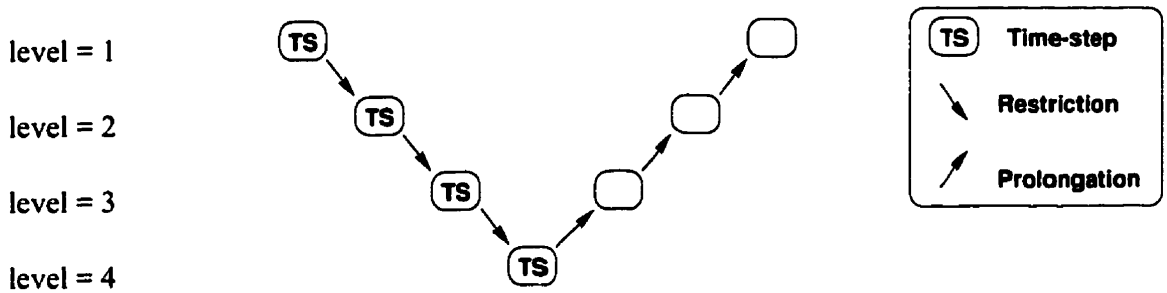
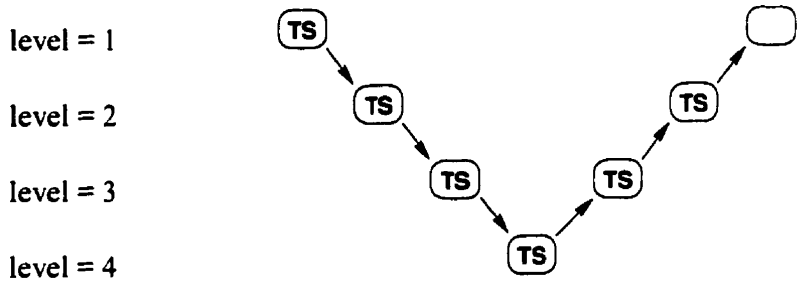


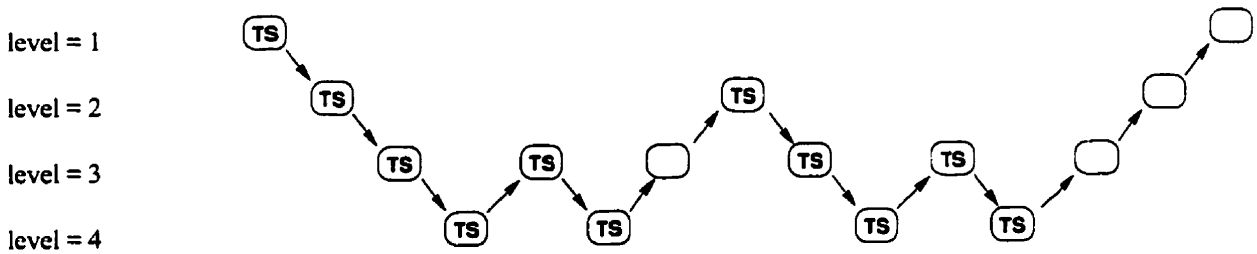
Figure 3.2: 2 Level Sawtooth Cycle



(a) Four-level Sawtooth Cycle



(b) Four-level V cycle



(c) Four-level W cycle

Figure 3.3: Types of Cycles

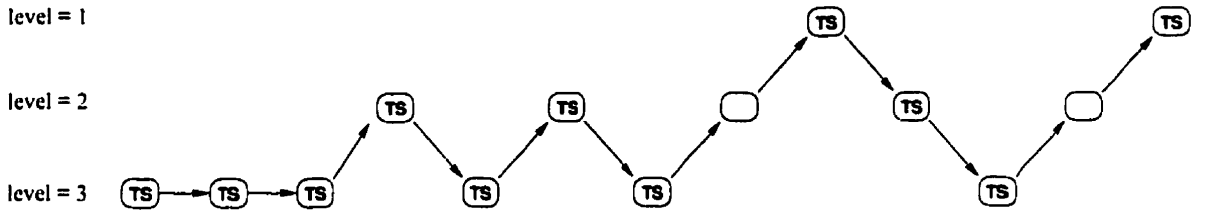


Figure 3.4: Full-multigrid

3.3 Restriction and Prolongation Transfers

3.3.1 Restriction

The transfer of a vector from a fine grid to a coarse grid is known as restriction. There are several types of restriction. The simplest is called simple injection, which results from the removal of every other grid line in the x and y directions. Other restrictions involve a weighted average, such as the weighted restriction shown in Fig. 3.5

The restriction of Q and the residual is done without the Jacobian scaling. The Jacobian of the transformation is quite different from one grid level to a coarser one. The restriction of the solution vector is done by simple injection, whereas the residual is restricted using the weighted restriction presented in Fig. 3.5.

3.3.2 Prolongation

Prolongation is used to transfer the solution vector Q from a coarse level to a fine level. The Q vector can be prolonged in either the computational or the physical domain. The approach taken here is to perform the transfer in the computational domain, as shown in Fig. 3.6.

3.4 Multigrid and Multiblock

Some aspects must be considered when applying the multigrid technique to a multiblock solver. There are two approaches that can be taken: local multigrid and global multigrid [9]. In the first method, also known as the vertical mode, the multigrid cycle

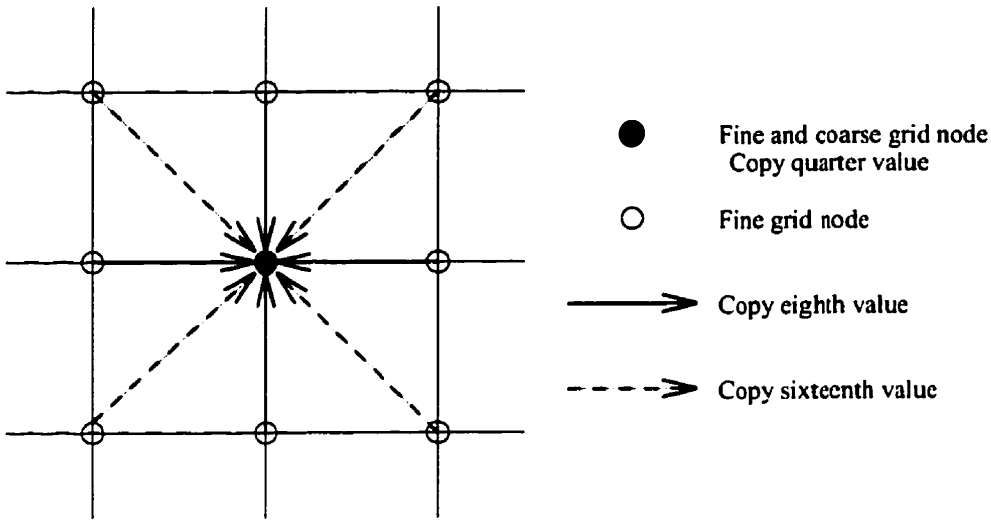


Figure 3.5: Weighted Restriction

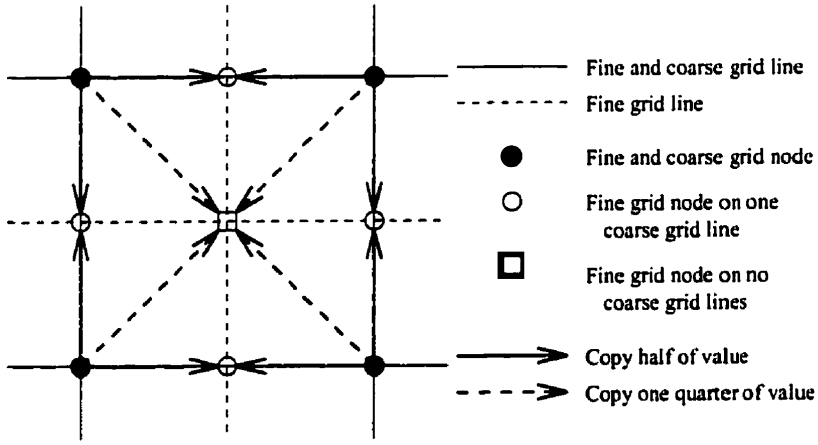


Figure 3.6: Prolongation

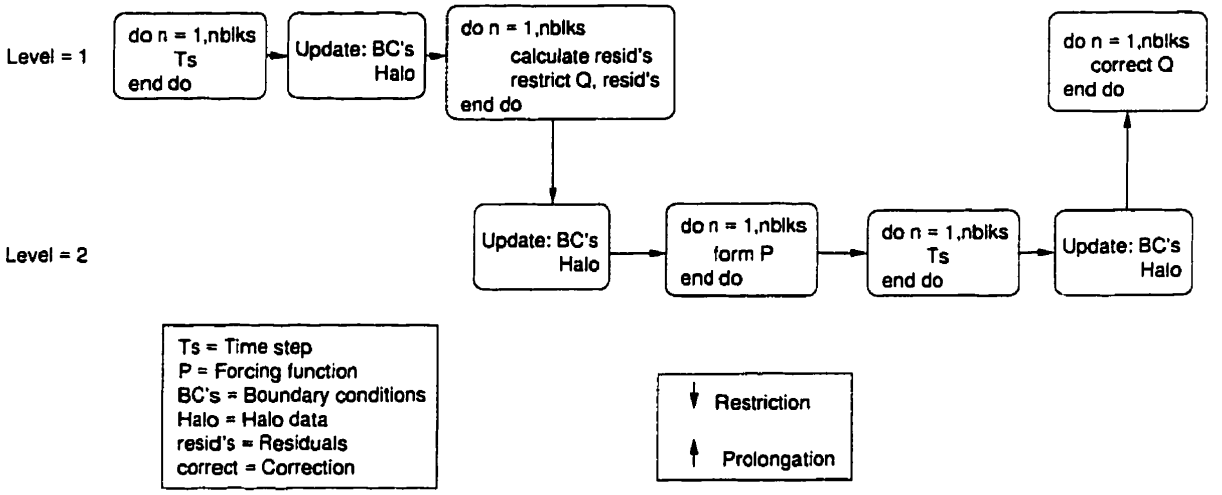


Figure 3.7: Global Multigrid (2 Level Cycle)

advances in each block independently. Data exchange between blocks only occurs in the finest level, and halo information is updated once all blocks have undergone a multigrid cycle. In global multigrid, also known as horizontal mode, there is comunication between blocks in all multigrid levels, not only on the finest. This allows transparency for wave propagation through boundary interfaces in all levels of the multigrid process. The approach taken in this paper is global multigrid, which has been proven by other researchers, such as Caughey [9], to be the more effective of the two modes. Fig. 3.7 illustrates the sequence to complete a 2-level sawtooth cycle. It is important to update boundary conditions and halo data after every smooth, restriction or prolongation.

Chapter 4

Results

This chapter is divided into validation test cases and high-lift applications. Inviscid cases are studied to validate the multigrid algorithm for a single-element airfoil. The multigrid-multiblock algorithm is also compared to the single block solver SC1 and with multigrid. In the high-lift section, the cases considered include a fully-attached flow over the NLR-7301 airfoil and flap, and case A2 from the AGARD Advisory Report No.303. A Silicon Graphics Origin 200 machine was used to run all test cases. The flags used to compile the code were O3 and mips4. The effectiveness of the sawtooth, V and W cycles is studied. In all test cases, four iterations are performed on all multigrid levels.

4.1 Validation Test Cases

4.1.1 Test Case 1: Comparison of Single- and Multi-block Multigrid

The first test case is an inviscid flow with a symmetrical airfoil (NACA0012). A 3-block grid, created using an elliptical grid generator [22], was employed. A plot of the grid is shown in Fig. 4.1. The blocks are 49×49, 129×49 and 49×49. Points are clustered near the leading edge and trailing edge of the airfoil, with arclength spacings of 0.001 chords. The off-wall spacing is 5×10^{-4} chords, and the distance to the outer boundary is 12 chords.

The test case studied was for a Mach number equal to 0.63 and and angle of attack of 2 degrees. A comparison of the convergence to steady state was studied for this test case between the multiblock solver and SC1 [23], a variation of ARC2D developed at UTIAS. The diagonal algorithm was used with second and fourth-order nonlinear dissipation and a

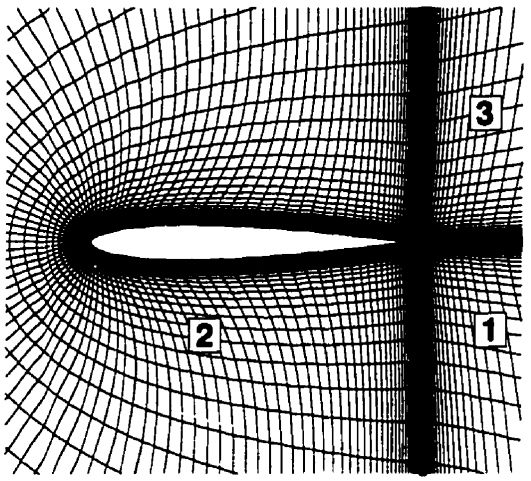


Figure 4.1: Close-up of 3-Block Grid (Case 1)

3 level multigrid sawtooth cycle. Local time-stepping was employed in both solvers. The reference time-step was 5. As shown in Fig. 4.2, the convergence to steady state of the multiblock solver is not very different from the convergence of the SC1 solver. The single block solver requires 62 cycles to converge to 1 x 10^{-13}, whereas the multiblock solver needs only four more cycles. The slight increase in the number of cycles performed is due to the effect of block interfaces.

4.1.2 Test Case 2: Inviscid Flow on a 3-Block Grid

In this case, the multigrid algorithm is studied further for test case 1. The use of the different multigrid cycles and full-multigrid, explained in earlier sections, is studied. The convergence histories of 2 and 3-level multigrid sawtooth cycles are presented in Fig. 4.3. It can be observed that the multigrid algorithm speeds up convergence approximately by a factor of 3. Figures 4.4 and 4.5 show the lift and drag residual histories respectively obtained using the 3 level sawtooth cycle. The C_l converges faster by also a factor of three. The effect of full-multigrid was studied by comparing a 3-level sawtooth cycle to a full-multigrid cycle. As seen in Fig. 4.6, full-multigrid did not significantly accelerate the

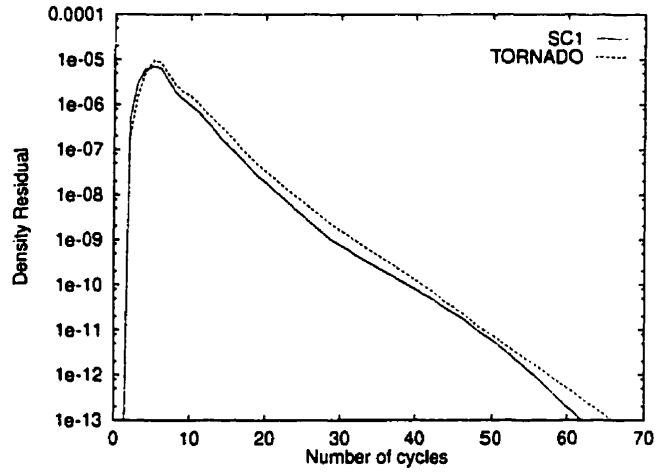


Figure 4.2: 3-level Sawtooth Cycle (Single block (SC1) vs Multiblock (TORNADO))

convergence. The convergence histories obtained by employing the sawtooth, V, and W cycles with three levels and four smoothings per level are presented in Fig. 4.7 and Fig. 4.8. For this particular case, there is no advantage in using the W cycle over the sawtooth or V cycles. The latter two converge to machine zero within the same time. The W cycle is not as effective as one could expect due to the coarse grid employed.

Four-level multigrid was also investigated, and did not converge faster than three-level multigrid. More than four subiterations per level were required at all levels to avoid instability of the solver.

4.1.3 Test Case 3: Inviscid Flow on an H-Mesh

This test case was conducted to validate the code for an H-mesh over a NACA 0012 airfoil at a Mach number equal to 0.63 and angle of attack of 2 degrees. The difference between this case and the previous one is the treatment of the leading edge. The grid was generated using the multiblock generator [22], and has 6 blocks. The corner blocks are 49×49 nodes and the two remaining with sides on the lower and upper surface of the airfoil are 65×49 nodes. The off-wall spacing is 1×10^{-4} chords, and clustering at the leading and trailing edge is 0.001 chords. The distance to the outer boundary is 10 chords. A close-up of the grid is shown in Fig. 4.9, obtained by removing every other line in each direction.

The benefits of using the multigrid algorithm with a sawtooth cycle are shown in

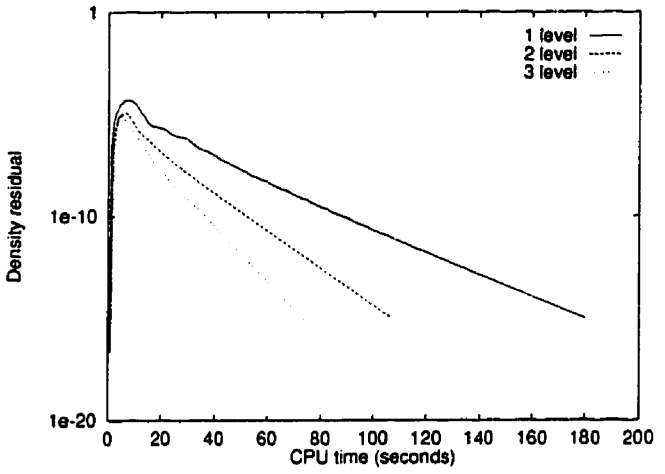


Figure 4.3: Residual History of Case 2 with Different Sawtooth Cycles

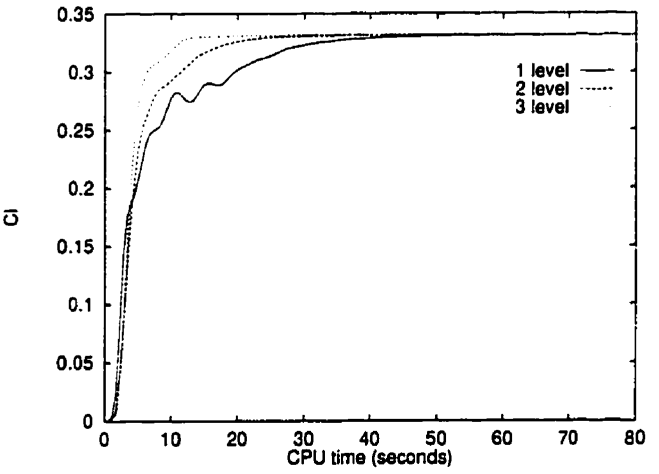


Figure 4.4: Lift Convergence of Case 2 Using 2 and 3 Level Sawtooth Cycles

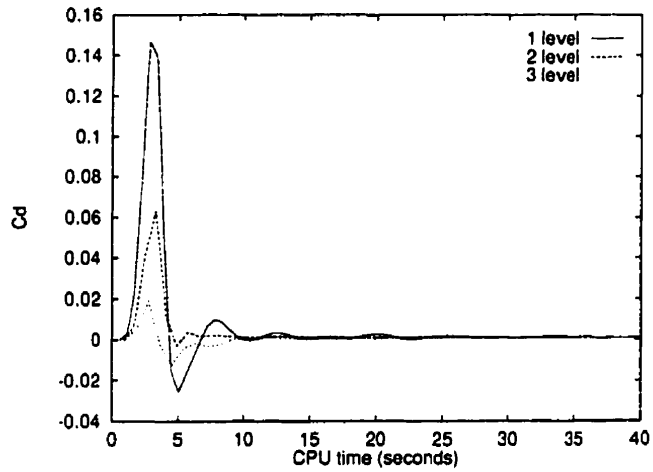


Figure 4.5: Drag Convergence of Case 2 with Different Sawtooth Cycles

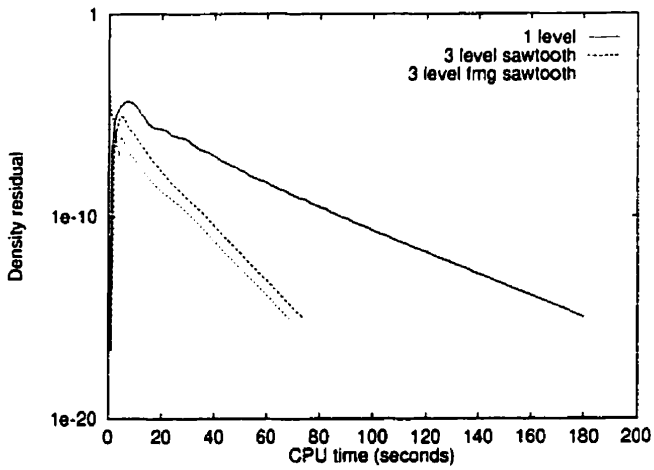


Figure 4.6: Residual History of Case 2 with Full-Multigrid

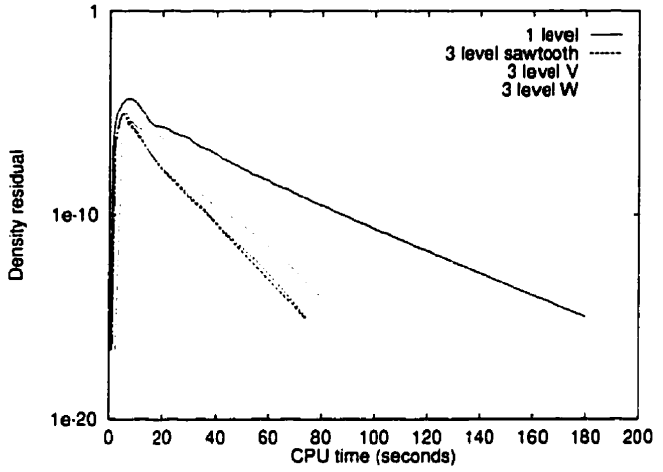


Figure 4.7: Residual History of Case 2 with Sawtooth, V and W Cycles

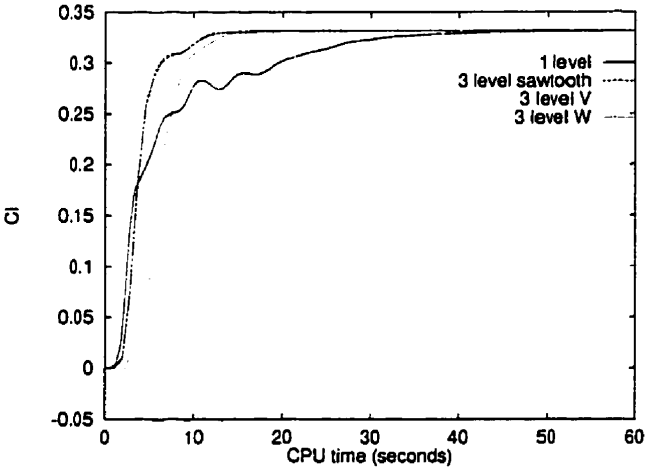


Figure 4.8: Lift Convergence of Case 2 with Sawtooth, V and W Cycles

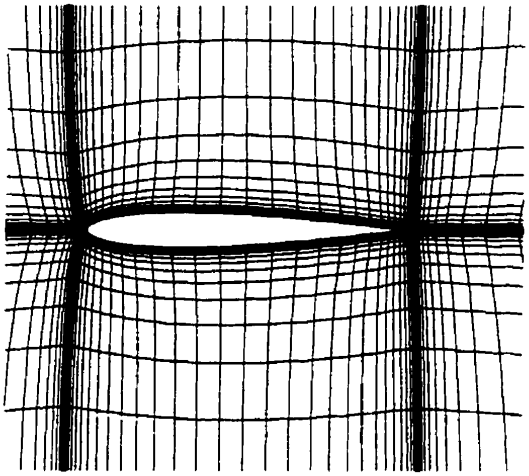


Figure 4.9: Close-up of Case 2 Grid

Fig. 4.10. There is a great speed-up, approximately a factor of 6, when using multigrid. The C_l convergence rate also improves by a large factor, approximately 4 (see Fig. 4.11). As in the previous test case, the use of full-multigrid contributes to a shorter run time (see Fig. 4.12). In this case, the W cycle proved to be the most effective. As shown in Fig. 4.13, the W cycle had the best convergence rate, followed by the V cycle. In the V cycle, four iterations were performed after each correction was made. In the case of the C_l, the convergence of the W and V cycles is almost identical, as shown in Fig. 4.14.

4.2 High-Lift Applications

4.2.1 Test Case 4: NLR-7301 Airfoil and Flap

Tests were performed on the NLR-7301 airfoil and flap configuration previously studied using TORNADO [15], as shown in Fig. 4.15 (obtained by removing every other line in each direction), which has a 32% chord flap with an angle of 20 deg. The overlap is 5.3% chord and the nominal gap setting is 2.4% chord. The Mach number and the Reynolds number were 0.185 and 2.51×10^6 respectively. The angle of attack was set at 13.1 deg,

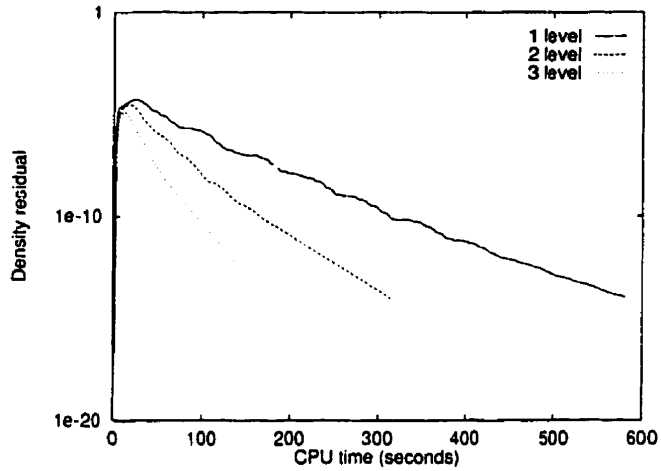


Figure 4.10: Residual History of Case 3 Using a Sawtooth Cycle

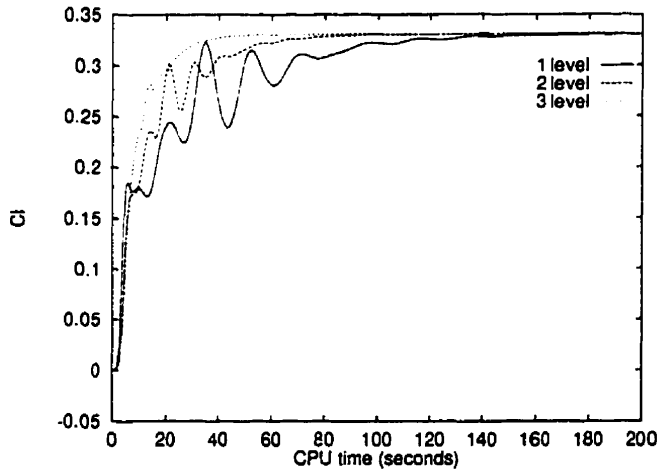


Figure 4.11: Lift Convergence of Case 3 Using a Sawtooth Cycle

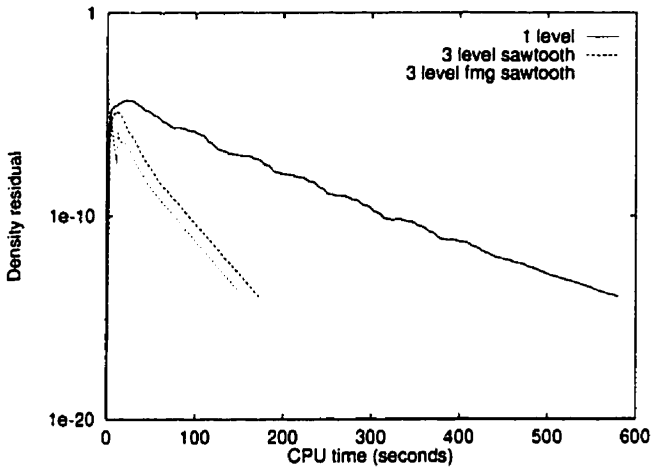


Figure 4.12: Residual History of Case 3 Using Full-Multigrid

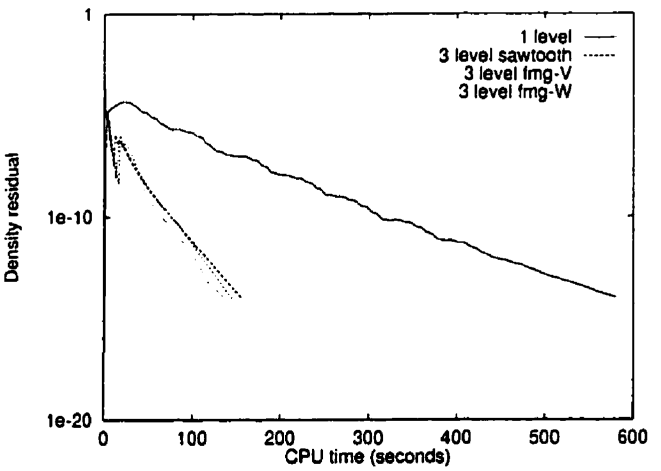


Figure 4.13: Residual History of Case 3 with Sawtooth, V and W Cycles

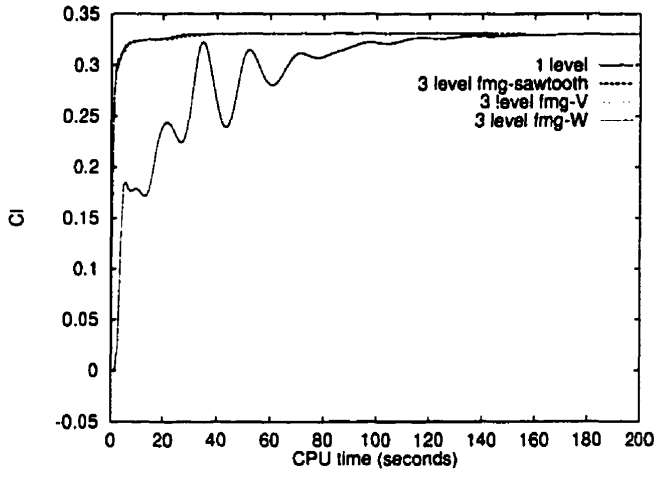


Figure 4.14: Lift Convergence of Case 3 with Sawtooth, V and W Cycles

which produces close to maximum lift. At this angle of attack, the flow is fully attached. The transition was fixed at 3.0% on the upper surface of the airfoil and at 78.0% on the lower surface. On the flap, transition was fixed at 58.8% and 95.0% on the upper and lower surfaces respectively. See tests conducted by Van den Berg in Ref. [24].

The grid used for this test case has a total of 91,409 nodes, and the off-wall spacing is 1×10^{-6} chord. Lines were clustered near leading and trailing edges of both surfaces, the main airfoil and the flap, accordingly. The distance to the outer boundary is 10 chords.

The residual and force convergence histories for a sawtooth cycle, using 2 and 3 levels, are presented in Figures 4.16-4.19. The speed-up factor in terms of the density residual and the C_l were approximately 2 and 3, respectively.

The sawtooth, V and W cycles were also compared, once the effectiveness of the multigrid algorithm, combined with the turbulence model was confirmed. In Fig. 4.20 and 4.21, it can be observed that the W cycle is the most effective in terms of the density residual and the coefficient of lift. The V cycle, which had four smooths per level after every restriction and prolongation, also accelerated the convergence to steady-state faster than the sawtooth cycle. The speed-up factor in terms of the C_l for the sawtooth, V and W cycles were approximately 3, 4 and 4.5 respectively.

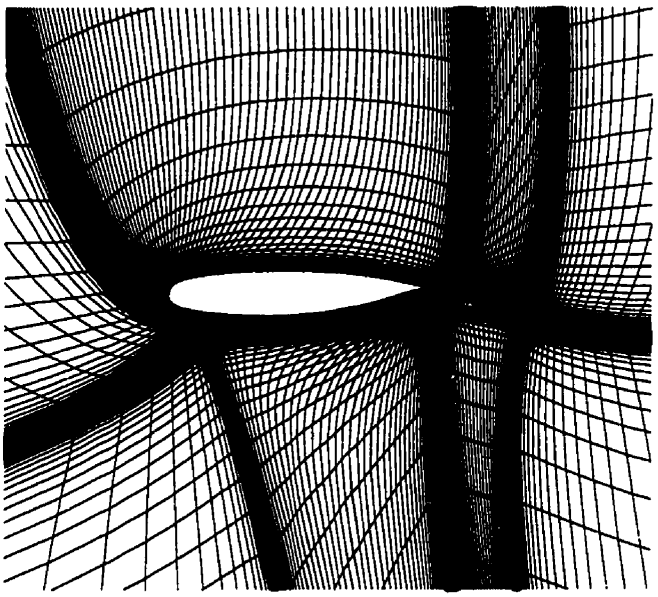


Figure 4.15: Close-up of the NLR-7301 Grid

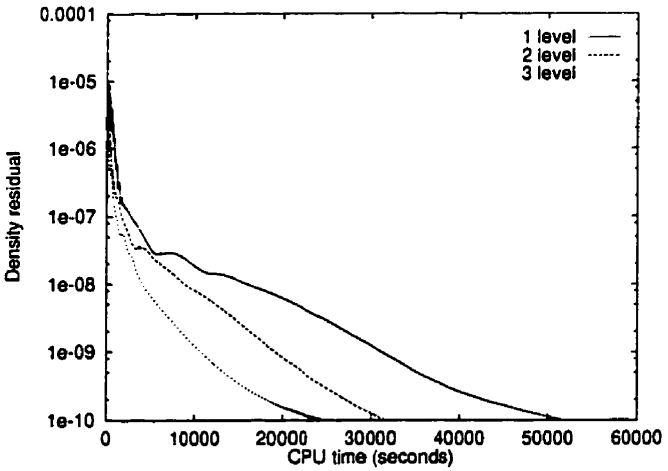


Figure 4.16: Residual History of the NLR-7301 Case with a Sawtooth Cycle

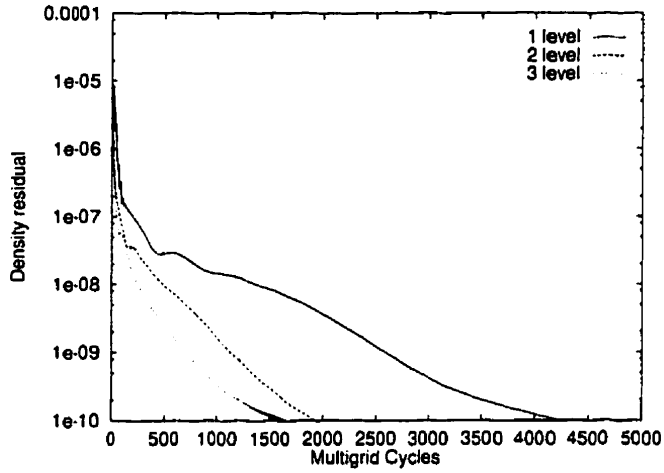


Figure 4.17: Residual History of the NLR-7301 Case with a Sawtooth Cycle (in Multigrid Cycles)

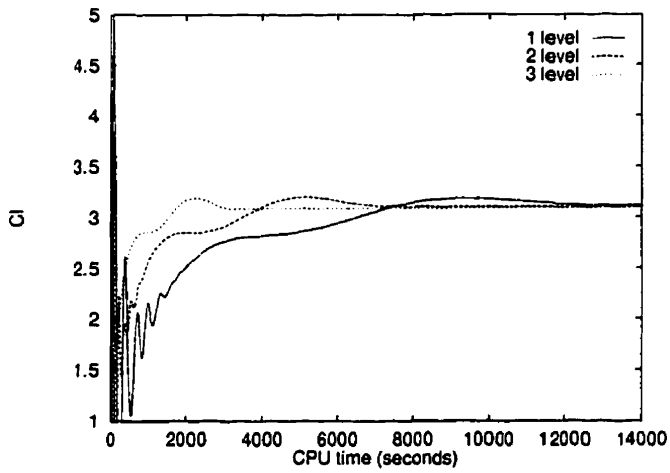


Figure 4.18: Lift Convergence of the NLR-7301 Case with a Sawtooth Cycle

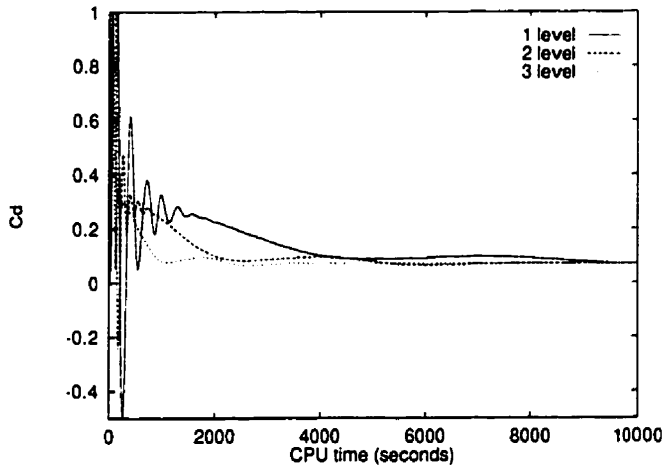


Figure 4.19: Drag Convergence of the NLR-7301 Case with a Sawtooth Cycle

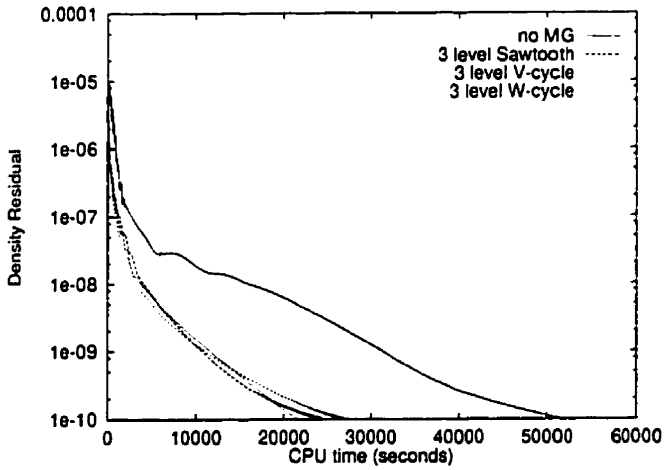


Figure 4.20: Residual History of the NLR-7301 Case with a Sawtooth,V and W Cycles

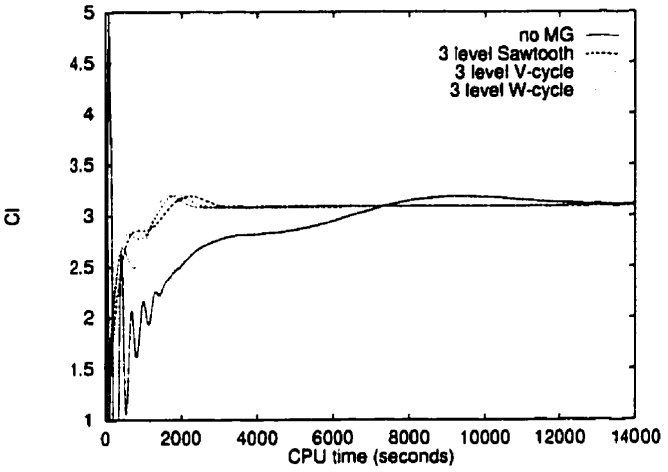


Figure 4.21: Lift Convergence of the NLR-7301 Case with a Sawtooth.V and W Cycles

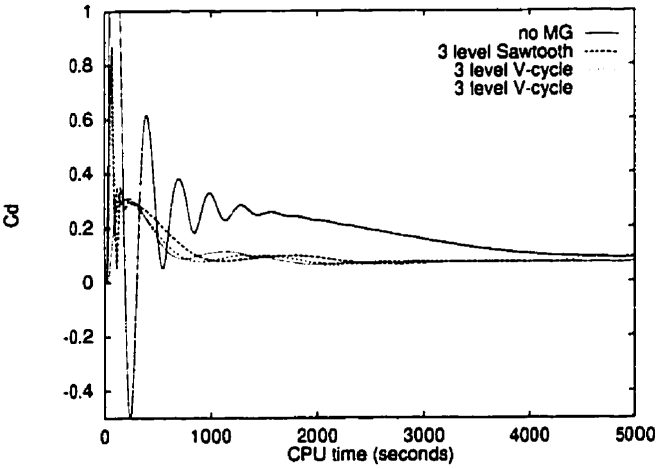


Figure 4.22: Drag Convergence of the NLR-7301 Case with a Sawtooth, V and W Cycles

α	*Slat*		*Airfoil*		*Flap*	
	Upper	Lower	Upper	Lower	Upper	Lower
4	100%	25%	0.3%	11.0%	0.7%	0.1%

Table 4.1: Specified transition locations (percent of element chord)

4.2.2 Test Case 5: Case A-2 from the Advisory Report No. 303

The grid employed in this test case is a C-H-mesh configuration with a total of 46 blocks (see Fig. 4.23, formed by the removal of every other grid line of the fine grid). An inner C-grid surrounds each airfoil element with thin blocks approximating the thickness of the boundary layer. The off-wall spacing is $1 \times 10^{-6}c$, but increases locally to $1 \times 10^{-5}c$ in the coves of the slat and the flap. The outer boundary is located 10 chords away from the airfoil. The grid has approximately 140,000 points. The Mach number and the Reynolds number for this test case were 0.2 and 3.52×10^6 respectively. The angle of attack is 4.0 deg. Transition points for the slat, main airfoil and flap are shown on the Table 4.1.

The residual, C_l, and C_d convergence histories are shown in Fig. 4.24-4.27 for a 3-level multigrid with a sawtooth cycle, which converges faster than the single-grid solver by a factor of approximately 3.5, using the C_l as the indicator.

The sawtooth cycle was also compared to the V and W cycles. The W cycle is the most effective in terms of the density residual and the coefficient of lift (see Fig. 4.28-4.31). As shown in Fig 4.30, it is approximately 5 times faster than the single-grid solver using the C_l as a reference.

The single-grid and multigrid solvers have difficulties converging this test case. There is a high residual at an interface grid point located behind the slat. This point, shared by 6 blocks, is updated by a zeroth-order extrapolation. A different method must be developed to eliminate the high residual at this point.

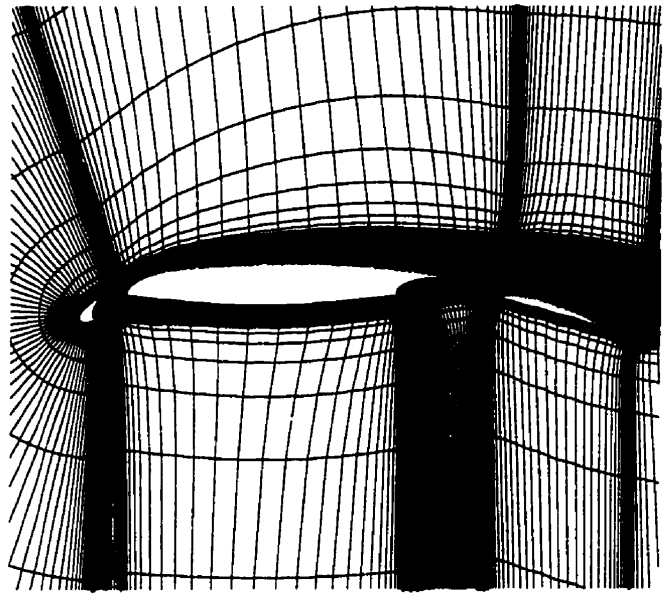


Figure 4.23: Close-up of the A-2 Grid

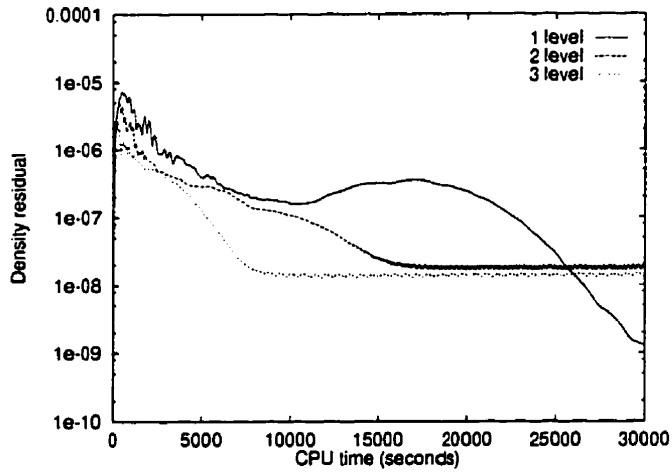


Figure 4.24: Residual History of A-2 Case with a Sawtooth Cycle

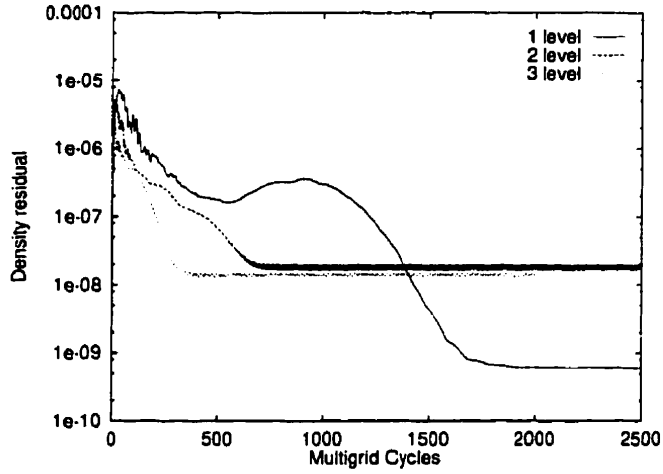


Figure 4.25: Residual History of A-2 Case with a Sawtooth Cycle (in Multigrid Cycles)

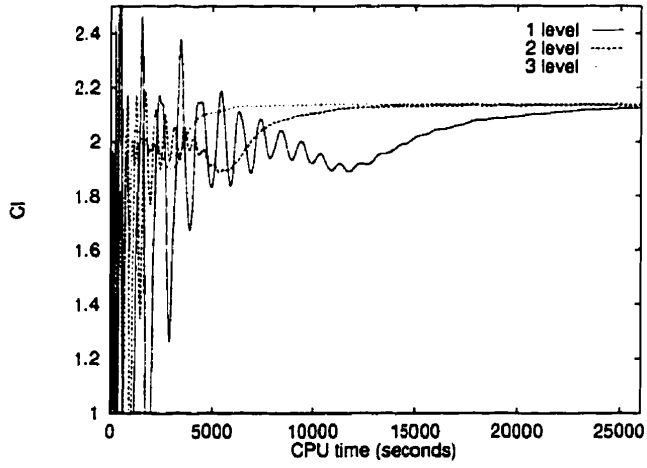


Figure 4.26: Lift Convergence of the A-2 Case with a Sawtooth Cycle

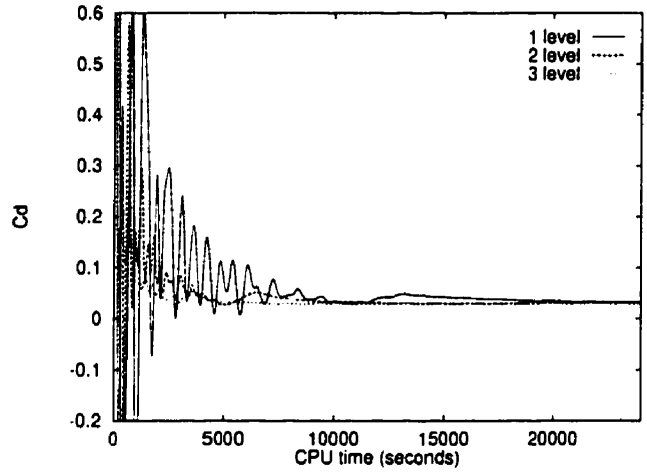


Figure 4.27: Drag Convergence of the A-2 Case with a Sawtooth Cycle

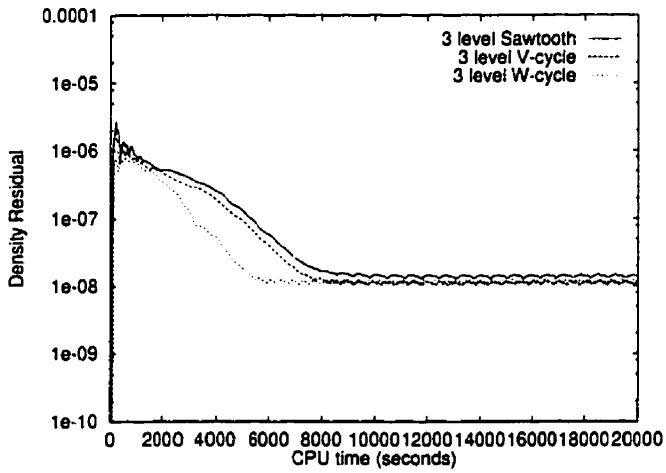


Figure 4.28: Residual History of A-2 Case with Sawtooth, V and W Cycles

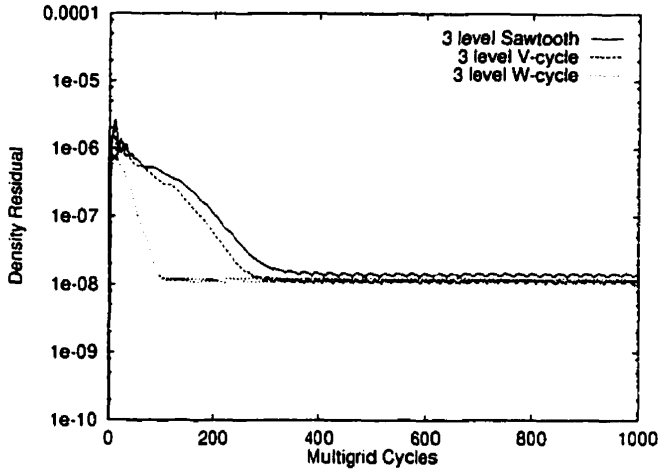


Figure 4.29: Residual History of A-2 Case with Sawtooth, V and W Cycles (in Multigrid Cycles)

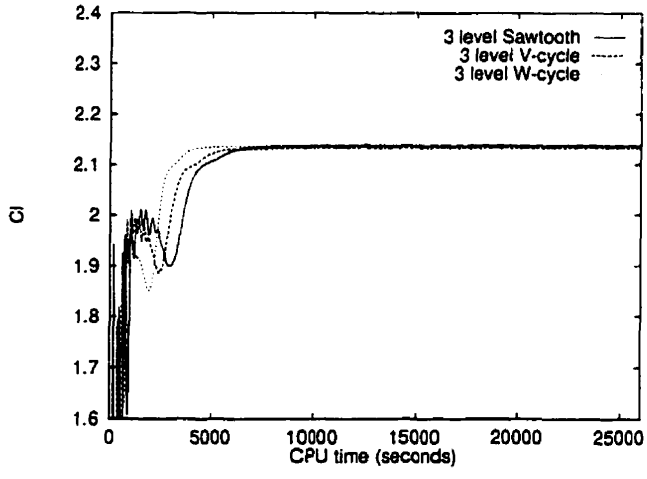


Figure 4.30: Lift Convergence of the A-2 Case with Sawtooth, V and W Cycles

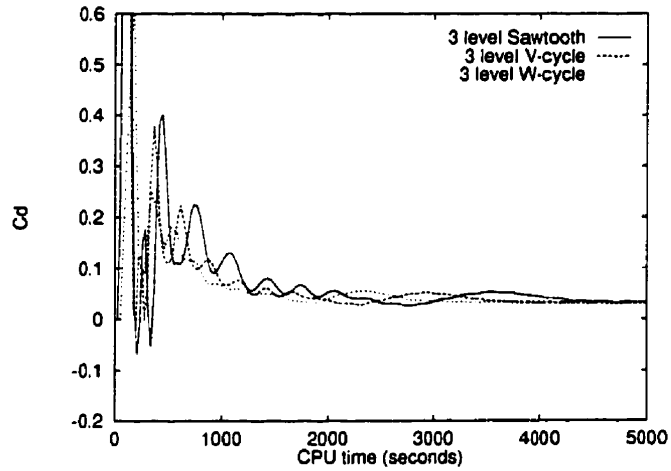


Figure 4.31: Drag Convergence of the A-2 Case with Sawtooth, V and W Cycles

Chapter 5

Conclusions and Recommendations

An efficient global-multigrid algorithm has been added to the TORNADO multiblock solver. The multiblock-multigrid solver is as efficient as the single-block multigrid of the SC1 solver. In the high-lift cases, the best multigrid cycle is the W cycle, followed by the V and sawtooth cycles. Three-level multigrid proved to be most effective at the removal of the error. With the addition of multigrid, TORNADO converges to steady state three to six times faster than previously.

It would be worth studying the effect on convergence of not applying multigrid to some of the blocks of the grid. In some cases, the error can be easily removed from some blocks and applying the multigrid cycle is not necessary. It would be important to develop a scheme to determine if multigrid should not be applied to some blocks of the grid. This would make the algorithm more effective and no unnecessary effort would be devoted to blocks where low-frequency errors are not present.

References

[1] Beam, R.M. and Warming, R.F., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," *J. Comp. Phys.*, Vol. 22, 1976, pp. 87–110.

[2] Steger, J.L., "Implicit Finite Difference Simulation of Flow about Arbitrary Geometries with Application to Airfoils," *AIAA paper*, 1977, pp. 77-665.

[3] Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," Lecture Notes For The Von Karman Institute For Fluid Dynamics Lecture Series: *Numerical Techniques For Viscous Flow Computation In Turbomachinery Bladings*, Jan. 1986.

[4] Jameson, A., "Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method," *Applied Mathematics and Computation*, Vol. 13, 1983, pp. 327–356.

[5] Jameson, A., "Multigrid Solutions of the Euler Equations using Implicit Schemes," *AIAA journal*, Vol. 24, No. 11, 1986, pp. 1737–1743.

[6] Martinelli, L. and Jameson, A. and Grasso, F., "A Multigrid Method for the Navier-Stokes Equations," AIAA paper 86-0208, 1986.

[7] Jameson, A. and Schmidt, W. and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping," AIAA Paper 81-1259, June 1981.

[8] Caughey, D.A., "Diagonal Implicit Multigrid Algorithm for the Euler Equations," *AIAA Journal*, Vol. 26, No. 7, July 1988.

[9] Yadlin, Y. and Caughey, D.A., "Block Multigrid Implicit Solution of the Euler Equations of Compressible Fluid Flow," *AIAA Journal*, Vol. 29, No. 0275, May 1991.

[10] Wang, L. and Caughey, D.A., "Multiblock/Multigrid Euler Method to Simulate 2D and 3D Compressible Flow," *AIAA Journal*, Vol. 32, No. 3, March 1993.

[11] Chisholm, T., *Multigrid Acceleration of an Approximately-Factored Algorithm for Steady Aerodynamic Flows*, Master's thesis, University of Toronto, Jan. 1997.

[12] Jespersen, D.C., Pulliam, T. and Buning, P., "Recent Enhancements to OVERFLOW," AIAA paper 97-0644, January 1997.

[13] Thompson, J.F., "A Composite Grid Generation Code for General 3-D Regions -the EAGLE code," *AIAA Journal*, Vol. 26, No. 3, 1988.

[14] Nelson, T. E., *Numerical Solution of the Navier-Stokes Equations for High-Lift Airfoil Configurations*, Ph.D. thesis, University of Toronto, May 1994.

[15] Godin, P. and Zingg, D.W. and Nelson, T.E., "High-Lift Aerodynamic Computations with One- and Two-Equation Turbulence Models," *AIAA Journal*, Vol. 35, No. 2, Feb. 1997, pp. 237–243.

[16] Nelson, T. E. and Zingg, D. W. and Johnston, G. W., "Compressible Navier-Stokes Computations of Multielement Airfoil Flows Using Multiblock Grids," *AIAA Journal*, Vol. 32, No. 3, March 1994.

[17] Fejtek, I., "Summary of Code Validation Results for a Multiple Element Airfoil Test Case," AIAA paper 97-1932, June 1997.

[18] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, January 1992.

[19] Jespersen, D.C., "Multigrid Methods for Partial Differential Equations," *Studies in Numerical Analysis*, Vol. 24, 1985, pp. 278–318.

[20] Wesseling, P., "Introduction to Multigrid Methods," ICASE Report No. 95-11, Feb. 1995.

[21] Maksymiuk, C.M., Swanson, R.C. and Pulliam T.H., "A Comparison of Two Central Difference Schemes for Solving the Navier-Stokes Equations," NASA Tech. Memo 102815, 1990.

[22] Wilkinson, A.R. and Zingg, D.W., "TORNADO User's Guide," Nov. 1993.

[23] De Rango, S. and Zingg, D.W., "SC1 User's Guide," Jan. 1997.

[24] Van den Berg, B., "Boundary Layer Measurements on a Two-Dimensional Wing with Flap," National Aerospace Lab., NLR TR 79009, Jan 1979.

Appendix A

Multigrid Pseudo-codes

Pseudo-code is presented for a multiblock solver with global multigrid. This multigrid subroutine, called MG, can be used for any of the multigrid cycles: Sawtooth, V cycle, and W cycle. Subscripts denote the level and block under consideration, whereas superscripts indicate the stage. "blk" and "nblk" represent the specific block and the total number of blocks in the grid repectively. "num_smooths" indicates the number of iterations per multigrid level.

Algorithm 1 subroutine MG

Require: ii,Q^0,Q^+,Q,S^0,P
  if (level .eq. bottom_level) **then**
    for (blk = 1,nblk) **do**
      $Q^+_{ii,blk} = Q^0_{ii,blk}$
      $S_{ii,blk}$ = RHS($Q^+_{ii,blk}$) + $P_{ii,blk}$
    end for
    for (blk = 1,nblk) & (k=1,num_smooths$_{ii}$) **do**
      $Q^+_{ii,blk}$ = SMOOTH ($Q^+_{ii,blk}$,$S_{ii,blk}$)
    end for
  else if (level .ne. bottom_level) **then**
    $Q_{ii,blk} = Q^0_{ii,blk}$
    if (cycle .eq. w-cycle) **then**
      nend = 2
    else
      nend = 1
    end if
    for (n = 1,nend) **do**
      for (blk = 1,nblk) & (k=1,num_smooths$_{ii}$) **do**
        $Q_{ii,blk}$ = SMOOTH ($Q_{ii,blk}$,$S_{ii,blk}$)
      end for
      for (blk = 1,nblk) **do**
        $Q^0_{ii+1,blk}$ = RESTRCT ($Q_{ii,blk}$)
      end for
      for (blk = 1,nblk) **do**
        $P_{ii+1,blk}$ = RESTRCT (RHS ($Q_{ii,blk}$)) - RHS($Q^0_{ii+1,blk}$)
      end for
      call MG (ii+1,Q^0,Q^+,Q,S^0,P)
      for (blk = 1,nblk) **do**
        $Q^+_{ii,blk} = Q_{ii,blk}$ + PLNG ($Q^+_{ii+1,blk}$ - $Q_{ii+1,blk}$)
      end for
      if (cycle .eq. V-cycle) **then**
        for (blk = 1,nblk) **do**
          $S_{ii,blk}$ = RHS ($Q^+_{ii,blk}$) + $P_{ii,blk}$
        end for
        for (blk = 1,nblk) & (k=1,num_smooths$_{ii}$) **do**
          $Q^+_{ii,blk}$ = SMOOTH ($Q^+_{ii,blk}$,$S_{ii,blk}$)
        end for
      end if
      if (cycle .eq. W-cycle .and. n.eq.1) **then**
        for (blk = 1,nblk) **do**
          $Q_{ii,blk} = Q^+_{ii,blk}$
        end for
      end if
    end for
  end if
  END