Multigrid Acceleration of an Approximately-Factored Algorithm for Steady Aerodynamic Flows

by

Todd Chisholm

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

The usefulness of any Navier-Stokes solver is directly related to the speed at which it converges. Multigrid has proven to be effective in dramatically reducing solution times for a range of solvers. In this thesis, the application of multigrid to an approximately-factored implicit Navier-Stokes solver for airfoils is discussed. Optimization of the solver and the multigrid process is carried out. The best type of cycle, restriction method, and number of smoothing passes at each level is determined. Six test cases are used to ensure the robustness of multigrid. Compared to a single-grid solver, a multigrid solver converges to steady state in one-quarter to one-sixth the time.

Acknowledgements

I would like to thank Professor D. W. Zingg for his supervision and guidance throughout my research. The members of his group provided invaluble advice. Dr. Dennis Jespersen gave insight when it seemed that multigrid would never work. Funding from de Havilland made the completion of this thesis possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The multigrid method is used to accelerate the numerical solution of partial differential equations. Multigrid theory was originally developed to be used in the solution of elliptic differential equations. Since then it has been applied to other types of systems, such as the compressible Navier-Stokes equations governing viscous fluid flows. Consider an approximate method for solving these equations. Normally an iterative method is applied to the spatially discretized equations, obtaining the final result in some number of steps. Iterative solvers can be regarded as error smoothers. They tend to reduce different error frequencies at different rates. Usually, the higher frequencies are eliminated much more quickly than the lower. Multigrid methods exploit this characteristic. Note that frequency in this context does not refer to the physical value of cycles per unit distance, but instead refers to cycles per node. Consider an error of one frequency only in a one dimensional system, with uniform node spacing. If every other node is removed, the frequency of this error is doubled. If the smoother operates more quickly on higher frequencies, then the error can be removed more effectively on the second grid, with fewer nodes. However the first, finer, grid is still necessary to achieve the required flow resolution. Multigrid combines at least two grids, using the coarser grids to help eliminate the long frequency error. Thus the effectiveness of multigrid is in a large part based on the error damping characteristics of the smoothing method, in this case the approximately factored method.

Considerable effort has been expended in adapting multigrid to different solvers. It has been particularly useful when an explicit method, such as a Runge-Kutta method, is used. Martinelli [1] has used this method to solve transonic and supersonic flows around airfoils. Maksymiuk, Swanson, and Pulliam [2] present a comparison of two schemes for solving for viscous flow around airfoils. They conclude that the Runga-Kutta solver using multigrid is considerably faster than an approximate factorization implicit method without multigrid. Hulshoff [3] and Arnone and Swanson [4] have solved rotor and cascade flows with a Runge-Kutta method accelerated using multigrid.

Multigrid with implicit time stepping has also been proven effective. Jameson and Yoon [5] and Caughey [6] have used multigrid with the approximate factorization and alternating direction implicit methods to solve the Euler equations around airfoils. O'Callahan and Thompson [7] provided a comparison of multigrid with three different methods: approximate factorization, line gauss-seidel and zebra. They solved the Euler equations on a transonic test case. They achieved convergence rate increases of better than three times. A multiblock two and three dimensional Euler solver using the alternating direction implicit method with multigrid was presented by Wang and Caughey [8].

The solution of the Navier-Stokes equations using implicit methods has also been aided by using multigrid. Varma and Caughey [9] used the alternating direction implicit method to solve viscous flow over airfoils. Jespersen, Pulliam, and Buning [10] describe the results of adding multigrid to OVERFLOW, a three dimensional Navier-Stokes code, which uses a variety of implicit methods.

This thesis discusses the application of multigrid to the approximately-factored implicit Navier-Stokes solver ARC2D. The new code is called ARC2D-MG. Results are presented for the flow around the NACA 0012 and RAE 2822 airfoils under a variety of flow conditions. The convergence histories of ARC2D and ARC2D-MG are compared to establish the benefits of multigrid in each case.

Chapter 2

ARC2D

This section discusses the governing equations and the basic numerical methods which ARC2D uses, concentrating on those parts relevant to multigrid. ARC2D may solve either the Euler equations, in the case of inviscid flows, or the thin-layer Navier-Stokes Equations in the viscous case. This paper will deal exclusively with the Navier-Stokes equations. To approximate the spatial derivatives, a second-order centered difference is used. Time marching is accomplished with an implicit method. Approximate factorization allows the use of banded matrix solvers, which results in drastic time savings per iteration. Computational work is further decreased through the use of diagonalization. Pulliam gives an overview of the operation of ARC2D[11].

2.1 The Thin-Layer Navier Stokes Equations

The Navier-Stokes equations may be used to predict airflow with viscosity. In the case of the solution of flow conditions around an airfoil, a good approximation may be made by using the thin-layer Navier-Stokes equations. These assume that the viscous terms caused by derivatives along the body are negligible. This assumption does not hold for low Reynolds numbers, or in a case with large regions of separated flow.

The thin-layer equations in curvilinear coordinates are given below:

$$\partial_\tau \hat{Q} + \partial_\xi \hat{E} + \partial_\eta \hat{F} = Re^{-1} \partial_\eta \hat{P} \tag{2.1}$$

with $\hat{Q} = J^{-1}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix}$, $\hat{E} = J^{-1}\begin{bmatrix}\rho U\\ \rho u U + \xi_x p\\ \rho v U + \xi_y p\\ U(e+p) - \eta_t p\end{bmatrix}$, $\hat{F} = J^{-1}\begin{bmatrix}\rho V\\ \rho u V + \eta_x p\\ \rho v V + \eta_y p\\ V(e+p) - \xi_t p\end{bmatrix}$

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

$$\begin{aligned}
m_1 &= \mu(4\eta_x u_\eta - 2\eta_y v_\eta)/3\\
m_2 &= \mu(\eta_y u_\eta + \eta_x v_\eta)\\
m_3 &= \mu(-2\eta_x u_\eta + 4\eta_y v_\eta)/3\\
m_4 &= \mu Pr^{-1}(\gamma-1)^{-1}\eta_x \partial_\eta(a^2)\\
m_5 &= \mu Pr^{-1}(\gamma-1)^{-1}\eta_y \partial_\eta(a^2)\\
U &= \eta_t + \eta_x u + \eta_y v\\
V &= \xi_t + \xi_x u + \xi_y v
\end{aligned}$$

J is the Jacobian matrix:

$$J^{-1} = (x_\eta y_\xi - x_\xi y_\eta) \tag{2.2}$$

The pressure is given by:

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

2.2 Time Marching Method

ARC2D uses first order implicit Euler time marching. When this is applied to equation 2.1 the following results:

$$\hat{Q}^{n+1} - \hat{Q}^n + h\left(\partial_\xi \hat{E}^{n+1} + \partial_\eta \hat{F}^{n+1} - Re^{-1}\partial_\eta \hat{P}^{n+1}\right) = 0 \tag{2.4}$$

where h is the time step. The flux vectors $\hat{E}$, $\hat{F}$, and $\hat{P}$ are linearized as follows:

$$
\begin{aligned}
\hat{E}^{n+1} &= \hat{E}^n + \hat{A}^n \Delta \hat{Q}^n + O(h^2) \\
\hat{F}^{n+1} &= \hat{F}^n + \hat{B}^n \Delta \hat{Q}^n + O(h^2) \\
Re^{-1} \hat{P}^{n+1} &= Re^{-1} \left[\hat{P}^n + \hat{M}^n \Delta \hat{Q}^n \right] + O(h^2)
\end{aligned}
\tag{2.5}
$$

where $\hat{A} = \frac{\partial \hat{E}}{\partial \hat{Q}}$, $\hat{B} = \frac{\partial \hat{F}}{\partial \hat{Q}}$, $\hat{M} = \frac{\partial \hat{P}}{\partial \hat{Q}}$, and $\Delta \hat{Q} = \hat{Q}^{n+1} - \hat{Q}^n$

To speed up the solution process, approximate factorization is used. This reduces the left hand side matrix to two block tridiagonal matrices. These are then inverted in two simple steps, instead of the time consuming inversion of a sparse matrix.

$$
\begin{aligned}
&\left[I + h \delta_\xi \hat{A}^n \right] \left[I + h \delta_\eta \hat{B}^n - h Re^{-1} \delta_\eta \hat{M}^n \right] \Delta \hat{Q}^n = \\
&-h \left[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n - Re^{-1} \delta_\eta \hat{P}^n \right]
\end{aligned}
\tag{2.6}
$$

Diagonalization also helps to simplify the equations. The Jacobian matrices may be diagonalized as follows:

$$\Lambda_\xi = T_\xi^{-1} \hat{A} T_\xi \tag{2.7}$$

$$\Lambda_\eta = T_\eta^{-1} \hat{B} T_\eta \tag{2.8}$$

These decompositions are applied to 2.6 giving:

$$
\begin{aligned}
&\left[T_\xi T_\xi^{-1} + h \delta_\xi \left(T_\xi \Lambda_\xi T_\xi^{-1} \right) \right] \left[T_\eta T_\eta^{-1} + h \delta_\eta \left(T_\eta \Lambda_\eta T_\eta^{-1} \right) \right] \Delta \hat{Q}^n = \\
&-h \left[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n - Re^{-1} \delta_\eta \hat{P}^n \right]
\end{aligned}
\tag{2.9}
$$

The final form is obtained when the eigenvector matrices T_η and T_ξ are factored out of the spatial derivatives:

$$T_\xi \left[I + h \delta_\xi \Lambda_\xi \right] T_\xi^{-1} T_\eta \left[I + h \delta_\eta \Lambda_\eta \right] T_\eta^{-1} \Delta \hat{Q}^n = -h \left[\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n - Re^{-1} \delta_\eta \hat{P}^n \right] \tag{2.10}$$

Variable time stepping is used to further increase the convergence rate, by roughly equalizing the Courant numbers of each cell. Since the grids used have highly stretched cells, the majority of the Courant number variation may be eliminated by scaling with the Jacobian:

$$h = \frac{h_{ref}}{1 + \sqrt{J}} \tag{2.11}$$

To further eliminate fluctuation of the Courant number, the time step may be scaled by both the Jacobian, and characteristic speeds:

$$h = \frac{h_{ref}}{|U| + |V| + a\sqrt{\xi_x^2 + \xi_y^2 + \eta_x^2 + \eta_y^2}} \tag{2.12}$$

In general, the former expression is used, since it seems adequate in smoothing the Courant number, and it requires less overhead.

2.3 Boundary Conditions

ARC2D may use both O-grid and C-grid topologies. The latter type will be used exclusively in this thesis. As shown in figure 2.1, there are four boundary types seen in ARC2D. The first two are the far field and outflow boundaries. They are both handled in the same manner, since there is free flow into and out of the region here. This boundary condition is enforced by the use of Riemann invariants. The second type of boundary condition is seen at the body. In viscous flows, the velocity at the surface is set to zero. The final boundary occurs at the wake cut. This may be solved explicitly or implicitly by integrating across the cut. The latter method is used in ARC2D-MG.

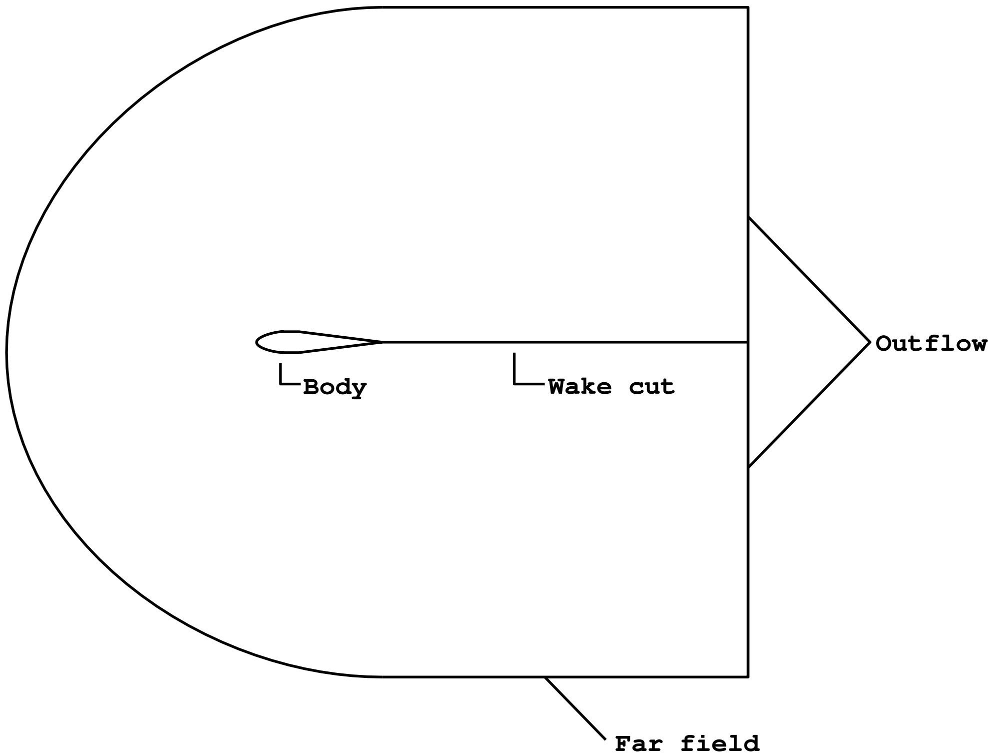

Figure 2.1: ARC2D boundary types

Chapter 3

Multigrid

3.1 Multigrid Cycles

Jameson provides an introduction to the operation and analysis of multigrid [12]. Much of the following is based on this paper. Before all the details of a multigrid cycle are considered, it is best to discuss the simplest possible cycle. Consider a method using two levels, solving for airflow in a one dimensional system. The discretized and implicit time differenced equations on the top or finest grid can be represented by the following:

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

The matrix A represents the matrix portion of the appropriate equations after they have been spatially discretized and linearized. RHS($\hat{Q}$) is the residual vector of these equations. $\hat{Q}$ holds the flow variables at each node.

Because the Navier-Stokes equations are non-linear, full approximation storage multigrid must be used. This requires that two vectors be copied from the fine grid to the coarse grid. The first, $\hat{Q}$, contains the flow variables. The second vector, $\hat{R}$, is the fine-grid residual, and ensures that the coarse grid is actually correcting the fine grid solution. The transferring of a vector from a fine grid to a coarse is termed restriction, while the reverse is prolongation. The following pseudo-code defines one iteration of the simplest possible multigrid cycle. A comprehensive description of the vectors and functions used is given in Appendix A.

- Form $\hat{S}_1^0$, the right hand side for $\hat{Q}_1^0$

  $\hat{S}_1^0 = \text{RHS}(\hat{Q}_1^0)$

- Perform one or more smoothing steps on the fine grid

  $\hat{Q}_1^i = \text{SMTH}(\hat{Q}_1^0,\hat{S}_1^0)$

- Form $\hat{R}_1^i$, the right hand side for $\hat{Q}_1^i$

  $\hat{R}_1^i = \text{RHS}(\hat{Q}_1^i)$

- Form $\hat{Q}_2^0$, the restricted solution

  $\hat{Q}_2^0 = \text{RSTRCT}(\hat{Q}_1^i)$

- Form $\hat{R}_2^r$, the restricted residual

  $\hat{R}_2^r = \text{RSTRCT}(\hat{R}_1^i)$

- Form $\hat{S}_2^0$, the right hand side for $\hat{Q}_2^0$

  $\hat{S}_2^0 = \text{RHS}(\hat{Q}_2^0)$

- Form $\hat{D}_2$, the forcing function to be used on the coarse grid

  $\hat{D}_2 = \hat{R}_2^r$ - $\hat{S}_2^0$

- Form $\hat{R}_2^0$, the residual used in the smoothing steps

  $\hat{R}_2^0 = \hat{S}_2^0 + \hat{D}_2$

- Form $\hat{Q}_2^+$, by performing one or more smoothing steps on the coarse grid

  $\hat{Q}_2^+ = \text{SMTH}(\hat{Q}_2^0,\hat{R}_2^0)$

- Correct the top level solution

  $\hat{Q}_1^+ = \hat{Q}_1^i + \text{PRLNG}(\hat{Q}_2^+ - \hat{Q}_2^0)$

In the above and all of the following, a vector's subscript refers to the level on which that vector is defined. Level one is the top, or finest level. If a vector has a zero as a superscript, then the vector occurs before smoothing. A superscript of i represents a vector after smoothing. A + superscript on $\hat{Q}$ shows that the vector has been corrected by the lower levels. The vector $\hat{R}$ has been introduced above. It is defined as:

$$\hat{R} = \hat{S} + \hat{D} \tag{3.2}$$

In the smoothing iteration on the coarse grid, $\hat{R}_2^0$ becomes the new right hand side, instead of $\hat{S}_2^0$. This gives the following modified equation which is passed to the smoothing routine:

$$A\Delta\hat{Q} = \hat{R} \tag{3.3}$$

In the above, four functions are introduced. PRLNG is the prolongation operator. It raises $\hat{Q}$ to the fine grid. RSTRCT performs the opposite function, lowering a vector to the coarse grid. There are a number of ways to implement these functions. These are discussed in section 3.2. SMTH($\hat{Q}$, $\hat{R}$) performs one smoothing pass on the vector $\hat{Q}$, using $\hat{R}$ as the residual vector. Note that on the finest level, the vector $\hat{S}$ is used as the residual vector. At all subsequent levels, the vector $\hat{R}$, which is the sum of the right hand side and the forcing function, is used as the residual vector. RHS($\hat{Q}$) forms the right hand side, or the vector $\hat{S}$.

The last step in the multigrid process is known as correcting, since the prolonged change from the lower grids is added to the top level. This process reveals that $\hat{D}_2$ is formed by subtracting $\hat{R}_2^r$ from $\hat{R}_2^0$. In the case of the simple routine given above, $\hat{D}$ may be expanded, and since only one smoothing iteration is performed on the second level, $\hat{S}_2 = \hat{S}_2^0$. Equation 3.3 then becomes:

$$A_2\Delta\hat{Q}_2 = \hat{R}_1^r \tag{3.4}$$

In this form it is easier to see how $\hat{D}$ drives the lower grid to correct the upper. However, the system sent to the smoother should be left in the form given in equation 3.3. This is in case multiple iterations are performed on the lower level, in which case $\hat{R}_2 \neq \hat{R}_2^0$.

It is easy to see that it may be possible to further increase the effectiveness of the multigrid process by adding coarser grids. It is common to use up to four levels. In fact, the process is easiest to implement using a recursive subroutine. When more than two levels are being used, the smoothing, restricting, and prolonging may be used in different combinations than the basic cycle given above. There are three common sequences used. The first is a V-cycle. See figure 3.2. This simple case involves smoothing then restricting to the lowest level, then smoothing and prolonging back to the top level. An even simpler sequence is shown in figure 3.1. This is the sawtooth cycle. It is similar to the V-cycle,

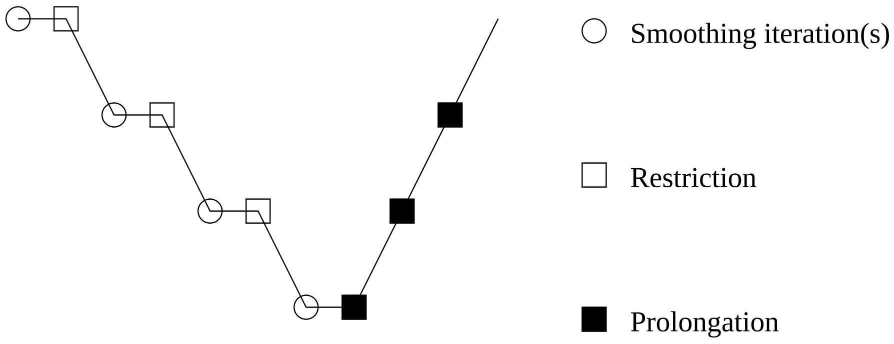

Figure 3.1: Sawtooth cycle

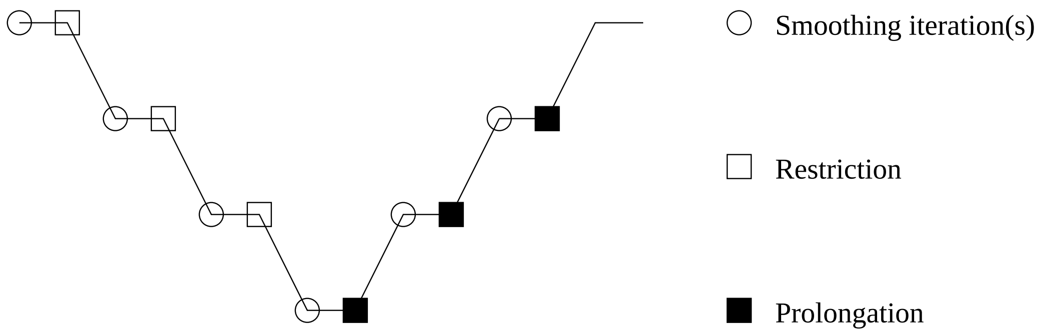

Figure 3.2: V-cycle

but does not smooth on the steps from the coarsest to the finest level. The final common sequence, called the W-cycle, may be seen in figure 3.3.

Appendix B gives comprehensive pseudo-code with an arbitrary number of levels, and a variety of cycles. It also allows for other parameters which will be discussed in section 4.3.

3.2 Restriction and Prolongation Methods

Recall that vectors need to be transferred both from a grid to the next coarser grid and back. The functions which perform this are labeled $RSTRCT$ and $PRLNG$ respectively. In ARC2D-MG, the coarse grids are formed by removing every other node in each direction, so that every node on the coarse grid has a corresponding node on the fine grid.

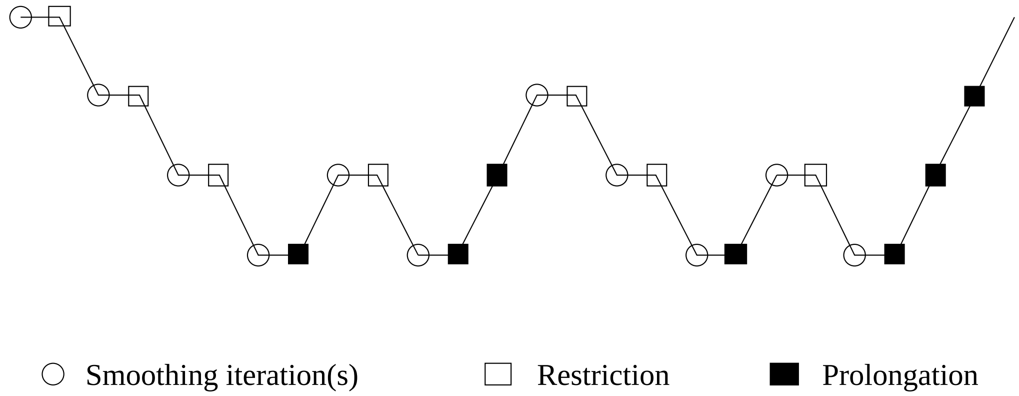

Figure 3.3: W-cycle

3.2.1 Jacobians

If the vector being either restricted or prolonged is $\hat{Q}$, then the inclusion of J^{-1} must be considered. Refer to section 2 for a discussion of these variables. The restriction and prolongation operators may or may not remove J_k^{-1}, on the operand, then multiply the result by J_{k+1}^{-1}, if the operation is restriction, or J_{k-1}^{-1} if the operation is prolongation. If J^{-1} is removed, then the operation is effectively being performed on $\vec{Q}$. This is also a consideration when the residual is restricted.

3.2.2 Restriction

Restriction offers two options. The simplest case uses the nodes on the fine grid that correspond directly to the coarse grid. See figure 3.4 for the one dimensional case. This is known as simple injection. The second method uses a weighting scheme over all the nodes in the grid. See figures 3.5 and 3.6. The effect of using different restriction methods is discussed in section 4.5.

3.2.3 Prolongation

When prolongation on a non-uniform grid is carried out, it is necessary to decide whether computational or physical space is to be used. There are three positions of fine grid nodes, as shown in 3.7. The first, shown as a filled circle, has a corresponding coarse grid node,

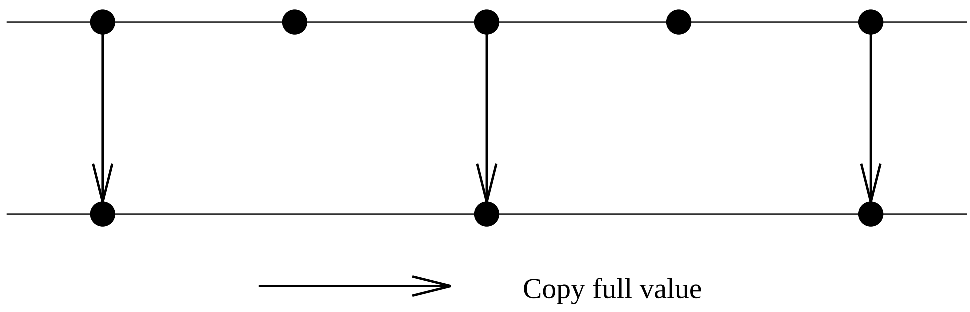

Figure 3.4: One dimensional restriction using simple injection

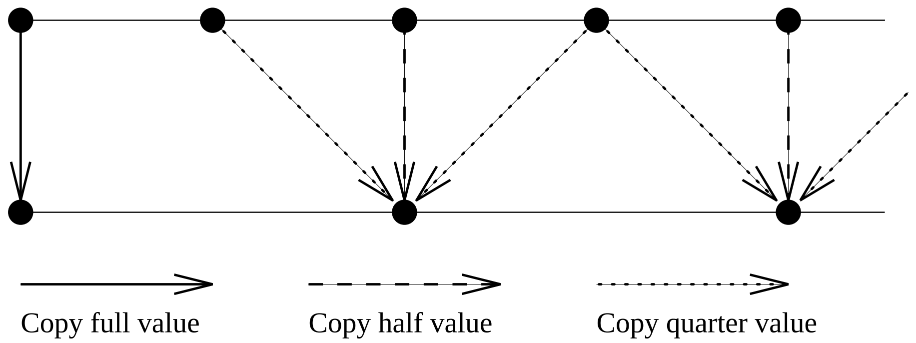

Figure 3.5: One dimensional restriction using weighting

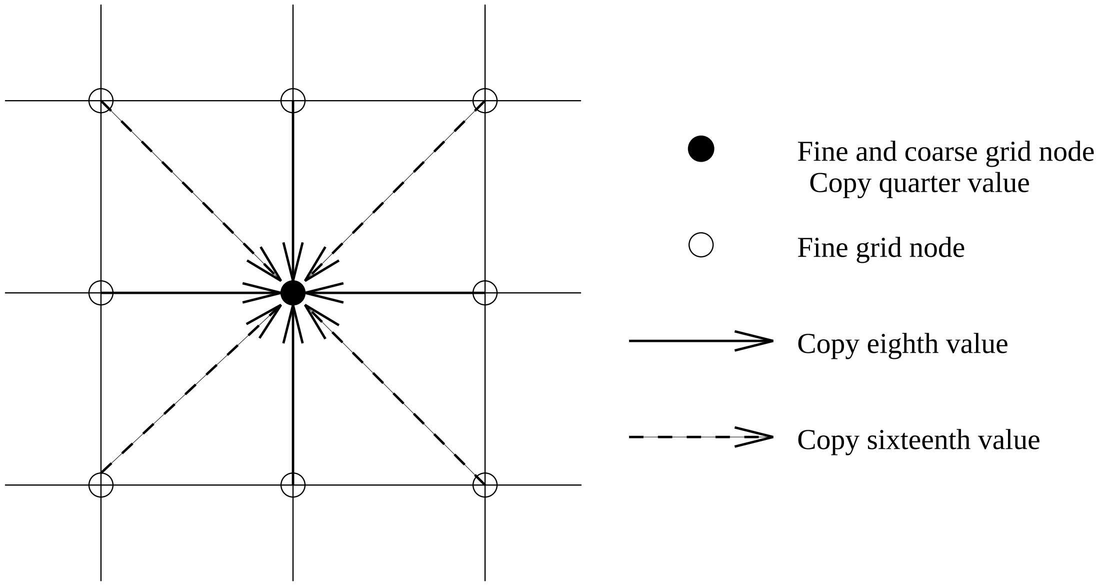

Figure 3.6: Two dimensional restriction using weighting

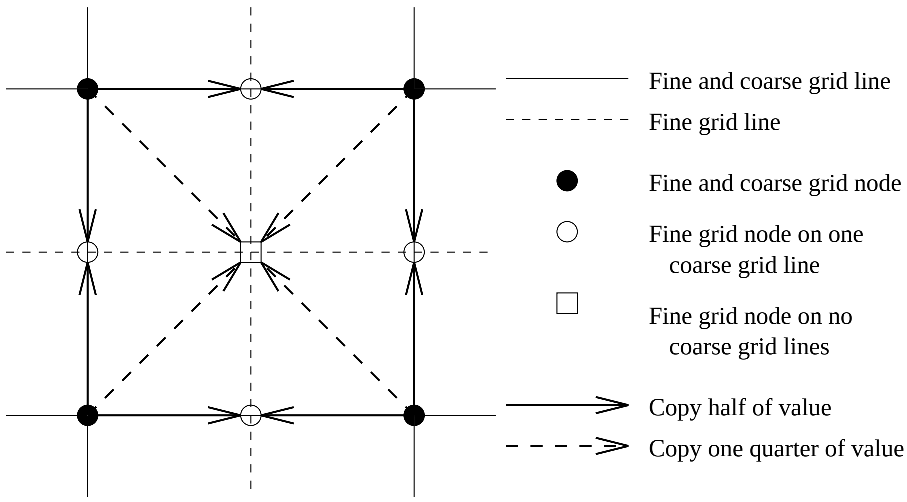

Figure 3.7: Two dimensional prolongation in computational space

and the value of that node is simply copied to the fine grid. The second type, shown as an open circle, lies on one coarse grid line, between two coarse grid nodes. The average of the surrounding two nodes gives the value for these nodes. The final node type, shown as a square, is not on any coarse grid line, and lies between four coarse grid nodes. The average of these four nodes is then used.

If the prolongation is to be carried out in physical space, then a distance weighted average is used, as shown in 3.8. In the case of the fine grid node with a corresponding coarse grid node, the value is directly copied, as usual. If the fine grid node lies on one coarse grid line, then the distance from each of the two surrounding nodes to the fine grid node is calculated. Call these D_1 and D_2. Then if we call the surrounding node values Q_1 and Q_2, then the fine grid node value Q_f is given by:

$$Q_f = Q_1 \frac{D_2}{D_1 + D_2} + Q_2 \frac{D_1}{D_1 + D_2} \tag{3.5}$$

In the case of a fine grid node which is not on any coarse grid lines, the distances from the four surrounding coarse grid nodes to the fine grid node must be found. The inverse of these is taken. Call them D_1^{-1} to D_4^{-1}. The sum of these is $\sum D^{-1}$. Then the fine grid node value is given by:

$$Q_f = \frac{Q_1 D_1^{-1} + Q_2 D_2^{-1} + Q_3 D_3^{-1} + Q_4 D_4^{-1}}{\sum D^{-1}} \tag{3.6}$$

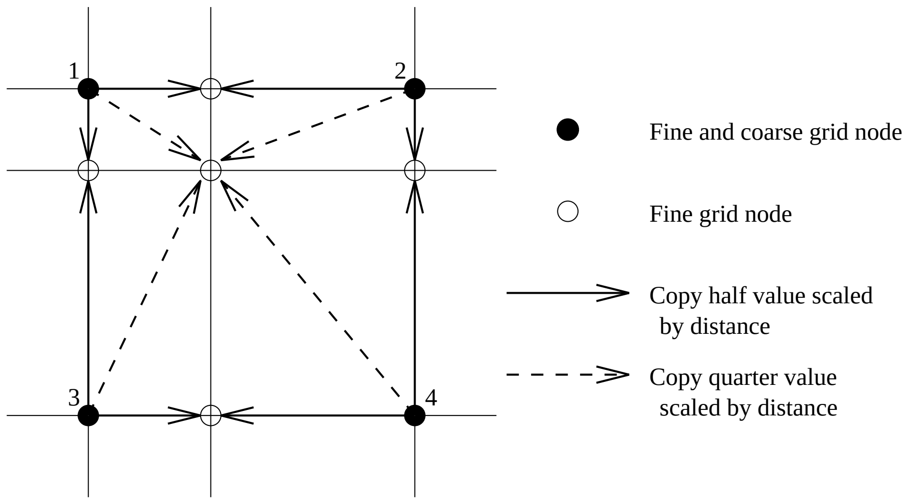

Figure 3.8: Two dimensional prolongation in physical space

The coefficients above are constant, and add very little overhead to the code.

Chapter 4

ARC2D-MG

4.1 Testing of Multigrid Method

The effectiveness of the MG method is analyzed in two simple ways. First, the residual vs. CPU time required is examined. The residual refers to the L2 norm of $\hat{S}$, the right hand side, on the finest level. Second, the coefficients of lift or drag vs. CPU time may be compared. The results are compared with the appropriate single level solver.

To ensure that the new code is robust, a number of different cases are shown, varying the grid, airfoil, angle of attack, and Mach number. Note that if a converged solution is obtained, it is independent of the method used.

4.2 Test Cases

Six cases were considered when testing the multigrid ARC2D code. They are shown in Table 4.2. A wide range of flow situations was used. Case one is relatively easy to solve, with a low angle of attack, but a somewhat high Mach number, leading to a very weak shock. Cases two and three are low speed flows. Case three has a very high angle of attack, giving high lift. Both of these cases are time consuming to solve. Cases four, five, and six all have stronger shocks, due to high Mach numbers and angles of attack. Cases four and five are solved on the RAE 2822 airfoil. All other cases utilize the NACA 0012. These cases were taken from Zingg's grid study [13].

Case	Airfoil	Mach	A of A	Re	Trans Lo	Trans Up	Grid
1	NACA 0012	0.7	1.49	9×10^6	0.05c	0.05c	2
2	NACA 0012	0.16	6.00	2.88×10^6	0.8c	0.05c	1
3	NACA 0012	0.16	12.0	2.88×10^6	0.95c	0.01c	1
4	RAE 2822	0.729	2.31	6.5×10^6	0.03c	0.03c	3
5	RAE 2822	0.754	2.57	6.2×10^6	0.03c	0.03c	3
6	NACA 0012	0.7	3.00	9×10^6	0.05c	0.05c	2

Table 4.1: ARC2D-MG test cases

Grid	Airfoil	Dimensions	Body	Wake	Off Wall	Nose	Tail
1	NACA 0012	249x97	100	25	10^{-6}	2×10^{-4}	2×10^{-4}
2	NACA 0012	385x65	144	49	2^{-6}	5×10^{-3}	5×10^{-4}
3	RAE 2822	385x65	144	49	2^{-6}	5×10^{-3}	5×10^{-4}

Table 4.2: ARC2D-MG test grids

Three grids are used in the study, in order to help demonstrate the dependence of multigrid on the grid configuration. The first is used for the low speed cases two and three. The second is used for the high speed cases solving around the NACA 0012. The third is for the RAE 2822 solutions. Figures 4.1, 4.2, and 4.3 show the grids. All were created using HGRD, a hyperbolic grid generator. Table 4.2 gives information required by HGRD.

4.3 Optimization of Multigrid

In order to ensure that multigrid is providing the most benefit, a number of optimizations were carried out. They are detailed in the following sections. In general, case one was used as the test case, because of the ease in solving it, as well as the moderate nature of the flow conditions. If it was discovered that this case benefited from the change of parameter, then it was applied to the rest of the cases. The purpose of the optimization was to determine the single set of parameters which would provide the fastest solution for all the test cases. This will hopefully ensure that the code may be used for any case, without requiring special knowledge to adjust any multigrid parameters.

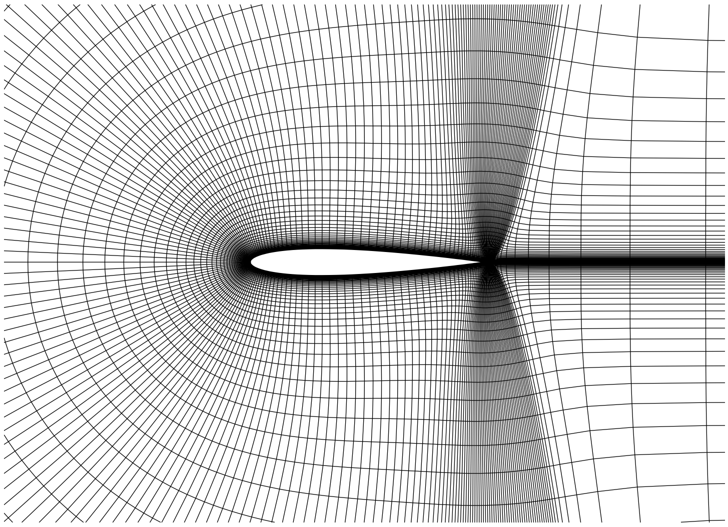

Figure 4.1: Grid 1

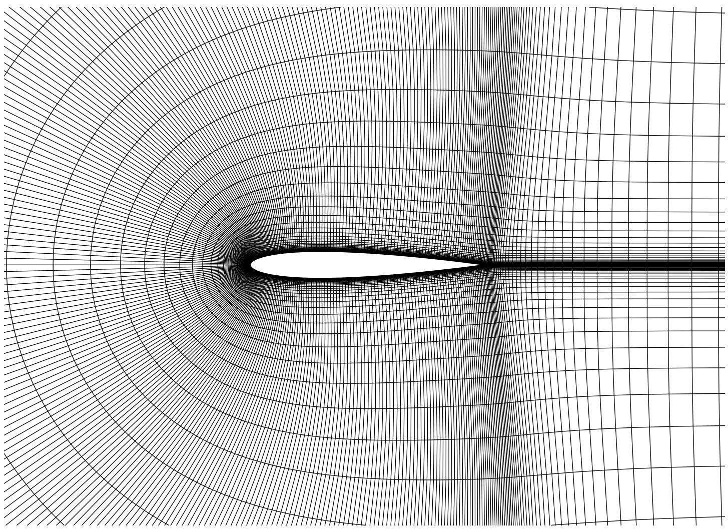

Figure 4.2: Grid 2

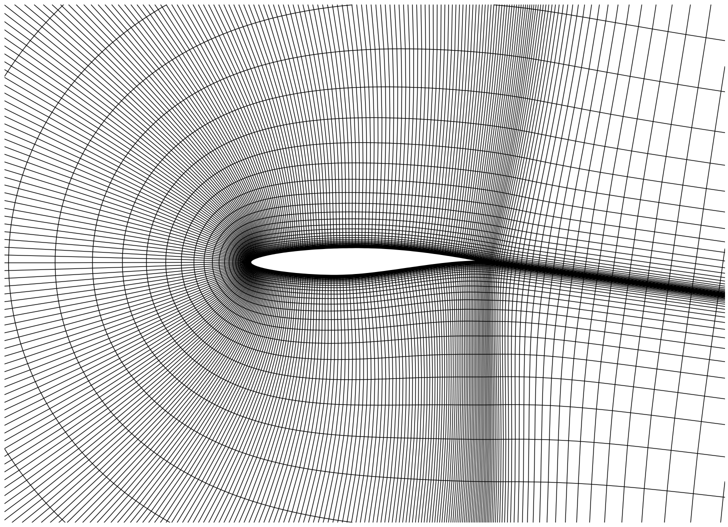

Figure 4.3: Grid 3

4.4 Smoothing Passes

If one examines a timing profile of a three-level multigrid code, it can be seen that a very large portion of time is spent creating the right hand side. This must be done twice when going from a fine to coarse grid. One way to decrease overhead significantly is to decrease the number of transfers. This can be done by doing multiple smoothing passes on each level before restricting. Some caution should be taken though. By performing more smoothing passes, the effectiveness of multigrid may be reduced. To determine if this is the case, we run the solver five times, beginning with one smoothing pass per level, up to five smoothing passes, and compare the number of multigrid cycles required to converge to a residual of less than 10^{-10}. The results are shown in Figure 4.4 and Table 4.3, for a sawtooth cycle applied to case one.

It is obvious that this is an important parameter. The best choice is four smoothing passes, which reduces the convergence time by over one third. The third column shows the number of multigrid cycles required to reach a residual of less than 10^{-10}. The fourth column is this number multiplied by the number of smoothing passes per level per cycle.

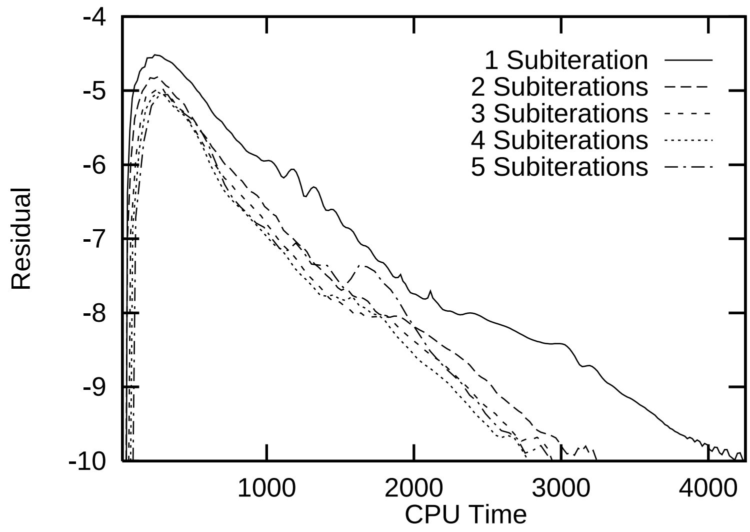

Figure 4.4: Effect of number of smoothing passes on convergence history

Smoothing Passes	CPU Time	Cycles	Normalized Iterations
1	4252	251	251
2	3269	127	254
3	2946	83	249
4	2815	64	256
5	2970	55	275

Table 4.3: Effect of number of smoothing passes

giving a comparison of the number of smoothing passes performed on the fine grid. It is interesting to note that this number is relatively constant until five smoothing passes are used. This indicates that the convergence rate relative to the number of iterations is not adversely affected when using four smoothing passes. Considerable time is saved by reducing the number of restrictions and prolongations.

4.5 Restriction and Prolongation

The methods of restriction and prolongation of the solution vector $\vec{Q}$ and the restriction of the residual vector $\vec{R}$ are considered next. It was discovered that the method applied to

$\vec{Q}$ is unimportant, as long as the Jacobian is removed from the solution before restriction. Simple injection and weighted injection in both computational and physical space were tried. There was no significant difference in convergence time. On the other hand, the residual must be restricted using the weighted injection method in computational space, operating on the residuals without Jacobians, in order to ensure convergence.

4.6 Boundary Conditions

ARC2DMG is particularly sensitive to when boundary conditions are applied. During a one-level code, they are applied after each smoothing iteration. However, during one cycle of a multi-level cycle, the boundary subroutine may be called every time a grid is created or changed. For a two-level code, this could be after each of the following:

- Top-level smoothing $\hat{Q}_1^i$
- Restriction to second level $\hat{Q}_2^0$
- Bottom-level smoothing $\hat{Q}_2^+$
- Top level correction $\hat{Q}_1^+$

The vector shown on the right is sent to the boundary routine. Setting the boundary conditions at the correct parts of the multigrid routine is critical to success. In general it is best to call these routines every time a new solution vector is created. This happens after a restriction, correction, or smoothing pass.

4.7 Full Multigrid

In an effort to reduce the initial error, ARC2D uses grid sequencing. This involves performing a number of iterations on multiple grids, starting with the coarsest grid, then moving to successively finer grids. Once the solution is transferred from a coarse grid to a finer grid, the coarse grid is not reused. The coarse grids, by virtue of the fact that they are faster to solve and allow higher time steps, tend to remove the initial error quickly.

Coarse Grid Iterations	Middle Grid Iterations	CPU Time
15	15	3917
20	20	3922
30	30	3688
40	40	3770
50	50	3981

Table 4.4: Comparison of number of FMG iterations

Using full multigrid, the same advantages of grid sequencing may be realized. The process is similar. A one level solver is used on the coarsest grid. After a number of iterations, the solution is transferred to the next grid, and a two level multigrid solver is used. This is continued until the finest level is reached, where the familiar multigrid process is used.

It is important to establish how many multigrid iterations are to be performed at each level. Table 4.4 shows the CPU time required to reduce the residual to less than 10^{-10}. This shows that an appropriate number of iterations is about thirty, and that the code is not particularly sensitive to this value.

Figure 4.7 shows the residual convergence histories of case one with and without FMG. A speedup of 25% is achieved through the use of FMG. Similar results are seen with other cases.

4.8 Multigrid Cycles

The type of cycle used can have a significant effect on the convergence rate. Three cycles are considered; the W-cycle, the sawtooth, and various V-cycles. Since the number of smoothing passes may be adjusted, a variety of V-cycles where tried. Figure 4.6 compares five cycles; a W-cycle, a sawtooth, and three V-cycles. The V-cycles are designated with two numbers. They indicate the number of smoothing passes performed on the way down and the way up, respectively. Examining the results shows that the W-cycle is the fastest choice. It will be seen that in many cases, the sawtooth cycle is very close to the W-cycle in performance.

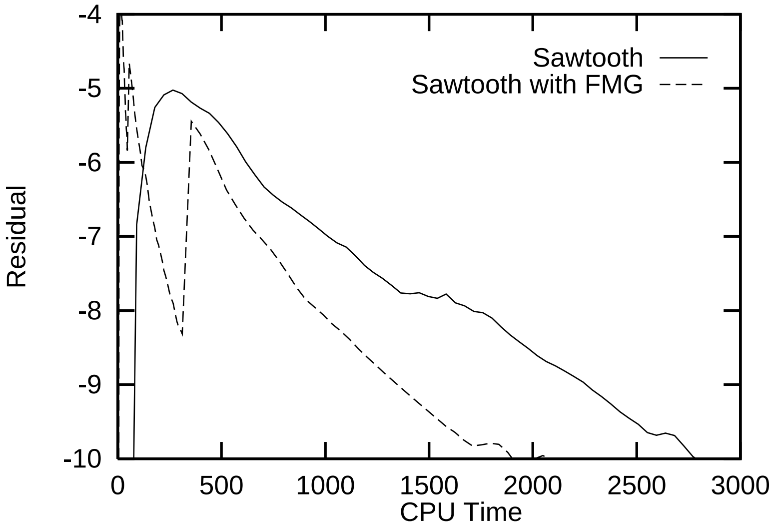

Figure 4.5: Effect of full multigrid on residual convergence, case one

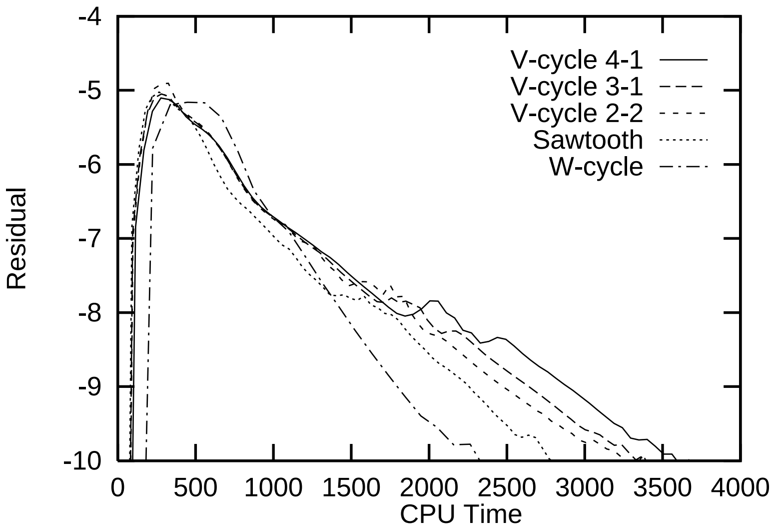

Figure 4.6: Comparison of multigrid cycles, case one

4.9 Correction Factor

Recall that the solution at any level which is not the coarsest level is corrected by:

$$\hat{Q}_k^+ = \hat{Q}_k^i + P(\hat{Q}_{k+1}^+ - \hat{Q}_{k+1}^0) \tag{4.1}$$

Jespersen [14] suggests including a correcting factor, σ:

$$\hat{Q}_k^+ = \hat{Q}_k^i + \sigma P(\hat{Q}_{k+1}^+ - \hat{Q}_{k+1}^0) \tag{4.2}$$

This allows the possibility of amplifying or attenuating the correction returned from lower levels. To test the usefulness of this, the code was executed twice, with σ equal to 0.9 then 1.1. Both resulted in slightly slower execution times.

4.10 Variable Time Stepping

Throughout these tests, variable time stepping has been based only on the geometry of the grid, with the time step given by equation 2.11. The alternative method, equation 2.12 was briefly tried, with the result being an increase in convergence time. For the single grid case, Δt_{ref} is optimized to five. This value is found to give good results for a variety of cases. However, it may be that the best value for multigrid is somewhat different. Figure 4.10 shows the effect of varying Δt_{ref} on case four using a sawtooth cycle. If a time step of ten is used, the convergence is accelerated by almost 20%. However, when a time step of ten is used on case six, with a W-cycle and FMG, convergence is slowed by almost 10%. When case eight is solved with similar circumstances, twice as much time is required. This seems to be a parameter best tuned to each case. In keeping with our desire to have a code which requires no adjustment though, Δt_{ref} will be kept at five, which seems to be adequate for all cases.

4.11 Dissipation

Recall that multigrid operates best when there is more damping of the high frequency components than of the lower. Jameson and Yoon [5] suggest that the frequency damping characteristics of the method may be modified by having the implicit and explicit

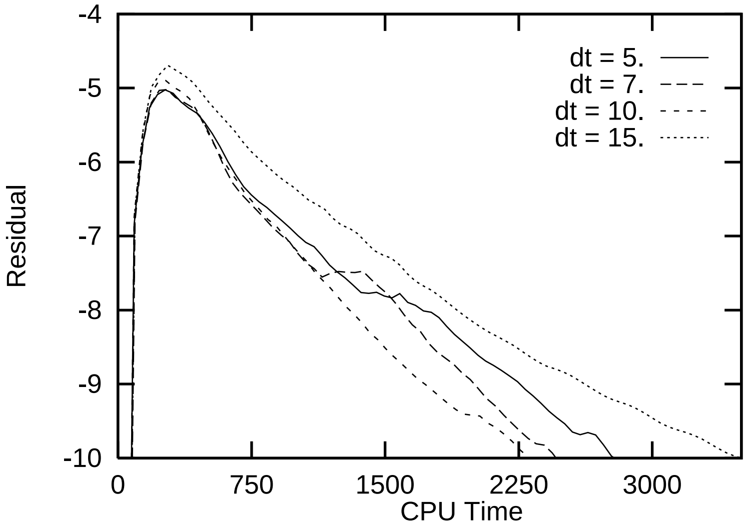

Figure 4.7: Effect of changing time step, case one

dissipation not equal. However, we desire to keep the right hand side of the equations equal to the original ARC2D, so that the fully converged solutions are equal. Thus, only the implicit dissipation may be modified. Case one with a W-cycle and FMG was solved twice, once with the fourth order implicit dissipation raised, and once lowered by about 10%. Both trials resulted in an increase in convergence time.

Chapter 5

Results

The fully optimized code was used to solve all six cases. Full multigrid was used along with both a sawtooth cycle and a W-cycle, using four smoothing passes on three levels. As a reference, ARC2D was executed using three level grid sequencing. Fifty iterations were performed on the first two levels, which seemed to be optimal for most cases. Parameters were not adjusted on a case by case basis for either ARC2D or ARC2D-MG. Both codes share the following parameters and settings:

- Geometry based variable time stepping (JACDT = 1)
- $\Delta t_{ref} = 5.0$
- Wake cut solved implicitly (CMESH = TRUE)
- Viscosity only in normal direction (VISETA = TRUE, VISXI = FALSE)
- Circulation correction on (CIRCU = TRUE)

The convergence histories of both the coefficient of lift and the L_2 norm of the residual are shown in figures 5.1 to 5.6. Table 5.1 shows how many times faster ARC2D-MG is compared to ARC2D. Each number is the time required by the grid sequenced solution divided by the time required by a W-cycle with FMG. It compares them at four instances. The first two are found when the coefficient of lift settles within 0.25% and 0.1% of its final value. The last two show when residuals of less than 10^{-10} and 10^{-12} are achieved.

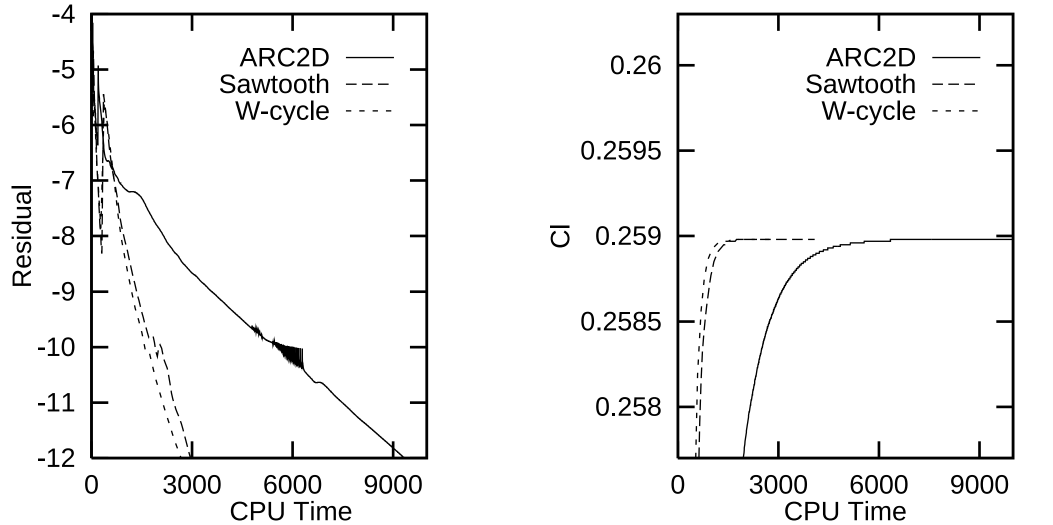

Figure 5.1: Lift and residual convergence histories, case one

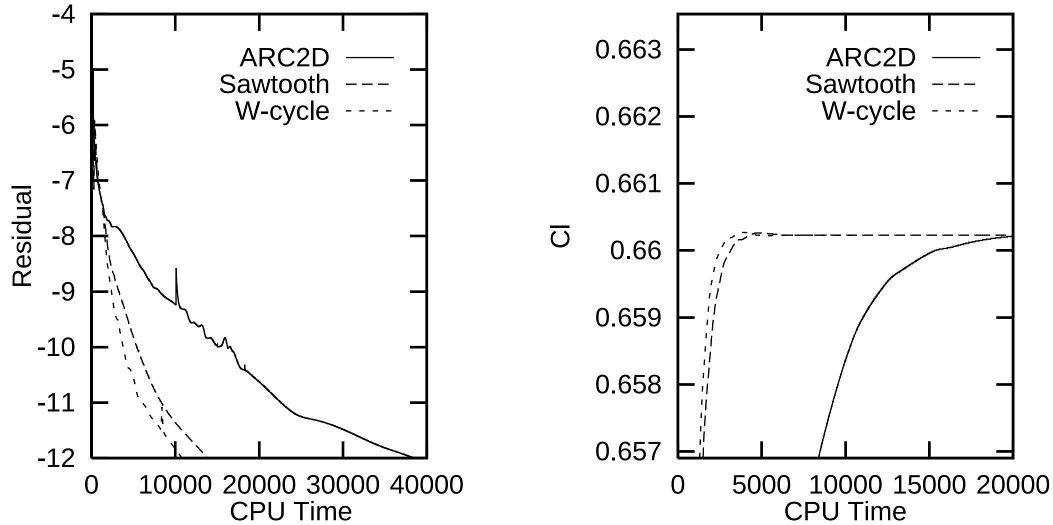

Figure 5.2: Lift and residual convergence histories, case two

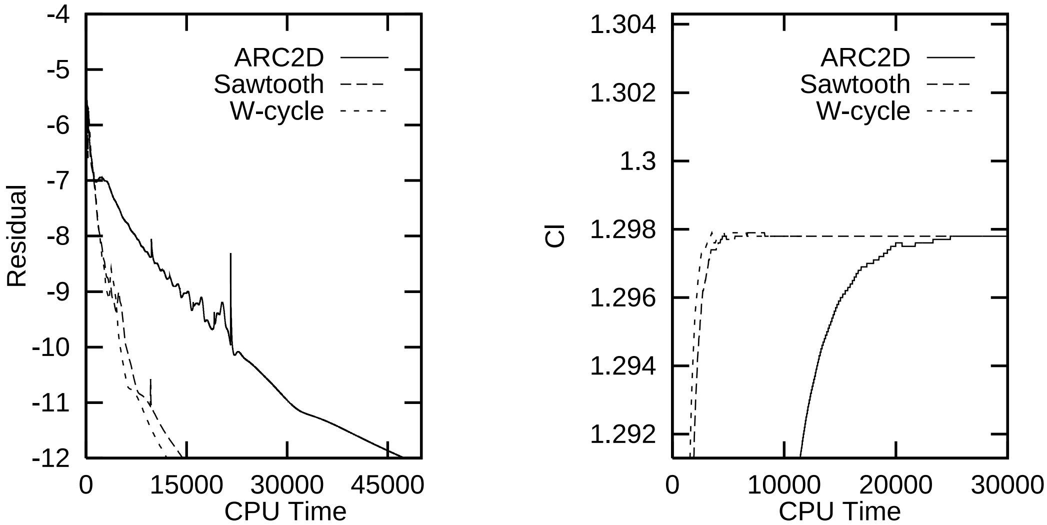

Figure 5.3: Lift and residual convergence histories, case three

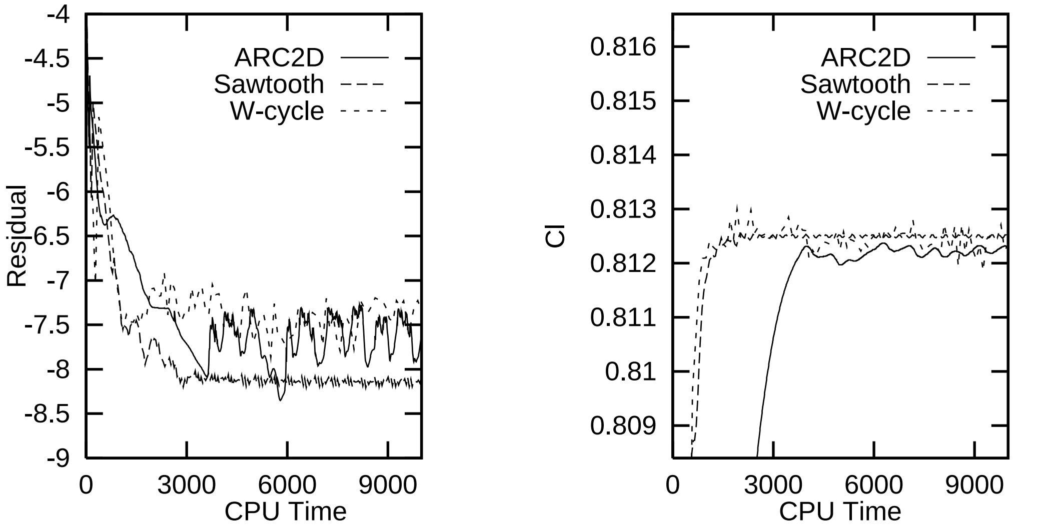

Figure 5.4: Lift and residual convergence histories, case four

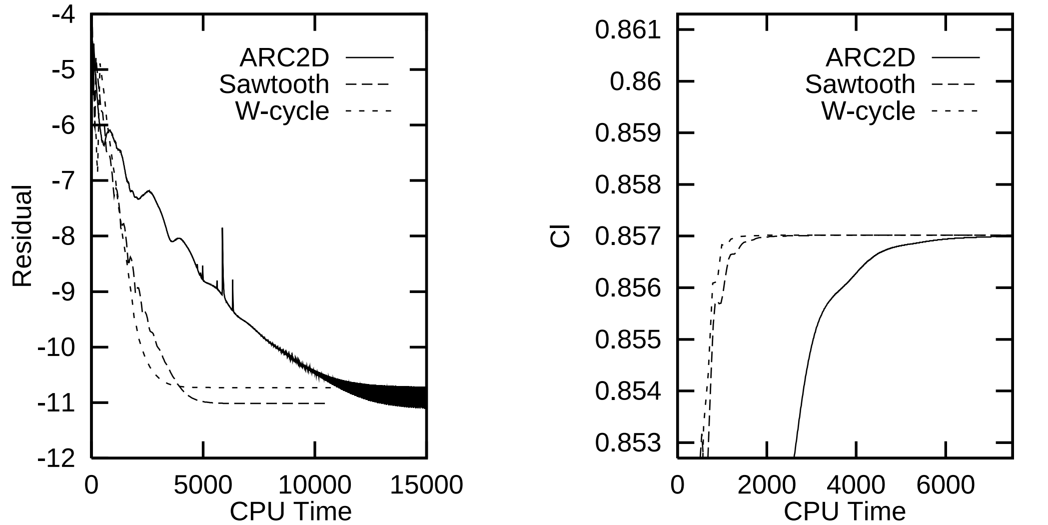

Figure 5.5: Lift and residual convergence histories, case five

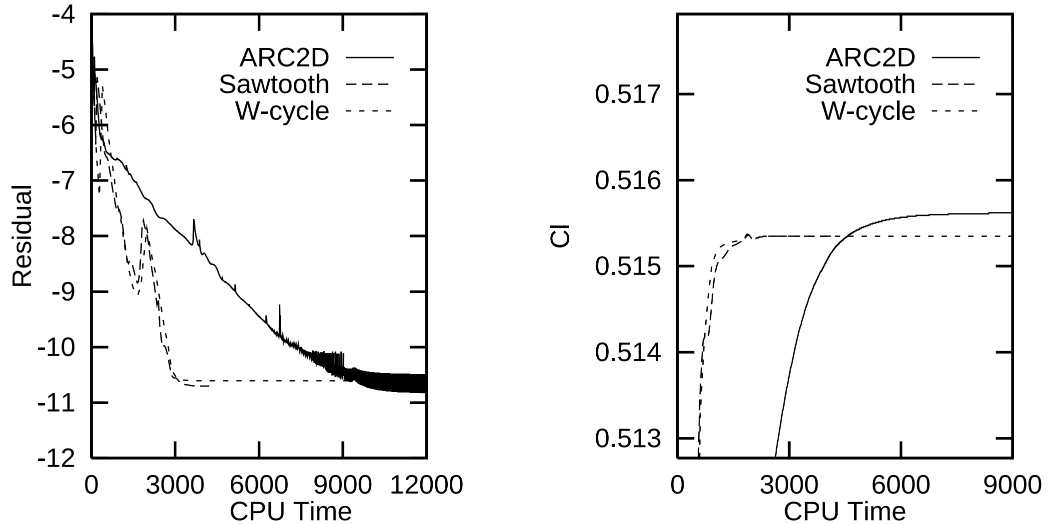

Figure 5.6: Lift and residual convergence histories, case six

Case	Cl<.25%	Cl<.1%	R< 10^{-10}	R< 10^{-12}
1	3.8	4.2	3.5	3.2
2	6.6	6.3	4.3	3.5
3	6.5	6.3	4.2	3.9
4	4.9	4.1	—	—
5	4.4	4.4	3.6	—
6	5.5	4.9	2.7	—

Table 5.1: Speedup of lift and residual convergence

5.1 Discussion

Due to the fact that the right hand side is the same in both ARC2D and ARC2D-MG, it is expected that fully converged solutions, and therefore the coefficients of lift, will be the same. This has happened in all cases except four and six, where total convergence was not achieved. The turbulence model is likely to blame for the failure to fully converge. Case five also does not converge. If the residual histories of these three cases is examined, an advantage of the sawtooth cycle becomes apparent. It consistently converges more completely than either the W-cycle or ARC2D. The solution seems to be more steady, since the fluctuations of the residual and lift coefficient seen in case four are much less pronounced when a sawtooth cycle is used.

Table 5.1 shows that multigrid is indeed providing substantial speedups. If the first two columns are compared to the second two, it is obvious that the coefficient of lift is benefitted more than the residual. This indicates that multigrid is working as expected, because lift is more sensitive to low frequency error, which is being removed more rapidly.

Chapter 6

Conclusions

Table 5.1 clearly shows that multigrid works well with the approximately factored method. If the coefficient of lift convergence is used as an indicator, then the time to convergence of ARC2D-MG is between one-quarter and one-sixth of the time required by ARC2D. Multigrid is most effective with the shock-free cases. This may be explained by considering the initial error of these cases compared to the transonic cases. Because there is no shock, the frequency components of the error tend to be comprised of lower frequencies, which multigrid is adept at handling.

The optimized parameters of ARC2D are best left alone when multigrid is applied. Four smoothing passes should be performed at each level, and only on the way down the cycle. In general, a W-cycle gives the fastest results. Multigrid will work best when the high and low frequency errors are damp at the same rate. The W-cycle is fastest likely because it balances these rates. However, the sawtooth cycle gives convergence rates very close to the W-cycle, and tends to converge further in the cases where the residual does not go to machine zero.

Appendix A

List of Symbols

A.1 Vectors

After spatial discretization, implicit time differencing, local time linearization, approximate factorization, and diagonalization, the Euler or Navier-Stokes equations take the following matrix form:

$$A\Delta\hat{Q} = \hat{S} \tag{A.1}$$

A	the matrix left hand side
$\Delta\hat{Q}$	Unknown vector
$\hat{S}$	Right hand side

When a multigrid routine is to be used, the following vectors are also necessary:

$\hat{D}$	Driving vector
$\hat{R}$	Right hand side plus driving vector

During a multigrid routine, $\hat{R}$ is sent to the smoothing routine instead of $\hat{S}$.

A subscript on any of the above vectors indicates what level the vector is defined on. Level one is the finest grid.

$\hat{Q}_k^0$	The flow variables on level k	Before smoothing
$\hat{Q}_k^i$		After smoothing
$\hat{Q}_k^+$		After correction
$\hat{S}_k^0$	The right hand side	Before smoothing
$\hat{S}_k^i$		After after smoothing
$\hat{R}_k^0$	Right hand side plus driving vector	Before smoothing
$\hat{R}_k^i$		After smoothing
$\hat{R}_k^r$		After smoothing and restriction
$\hat{D}_k$	Driving vector for level k	

A.2 Operators

RSTRCT	Restrict. Operates on $\hat{Q}$, $\hat{R}$, and $\hat{S}$.
PRLNG	Prolong. Operates on $\hat{Q}$.
RHS	Make right hand side. Operates on $\hat{Q}$ and creates $\hat{S}$.
SMTH	Perform one smoothing pass. Operates on $\hat{Q}$ and $\hat{R}$.

A.3 Relations

$$
\begin{aligned}
\hat{Q}_{k+1}^0 &= R(\hat{Q}_k^i) \\
\hat{S}_k^0 &= RHS(\hat{Q}_k^0) \\
\hat{S}_k^i &= RHS(\hat{Q}_k^i) \\
\hat{R}_k^0 &= \hat{S}_k^0 + \hat{D}_k \\
\hat{R}_k^i &= \hat{S}_k^i + \hat{D}_k \\
\hat{R}_k^r &= R(\hat{R}_k^i) \\
\hat{D}_k &= \hat{R}_k^r - \hat{S}_k^0 \\
\hat{D}_1 &= \hat{0}
\end{aligned}
$$

Appendix B

Multigrid Pseudo-code

This detailed multigrid process is given in pseudo-code. Since the method lends itself to recursion, the main routine, which handles the multigridding, will be presented as a single recursive routine, called MG.

- function MG($\hat{Q}_k^0$,$\hat{D}_k$)
- $\hat{Q}_k^i = \hat{Q}_k^0$
- Do i = 1, subiterations-down
  - $\hat{Q}_k^i$ =SMTH($\hat{Q}_k^i, RHS(\hat{Q}_k^i - \hat{D}_k)$
  - Call BC($\hat{Q}_k^i$)
- Enddo
- If (lowest-level = TRUE) then
  - $\hat{Q}_k^+ = \hat{Q}_k^i$
- Else
  - $\hat{Q}_{k+1}^0 = R(\hat{Q}_k^i)$
  - Call BC($\hat{Q}_{k+1}^0$)
  - $\hat{S}_{k+1}^0$ =RHS($\hat{Q}_{k+1}^0$)
  - $\hat{R}_k^r$ =RSTRCT($RHS(\hat{Q}_k^i) + \hat{D}_k$)
  - $\hat{D}_{k+1} = \hat{R}_k^r - \hat{S}_{k+1}^0$
  - $\hat{Q}_{k+1}^+$ =MG($\hat{Q}_{k+1}^0, \hat{D}_{k+1}$)
  - $\hat{Q}_k^{ii} = \hat{Q}_k^i + P(\hat{Q}_{k+1}^+ - \hat{Q}_{k+1}^0)$

- Call BC($\hat{Q}_k^{ii}$)
- Do i = 1, subiterations-up
  * $\hat{Q}_k^{ii}$ =SMTH($\hat{Q}_k^{ii}, RHS(\hat{Q}_k^{ii}) - \hat{D}_k$)
  * Call BC($\hat{Q}_k^{ii}$)
- Enddo
- If (W-Cycle = TRUE) then
  * $\hat{Q}_{k+1}^0 = R(\hat{Q}_k^{ii})$
  * Call BC($\hat{Q}_{k+1}^0$)
  * $\hat{S}_{k+1}^0$ =RHS($\hat{Q}_{k+1}^0$)
  * $\hat{R}_k^r$ =RSTRCT($RHS(\hat{Q}_k^{ii}) + \hat{D}_k$)
  * $\hat{D}_{k+1} = \hat{R}_k^r - \hat{S}_{k+1}^0$
  * $\hat{Q}_{k+1}^+$ =MG($\hat{Q}_{k+1}^0, \hat{D}_{k+1}$)
  * $\hat{Q}_k^+ = \hat{Q}_k^{ii}$+PRLNG($\hat{Q}_{k+1}^+ - \hat{Q}_{k+1}^0$)
  * Call BC($\hat{Q}_k^+$)
- Else
  * $\hat{Q}_k^+ = \hat{Q}_k^{ii}$
- Endif
- Return $\hat{Q}_k^+$

A few variables above should be explained. 'subiterations-down' and 'subiterations-up' give the number of smoothing passes on the way down and up, respectively. 'lowest-level' is a flag indicating whether the current level, k, is the coarsest grid. 'w-cycle' indicates the cycle used is a W-cycle, or either sawtooth or V-cycle. It will be a sawtooth if 'subiterations-up' is equal to zero. The subroutine 'BC()' calls the boundary condition routine of ARC2D on its argument.

Bibliography

[1] L. Martinelli. *Calculations of Viscous Flows with a Multigrid Method.* PhD thesis, Princeton University, October 1987.

[2] C. M. Maksymiuk R. C. Swanson and T. H. Pulliam. A comparison of two central difference schemes for solving the Navier-Stokes equations. Technical report, NASA TM-102815, July 1990.

[3] S. J. Hulshoff. *An Euler Solution Algorithm for Steady Helicopter-Rotor Flows.* PhD thesis, University of Toronto, 1994.

[4] A. Arnone and R. C. Swanson. A Navier-Stokes solver for cascase flows. Technical report, NASA CR-181682, July 1988.

[5] A. Jameson and S. Yoon. Multigrid solution of the Euler equations using implicit schemes. *AIAA Journal*, 24(11):1737–1743, November 1986.

[6] D. A. Caughey. Diagonal implicit multigrid algorithm for the Euler equations. *AIAA Journal*, 26(7):841–851, July 1988.

[7] J. A. O'Callahan and D. S. Thompson. A comparison of implicit multigrid Euler solvers implemented on a multiprocessor computer, June 1991. AIAA paper 91-1580-CP.

[8] L. Wang and D. A. Caughey. Multiblock/multigrid Euler method to simulate 2d and 3d compressible flow, January 1993. AIAA paper 93-0332.

[9] R. R. Varma and D. A. Caughey. Diagonal implicit multigrid solution of compressible turbulent flows, June 1991. AIAA paper 91-1571-CP.

[10] D. Jespersen, T. Pulliam, and P. Bunting. Recent enhancements to overflow, January 1997. AIAA paper 97-0644.

[11] T. H. Pulliam. Efficient solution methods for the Navier-Stokes equations. Lecture Notes for The Von Karman Institute, January 1986.

[12] A. Jameson. *Multigrid Algorithms for Compressible Flow Calculations.* Princeton University.

[13] D. W. Zingg. Grid studies for thin-layer Navier-Stokes computations of airfoil flowfields, January 1992. AIAA paper 92-0184.

[14] D. Jespersen. A multigrid algorithm. Private Communication, March 1996.