Recent Progress in Numerical Techniques for Flow Simulation

H. Lomax*
NASA Ames Research Center, Moffett Field, Calif.

Recent developments in the use of numerical methods for fluid flow simulation show an increasing tendency to use numerical operators that can, in various ways, be factored. Use of methods having this property often increases the accuracy and efficiency of computer codes. Tracing the factorization property provides a unity to the basic concepts involved in the development of cyclic reduction, predictor-corrector, splitting, fast Fourier transform, and pseudospectral methods.

Nomenclature

$B(a,b,c,\dots)$	= banded matrix, number of arguments must be odd and central one denotes the main diagonal vector
$B(a;b;c;\dots)$	= banded block matrix whose arguments are matrices (order of matrices in arguments corresponds to number of dependent variables)
$B_p(a,b,c,\dots)$	= banded matrix formed from operators with periodic end conditions
$D(a)$	= diagonal matrix, a is the diagonal vector (a can be a vector of scalars or matrices depending on context)
D_+, D_-, D_0	$= r(u_{m+1}-u_m),\quad r(u_m-u_{m-1}),\quad (\tfrac{1}{2})\, r(u_{m+1}-u_{m-1})$
h, h_x, h_y, h_t	= step size, subscript if required, refers to coordinate
r, r_x, r_y, r_t	$= 1/h,\ 1/h_x,\ 1/h_y,\ 1/h_t$
u	= dependent variable or vector of dependent variables, depending on context
u_j, u_k, u^n	$= u(x),\ u(y),\ u(t)$
$u_{j+m}, u_{k+m}, u^{n+m}$	$= u(x+mh_x),\ u(y+mh_y),\ u(t+mh_t)$
$\partial_x, \partial_{xx}$	= partial derivative operators for first and second derivatives
δ_x, δ_{xx}	= finite difference approximations to ∂_x and ∂_{xx}

I. Introduction

If some liberty is taken with the word recent so that, in fact, it covers a span of about 10 years, substantial improvements in the numerical simulation of fluid flows are apparent to us all. These improvements are evident in many areas. Certainly the computers have grown bigger and faster and the dissemination of their output in graphic and motion picture form has become more readily available. Further, the fundamental understanding of the proper relation between the numerics and the physics of fluid dynamic problems has been considerably improved. But perhaps the most important advance of all is in the confidence now being placed in the simulations. For example, many of us believe that numerical calculations of unsteady, fully separated, laminar, incompressible flows can now be carried out on modern computers and made to be as reliable as their experimental counterparts.[1] How much this is being used to advance the knowledge of fluid dynamics is another matter, but the possibility is there.

This report is limited to a mathematical (rather than physical) review of a part of the recent developments in numerical methods used for flow simulations. This is because the author believes that the use and understanding of sophisticated numerical algorithms should become as natural to the physicist as are the use and understanding of calculus. The reason for this belief is not complicated. Not many years ago the physicist knew very little about numerical analysis and nothing about computer programing. The situation has changed. He now knows a great deal about computer programing and somewhat more about numerical analysis. But the important observation is that the numerical techniques he is using were quite often developed by himself. If not, they were often developed by another physicist working in the same field. Such is the case because in many difficult problems the physics dictates the numerical approach. Thus, a person working in computational fluid dynamics cannot depend upon the mathematician to invent his algorithm; he must be prepared to invent it himself. And in inventing it he should, of course, make use of the best tools in the trade.

Pursuing this line of thought we are led to ask the following question: If the physicist is committed to the use of the computer, what language should he be using to express himself? It is certainly not obvious that FORTRAN is the best scientific language, yet for many years it has carried almost the full brunt of man–computer scientific communication. If a better language (or more likely, set of languages) is to evolve, quite probably the physicist will not only have to invent it but also implement it. A major problem in this direction is the evolution of a notation that is acceptable both from the physical and numerical points of view. In the field of computational fluid dynamics, at least, this may be feasible. Although it was only a trivial exercise, some thought was given in writing this paper to the use of a notation that could be translated by an appropriately written compiler.

Finally, a few words should be said regarding the content of the following sections. Although the topics discussed apply to a fairly broad range of numerical analysis, there is a unifying theme. This theme is an attempt to trace th role of numerical operators and their factorization through the numerous computer codes that are being used to simulate various viscous and inviscid, and compressible and incompressible fluid flows.

II. Difference Schemes as Matrices

One Dimension

The importance of a usable and yet readable notation has been discussed in the introduction. One reasonable set of

Presented at the AIAA 2nd Computational Fluid Dynamics Conference, Hartford, Conn., June 19-20, 1975 (no preprints, pp. 1-9 bound volume Conference papers) submitted June 20, 1975, revision received November 24, 1975.

Index category: Computer Technology and Computer Simulation Techniques.

*Chief, Computational Fluid Dynamics Branch. Member AIAA.

descriptions is discussed following. In terms of symbols presented in the nomenclature, a forward difference scheme can be described in several ways. Thus

$$\delta_x u = D_+ u \equiv (u_{j+1} - u_j)/h = (u_{j+1} - u_j)r = B(o,-r,r)u \quad (1)$$

The meaning of the terms become more precise as the expression proceeds from left to right. The symbol δ_x refers to any finite difference approximation to the differential operator $\partial_x \equiv \partial/\partial_x$. However, the symbol δ_x, as we use it, is not unique. Certainly $\partial_x \partial_x = \partial_{xx}$ is without ambiguity. However, $\delta_x \delta_x$ (meaning the *same* difference operator applied twice) is only one of many ways to form δ_{xx}. In fact, it is well known that a second-order expression for δ_{xx} can be computed by successive applications of first derivative formulas that are not only different but individually of first order. The discussion of Eq. (11) bears on this point. The symbol D_+ refers to a particular local approximation at the point-$x = jh_x = j/r_x = jh = j/r$ (the subscripts can be omitted when there is no resulting ambiguity). The symbol B represents a matrix referring to the approximation along a complete line-*including the end points*. The letter B is chosen to represent a banded matrix; the number of arguments (if specified) must be odd and the central one designates the diagonal. If B appears without subscript, it refers to a "normal" runoff at the ends of the diagonals as in the top of expression (2).

$$B(0,-r,r) \equiv \begin{bmatrix} -r & r & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & 0 & -r & r \\ 0 & 0 & 0 & -r \end{bmatrix}$$

$$B_p(0,-r,r) \equiv \begin{bmatrix} -r & r & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & 0 & -r & r \\ r & 0 & 0 & -r \end{bmatrix} \quad (2)$$

In this case a boundary condition would be required for the bottom row. If B_p appears, the data being differenced are periodic and no boundary conditions outside the matrix itself are required. Clearly the 3-point, central-difference approximation to ∂_{xx} can be written $\delta_{xx} = r^2 B(1,-2,1)$ for Dirichlet end conditions and as $r^2 B_p(1,-2,1)$ if the data are periodic. Most conventional accuracy and stability analyses of partial differential equations (e.g., Neumann or Fourier) are based on the assumption that the difference operators are periodic.

The matrix operator notation is useful in understanding, constructing, and applying difference schemes. For example, the FORTRAN subroutine MULT (UX,A,B,C,U) could easily (and efficiently) be written to carry out the operation $U_x = B(a,b,c,)U$; that is, perform a matrix multiple of the array U by a tridiagonal and put the results in the array UX. An extremely important extension of this concept is that another FORTRAN subroutine INVT(U,A,B,C,UX) can be efficiently coded to carry out an inverse operation representing $U = B^{-1}(A,b,c,)U_x$, such as the Gaussian elimination of a tridiagonal. Notice, that if B represents some δ_x, B^{-1} represents, in some sense, $\int dx$. The qualification is used since *the end conditions* are built into B. For example, the inverse of B in expression (2) represents $\int\}d\bar{x}$, while the inverse of B_p does not exist.

Higher Order Accuracy, Hermitian Methods and Pade Approximations

An application of the inverse matrix (or operator) that makes use of factorization appears in the recent practical applications[2] of compact higher order difference approximations. One well known approximation to ∂_{xx}, that results in an error of $O(h^4)$, can be written

$$\delta_{xx} = r^2 B(-1,16,-30,16,-1)/12$$

This represents the "conventional" 5-point central-difference scheme for ∂_{xx}. An alternative expression having an error of $O(h^4)$ can be constructed using a Hermitian formula. Hermitian methods[3] connect the values of the function as well as its derivatives at the data locations. Thus

$$1/12[(\partial_{xx}u)_{j-1} + 10(\partial_{xx}u)_j + (\partial_{xx}u)_{j-1}] \simeq r^2(u_{j-1} - 2u_j + u_{j+1}) \quad (3a)$$

also represents ∂_{xx} at the point j with an error $O(h^4)$. In matrix notation this can be written $B(1,10,1)u_{xx} \simeq 12\ r^2\ B(1,-2,1)u$ or

$$\delta_{xx}u = 12r^2 B^{-1}(1,10,1)B(1,-2,1)u = \frac{D_+D_-}{1 + h^2 D_+ D_-/12}u, \quad O(h^4) \quad (3b)$$

The term to the right of the second equal sign is expressed in the form of a Pade approximation. A number of Pade approximations have been systematically tabulated by Kopal.[4] In FORTRAN the evaluation of δ_{xx} in Eq. (3b) could be made by

	A	M	D
CALL MULT (UXX,1,-2,1,U)	2	1	0
CALL INVT (UXX,1,10,1,UXX)	3	2	1
CALL MULT (UXX,0,12*R*R,0,UXX)	0	1	0

(3c)

The numbers tabled on the right give the theoretical minimum number of adds, mutiplies, and divides for each operation. Notice that the arithmetic required for the 5-point scheme given previously is (A,M,D) ~ (4,3,0) so that the arithmetic is comparable. An advantage of using Eq. (3b) can arise when applying the boundary conditions, since it is more compact than the 5-point scheme. A Hermitian and Pade approximation for the first derivative is

$$\delta_x = 3rB^{-1}(1,4,1)B(-1,0,1) = \frac{D_0}{1 + h^2 D_+ D_-/6}, \quad O(h^4) \quad (4)$$

Still another commonly used technique that can be expressed readily in matrix form is the spline approximation (see Rubin and Graves[5] for an application). The conventional, equispaced, cubic spline leads directly to

$$\delta_{xx} = 6r^2 B^{-1}(1,4,1)B(1,-2,1) = \frac{D_+D_-}{1 + h^2 D_+ D_-/6}, \quad O(h^2) \quad (5)$$

Notice the order of the error in this approximation is lower than that given by Eq. (3b). How much this is offset in practical use by the increased global continuity of the spline fit is not known. The spline approximation to the first derivative is identical to Eq. (4).

Two Dimensions

Most problems in fluid mechanics have more than one dependent variable. In such cases, the matrix notation in the

previous section should be generalized to include a vector (and corresponding matrix) which has a length (and order) corresponding to the number of dependent variables used in the analysis. To avoid an unwiedly complication, (at the risk, of course, of occasional ambiguity) we let the meaning be taken from the context. Thus u can represent either a scalar or a vector of dependent variables. However, we can be more precise with the matrix definition if we separate the arguments by semicolons when they refer to matrices. We can further simplify this notation by omitting the I if an argument is a diagonal of the same scalar. Thus, $B(O;-r;r)U$ is a block matrix, vector notation representing

$$B(O;-r;r)U_j \equiv \begin{bmatrix} -rI & rI & O & \cdots \\ O & -rI & rI & \\ O & O & -rI & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

where U_j is a vector of dependent variables.

The representation of two space dimensions is more formidable. We adopt a block matrix notation in which the block sizes depend upon the mesh size. In particular, we use $[B]$ for a block banded matrix. For our finite difference applications it is convenient to designate how the blocks are formed; that is, whether they apply to vectors formed by data along rows or columns in the two-dimensional mesh. Consider a mesh of 12 points, 4 in the $x(=jh_x)$ direction, and 3 in the $y(=kh_y)$ direction. The difference operator $B_o \equiv B[(-r_x/2), O, (r_x/2)]$ applied to each line of x, can be expressed either as $\delta_x = [B_k](O,B_o,O)$ or as $[B_j][(-r_x/2)I, O, (r_x/2)I]$, depending on whether the blocks refer to lines of x data or columns of y data, respectively. In our example, the order of $[B_k]$ and I is 3, and the order of $[B_j]$ and B_o is 4 for the 12-point total. The notation $[B_j]p[(-r_x/2)I, O, (r_x/2)I]$ designates periodic end conditions. It is important to notice that if $B_o \equiv B(a,b,c,)$, $[B_j](aI, bI, cI) = [B_k](O,B_o,O)$ and under these conditions $[B_k]^{-1} = [B_k](O,B_o^{-1},O)$. The notation accounts for a permutation of rows and columns in matrix algebra and for different address spacing in FORTRAN manipulations.

III. Direct Solutions

One of the most elegant uses of factorization in numerical methods occurs in the direct solution of block matrix equations arising from difference approximations to partial differential equations. The form we allude to is the cyclic reduction of special forms of elliptic equations.[6-9] Consider the Poisson equation $(\partial_{xx}+\partial_{yy})u=f(x,y)$. Let $\delta_{xx} = -r_x^2 B(-1,2,-1)$ and $\delta_{yy} = -r^2 B(-1,2,-1)$. Then

$$-(\delta_{xx}+\delta_{yy}) = r_y^2[B_k](-I,B_o,I) \tag{6}$$

where

$$B_o \equiv h_y^2 r_x^2 B[-1, 2(1+h_x^2+h_y^2), -1]$$

This is the classical 5-point star approximation.[10] To illustrate the use of factorization in this case, we take three lines of x data (with an arbitrary number of points in each line) and write the Poisson approximation in the form

$$\begin{bmatrix} B_0 & -I & O \\ -I & B_0 & -I \\ O & -I & B_0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \tag{7}$$

If there are 100 points along x each U_m is 100 elements long and the order of the matrix is 300. A "conventional" solution would be tedious. Instead, multiply the central line of blocks by B_o and add the top and bottom lines to it. There results

$$\begin{bmatrix} B_0 & -I & O \\ O & B_0^2-2I & O \\ O & -I & B_0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ B_0F_2+F_1+F_3 \\ F_3 \end{bmatrix} \tag{8}$$

This is cyclic reduction and it uncouples the central block from the other two. We still have a problem, however, because B_o^2-2I is a pentadiagonal and in practical cases, with more x-lines, more of the diagonals would be filled if the process were extended.

To overcome the difficulty we simply make use of the well known fact that *polynomials of a matrix* can be *factored* and the *factors commute*; thus,

$$B_o^2-2I = (B_o+\sqrt{2}I)(B_o-\sqrt{2}I) = (B_o-\sqrt{2}I)(B_o+\sqrt{2}I)$$

But this means that we can express U_2 in the form

$$U_2 = (B_o-\sqrt{2}I)^{-1}(B_o+\sqrt{2}I)^{-1}(B_oF_2+F_1+F_3)$$

We further observe that computation of the two inverse operators in this application requires only scalar tridiagonal inversions (this is not, of course, necessary for the factorization), and their actual implementation in FORTRAN is achieved by two successive calls to subroutine INVT with different entries in the third argument. The final solution for U_1 and U_3 requires one more call to INVT for each vector.

Three further points should be mentioned before we leave this section. First, if the two-dimensional operator has the form $[B_k]$ (Ia,B_o+Ib,Ic) where a,b, and c can be vectors, the factorization can still be made.[11] The central term of the matrix becomes

$$(B_o+b_1)(B_o+b_2)(B_o+b_3)$$
$$-a_2c_1(B_o+b_3)-a_3c_2(B_o+b_1)$$

which factors into $(B_o+\sigma_1 I)(B_o+\sigma_2 I)(B_o+\sigma_3 I)$ and leads to three calls to INVT for the solution for U_2. Second, for conventional cyclic reduction the number of lines must equal 2^n-1. This restriction has been removed, so that for very little extra work the number of lines is not restricted to this set.[12] Finally, if the elliptic equation has a nonlinear term multiplying $\partial_{xx}u$ (as, for example, the small perturbation equation for subcritical flow) the difference equation could not be simplified beyond the form

$$\begin{bmatrix} B_1 & -i & \\ -I & B_2 & -I \\ & -I & B_3 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}$$

Cyclic reduction would then give, for the central row of blocks

$$(B_2^2-2I)U_2 = B_2F_2+F_1+F_3$$
$$+\{(B_2-B_1)U_1-(B_3-B_2)U_3\}$$

This could be iterated, of course, but that is not very interesting. It is much more interesting to notice that the expression in { } can be shown to represent higher order terms involving $h_y^2\delta_{yy}u$. To the extent to which such terms can be neglected, a direct solution to a nonlinear equation can be calculated. In such applications it is important that we are not dependent on the "black boxes" of sophisticated numerical algorithms.

IV. Marching Methods as Operators

Consider next the solution of initial value problems. In particular, consider the equation

$$\partial_t u = A\partial_x u \tag{9}$$

where u represents a vector of dependent variables and A is a matrix of constants. In many practical applications, A is, of course, nonlinear. The appearance of nonlinear terms increases the difficulty of developing algorithms with higher-order accuracy. One way around this difficulty is to express the equations in conservation law form. In a variety of applications this makes it relatively easy to approximate the space derivatives with high order methods. Accuracy in the time integration is usually held to second order, especially for implicit techniques.

Equation (9) can be used to form $\partial_{tt}u = A\partial_x A\partial_x u = A^2\partial_{xx}u$ and it follows that

$$u(t+h) = [I + hA\partial_x + \tfrac{1}{2}h^2A^2\partial_{xx}]u(t) + O(h^3)$$

This defines a differential marching operator of $O(h^3)$ (where, without ambiguity, $h = h_t$)

$$\mathcal{L}_x(h) = I + hA\partial_x + \tfrac{1}{2}h^2A^2\partial_{xx} \tag{10}$$

We seek to approximate this by a difference operator with error $O(h_t^3)$ such that

$$\mathcal{L}_x(h) \approx L_x(h_t) = I + h_tA\delta_x + \tfrac{1}{2}h_t^2A^2\delta_{xx}$$

The error $O(h_x^n)$ in x depends upon the choice of δ_x and δ_{xx} used to approximate ∂_x and ∂_{xx}. Consider, for example, the MacCormack predictor-corrector combination[13]

$$\left.\begin{aligned} \tilde{u}_j^{n+1} &= u_j^n + h_tAr_x(u_{j+1}^n - u_j^n) \\ u_j^{n+1} &= 0.5[u_j^n + \tilde{u}_j^{n+1} + h_tAr_x(\tilde{u}_j^{n+1} - \tilde{u}_{j-1}^{n+1})] \end{aligned}\right\} \tag{11}$$

which uniquely defines the operator $L_x(h_t)$ for this case. Notice that for periodic boundary conditions

$$L_x(h_t) = I + 0.5h_tAr_x[B_p(0;-1;1) + B_p(-1;1;0)] + 0.5h_t^2A^2r_x^2B_p(-1;1;0)B_p(0;-1;1)$$

The beauty of the sequence in Eq. (11) is apparent in the matrix constructions of L_x. The sum of the two first-order operators for δ_x gives $0.5r_xB_p(-1;0;1)$, a second-order-accurate approxiamtion to ∂_x. Their product gives for δ_{xx} the term $0.5r_x^2B_p(1;-2;1)$ which is a second order accurate approximation to ∂_{xx}.

V. Splitting

Splitting techniques (also referred to as the method of fractional steps) appear to have originated with the work of Douglas[14] and Peaceman and Rachford[15] in 1955. These techniques were presented in quite general mathematical form in 1964 by Douglas and Gunn[16] and by several Russian authors in the 1960s, notably Yanenko and Marchuk. (The reader is referred to the English translations of the books by Yanenko[17] and Marchuk.[18]) In the last few years they have begun to appear with increasing frequency in American-generated computer codes that apply to simulations of quite complicated fluid flows.

The concept of splitting is sometimes presented as a reduction of a multi-dimensional problem to a series of one-dimensional problems. This is an oversimplification, in that, terms involving derivatives in the same dimension can also be split. Indeed, when cast in matrix form, it is clear that even a single term can be split. The number of possibilities is enormous which accounts for the voluminous literature on the subject. A more mathematical description of the splitting concept is that a marching operator can be *factored*. Care must be taken to maintain the desired order of accuracy when applying the boundary conditions for split or factored algorithms.

The accuracy of a splitting method, as well as confidence in the technique itself, can be ascertained by applying the method to the scalar equation (given with its exact solution)

$$\frac{du}{dt} = \sigma_1 u + \sigma_2 u + \ldots + e^{\mu t};$$

$$u = c_o e^{(\sigma_1+\sigma_2+\ldots)t} + e^{\mu t}/(\sigma_1 + \sigma_2 + \ldots - \mu) \tag{12}$$

where $\sigma_1,\ldots,\sigma_m,\ldots$ and μ are complex constants. For application to practical nonlinear problems the same order of accuracy for both the complementary and particular solution must be assured by the splitting method being tested, and products of σ should not be allowed to commute.

It is not necessary to specify the precise form of the space differencing scheme to bring out the essential idea of splitting. Consider the example

$$\partial_t = A_x\partial_x u + A_y\partial_y u \tag{13a}$$

where u is a vector of dependent variables and A_x and A_y are matrices with constant elements. The second time derivative is

$$\partial_{tt}u = (A_x^2\partial_{xx} + (A_xA_y + A_yA_x)\partial_{xy} + A_y^2\partial_{yy})u$$

and from this we can construct the second-order approximation

$$u^{n+1} = \{I + h_t(A_x\partial_x + A_y\partial_y) + \tfrac{1}{2}h_t^2[A_x^2\partial_{xx} + (A_xA_y + A_yA_x)\partial_{xy} + A_y^2\partial_{yy}]\}u^n + O(h_t^3) \tag{13b}$$

Explicit Forms

Using the second-order differential operator in Eq. (10) for both the x and y derivatives, one can easily show that the equation

$$u^{n+1} = \mathcal{L}_x(h_t)\mathcal{L}_y(h_t)u^n$$

reproduces the first two terms in the { } of Eq. (13b) but fails on the coefficient of h_t^2 because of the commuting condition. It is also easy to show by simple algebra that

$$u^{n+2} = \mathcal{L}_y(h_t)\mathcal{L}_x(h_t)\mathcal{L}_x(h_t)\mathcal{L}_y(h_t)u^n \tag{13c}$$

reproduces all three terms in { } and represents, therefore, a second-order-accurate solution to Eq. (13a) at every other time step. On this basis, Eqs. (13b) and (13c) are equally valid. The role of factoring is obvious, and Eq. (13c) represents an easily programmable numerical algorithm if we replace the differential operators by some appropriate difference operators and note that $\delta_x\delta_y = \delta_y\delta_x$.

Another approximation to Eq. (13a) can be written

$$u^{n+2} = \mathcal{L}_y(h_t)\mathcal{L}_x(2h_t)\mathcal{L}_y(h_t)u^n$$

which one can also show is valid to second order. In fact,

$$u^{n+2N} = \left[\sum_1^N \mathcal{L}_y(h_t)\right]\mathcal{L}_x(2Nh_t)\left[\sum_1^N \mathcal{L}_y(h_t)\right]$$

is valid to second order, and it is in this form that splitting is often used in explicit applications. The usefulness of this kind of factorization arises when the mesh sizes in the x and y directions are greatly different.[19]

Implicit Forms

A fully implicit solution to Eq. (13a) can also be written in split form. First, apply the Crank-Nicolson (second-order accurate in time) approximation.

$$(u^{n+1}-u^n)r_t=(A_x\partial_x+A_y\partial_y)(u^{n+1}+u^n)/2$$

Rearranging terms gives

$$[I-0.5h_t(A_x\partial_x+A_y\partial_y)]u^{n+1}$$
$$=[I+0.5h_t(A_x\partial_x+A_y\partial_y)]u^n+O(h_t^3) \quad (13d)$$

which has the same order of accuracy as Eq. (13b). If *both sides* of Eq. (13d) are factored, we have

$$(I-0.5h_tA_x\partial_x)(I-0.5h_tA_y\partial_y)u^{n+1}$$
$$=(I+0.5h_tA_x\partial_x)(I+0.5h_tA_y\partial_y)u^n+O(h_t^3) \quad (13e)$$

Notice that Eq. (13e) reproduces the unfactored expression (13d) except for the term $(0.25r_t)^2A_xA_y$ which appears with a + sign on both sides. But this also represents an error of $O(h_t^3)$ so that Eq. (13e) is, on the basis of order, as valid an approximation to Eq. (13a) as Eqs. (13b, c, or d). If in applying Eq. (13e) the order of operations on both sides is strictly adhered to in a single step, commuting is not required to maintain the second order accuracy and the sequence does not have to be alternated. Of course, to be useful, the operators $(I-0.5h_tA_x\partial_x)^{-1}$ and $(I-0.5h_tA_y\partial_y)^{-1}$ must be efficiently programable. In the case where the A's are scalars, this leads to only scalar tridiagonal solvers, even for fourth-order accuracy in the space derivatives. Applications of this type appear in studies of the incompressible Navier-Stokes equations. If A is a matrix, block tridiagonal solvers can be used and applications appear in studies of the compressible Navier-Stokes equations and the Eulerian equations.

Mixed Implicit, Explicit Forms

There is nothing to prevent mixed application of implicit and explicit operations in splitting techniques. Consider, for example, the equation

$$\partial_t u=Au+A_x\partial_x u \quad (14a)$$

which has the second derivative

$$\partial_{tt}u=[A^2+(AA_x+A_xA)\partial_x+A_x^2\partial_{xx}]u$$

First-order accuracy is obtained by the operators

$$(I-0.5h_tA)u^{n+1}=(I+0.5h_tA)\mathcal{L}_x(h_t)u^n \quad (14b)$$

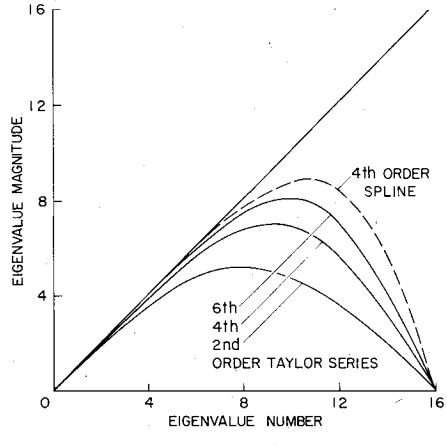

Fig. 1 Eigenvalue spectrums of various methods approximating a first derivative with periodic end conditions.

where $\mathcal{L}_x(h_t)$ is given by Eq. (10). This follows from the fact that

$$\mathcal{L}(h_t)=(I-0.5h_tA)^{-1}(I+0.5h_tA)$$
$$=I+h_tA+\tfrac{1}{2}h_t^2A^2+O(h_t^3)$$

for any matrix A. Second-order accuracy follows at every other time step if the $\mathcal{L}$ and $\mathcal{L}_x$ operators are alternated as in Eq. (13c).

VI. Extension of Matrix Concept

Let us return to the matrix form of a difference scheme. Consider, for example, the three-point central difference approximation to a first derivative with periodic end conditions

$$\delta_x u=\tfrac{1}{2}r(u_{j+1}-u_{j-1})=B_p(-r/2,o,r/2)u+O(h^2) \quad (15a)$$

By simple matrix algebra this can be written

$$W^{-1}\delta_x u=W^{-1}B_pWW^{-1}u$$

where W is an arbitrary nonsingular matrix. However, if W is chosen so that $W^{-1}B_pW=D$ where D is a diagonal matrix having the eigenvalues of B_p for its elements, Eq. (15a) can also be written in the equivalent form

$$\delta_x u=\tfrac{1}{2}r(u_{j+1}-u_{j-1})=[WDW^{-1}]u \quad (15b)$$

and this is the form we shall inspect. It is well known that the columns of W are the eigenvectors of B_p.

First of all, notice that the matrix B_p is *circulant*. That is to say, its elements are constant along diagonals and when a diagonal meets the right edge it reappears one row lower at the left edge. The five- and seven-point central difference approximations to a first derivative with periodic boundary conditions can also be expressed in matrix notation. They are

$$\delta_x=\frac{1}{12}rB_p(1,-8,0,8,-1)+O(h^4) \quad (15c)$$

and

$$\delta_x=\frac{1}{60}rB_p(-1,9,-45,0,45,-9,1)+O(h^6) \quad (15d)$$

respectively. These and all other matrices formed by periodic difference operators are circulant. It is a remarkable fact that the eigenvectors of all circulant matrices (with the same order) are identical. This leads to the property that all linear difference scheme approximations to a first (or any other) derivative with periodic boundary conditions can be expressed as $\delta_x=WDW^{-1}$ in which W and W^{-1} are identical for all cases. The diagonal matrix D is the "signature" of a method and is the only term in the factored operator that varies from method to method.

All classical central (antisymmetric) differencing schemes for the first derivative can be written in the general form[20]

$$h\delta_x=\sum_{m=-N}^{N}\beta_{m,N}u_m+O(h^{2N}) \quad (15e)$$

where

$$\beta_{m,N}=\frac{N!N!(-1)^{m+1}}{m(N+m)!(N-m)!},\quad B_{0,N}=0$$

of which the equations in (15a, c, and d) are examples for $N=1,2$, and 3, respectively. Since the eigenvalues of a circulant matrix are easy to find, Eq. (15e) can be used to compare the elements of D for a δ_x scheme having any order of accuracy. Figure 1 shows the results for $N=1,2$, and 3. As N increases (i.e., the order of accuracy increases) the elements are approaching a straight line. The Hermitian, Pade, and spline differencing methods are also represented by circulant matrices when applied to periodic boundary conditions. The D that derives from the scheme represented by Eq. (4) is also shown in the figure.

VII. Practical Use of Matrix Extension

It is surprising that the matrix formulation WDW^{-1} can be useful from a computational point of view. This is not immediately obvious because W and W^{-1} are competely dense matrices. If

$$\omega_M = e^{2\pi i/M}, \quad W_M(j,k) = \omega_M^{jk}$$

for $0 \le j,k \le M-1$ where M is the number of points being differenced. For example

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{2\pi i/4} & e^{4\pi i/4} & e^{6\pi i/4} \\ 1 & e^{4\pi i/4} & e^{8\pi i/4} & e^{12\pi i/4} \\ 1 & e^{6\pi i/4} & e^{12\pi i/4} & e^{18\pi i/4} \end{bmatrix} \tag{16a}$$

Once again the key to the practical use of the matrix concept is factorization. One can show that

$$W_4 = \underbrace{\left[\begin{array}{cc:cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]}_{S_4^{-1}} \underbrace{\left[\begin{array}{cc:cc} 1 & 1 & & \\ 1 & -1 & & 0 \\ \hdashline & & 1 & 1 \\ & G & 1 & -1 \end{array}\right]}_{W_4^{(3)}}$$

$$\times \underbrace{\left[\begin{array}{cc:cc} 1 & & & 0 \\ & 1 & & \\ \hdashline & & 1 & \\ 0 & & & i \end{array}\right]}_{W_4^{(2)}} \underbrace{\left[\begin{array}{cc:cc} 1 & & 1 & \\ & 1 & & 1 \\ \hdashline 1 & & -1 & \\ & 1 & & -1 \end{array}\right]}_{W_4^{(1)}} \tag{16b}$$

Notice that there are four adds in $W_4^{(1)}$, four more in $W_4^{(3)}$ and one complex shift in $W_4^{(2)}$. There are no arithmetic operations involved in the S_4^{-1} operation which is referred to as a shuffle. The factorization of W_8 can be written

$$S_8 W_8 = \left[\begin{array}{cc:cc} W_4 & & 0 & \\ & & & \\ \hdashline 0 & & W_4 & \end{array}\right] \left[\begin{array}{cc:cc} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ \hdashline & & & \\ & & & \\ & & & \\ & & & \end{array}\right.$$

$$\left.\begin{array}{cc:cc} & & 1 & \\ & & & 1 \\ & & & \\ & & & \\ \hdashline 1 & & & \\ & c_1 & & \\ & & i & \\ & & & c_2 \end{array}\right]$$

$$\times \left[\begin{array}{cc:cc} 1 & & 1 & \\ & 1 & & 1 \\ & & 1 & \\ & & & 1 \\ \hdashline 1 & & -1 & \\ & 1 & & -1 \\ & & 1 & \\ & & & -1 \end{array}\right] \tag{16c}$$

where $c_1 = (\sqrt{2} + i\sqrt{2})/2$, $c_2 = (-\sqrt{2} + i\sqrt{2})/2$, and S_8 (like S_4) is an even row interchange. The generalization is clear; in fact, the only new numbers to be determined are the complex factors in the lower right block of $W_M{}^{(2)}$. It should be mentioned that $W_M^{-1} = W_M^*/M$ and $c_2 = -c_1^*$ where the superscript * denotes the complex conjugate.

The reader is probably aware that Eq. (16a) is the transform matrix of the discrete Fourier transform and the factorizations in Eqs. (16b and c) are one way of viewing the well publicized fast Fourier transform (FFT) algorithms. The author is indebted to Dr. W.A. Mersman for the particular factorization previously shown. (The literature on the FFT algorithms is extensive, for example, see Ref. 21.) All of the derivative operations in Eqs. (15) can be carried out by two applications of an FFT and a vector product. The concepts presented before have been very well described, in a somewhat different way, by Fornberg.[20]

The essential part of the practical usefulness of the matrix diagonalization of a derivative depends upon the nature of the eigenvectors in W. If these are extremely well ordered, as they are for periodic application to lines with 2^N points, they can provide an efficient method for computing derivatives with high-order accuracy. Much work has been done to eliminate the 2^N requirements in FFT applications. This amounts to finding different factors to the diagonalizing matrix W and its inverse.

It is important to notice that the development of the basic concept does not depend on the periodic boundary conditions. Consider for example the nonperiodic banded matrix $B_o \equiv B(1,0,1)$. An approximation to the second derivative that is $O(h^2)$ is $\delta_{xx} = r^2(B_o - 2I) = r^2 B(1,-2,1)$ if the function is specified on both ends (Dirichlet boundary conditions). One can easily show that the approximation

$$\delta_{xx} = \frac{1}{12} r^2 (-B_o^2 + 16B_o - 28I)$$

is $O(h^4)$ and *except at the end points*, represents the banded matrix $r^2 B(-1,16,-30,16,-1)/12$. In order to maintain the accuracy at the ends of a line, higher order derivative conditions must be supplied there. The point of this example depends upon the fact that the eigenvectors of B_o can be written in the form

$$W_M(j,k) = \sin[jk\pi/(M+1)], \quad 1 \le j, \ k \le M$$

This is the sine transform. It follows at once that if W diagonalizes B_o, it diagonalizes any matrix polynomial with constant coefficients involving B_o and I. Therefore, with a fast sine transform algorithm, higher order accurate nonperiodic approximations to δ_{xx} can be computed in the same way that periodic ones are computed using the Fourier transform. Of course, sensitivity to the boundary conditions is an important consideration that is problem dependent.

VIII. Pseudospectral Methods

Numerical studies of the Navier-Stokes equations for incompressible flow are sometimes carried out using spectral or pseudospectral methods. For an excellent discussion of this topic see Orszag and Israeli[22] and the extensive bibliography therein. If the boundary conditions are periodic, in one of the independent variables, the spectral method usually consists of expanding the dependence of the flow on that independent variable into a discrete Fourier series. The equations are manipulated in the transformed variables and physical results are later determined by means of an inverse Fourier transform. When the spectral method is applied to a term with a nonconstant coefficient, a convolution summation must be computed. Even with the use of FFT algorithms, this procedure can be complicated to implement and relatively time consuming to compute.[23]

The pseudospectral method was designed to avoid the convolution calculation as well as to simplify and generalize the coding applications. We describe it here for periodic boundary conditions. (More general boundary conditions can be treated using polynomial approximations other than the trigonometric[22]). Consider the term $\partial_x u(x)$. In a numerical simulation u is a given array of numbers representing a dependent variable at a prescribed set of points along the x coordinate. To apply the method, we first make a Fourier transform of u. This forms an (complex) array of numbers that corresponds to the coefficients in a Fourier series for each wave no. k. Each number in this array is then multiplied by $\sqrt{-1}$ times the appropriate value of $\pm k$. This approximates the ∂_x operation in "k space." Finally, the inverse transform of the array is taken. The resulting set of numbers is an approximation to $\partial_x u(x)$ at the prescribed values of x and is used as such in equations used to model the physical phenomena. This is the pseudospectral method.

It is interesting to compare this periodic form of pseudospectral analysis with the extended matrix evaluation of finite difference schemes described in Sec. VII. We have seen that Taylor series, Hermitian series, splines, etc., can be used to approximate derivative calculations along lines of data with periodic boundary conditions, and that these approximations lead to circulant matrices. Circulant matrices, in turn, can always be factored into the form WDW^{-1} where W^{-1} and W are identical to a Fourier transform and its inverse. The remaining diagonal matrix D is determined by the particular method chosen and it is the only set of numbers that varies from method to method. A derivative found by the Fourier pseudospectral method fits directly into this concept. Its particular signature is a diagonal matrix with elements corresponding to $\pm ik$. These elements plot along the straight line in Fig. 1, the line which is the asymptote of the lines representing the conventional, Taylor-series, finite-difference approximation as their order of accuracy goes to infinity. In this sense the pseudospectral method can be said to have infinite-order accuracy for problems with periodic boundary conditions. On the basis of this observation one can ask: Where does the error in the Fourier pseudospectral method appear? The answer is in aliasing brought about by the truncation of terms in the series expansion. A discussion of this effect[24] is outside the scope of this paper.

IX. Conclusions

The use of numerical operators and factorization appears in many recent codes that are being used to simulate compressible, incompressible, viscous (both laminar and turbulent), inviscid, and chemically reacting flows. Many of the techniques that have been described make use of an underlying orderly structure in the numerical approach. It appears that increased orderliness in the numerical process leads to increased speed in an optimized algorithm. This should apply in assessing the value of finite element approaches. After the variation or Galerkin principles have been applied to derive the approximating formulas, the actual implementation of a finite element method can be expressed in terms of matrix operators. The efficiency of a particular method will depend on the sparseness of these operators *after factorization*. Remember, in this regard, that the initial matrix in the fast Fourier transform approach was everywhere dense.

Finally, it may be argued that the concepts discussed in the body of this report should be familiar to anyone working in computational fluid dynamics. It does not follow, however, that any of them should be used in constructing a code for a particular flow simulation. The best computer code is the one that gives reliable answers to fluid dynamic problems in the shortest time. The time, it is true, may be measured by CPU cycles or calendars, depending on the application; but the basic charter of computational fluid dynamics is to advance the knowledge of fluid dynamics, not to develop elegant algorithms per se. An example pertaining to these remarks lies in the study of unsteady flows having shock waves and contact surfaces. Often, a code designed for these problems will treat each discontinuity with special consideration[25] and minimize the number of grid points between them. The kind of orderliness required for matrix factorization is lost, but a very efficient computer program can be produced.

References

[1] Mehta, U. B. and Lavan, Z., "Starting Vortex, Separation Bubbles and Stall; A Numerical Study of Laminar Unsteady Flow Around an Airfoil," *Journal of Fluid Mechanics*, Vol. 67, No. 2, pp. 227-256, 1975.

[2] Kreiss, H. O. and Oliger, J., "Comparison of Accurate Methods for the Integration of Hyperbolic Equations," *Tellus*, Vol. 24, 1972, pp. 199-215.

[3] Collatz, L., *The Numerical Treatment of Differential Equations*, 3rd ed., Springer Verlag, Berlin, 1960.

[4] Kopal, Z., *Numerical Analysis*, 2nd ed., Wiley, New York, 1961.

[5] Rubin, S. G. and Graves, R. A., Jr., "Viscous Flow Solutions with a Cubic Spline Approximation," *Computers and Fluids*, Vol. 3, No. 1, 1975, pp. 1-37.

[6] Buzbee, B. L., Golub, G. H., and Nielson, C. W., "On Direct Methods for Solving Poisson's Equations," *SIAM Analysis*, Vol. 7, No. 4, 1970, pp. 627-56.

[7] Buneman, O., "A Compact Non-Iterative Poisson Solver," SUIPR Rept. No. 294, 1969, Inst. Plasma Research, Stanford Univ., Calif.

[8] Hockney, R. W., "The Potential Calculation and Some Applications," *Methods in Computational Physics*, ed. Alder, B., Fernback, S., and Rotenberg, M., Vol. 9, Academic Press, New York, 1970, pp. 135-211.

[9] Dorr, F. W., "The Direct Solution of the Discrete Poisson Equation on a Rectangle," *SIAM Rev.*, Vol. 12, 1970, pp. 248-263.

[10] Forsythe, G. E. and Wasow, W. R., *Finite-Difference Methods for Partial Differential Equations*, Wiley, New York, 1960.

[11] Swartztrauber, P. N., "A Direct Method for the Discrete Solution of Separable Elliptic Equations," *Journal of Numerical SIAM Analysis*, Vol. 11, No. 6, 1974, pp. 1136-1149.

[12] Sweet, R. A., "A Generalized Cyclic Reduction Algorithm, *SIAM Journal of Numerical Analysis*, Vol. 11, No. 3, 1974, pp. 506-520.

[13] MacCormack, R. W., "Numerical Solution of the Interaction of a Shock Wave with a Laminar Boundary Layer," *Lecture Notes in Physics*, Vol. 8, Springer-Verlag, 1971, p. 151.

[14] Douglas, J., "On the Numerical Integration of $U_{xx}.U_{yy} = U_t$ by Implicit Methods," *Journal of the Society of Industrial and Applied Mathematics*, Vol. 3, 1955, pp. 42-65.

[15] Peaceman, D. and Rachford, H., "The Numerical Solution of Parabolic and Elliptic Differential Equations," *Journal of the Society of Industrial and Applied Mathematics*, Vol. 3, 1955, pp. 28-41.

[16] Douglas, J. and Gunn, J., "A General Formulation of Alternating Direction Methods," *Numerical Mathematics*, Vol. 6, No. 5, 1964.

[17] Yanenko, N. N., *The Method of Fractional Steps*, Springer-Verlag, Berlin, 1971.

[18] Marchuk, G. I., *Numerical Methods in Weather Prediction*, Academic Press, New York, 1974.

[19] MacCormack, R. W. and Baldwin, B. S., "A Numerical Method for Solving the Navier-Stokes Equations with Application to Shock-Boundary Layer Interactions," AIAA Paper 75-1, Jan. 1975, Pasadena, Calif.

[20] Fornberg, B., "On a Fourier Method for Integration of Hyperbolic Equations," *SIAM Journal of Numerical Analysis*, Vol. 12, No. 4, 1975, pp. 509-528.

[21] Brigham, E. O., *The Fast Fourier Transform*, Prentice Hall, New Jersey, 1974.

[22] Orszag, S. A. and Israeli, M., Numerical Solution of Viscous Incompressible Flows, *Annual Review of Fluid Mechanics*, Vol. 6, 1974, pp. 281-317.

[23] Orszag, S. A., "Comparison of Pseudospectral and Spectral Approximation," *Studies in Applied Mathematics*, Vol. 51, No. 3, 1972, pp. 253-259.

[24] Orszag, S. A., "Numerical Simulation of Incompressible Flows Within Simple Boundaries: Accuracy," *Journal of Fluid Mechanics*, Vol. 49, part 1, 1971, pp. 75-112.

[25] Moretti, G., Computations of Unsteady Flows: A Realistic View of the State-Of-The-Art and of Perspectives for the Future, presented at the Symposium on Unsteady Aerodynamics, Tucson, Arizona, 1975 (to be published).