VII. Time Accuracy, Steady States, Convergence and Stability

7.1 Time Accuracy vrs Steady-State Computation

The implicit algorithm is designed to be time accurate where second order accuracy can be maintained and the equations are integrated through time from some meaningful initial condition to the the solution at time T. In this case the time step is chosen to be commensurate with some time scale of the problem. The evolution of the solution through time is physically realistic and good solution accuracy is dependent on the mesh spacing and boundary conditions.

The equations can also be applied to steady-state problems. Typically we employ the first order scheme in time and attempt to accelerate the algorithm by various non-time-like maneuvers. The equations are then integrated from an arbitrary initial condition to a time asymptotic state. Any procedure which drives us to the steady-state faster must also be stable and accurate at convergence. It might be expected that large time steps could be used to drive the solution to the steady-state faster. As we shall see, based on linear analysis large time steps can increase the convergence rate, but for factored forms the limit of the amplification factor (a measure of the maximum convergence rate) as $h = \Delta t \to \infty$ is 1.

A. Effect of Factorization Errors On Convergence

Let us divide the total solution into the transient (time-like) and particular (steady-state) parts. The goal of any fast steady-state algorithm is to eliminate the transient as quickly as possible. We can examine the ability of the implicit scheme to eliminate transients by investigating the model problem, Eq. (5.12). In this case, instead of Eq. (5.14) we take

$$w = \widetilde{w}(t)\, e^{i\beta\, x} \tag{7.1}$$

and treat the spatial derivative ∂_x analytically, then examine the temporal differencing schemes in one and two dimensions. This gives us the purely transient one dimensional model problem

$$\widetilde{w}_t + \lambda_x\, \widetilde{w} = 0 \tag{7.2}$$

with $\lambda_x = i\beta\lambda$.

The delta form of the first order implicit algorithm is

$$(1 + h\,\lambda_x)\, \Delta\widetilde{w}^n = -h\lambda_x\, \widetilde{w}^n \tag{7.3}$$

which can be rewritten as

$$\begin{gathered} \widetilde{w}^{n+1} = \left[\frac{1}{(1+h\lambda_x)}\right] \widetilde{w}^n \\ \text{or} \\ \widetilde{w}^n = \left[\frac{1}{(1+h\lambda_x)}\right]^n \widetilde{w}^0 \end{gathered} \tag{7.4}$$

where $\widetilde{w}^0$ is some initial value. The term in the brackets is the amplification factor, σ.

For $h \to \infty$, $\widetilde{w}^n \to 0$ and the transient can actually be eliminated directly for large h.

In contrast, let us examine a two-dimensional factored implicit scheme for the two-dimensional transient problem

$$\widetilde{w}_t + \lambda_x \widetilde{w} + \lambda_y \widetilde{w} = 0 \; . \tag{7.5}$$

This is the two-dimensional counterpart to Eq. (7.2). Applying the first order implicit approximate factorization delta algorithm to Eq. (7.5) we have

$$[1 + h\,\lambda_x]\;[1 + h\,\lambda_y]\;\Delta\widetilde{w}^n = -h\;(\lambda_x + \lambda_y)\;\widetilde{w}^n \tag{7.6}$$

Expanding $\Delta\widetilde{w}^n = \widetilde{w}^{n+1} - \widetilde{w}^n$ and combining terms we have

$$\widetilde{w}^n = \left[\frac{\left(1 + h^2\;\lambda_x\lambda_y\right)}{\left(1 + h\;\lambda_x + h\;\lambda_y + h^2\;\lambda_x\lambda_y\right)}\right]^n \widetilde{w}^0 \tag{7.7}$$

and so $|\sigma| \to 1$ as $h \to \infty$.

A close examination of this result shows that the factorization has destroyed the good convergence characteristics at large time steps. The factoring error term has introduced a h^2 term in the numerator of the amplification factor. Therefore the factored schemes do not have good convergence characteristics for large time steps. Actually, there is a range of moderately large time steps where the amplification factor is a minimum, see for instance Abarbanel, Dwoyer, and Gottlieb [29]. Convergence can therefore be accelerated by using a time step which minimizes the amplification factor.

Note that for the delta form of the algorithm (either factored or unfactored) the steady-state solution is independent of the time step, h. (There are numerical schemes where this is not the case, such as Lax-Wendroff.) Therefore, the time step path to the steady-state does not affect the final solution and we can envision using time step sequences or spatially variable time steps to accelerate convergence.

B. Space Varying Δt

Manipulation of the time step can have a substantial influence on convergence even within the framework of the factored algorithms. If only a steady state solution is required, one can let h (or Δt) change in space. This approach can be view as a way to condition the iteration matrix of the relaxation scheme defined via Eq. (5.22) or Eq. (5.26). Use of a space varying Δt can also be interpreted as an attempt to use a more uniform Courant number throughout the field. In any event, changing Δt can be effective for grid spacings that vary from very fine to very coarse

- a situation usually encountered in aerodynamic simulations where grids contain a wide variety of length scales.

A space varying Δt has been used in both explicit and implicit schemes (e.g. Shang and Hankey [30], McDonald and Briley [31], Shirnivasan et al [32], Coakley [33], Jameson [6], etc). As a rule one wishes to adjust Δt at each point proportional to the grid spacing and the characteristic speed of the flow. Something like the Courant number restriction (which for the Euler equations in multi-dimensions is a bit of an approximation).

For highly stretched grids the space variation of the grid is the most important parameter to scale with. In subsonic and transonic flow the characteristic speeds have moderate variation and we have found that a purely geometric variation of Δt is adequate, specifically

$$\Delta t = \frac{\Delta t|_{ref}}{1. + \sqrt{J}} \tag{7.8a}$$

To illustrate the advantage of using a variable time step, Fig. 6 shows the degradation in convergence rate when a constant step size is substituted for the variable time step in a NACA 0012 test case. For this comparison all other possible parameters were held constant and no other changes were employed. We should note at this time that the above variation of time step has often worked poorly in viscous flow until the numerical dissipation terms were also put in implicitly as described later. Also other forms of the variable step size sometimes perform better than Eq. (7.8a1), for example

$$\Delta t = \frac{\Delta t|_{ref}}{|U| + |V| + a\sqrt{\xi_x^2 + \xi_y^2 + \eta_x^2 + \eta_y^2}} \tag{7.8b}$$

which is approximately a constant CFL condition. However, Eq. (7.8b) is more costly to compute then Eq. (7.8a).

C. Mesh Sequences

For inviscid airfoil calculations on a grid of O(250 x 50) practical convergence is usually obtained in 500-600 fine grid iterations when the flow field has been started from an initial condition of uniform free stream flow. Typically the first 100 to 200 iterations on the fine mesh are needed to get past the initial transients which can be a substantial portion of the total solution time. For instance, in the above test case it takes on the order of 600 fine grid iterations for a tight convergence criteria (e.g. lift to 5 decimal places) , 200 of which are spent on clearing out the impulsive start. One way to accelerate convergence to a steady state is to obtain a good initial guess for a fine mesh by first iterating on a sequence of coarse grids and then interpolating the solution up to the next refined grid. Such a mesh sequence procedure can often reduce the amount of time required to obtain a solution to plotable accuracy by

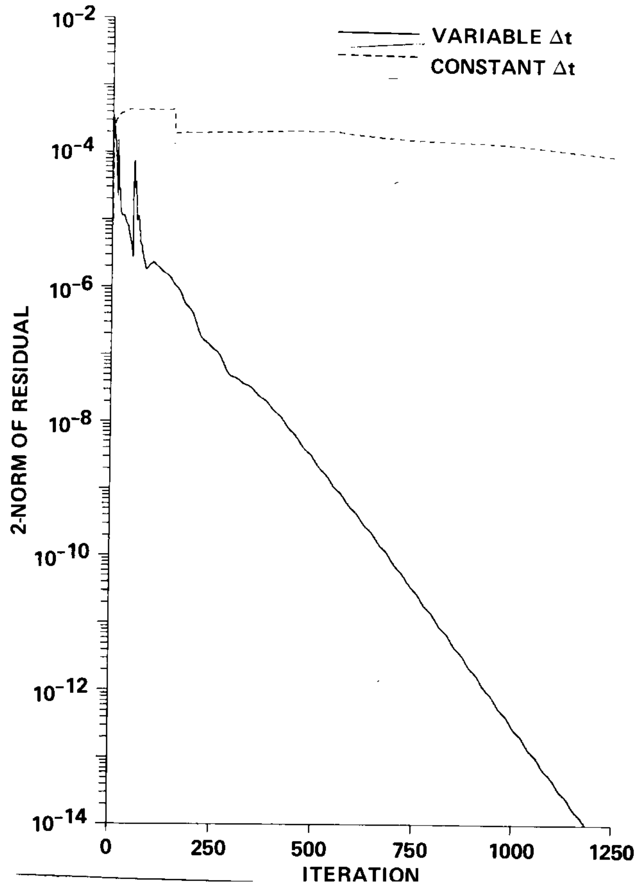

FIGURE 6. Convergence Improvement Due to Variable Time Step.

a factor of two. Also, because a coarse grid tends to damp high frequency waves, using a mesh sequence procedure can improve the overall robustness of the code.

A mesh sequencing procedure can be implemented in an optionally called stand alone routine. If a sequence of m grids are used, a coarsened grid is cut from each previous grid by halving the number of points in the ξ-direction and by regenerating a new η-distribution of points in the η-direction using a fewer number of points. The η-distribution is regenerated because in viscous flows the spacing near the wall would be too coarse if the halving procedure is used. A finite number of iterations (perhaps 50) are carried out on each coarsened grid at which point the approximate solution is interpolated onto a more refined grid. The finest grid is then iterated to convergence. The result is faster convergence to practical levels and a more robust starting procedure.

For a NACA 0012 test case a sequence of 3 grids has been used; 48 by 18 and

96 by 25 and the final grid of 192 by 33 points. The convergence of C_l is shown in Fig. 7 to indicate the overall improvement in convergence due to the use of mesh sequencing in comparison to the use of a fine grid only. Both cases were started with a free stream initial condition.

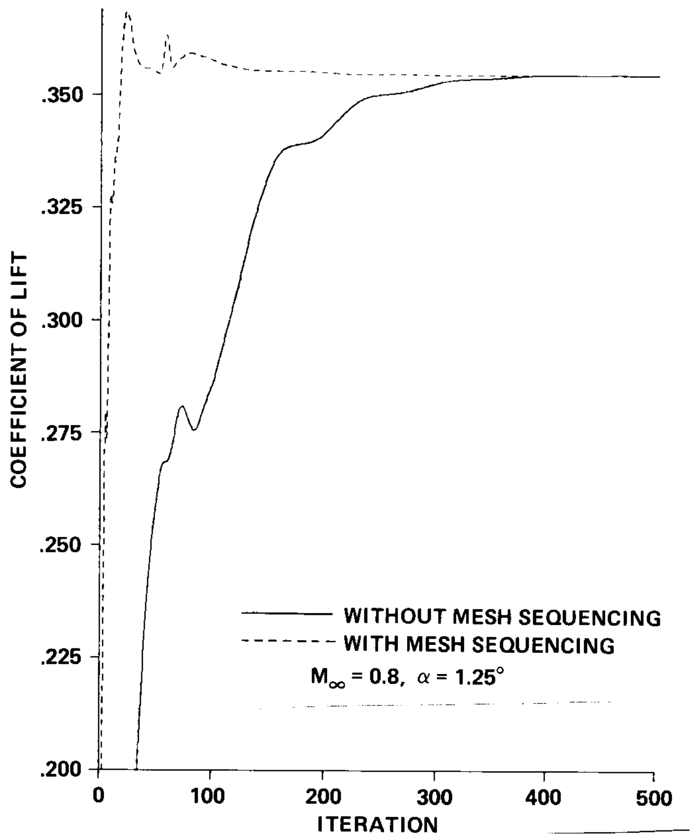

FIGURE 7. Improvement In Total Convergence of Lift Due to Mesh Sequencing.

7.2 Effect of Dissipation Model on Convergence and Stability

As discussed in Section VI., based on linear theory the use of explicit dissipation produces an explicit stability bounds unless implicit dissipation is added. The second-difference dissipation, Eq. (6.2b), will stabilize the fourth-difference dissipation if the coefficients are chosen properly. Ideally though, it would be better to treat the explicit dissipation in a fully implicit manner. That is, use implicit fourth-difference dissipation which is an exact linearization of the explicit fourth-difference dissipation. In fact, although the implicit second-difference dissipation stabilizes the fourth-difference dissipation it can have a detrimental effect on the convergence rates of an implicit algorithm for steady-state computations.

Consider the model problem in one-dimension (equivalent to Eq.(5.12) with a convenient change in notation),

$$q_t + aq_x = 0 \tag{7.9}$$

Applying the first-order time accurate Euler implicit scheme in delta form to Eq. (7.9) and adding explicit fourth-difference dissipation ($\beta_4 > 0$), implicit second-difference dissipation ($\alpha_2 > 0$), and implicit fourth-difference dissipation ($\alpha_4 > 0$) gives the algorithm

$$\left[1 + ha\delta_x - h\alpha_2\nabla_x\Delta_x + h\alpha_4(\nabla_x\Delta_x)^2\right](q^{n+1} - q^n) = -h\left(a\delta_x + \beta_4(\nabla_x\Delta_x)^2\right)q^n \tag{7.10}$$

Fourier analysis using $q^n = w^n e^{ik_j j\Delta x}$ (with k_j the wave number in x) produces

$$\left[1 + ha\lambda_x - h\alpha_2\mu_x + h\alpha_4\mu_x^2\right](w^{n+1} - w^n) = -h\left(a\lambda_x + \beta_4\mu_x^2\right)w^n \tag{7.11}$$

where $\lambda_x = 2i\sin(k_j\Delta x)/\Delta x$ represents the Fourier signature for the central difference δ_x, $\mu_x = -2 + 2\cos(k_j\Delta x)$ the signature of the second-difference dissipation operator $\nabla_x\Delta_x$, and μ_x^2 for the fourth-difference dissipation.

The amplification factor for $w^{n+1} = \sigma w^n$ is then

$$\sigma = \frac{1 + h\left((\alpha_4 - \beta_4)\mu_x^2 - \alpha_2\mu_x\right)}{1 + h\left(a\lambda_x - \alpha_2\mu_x + \alpha_4\mu_x^2\right)} \tag{7.12}$$

The choices which will be investigated are

1. $\beta_4 \neq 0$ and $\alpha_4 = \alpha_2 = 0$, explicit dissipation only.
2. $\beta_4 \neq 0$, $\alpha_2 \neq 0$, and $\alpha_4 = 0$, explicit fourth-difference dissipation and implicit second-difference dissipation, no implicit fourth-difference dissipation.
3. $\beta_4 \neq 0$, $\alpha_4 \neq 0$ and $\alpha_2 = 0$, explicit and implicit fourth-difference dissipation with no implicit second-difference dissipation.

For case 1, explicit dissipation only, Eq. (7.12) becomes

$$\sigma = \frac{1 - h\beta_4\mu_x^2}{1 + ha\lambda_x} \tag{7.13a}$$

Now, since λ_x is pure imaginary and has a minimum of 0, and $-4 \leq \mu_x \leq 0$ the explicit stability bound is $h\beta_4 < \frac{1}{8}$. This is a limit on the product of h and β_4 and therefore one can always find a combination which will be stable. But, for arbitrary h, especially in the case where large h are used to accelerate convergence, this bound is too restrictive.

In the second case, implicit second-difference dissipation can eliminate the above stability bound. The amplification factor σ is now

$$\sigma = \frac{1 - h\beta_4\mu_x^2 - h\alpha_2\mu_x}{1 + ha\lambda_x - h\alpha_2\mu_x} \tag{7.13b}$$

The numerator term λ_x can only improve the stability bounds since it is pure imaginary, so it is taken at its minimum, 0. Let $\alpha_2 = 2\beta_4$ and apply the stability condition $|\sigma| \leq 1$ which results in the condition $-2 \leq -h\beta_4\mu_x(4 + \mu_x)$. Since $\mu_x \leq 0$, the condition can be rewritten as $-2 \leq h\beta_4|\mu_x|(4 + \mu_x)$ which is satisfied because $-4 \leq \mu_x$. Therefore, using $\alpha_2 = 2\beta_4$ leads to unconditional stability. The disadvantage of this is form is evident from the amplification factor, Eq. (7.13b). Even though the scheme is unconditionally stable, $\sigma \to 1$ as $h \to \infty$. In fact, the amplification factor has a minimum at a finite h and then asymptotes rapidly to 1 as h increases. For this reason, large h cannot be used to accelerate convergence even in this simple one-dimensional example.

In contrast, the third case of implicit and explicit dissipation is unconditionally stable and has good convergence characteristics for large h. The amplification factor σ for $\alpha_4 = \beta_4$ and $\alpha_2 = 0$ is

$$\sigma = \frac{1}{1 + h\left(a\lambda_x + \alpha_4\mu_x^2\right)} \tag{7.13c}$$

which is unconditionally stable and $\sigma \to 0$ as $h \to \infty$.

The analysis for two and three dimensions is straightforward and gives similar results for the unfactored forms. The optimal algorithm is a fully implicit one. In general, optimal stability and convergence only occurs for the fully implicit form of the algorithm.

We demonstrate the improved convergence and stability bounds in Fig. 8.

The curves in Fig. 8 are convergence histories for a transonic airfoil computation showing the effect of a fully implicit treatment of the artificial dissipation. The upper curve is the result of second order constant coefficient implicit dissipation, Eq. (6.2b), with nonlinear explicit dissipation, Eq. (6.12). A much faster convergence rate is obtained in this problem when the second order implicit dissipation is replaced by an implicit linearization of the nonlinear dissipation of Eq. (6.12), see Ref. [28] for more details. Also the maximum allowable time step is at least 10 times larger for the fully implicit scheme.

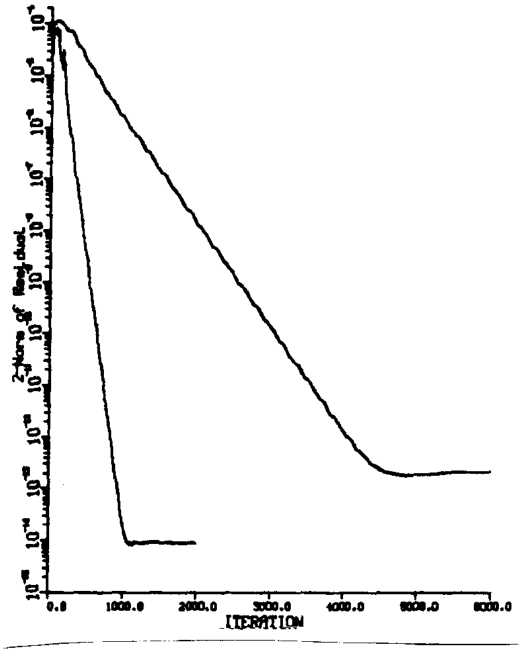

FIGURE 8. Improvement in Convergence Rate Due to Implicit Treatment of Artificial Dissipation.

VIII. ARC2D - ARC3D Algorithms

General purpose centrally space differenced implicit finite difference codes in two [2] and three [3] dimensions have been developed at NASA Ames and have been widely distributed since their introduction in 1977 and 1978. These codes, now referred to ARC2D and ARC3D, can run either in inviscid or viscous mode for steady or unsteady flow. They use general coordinate systems and can be run on any smoothly varying curvilinear mesh, even a mesh that is quite skew. Because they use well ordered finite difference grids, the codes can take advantage of vectorized computer processors and have been implemented for the Control Data 205 and the CRAY 1-S and X-MP. On a single processor of the X-MP a vectorized version of the code runs approximately 20 times faster than the original code which was written for the Control Data 7600.

Traditionally gains in computational efficiency due to improved numerical algorithms have kept pace with gains due to increased computer power. Since the ARC2D and ARC3D codes were introduced, a variety of algorithmic changes have been individually tested and have been shown to improve overall computational efficiency. These include use of a spatially varying time step (Δt), use of a sequence of mesh refinements to establish approximate solutions, implementation of various ways to reduce inversion work, improved numerical dissipation terms, and more implicit treatment of terms. Although the various individual algorithm improvements can interact with each other, sometimes adversely making optimization difficult, their combined effect has lead to an order of magnitude gain in computational efficiency for steady state applications. This is a gain equivalent to that achieved with computer hardware. Unsteady flow calculations have also benefited from some of the above improvements.

We now summarize the two basic algorithms used in the code ARC2D, see Section VIII for details of ARC3D. The standard algorithm is used mainly for time accurate calculations. The equations are reproduce from Eq. (5.22)

$$\begin{aligned} \left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n - hRe^{-1}\delta_\eta \widehat{M}^n\right]\Delta\widehat{Q}^n &= \widehat{R}^n \\ \widehat{R}^n &= -h\left[\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n - Re^{-1}\delta_\eta \widehat{S}^n\right] \end{aligned} \tag{8.1}$$

This scheme consists of first forming the right hand side, $\widehat{R}^n$ then performing two block tridiagonal inversions. Central differences are used for the flux and Jacobian differences. The dissipation models used are the implicit second order, Eqs. (6.1b), added to the appropriate implicit operator to keep the band width tridiagonal and the explicit nonlinear term Eq. (6.12). Since this scheme is used for time accurate calculation the typical time step will be small enough to assure stability even though the explicit dissipation operator is not properly linearized. The advantage of this scheme is time accuracy while the disadvantage is substantial computational work.

In most instances we are interested in steady state computations. In that case we can take advantage of simplifications such as the diagonal algorithm as long as the scheme converges and we do not distort the steady state equations. The diagonal algorithm used in ARC2D is

$$T_\xi\left[I + h\,\delta_\xi\,\Lambda_\xi\right]\widehat{N}\left[I + h\,\delta_\eta\,\Lambda_\eta\right]T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \tag{8.2}$$

In this case we always employ the nonlinear dissipation models, Eq. (6.12) with a linearization of the terms which necessitates the use of scalar pentadiagonal solvers. (Note that the implicit artificial dissipation terms are placed inside the bracketed terms and are operated on by the similarity transformations). This form of the diagonal scheme gives us a very efficient code in terms of computational work and enhanced stability and convergence due to the proper linearization of the explicit steady state equations. The time accuracy though is at most first order. We also employ the variable time step Eq. (7.8) and mesh sequencing to accelerate convergence.

We shall employ ARC2D and ARC3D to demonstrate various aspects of algorithm improvements, accuracy, and application in the remainder of these notes.