Development of Implicit Methods in CFD NASA Ames Research Center 1970's - 1980's

Thomas H. Pulliam

NASA Ames Research Center

Abstract

The focus here is on the early development (mid 1970's-1980's) at NASA Ames Research Center of implcit methods in Computational Fluid Dynamics (CFD). A class of implicit finite difference schemes of the Beam and Warming approximate factorization type will be addressed. The emphasis will be on the Euler equations. A review of material pertinent to the solution of the Euler equations within the framework of implicit methods will be presented. The eigensystem of the equations will be used extensively in developing a framework for various methods applied to the Euler equations. The development and analysis of various aspects of this class of schemes will be given along with the motivations behind many of the choices. Various acceleration and efficiency modifications such as matrix reduction, diagonalization and flux split schemes will be presented.

1. Introduction

The development employed here is based on the implicit approximate factorization algorithm of Beam and Warming [1]. A particular application in two dimensions was first presented by Steger [2] and for three dimensions by Pulliam and Steger [3]. While there have been numerous developments and variant implementations of implicit methods over the years, the original work of Beam, Warming and Steger stands out as the backbone of modern methods. I shall concentrate here on the theoretical development, application and assessment of the implicit algorithms of the Beam-Warming-Steger type

Email address: `Thomas.H.Pulliam@nasa.gov` (Thomas H. Pulliam)

as they were developed at NASA Ames Research Center in the early 1970's to the 1980's.

There are a number of considerations to weigh when choosing a numerical algorithm to apply to a set of partial differential equations. If we restrict ourselves to finite difference schemes then the possibilities are narrowed somewhat to the two classical approaches for time integration, explicit and implicit techniques. The merits of these have been extensively discussed in the literature. Explicit methods typically require less computational work and are simpler both in derivation, application and parallelization. Implicit methods, while computationally expensive, have less stringent stability bounds (classical stability analysis shows unconditional stability but in practice on nonlinear problems bounds are encountered).

Implicit numerical schemes are usually chosen because we wish to obtain solutions which require fine grid spacing for numerical resolution, and we do not want to limit the time steps by employing a conditionally stable explicit scheme. Explicit schemes are very useful and schemes such as MacCormack's explicit algorithm [4] were used extensively in the 1970's - 1980's. The extra work required for an implicit scheme is usually offset by the advantages obtained by the increased stability limits, and in general implicit schemes have been very useful and successful for a variety of inviscid and viscous flowfield calculations.

For unsteady, transient problems we wish to employ time accurate methods, initialize the flow with some realizable state and integrate forward in time with time steps commensurate with the unsteady phenomenon being simulated. Both implicit and explicit methods are capable of computing time accurately. In steady state calculation we wish to integrate from some arbitrary state to the asymptotic solution in any manner which will get us there with the least amount of computational work. Non-time-accurate techniques (for instance relaxation methods, variable time steps, matrix preconditioning, large time steps) can be employed as long as they are convergent and do not distort the steady state equations so as to produce inaccurate results. The methods presented below can be employed either for time accurate calculations or for steady state rapidly convergent solutions.

The algorithm to be presented is an implicit approximate factorization finite difference scheme which can be either first or second order accurate in time. Local time linearizations are applied to the nonlinear terms and an approximate factorization of the two-dimensional implicit operator is used to produce locally one-dimensional operators. This results in block tridiagonal matrices, which are easy to solve. The spatial derivative terms are approximated with finite differences of various orders of accuracy, involving stencil sizes from 3 to 9 points. Explicit and implicit artificial dissipation terms (which will not be discussed here, see Pulliam [5]) are added to achieve nonlinear stability. A spatially variable time step is used to accelerate convergence for steady-state calculations. A diagonal form of the algorithm is also discussed, which produces a computationally efficient modification of the standard algorithm where the diagonalization results in scalar tridiagonal or pentadiagonal operators in place of the block operators. This diagonal form of the algorithm produces a robust, rapid and versatile scheme for steady state calculations.

2. Euler Equations

The Euler equations are comprised of the inviscid compressible continuity, momentum and the energy equations. The Euler equations are of interest for a number of reasons. They are the next step after the potential equation in the hierarchy of equations which lead to the full Navier-Stokes equations. Besides being valid for use in applications where viscous effect are negligible, they are often used in analysis and development of algorithms which eventually are applied to the Navier-Stokes equations. Since the equations are capable of capturing, convecting and creating vorticity, they are often used to simulate vortical flows where either physical mechanisms (such as shocks) or artificial mechanisms (fixed stagnation points, numerical dissipation) account for the production of vorticity. In some cases, the resulting flows represent acceptable physical solutions and in others the validity of the Euler solution is in question relative to a more physical Navier-Stokes equation solution.

We shall restrict ourselves to the Cartesian form of the two dimensional (2-D) equations in strong conservation law form. Strong conservation law form is chosen because we wish to admit shock capturing. Typically the extension of ideas to three dimensions (3-D) is rather straightforward. It is usually a mistake to restrict oneself to just 1-D equations, since ideas developed for 1-D often are difficult to extend formally to multidimensions. In contrast, the extension from 2-D to 3-D is more easily accomplished.

2.1. Equations

The Euler equations in conservation law form are

$$\partial_t Q + \partial_x E + \partial_y F = 0 \tag{1}$$

where

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, E = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{bmatrix}, F = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{bmatrix} \tag{2}$$

Pressure is related to the conservative flow variables, Q, by the equation of state

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{3}$$

where γ is the ratio of specific heats, generally taken as 1.4. The speed of sound is c which for ideal fluids, $c^2 = \gamma p/\rho$.

$$\partial_t e + \partial_x(\rho u h) + \partial_y(\rho v h) = 0 \tag{4}$$

2.2. General Form

First, let us recast Eqs. 1 - 3 in a more general form

$$Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}, E = \begin{bmatrix} q_2 \\ q_2^2/q_1 + p(q) \\ q_2 q_3/q_1 \\ q_2\left(q_4 + p(q)\right)/q_1 \end{bmatrix}, F = \begin{bmatrix} q_3 \\ q_2 q_3/q_1 \\ q_3^2/q_1 + p(q) \\ q_3\left(q_4 + p(q)\right)/q_1 \end{bmatrix} \tag{5}$$

with

$$p(q) = (\gamma - 1)(q_4 - \frac{1}{2}\left(q_2^2 + q_3^2\right)/q_1) \tag{6}$$

In using Eqs. 5, 6 we will always assume that the q_i variables are linearly independent. This is important when we will be examining linearizations and Jacobians of the fluxes.

2.3. Flux Jacobians

The fluxes, E and F, defined in the previous section are nonlinear functions of Q. In stability analysis and design of numerical algorithms for the Euler equations, the flux Jacobians

$$A \equiv \frac{\partial E}{\partial Q}, \quad B \equiv \frac{\partial F}{\partial Q} \tag{7}$$

play a dominant role.

For example, if we attempt to use the 1st order implicit scheme (define $Q^n = Q(n \cdot \Delta t)$)

$$\frac{Q^{n+1} - Q^n}{\Delta t} + \partial_x E(Q^{n+1}) + \partial_y F(Q^{n+1}) = 0 \tag{8}$$

to integrate Eq. 1, the second and third terms are nonlinear in Q^{n+1}. We can linearize that term to 2nd order accuracy by a Taylor series expansion

$$E^{n+1} = E^n + A^n(Q^{n+1} - Q^n) + O(\Delta t^2) \tag{9}$$

and similarly for F^{n+1}, resulting in

$$\left[I + \Delta t \partial_x A^n + \Delta t \partial_y B^n\right](Q^{n+1} - Q^n) = -\Delta t \left(\partial_x E^n + \partial_y F^n\right)$$

which is now linear in the solution variable Q^{n+1}.

The easiest way to derive the flux Jacobians is to start with the general form of the fluxes given in Eq. 5. The elements of A are defined as

$$A_{i,j} = \frac{\partial E_i}{\partial q_j} \tag{10}$$

where the q_j are the independent variables. For example, $E_2 = \rho u^2 + p = q_2^2/q_1 + p(q)$ and the element $A_{2,1}$ is found to be

$$A_{2,1} = \frac{\partial E_2}{\partial q_1} = -\frac{q_2^2}{q_1^2} + \frac{(\gamma - 1)}{2}\left(\frac{q_2^2 + q_3^2}{q_1^2}\right) = \phi - u^2 \tag{11}$$

with $\phi = \frac{(\gamma-1)(u^2+v^2)}{2}$.

The Jacobian matrices for the two- dimensional Euler equations are

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \phi - u^2 & (3-\gamma)u & -(\gamma-1)v & (\gamma-1) \\ -uv & v & u & 0 \\ a_1 u & \left(\frac{\gamma e}{\rho} - \phi - (\gamma-1)u^2\right) & -(\gamma-1)uv & \gamma u \end{bmatrix} \tag{12}$$

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -uv & v & u & 0 \\ \phi - v^2 & -(\gamma-1)u & (3-\gamma)v & (\gamma-1) \\ a_1 v & -(\gamma-1)uv & \left(\frac{\gamma e}{\rho} - \phi - (\gamma-1)v^2\right) & \gamma v \end{bmatrix} \tag{13}$$

2.4. Homogeneous Property

The fluxes of the Euler equations have the very interesting and useful property of being homogeneous of degree 1, $E(sQ) = sE(Q)$. Since the fluxes are homogeneous of degree 1 they can be shown to satisfy $E = AQ$, $F = BQ$ exactly. Beam and Warming took advantage of this in the original development of their implicit approximate factorization algorithm [1]. Steger and Warming [6] also used this property as an integral part of their development of a flux split algorithm. We can use the homogeneous property here to show that, for instance,

$$\frac{\partial E}{\partial x} = \frac{\partial AQ}{\partial x} = A\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial x}Q \tag{14}$$

also

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial Q}\frac{\partial Q}{\partial x} = A\frac{\partial Q}{\partial x} \tag{15}$$

which implies that

$$\frac{\partial A}{\partial x}Q = 0 \tag{16}$$

One can verify this by using Eqs. 2 and 12,13. Similar expressions hold for any derivative of E and F. Using such relations we can form the quasilinear form of Eq. 1

$$\partial_t Q + A\partial_x Q + B\partial_y Q = 0 \tag{17}$$

2.5. Eigenvector Matrices

The flux Jacobians A and B each have real eigenvalues and a complete set of eigenvectors. Therefore, the Jacobian matrices can be diagonalized. Warming, Beam, and Hyett [7] consider a general matrix which is a linear combination of A and B,

$$\widehat{A} = \kappa_x A + \kappa_y B \tag{18}$$

The diagonalization similarity transformation is

$$\Lambda_\kappa = T_\kappa^{-1}\widehat{A}T_\kappa \tag{19}$$

where T_κ is the matrix whose columns are the eigenvectors of $\widehat{A}$ and T_κ^{-1} is the corresponding left eigenvector matrix.

$$\Lambda_\kappa = \begin{bmatrix} U & & & \\ & U & & \\ & & U + c\sqrt{\kappa_x^2+\kappa_y^2} & \\ & & & U - c\sqrt{\kappa_x^2+\kappa_y^2} \end{bmatrix} \tag{20}$$

$$T_\kappa = \begin{bmatrix} 1 & 0 & 1 & 1 \\ u & \widetilde{\kappa}_y & (u+\widetilde{\kappa}_x c) & (u-\widetilde{\kappa}_x c) \\ v & -\widetilde{\kappa}_x & (v+\widetilde{\kappa}_y c) & (v-\widetilde{\kappa}_y c) \\ \frac{\phi^2}{(\gamma-1)} & (\widetilde{\kappa}_y u - \widetilde{\kappa}_x v) & \left[\frac{\phi^2+c^2}{(\gamma-1)} + c\widetilde{\theta}\right] & \left[\frac{\phi^2+c^2}{(\gamma-1)} - c\widetilde{\theta}\right] \end{bmatrix} \tag{21}$$

$$T_\kappa^{-1} = \begin{bmatrix} (1-\phi^2/c^2) & (\gamma-1)u/c^2 & (\gamma-1)v/c^2 & -(\gamma-1)/c^2 \\ -(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v) & \widetilde{\kappa}_y & -\widetilde{\kappa}_x & 0 \\ \beta(\phi^2 - c\widetilde{\theta}) & \beta[\widetilde{\kappa}_x c - (\gamma-1)u] & \beta[\widetilde{\kappa}_y c - (\gamma-1)v] & \beta(\gamma-1) \\ \beta(\phi^2 + c\widetilde{\theta}) & -\beta[\widetilde{\kappa}_x c + (\gamma-1)u] & -\beta[\widetilde{\kappa}_y c + (\gamma-1)v] & \beta(\gamma-1) \end{bmatrix} \tag{22}$$

with $U = \kappa_x u + \kappa_y v$, $\phi^2 = \frac{1}{2}(\gamma-1)(u^2+v^2)$, and $\beta = 1/(2c^2)$, $\widetilde{\theta} = \widetilde{\kappa}_x u + \widetilde{\kappa}_y v$, and, for example, $\widetilde{\kappa}_x = \kappa_x/\sqrt{\kappa_x^2+\kappa_y^2}$.

We can recover the individual eigenvalue and eigenvector matrices for A and B by using $\kappa_x = 1, \kappa_y = 0$ for T_A, T_A^{-1}, Λ_A and $\kappa_x = 0, \kappa_y = 1$ for T_B, T_B^{-1}, Λ_B. An interesting relation exists between T_A and T_B of the form

$$\widehat{N} = T_A^{-1} T_B, \quad \widehat{N}^{-1} = T_B^{-1} T_A \tag{23}$$

where

$$\widehat{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\mu & \mu \\ 0 & \mu & \mu^2 & \mu^2 \\ 0 & -\mu & \mu^2 & \mu^2 \end{bmatrix} \quad \widehat{N}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \mu & -\mu \\ 0 & -\mu & \mu^2 & \mu^2 \\ 0 & \mu & \mu^2 & \mu^2 \end{bmatrix} \tag{24}$$

with $\mu = 1/\sqrt{2}$.

Note that the matrix $\widehat{N}$ is not a function of the flow variables and is in fact a constant matrix.

It is not possible to simultaneously diagonalize both of the flux Jacobians of the Euler equations. It is possible to simultaneously symmetrize the equations which is often useful in stability analysis. The eigenvector matrices T_κ and T_κ^{-1} will diagonalize one and symmetrize the other flux Jacobian matrix depending on the choice of κ_x and κ_y.

2.6. Flux Vector Splitting

In the early years of CFD, a number of schemes were developed based on upwind differencing. The flux split scheme of Steger and Warming [6] employed a decomposition of the flux vectors in such a way that each element can be stably differenced in an upwind fashion. These schemes all claim (with good justification) to be physically consistent since they follow in some sense the characteristics of the flow. They in general can be shown to produce sharp oscillation free shocks without added artificial dissipation. It should be noted that these schemes have an inherent amount of internal dissipation, due to the one sided differences.

The plus - minus flux split method of Steger and Warming [6] will be used here to introduce the concept of flux splitting. The approach taken is to split the eigenvalue matrix Λ of the flux Jacobians into two matrices, one with all positive elements and the other with all negative elements. Then the similarity transformations T_A or T_B are used to form new matrices A^+, A^- and B^+, B^-. Formally,

$$A = T_A \Lambda_A T_A^{-1} = T_A(\Lambda_A^+ + \Lambda_A^-)T_A^{-1} = A^+ + A^- \tag{25}$$

with

$$\Lambda_A^{\pm} = \frac{\Lambda_A \pm |\Lambda_A|}{2} \tag{26}$$

Here, $|\Lambda|$ implies that we take the absolute values of the elements of Λ. The two matrices, A^+ and A^- have by construction all positive and all negative eigenvalues, respectively.

New flux vectors can be constructed as

$$E = AQ = (A^+ + A^-)Q = E^+ + E^-, \quad F = BQ = (B^+ + B^-)Q = F^+ + F^- \tag{27}$$

The Euler equations can now be written

$$\partial_t Q + \partial_x E^+ + \partial_x E^- + \partial_y F^+ + \partial_y F^- = 0 \tag{28}$$

Different type of spatial differencing can now be used for each of the above flux vector derivatives. In the case of the $+$ terms a backward difference in space can be used (forward difference for the $-$ terms), the resulting scheme maintains linear stability for the resulting ODE.

A generalized flux vector can be defined as

$$\widehat{F} = T_\kappa \widehat{\Lambda} T_\kappa^{-1} Q \tag{29}$$

where

$$\widehat{\Lambda} = \begin{bmatrix} \widehat{\lambda}_1 & & & \\ & \widehat{\lambda}_2 & & \\ & & \widehat{\lambda}_3 & \\ & & & \widehat{\lambda}_4 \end{bmatrix}, \tag{30}$$

with $\widehat{\lambda}$ any definition of an eigenvalue.

For example, we can have $\lambda^\pm$ from Eq. 26 to get $E^\pm$, or we could use $\widehat{\lambda}_i = u : i = 1,2,3,4$ producing a E^u and $\widehat{\lambda}_i = 0 : i = 1,2$ with $\widehat{\lambda}_3 = c$, $\widehat{\lambda}_4 = -c$ producing a E^c. Note that then $E = E^u + E^c$.

The generalized flux vector, see Steger and Warming [6], is written as

$$\widehat{F} = \frac{\rho}{2\gamma} \begin{bmatrix} 2(\gamma-1)\widehat{\lambda}_1 + \widehat{\lambda}_3 + \widehat{\lambda}_4 \\ 2(\gamma-1)\widehat{\lambda}_1 u + \widehat{\lambda}_3(u + c\widetilde{\kappa}_x) + \widehat{\lambda}_4(u - c\widetilde{\kappa}_x) \\ 2(\gamma-1)\widehat{\lambda}_1 v + \widehat{\lambda}_3(v + c\widetilde{\kappa}_y) + \widehat{\lambda}_4(v - c\widetilde{\kappa}_y) \\ f_1 \end{bmatrix} \tag{31}$$

where $f_1 = (\gamma-1)\widehat{\lambda}_1(u^2+v^2) + \widehat{\lambda}_3[(u+c\widetilde{\kappa}_x)^2 + (v+c\widetilde{\kappa}_y)^2]/2 + \widehat{\lambda}_4[(u-c\widetilde{\kappa}_x)^2 + (v-c\widetilde{\kappa}_y)^2]/2 + (3-\gamma)(\widehat{\lambda}_3+\widehat{\lambda}_4)c^2/(2(\gamma-1))$ and

$$\widehat{\lambda}_1 = \widehat{\lambda}_2 = \kappa_x u + \kappa_y v, \quad \widehat{\lambda}_3 = \widehat{\lambda}_1 + c\sqrt{\kappa_x^2+\kappa_y^2}, \quad \widehat{\lambda}_4 = \widehat{\lambda}_1 - c\sqrt{\kappa_x^2+\kappa_y^2} \tag{32}$$

The original E and F can be recovered with the appropriate values of κ_x and κ_y.

There is a very fundamental consideration pertaining to the flux vectors defined above. In the case of the $\pm$ flux splitting and other splitting which are a direct result of the similarity transform, Eq. 29, it is more often the rule than the exception that the flux Jacobians of the resulting flux vectors are not equal to the similarity matrices, i.e.

$$\frac{\partial F^{\pm}}{\partial Q} \neq A^{\pm} \tag{33}$$

We shall not write out the exact Jacobians here, but we should note that for the $\pm$ splitting it has been shown that the eigenvalues of the exact Jacobians, while not $\Lambda^{\pm}$, are positive and negative appropriately for the $+$ and $-$ fluxes.

3. Implicit Numerical Algorithms

3.1. Implicit Time Differencing

Consider an implicit three point time differencing scheme of the form (Warming and Beam [1])

$$\Delta Q^n = \frac{\vartheta \Delta t}{1+\varphi}\frac{\partial}{\partial t}\left(\Delta Q^n\right) + \frac{\Delta t}{1+\varphi}\frac{\partial}{\partial t}Q^n + \frac{\varphi}{1+\varphi}\Delta Q^{n-1} + O\left[(\vartheta - \frac{1}{2} - \varphi)\Delta t^2 + \Delta t^3\right] \tag{34}$$

where $\Delta Q^n = Q^{n+1} - Q^n$ and $Q^n = Q(n\Delta t)$. The parameters ϑ and φ can be chosen to produce different schemes of either first or second order accuracy in time.

For $\vartheta = 1$ and $\varphi = 0$, we have the first order Euler implicit scheme, and for $\vartheta = 1$ and $\varphi = 1/2$, the three point implicit scheme.

Let us restrict ourselves to the first order scheme in time (although all of the subsequent development can easily be extended to any second order scheme formed from Eq. 34. Applying Eq. 34 with $\vartheta = 1$ and $\varphi = 0$ to Eq. 1, results in

$$Q^{n+1} - Q^n + \Delta t\left(\partial_x E^{n+1} + \partial_y F^{n+1}\right) = 0 \tag{35}$$

3.2. Local Time Linearizations

We wish to solve Eq. 35 for Q^{n+1} given Q^n. The flux vectors E and F are nonlinear functions of Q and therefore Eq. 35 is nonlinear in Q^{n+1}. The nonlinear terms are linearized in time about Q^n by a Taylor series such that

$$E^{n+1} = E^n + A^n \Delta Q^n + O(\Delta t^2) \quad F^{n+1} = F^n + B^n \Delta Q^n + O(\Delta t^2) \tag{36}$$

where $A = \partial E/\partial Q$, and $B = \partial F/\partial Q$ are the flux Jacobians and ΔQ^n is $O(\Delta t)$. The Jacobian matrices A and B are griver by Eq. 12, 13.

Note that the linearizations are second order accurate and so if a second order time scheme had been chosen the linearizations would not degrade the time accuracy.

Applying Eqs. 36 to Eq. 35 and combining the ΔQ^n terms produces the "delta form" of the algorithm

$$[I + \Delta t \partial_x A^n + \Delta t \partial_y B^n] \Delta Q^n = -\Delta t \left(\partial_x E^n + \partial_y F^n\right) \tag{37}$$

This is the unfactored form of the block algorithm. We shall call the right hand side of Eq. 37 the "explicit" part (RHS) and the left hand side the "implicit" part (LHS) of the algorithm.

3.3. Space Differencing

The next step is to take the continuous differential operators ∂_x and ∂_y and approximate them with finite difference operators on a discrete mesh.

Introducing a grid of mesh points (j,k), variables are defined at mesh points as

$$u_{j,k} := u(j\Delta x, k\Delta y) \tag{38}$$

The choice of the type and order of the spatial differencing is important both in terms of accuracy and stability. In most applications second order accuracy has proven to be sufficient provided the grid resolution is reasonable. The choices for differencing type include central and upwind operators. These choices are dictated by stability, accuracy, efficiency and programming issues.

Second order central difference operators can be used where, for example,

$$\delta_x^c u_{j,k} = \left(u_{j+1,k} - u_{j-1,k}\right)/(2\Delta x) \quad , \quad \delta_y^c u_{j,k} = \left(u_{j,k+1} - u_{j,k-1}\right)/(2\Delta y) \tag{39}$$

or upwind operators for the type dependent schemes

$$\begin{aligned} \delta_x^b u_{j,k} &= \left(3u_{j,k} - 4u_{j-1,k} + u_{j-2,k}\right)/(2\Delta x), \\ \delta_y^b u_{j,k} &= \left(3u_{j,k} - 4u_{j,k-1} + u_{j,k-2}\right)/(2\Delta y) \end{aligned} \tag{40}$$

with a similar forward differencing forms.

We will not focus here on the details of the difference schemes, boundary conditions, general geometry, etc. The pertinent aspect of the finite differences is the stencil size and form obtained when a difference scheme is chosen. In general, three to seven point operators can be employed, which has a direct impact on the bandwidth of the resulting implicit operators.

3.4. Matrix Form of Unfactored Algorithm

We now turn to examining the matrices we get when difference formulas are applied to the implicit algorithm. It is always instructive to examine the matrix structure of any finite difference equation. With the application of central differences to Eq. 37. it is easy to show that the implicit algorithm produces a large banded system of algebraic equations. Let the mesh size in x be J_{max} and in y be K_{max}. Then the banded matrix is a $(J_{max} \cdot K_{max} \cdot 4) \times (J_{max} \cdot K_{max} \cdot 4)$ square matrix. Let $h = \Delta t$, we have for the LHS matrix operator $\left[I + h\delta_x^c A^n + h\delta_y^c B^n\right]$ from Eq. 37, the matrix structure

$$\left[\begin{array}{cccccccccccc}
[\;] & [\;] & & & [\;] & & & & & & & \\
[\;] & [\;] & [\;] & & & [\;] & & & & & & \\
 & [\;] & [\;] & [\;] & & & [\;] & & & & & \\
 & & [\;] & [\;] & & & & [\;] & & & & \\
[\;] & & & & [\;] & [\;] & & & [\;] & & & \\
 & \ddots & & & \ddots & \ddots & \ddots & & & \ddots & & \\
 & & \left[-h\frac{B_{j,k-1}}{2\Delta y}\right] & & & \left[-h\frac{A_{j-1,k}}{2\Delta x}\right] & [I] & \left[h\frac{A_{j+1,k}}{2\Delta x}\right] & & & \left[h\frac{B_{j,k+1}}{2\Delta y}\right] & \\
 & & & [\;] & & & [\;] & [\;] & & & & [\;] \\
 & & & & [\;] & & & & [\;] & [\;] & & \\
 & & & & & [\;] & & & [\;] & [\;] & [\;] & \\
 & & & & & & [\;] & & & [\;] & [\;] & [\;] \\
 & & & & & & & [\;] & & & [\;] & [\;]
\end{array}\right] \tag{41}$$

where the variables have been ordered with j running first and then k.

The matrix is sparse but it would be very expensive (computationally) to solve the algebraic system. For instance, for a reasonable two-dimensional calculation of transonic flow past an airfoil we could use approximately 300 points in the x direction and 80 points in the y direction. The resulting algebraic system has a 96,000 $\times$ 96,000 matrix problem to be solved and although we could take advantage of its banded sparse structure it would still be very costly in both CPU time and memory. For a banded matrix with bandwidth b and rank N, the operation count estimate for inversion is $O(b^2 N)$. In two dimensions, the bandwidth (for Eq. 41) is approximately $4 \times K_{max}$ (4 for the block size and K_{max} from the differencing in k for each j line) for the above ordering. The rank is $N = 4 \times J_{max} \times K_{max}$, so that the total operation count for the inversion is on the order of $O(4^3 \times J_{max} \times K_{max}^3)$. For the above example grid sizes this gives $\approx 9.8 \times 10^9$ operations per time step. In two dimensions this is not too difficult for today's large scale computer systems. In three dimensions, the block sizes are 5, bandwidth is expanded by the extra dimension and the rank is also expanded leading to a large operation count which is still beyond the capabilities of modern computer systems.

3.5. Approximate Factorization

As we have seen, the integration of the full two-dimensional operator is expensive. One way to simplify the solution process is to introduce an approximate factorization of the two-dimensional operator into two one-dimensional operators. The implicit side of Eq. 37 can be written as

$$\begin{aligned} [I + h\delta_x A^n + h\delta_y B^n] \quad & \Delta Q^n = \\ [I + h\delta_x A^n]\ [I + h\delta_y B^n] \quad & \Delta Q^n - h^2 \delta_x A^n \delta_y B^n \Delta Q^n \end{aligned} \tag{42}$$

The cross terms (h^2 terms) are second order in time since ΔQ^n is $O(h)$. They can therefore be neglected without degrading the time accuracy of any second order scheme which we may choose.

The resulting factored form of the algorithm is

$$[I + h\delta_x A^n]\,[I + h\delta_y B^n]\,\Delta Q^n = -h\,[\delta_x E^n + \delta_y F^n] \tag{43}$$

We now have two implicit operators each of which is block tridiagonal. The structure of the block tridiagonal matrix $[I + h\delta_x^c A^n]$ is

$$\begin{bmatrix} [\;] & [\;] & & & & & & \\ [\;] & [\;] & [\;] & & & & & \\ & [\;] & [\;] & [\;] & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \left[-h\frac{A_{j-1,k}}{2\Delta x}\right] & [I] & \left[h\frac{A_{j+1,k}}{2\Delta x}\right] & & \\ & & & & [\;] & [\;] & [\;] & \\ & & & & & [\;] & [\;] & [\;] \\ & & & & & & [\;] & [\;] \end{bmatrix} \tag{44}$$

Rewriting Eq. 43 as a two stage process, we have

$$\begin{aligned} \text{Form} \quad & R^n = -h\,[\delta_x E^n + \delta_y F^n] & \\ \text{For all k lines} \quad & [I + h\delta_x A^n]\,\Delta\widehat{Q}^n = R^n & :: \text{ Block tridiagonal solve in } j \\ \text{For all j lines} \quad & [I + h\delta_y B^n]\,\Delta Q^n = \Delta\widehat{Q}^n & :: \text{ Block tridiagonal solve in } k \end{aligned} \tag{45}$$

The solution algorithm now consists of two one dimensional sweeps, one in the x and one in the y direction. The block matrix size is now at most rank $N = (4 \times \max[J_{max}, K_{max}])$ with bandwidth $b = 8$. Each step requires the solution of a linear system involving a block tridiagonal which is solved by block LU (lower-upper) decomposition. For the above example, there are K_{max} block tridiagonal inversions in x and J_{max} block tridiagonal inversions in y, resulting in $2 \times (J_{max} \times K_{max} \times 4) \times 8^2 \approx 12 \times 10^6$ operations, a decrease in work by a factor of about 800. The resulting solution process is much more economical than the unfactored algorithm in terms of computer memory and CPU time.

3.6. Diagonal Form

Most of the computational work for the implicit algorithm is tied to the block tridiagonal solution process. The computational work can be decreased by introducing a diagonalization of the blocks in the implicit operators as developed by Pulliam and Chaussee [8]. The eigensystem of the flux Jacobians A and B are used in this construction.

The eigenvalues and eigenvectors of the flux Jacobians A and B are given in Section 2.5 and we have

$$\Lambda_x = T_A^{-1} A T_A \quad \text{and} \quad \Lambda_y = T_B^{-1} B T_B \tag{46}$$

.

Here we take the factored algorithm in delta form, Eq. 43 and replace A and B with their eigensystem decompositions.

$$\left[T_A\, T_A^{-1} + h\, \delta_x \left(T_A\, \Lambda_x\, T_A^{-1}\right)\right]\ \left[T_B\, T_B^{-1} + h\, \delta_y \left(T_B\, \Lambda_y\, T_B^{-1}\right)\right]\ \Delta Q^n = R^n \tag{47}$$

At this point Eq. 43 and Eq. 47 are equivalent. A modified form of Eq. 47 can be obtained by factoring the T_A and T_B eigenvector matrices outside the spatial derivative terms δ_x and δ_y. The resulting equations are

$$T_A \left[I + h\, \delta_x\, \Lambda_x\right]\ N\ \left[I + h\ \delta_y\, \Lambda_y\right]\ T_B^{-1} \Delta Q^n = R^n \tag{48}$$

where $N = T_A^{-1} T_B$, see Eq. 23.

The explicit side of the diagonal algorithm (the steady-state finite difference equations) is exactly the same as in the original algorithm, Eq. 43. The modifications are restricted to the implicit side and so if the diagonal algorithm converges, the steady-state solution will be identical to one obtained with the unmodified algorithm. In fact, linear stability analysis would show that the diagonal algorithm has exactly the same unconditional stability as the original algorithm. (This is because the linear stability analysis assumes constant coefficients and diagonalizes the blocks to scalars, so the diagonal algorithm then reduces to the unmodified algorithm.) The modification (pulling the eigenvector matrices outside the spatial derivatives) of the implicit operator does affect the time accuracy of the algorithm. The eigenvector matrices are functions of x and y and therefore this modification reduces the time accuracy to at most first order in time and also gives time accurate shock calculations a nonconservative feature, i.e., errors in shock speeds and shock jumps. However, the steady-state is fully conservative since the steady-state equations are unmodified. Also, computational experiments by Pulliam and Chaussee [16] have shown that the convergence and stability limits of the diagonal algorithm are similar to that of the unmodified algorithm.

The diagonal algorithm reduces the block tridiagonal inversion to a set of 4×4 matrix multiplies and scalar tridiagonal inversions. In two dimension, reqrite Eq. 48

$$
\begin{array}{rcl}
\text{Form} & R^n = -h\left[\delta_x E^n + \delta_y F^n\right] & \\
\text{For all j and k} & S^n = T_A^{-1} R^n & :: \; 4 \times 4 \text{ matrix multiples} \\
\text{For all k lines} & S^n = \left[I + h\,\delta_x\,\Lambda_x\right]^{-1} S^n & :: \; 3 \text{ scalar tridiagonal solves} \\
\text{For all j and k} & S^n = N^{-1} S^n & :: \; 4 \times 4 \text{ matrix multiples} \\
\text{For all j lines} & S^n = \left[I + h\,\delta_y\,\Lambda_y\right]^{-1} S^n & :: \; 3 \text{ scalar tridiagonal solves} \\
\text{For all j and k} & \Delta Q^n = T_B R^n & :: \; 4 \times 4 \text{ matrix multiples}
\end{array} \qquad (49)
$$

Pulliam and Chaussee [8] showed that the operation count per grid point associated with the implicit side of the full block algorithm is 410 multiplies, 356 adds, and 10 divides, a total of 776 operations, while the diagonal algorithm requires 233 multiplies, 125 adds, and 26 divides or 384 operations. Adding in the explicit side and other overhead such as I/O (input/output) and initialization, the overall savings in computational work can be as high as 40%. Note the the computational work was further decreased by noting that the first two eigenvalues of the system are identical, Eq. 20. This allows us to combine the coefficient calculations and part of the inversion work for the first two scalar operators. Another advantage of the approximate factorization is that the solution process is easily parallelized. One can parallelize in y as you invert in x and parallelize in x for the y inversions.

3.7. Flux Split Forms

Flux split forms of the implicit algorithms have been exploited in many ways. Starting with Steger-Warming [6] flux vector splitting and the equations given in Section 2.6 we have

$$\partial_t Q + \delta_x^b E^+ + \delta_x^f E^- + \delta_y^b F^+ + \delta_y^f F^- = 0 \tag{50}$$

where the appropriate upwind differences are used. A first order implicit delta form for Eq. 50 is

$$\begin{aligned}\left[I + h\delta_x^b (A^+)^n + h\delta_x^f (A^-)^n + h\delta_y^b (B^+)^n + h\delta_y^f (B^-)^n\right] \Delta Q^n = \\ -h\left(\delta_x^b (E^+)^n + \delta_x^f (E^-)^n + \delta_y^b (F^+)^n + \delta_y^f (F^-)^n\right)\end{aligned} \tag{51}$$

Applying first, second or higher order upwind differencing would produce a sparse large bandwidth system similar to the unfactored block implicit scheme, Eq. 41 and work estimates which are excessive, especially in three dimensions. Approximate factorization similar to Section 3.5 would produce

$$\begin{aligned}\left[I + h\delta_x^b (A^+)^n + h\delta_x^f (A^-)^n\right]\left[I + h\delta_y^b (B^+)^n + h\delta_y^f (B^-)^n\right] \Delta Q^n = \\ -h\left(\delta_x^b (E^+)^n + \delta_x^f (E^-)^n + \delta_y^b (F^+)^n + \delta_y^f (F^-)^n\right)\end{aligned} \tag{52}$$

which produces work estimates similar to the approximate factorization of that Section.

An alternative factorization would be to keep the positive terms together and the negative terms together producing

$$\begin{aligned}\left[I + h\delta_x^b (A^+)^n + h\delta_y^b (B^+)^n\right]\left[I + h\delta_x^f (A^-)^n + h\delta_y^f (B^-)^n\right] \Delta Q^n = \\ -h\left(\delta_x^b (E^+)^n + \delta_x^f (E^-)^n + \delta_y^b (F^+)^n + \delta_y^f (F^-)^n\right)\end{aligned} \tag{53}$$

Now each implicit operators is block tridiagonal, e.g. the structure of

$$\left[I + h\delta_x^b (A^+)^n + h\delta_y^b (B^+)^n\right]$$

is a lower block diagonal matrix

$$\left[\begin{array}{ccccccccccccc}
[\ ] &&&&&&&&&&&& \\
[\ ] & [\ ] &&&&&&&&&&& \\
& [\ ] & [\ ] &&&&&&&&&& \\
&& [\ ] & [\ ] &&&&&&&&& \\
[\ ] &&&& [\ ] &&&&&&&& \\
& \ddots && \ddots & \ddots &&&&&&&& \\
&& \left[-h\frac{B^+_{j,k-1}}{\Delta y}\right] && \left[-h\frac{A^+_{j-1,k}}{\Delta x}\right] && \left[I + h\frac{A^+_{j,k}}{\Delta x} + h\frac{B^+_{j,k}}{\Delta y}\right] &&&&&& \\
&& \ddots && \ddots && \ddots &&&&&& \\
&&& [\ ] && [\ ] && [\ ] &&&&& \\
&&&& \ddots && \ddots && \ddots &&&& \\
&&&& [\ ] &&&& [\ ] & [\ ] &&& \\
&&&&&& \ddots &&& \ddots & \ddots && \\
&&&&&& [\ ] &&&&& [\ ] & [\ ]
\end{array}\right] \tag{54}$$

Similarly, the forward operators produce an upper block diagonal matrix. The advantage of this factorization is that each matrix operator can be inverted by a forward and then a backward sweep process, e.g. a Lower-Upper (LU) factorization solution process. The disadvantage of the LU schemes is encountered when parallelization is attempted and the natural recursion for the forward and backward sweeps preclude an easy parallel implementation.

4. Summary

The implicit schemes developed in the 1970's and 1980's by Richard Beam, Robert Warming and Joseph Steger at NASA Ames Research Center are the basic building blocks for a large segment of the CFD codes in use today. The advantages of approximate factorization are realized every day by modern codes in both the structured arena (e.g. OVERFLOW [9]) and even in the unstructured world where LU factorizations are frequently employed. We have not focused here on the analysis and application of these techniques, that work can be seen in the vast literature where the methods described here are widely employed. One just has to pick up any textbook which addresses CFD or any relevant journal to see numerous examples of the use of these pioneering concepts.

References

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

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

[3] Pulliam, T. H. and Steger, J. L. *Implicit Finite- Difference Simulations of Three Dimensional Compressible Flow*, **AIAA J** , Vol. 18 1980 page 159

[4] MacCormack, R. W. *The Effect of Viscosity in Hypervelocity Impact Cratering*, **AIAA Paper 69-354**, 1969.

[5] Pulliam, T. H., *Artificial Dissipation Models for the Euler Equations* **AIAA J.**, Vol 24, No 12 p. 1981

[6] Steger, J. L. and Warming, R. F. *Flux Vector Splitting of the Inviscid Gas Dynamic Equations with Applications to Finite Difference Methods* **J. Comp. Phys.** Vol 40, 263-293,1981

[7] Warming, R. F., Beam, R., and Hyett, B. J., *Diagonalization and Simultaneous Symmetrization of the Gas-Dynamic Matrices*, **Math Comp**, Vol 29, 1037, 1975

[8] Pulliam, T. H. and Chaussee D. S.,*A Diagonal Form of an Implicit Approximate Factorization Algorithm*, **Journal of Computational Physics**, Vol 39, No. 2, Feb. 1981

[9] Buning, P. G., Chiu, I. T., Obayashi, S., Rizk, Y. M., and Steger, J. L. 1988. *Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent,* **AIAA Paper 1988-4359**, AIAA Atmospheric Flight Mechanics Conference, Minneapolis, MN, 1988