FINITE-ELEMENT METHODS

Andrew Staniforth
The Meteorological Office
Bracknell, U.K.

Summary: Many fluid flow problems are mappable to a rectangle or a box; conformal mappings are particularly useful in this regard. We are concerned here with the efficient solution of such problems using finite elements. The central issue is the element choice. Rectangular (box) elements generally lead to more efficient algorithms than triangular (tetrahedral) elements. A synthesis of algorithms, based on bilinear (trilinear) elements is presented. The algorithms have the attributes of simplicity, accuracy, stability and straightforward incorporation of boundary conditions. For bilinear and trilinear elements, it is found that product and first-derivative terms are well-handled by the Galerkin FE method, but that it is advantageous to go outside of the Galerkin framework when treating second-derivative terms. It is particularly important to consider the form of the governing equations, vis-à-vis the choice of staggered, non-staggered and/or mixed-order elements, and to choose an appropriate time scheme.

1. INTRODUCTION

Many fluid flow problems of interest occur in regular domains. In the context of this paper we define a regular domain to be one that can be mapped (conformally or otherwise) to either a rectangle or a box. This class of problem includes the simulation of the earth's atmosphere including orographic effects. Often when one is interested in understanding more about physical phenomena (eg convection studies), the choice of geometry is arbitrary and is frequently taken to be regular for both convenience and efficiency.

The central issue when formulating a finite-element (FE) flow model is the choice of elements, since this choice has a very direct impact on the accuracy, stability, computational efficiency and simplicity of a given formulation.

To help clarify the use of the presented algorithms, we relate them to the integration of the shallow-water equations on a rotating sphere. These equations, although simple in form, are nevertheless very useful for illustrative purposes since they include time-dependence, two space dimensions, first and second derivatives, non-linear terms, stiffness and variable coefficients due to a co-ordinate transformation (mapping). On a polar-stereographic projection, true at 60 degrees N, they are (e.g. Staniforth and Mitchell, 1977)

$$\zeta_t/S = -\left[(QU)_x + (QV)_y\right], \tag{1}$$

$$D_t/S + \phi_{xx} + \phi_{yy} = \left(QV - K_x\right)_x - \left(QU + K_y\right)_y, \tag{2}$$

$$\phi_t + \Phi_0\, D/S = -\left[(\phi U)_x + (\phi V)_y\right], \tag{3}$$

where

$$\zeta = S\left(V_x - U_y\right) = \text{relative vorticity,} \tag{4}$$

$$D = S\left(U_x + V_y\right) = \text{divergence,} \tag{5}$$

$$Q = \zeta + f = \text{absolute vorticity,} \tag{6}$$

$$U = u/m,\ V = v/m \text{ and } S = m^2. \tag{7}$$

Here, x and y are the co-ordinates of the projection, u and v are the components of the wind vector along the axes of the co-ordinate system, ϕ is the perturbation geopotential height of the free surface about its mean value (Φ_0), $m=[1+\sin(\pi/3)]/[1+\sin(\textit{latitude})]$ is the map-scale factor, f is the Coriolis parameter and U and V are termed the wind images. Decomposing the wind images in terms of velocity potential χ and a stream function ψ, we have

$$U=\chi_x-\psi_y, \tag{8}$$

$$V=\chi_y+\psi_x, \tag{9}$$

which lead to the relations

$$\psi_{xx}+\psi_{yy}=\zeta/S, \tag{10}$$

$$\chi_{xx}+\chi_{yy}=D/S. \tag{11}$$

For a contained flow, $\psi=0$ and $\nabla\chi\cdot\mathbf{n}=0$ on the boundary, where $\mathbf{n}$ is the normal vector to the boundary.

Equations (1) and (2) are derived from the momentum equations and (3) is the continuity equation. For the corresponding problem in a plane geometry, the equations remain unchanged, except that $m=S=1$ and f is constant.

In section 2 we examine the impact that the element choice has on the structure of the matrices which result from application of the Galerkin finite-element method (GFEM), by comparting the computational effort required to evaluate derivatives and products when using (linear) rectangular and triangular elements. The key issue here is the efficient solution of the 'mass-matrix problem'.

A breakdown of the remainder of the paper is: section 3 - element order, spatial evolutionary error and code modularity; section 4 - staggered, non-staggered and mixed- order elements, and the form of the governing equations; section 5 - stability; section 6 - miscellaneous considerations; section 7 - time schemes; and section 8 - some conclusions.

2. COMPUTING DERIVATIVES AND PRODUCTS - THE 'MASS-MATRIX PROBLEM'

The essence of the GFEM (e.g. Strang and Fix, 1973) is to:

(i) expand the dependent variables of the problem in terms of a set of basis functions, each of which is a low-order polynomial of compact support (i.e. non-zero only over a small subdomain called an element);

(ii) insert these expansions into the governing equations and orthogonalise the error to the basis.

The first task is to geometrically subdivide the domain of the problem into a set of overlapping subdomains, and to examine the impact on efficiency of the choice of subdivision. To illustrate this point, let us examine the simple (but fundamental) operations of calculating first derivative and product terms over a rectangular domain, and contrast the impact of a subdivision of the domain into rectangles (Fig.1a) with that of arbitrarily chosen triangles (Fig. 1b) when using linear elements. Clearly triangularisation is more general than rectangularisation, being applicable to an arbitrary polyhedral domain, but here we only concern ourselves with problems in a Cartesian geometry. The key issues for problems that are mappable to a Cartesian geometry are:

(i) can we afford the generality of triangles in comparison to rectangles?

(ii) does either offer any advantage over competing methods such as finite differences?

As we will see, the answer to (i) is an overwhelming no, whereas the answer to (ii) is a qualified yes for rectangular FEs.

It remains to define the basis functions. For both triangular and rectangular elements we associate a basis function with each and every vertex (or node). The basis function association with a given node is defined to be unity at the node, to vary linearly to zero at neighbouring nodes and to be identically zero elsewhere.

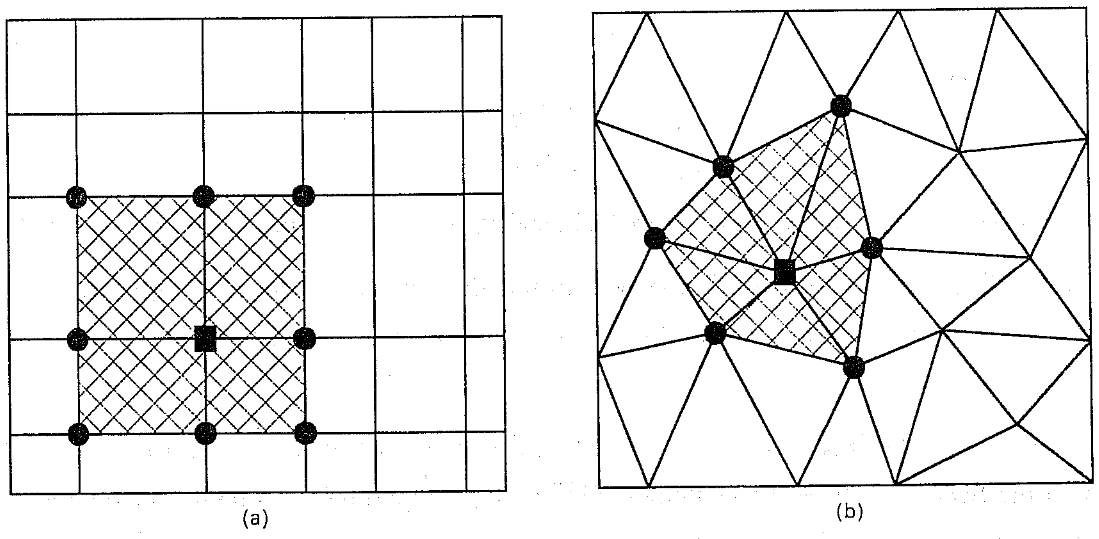

Figure 1. Two subdivisions of a rectangular domain: (a) rectangles, and (b) triangles. Basis functions associated with square nodes have value unity there, vary linearly to zero at neighbouring circular nodes, and are zero outside hatched areas.

In Fig.1, the given node is denoted by a square and neighbouring nodes by circles; the associated basis function is unity at square points, zero at circle points, and non-zero only within the hatched areas. For a rectangular element (Fig.1a), the algebraic variation of the basis function over a hatched rectangle is $(a+bx)(c+dy)$ where a, b, c and d are determined such that the basis function is unity at the given node and zero at the other three nodes of the rectangle. Similarly for triangular elements (Fig. 1b), except that the algebraic variation within a hatched triangle is $(a+bx+cy)$, since a triangle has one less node than a rectangle. For both rectangular and triangular elements the basis is an interpolatory one, since the coefficients in the expansion of a function in terms of the basis are just the values of the function at the nodes.

2.1 **First derivatives**

Consider the problem of evaluating

$$v = u_x, \tag{12}$$

where u is known at the set of nodal points and we require the value of v at these nodal points. The first step in the FE treatment of this problem is to expand u as

$$u(x,y) = \sum_i u_i e^i(x,y), \tag{13}$$

where $e^i(x,y)$ is the basis function associated with the ith node (assuming some ordering of nodes) the sum over i is the sum over all nodes and u is the value of $u(x,y)$ at the ith node: v is also expanded in a similar manner. After insertion of these expansions into (12) and orthogonalisation of the error to the basis (by multiplying by an arbitrary basis function and integrating over the domain), we obtain a set of linear equations of the form

$$P\mathbf{v} = Q\mathbf{u}, \tag{14}$$

where u and v are vectors of nodal values, P and Q are large sparse matrices and P is often referred to as the mass matrix or projection matrix. For rectangular elements with the usual row-wise ordering of elements, both P and Q are block tridiagonal and each block itself tridiagonal. For triangular elements there is no natural ordering and the structure will depend on the particular ordering chosen.

For *rectangular* elements, (14) may be written explicitly at the point (x_m, y_n) of the mesh (Fig. 1a) as

$$\frac{1}{36}\begin{bmatrix} h_{m-1}k_n v_{m-1,n+1} + 2(h_{m-1}+h_m)k_n v_{m,n+1} + h_m k_n v_{m+1,n+1} \\ +2h_{m-1}(k_{n-1}+k_n)v_{m-1,n} + 4(h_{m-1}+h_m)(k_{n-1}+k_n)v_{m,n} + 2h_m(k_{n-1}+k_n)v_{m+1,n} \\ +h_{m-1}k_{n-1}v_{m-1,n-1} + 2(h_{m-1}+h_m)k_{n-1}v_{m,n-1} + h_m k_{n-1}v_{m+1,n-1} \end{bmatrix}, \tag{15}$$
$$= \frac{1}{12}\left[k_n(u_{m+1,n+1} - u_{m-1,n+1}) + 2(k_{n-1}+k_n)(u_{m+1,n} - u_{m-1,n}) + k_{n-1}(u_{m+1,n-1} - u_{m-1,n-1})\right]$$

where $h_m = x_{m+1} - x_m$ and $k_n = y_{n+1} - y_n$. The evaluation of the right-hand side of (15) is explicit and straightforward to calculate, but it is not immediately clear how to solve efficiently for $v_{m,n}$ given the implicit way in which it appears.

The brute-force method of multiplying the right-hand side of (15) by the inverse of P is clearly not viable since this requires $O(M^2N^2)$ arithmetic operations and $O(M^2N^2)$ words of storage for an $M \times N$ mesh, even when P^{-1} has been precalculated. The operation count and memory requirements may both be reduced to $O(M^2N)$ if a 'banded-solver' is used that exploits the fact the elements of P are zero everywhere outside the tridiagonal band of blocks centred on the diagonal. Although this is better, it is still a factor of $O(M)$ more expensive than the theoretical optimum of $O(MN)$ operations and storage. Since typically $M = O(10^2)$, this is still a very expensive proposition, both in terms of arithmetic operations and storage. Cullen (1973) proposed a method that uses a truncated series to obtain an approximate solution, the accuracy of the solution depending on the number of terms taken. This method has the advantages of reducing memory requirements to optimum order and decreasing the operation count, but has the disadvantages of not being exact, not having an optimal operation count and only working for uniform grids. Iterative methods suffer from similar deficiencies except that they are applicable to non- uniform Cartesian meshes.

It must be emphasised that it is extremely important to be able efficiently to solve (14) if the FE method is to be competitive with the finite-difference (FD) method, since P is the identity matrix for FD methods and the problem is then trivial. The matrix P in the finite-element literature is often replaced by a diagonal matrix (or mass-lumped in FE parlance), where the diagonal element is obtained by taking the row sum of P. As we shall see later, this may be undesirable since it can lead to a significant reduction in the accuracy of the method.

2.2 Efficient solution of the 'mass-matrix problem' for rectangular elements

An efficient solution algorithm for (14) is easily derived for the rectangular element case (but not the triangular element case). In the engineering literature it has been used in conjunction with approximate factorisation techniques (e.g. Baker and Soliman, 1983) and is often referred to as the tensor-product method. The essence of this algorithm (Staniforth and Mitchell, 1977, 1978) is to

(i) Solve the set of non-dimensional tridiagonal problems

$$P^x \mathbf{s}_n = \mathbf{r}_n; \quad n = 1,2,\ldots,N, \tag{16}$$

for $\mathbf{s}_n$ along lines of constant y (i.e. n fixed) using Gaussian elimination (e.g. Ahlberg et al., 1967), where

$$\mathbf{r}_n = \begin{bmatrix} r_{1,n} \\ r_{2,n} \\ \cdot \\ \cdot \\ r_{M-1,n} \\ r_{M,n} \end{bmatrix}, \quad \mathbf{s}_n = \begin{bmatrix} s_{1,n} \\ s_{2,n} \\ \cdot \\ \cdot \\ s_{M-1,n} \\ s_{M,n} \end{bmatrix}, \quad P^x = \frac{1}{6}\begin{bmatrix} 2h_1 & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & h_{M-2} & 2(h_{M-2}+h_{M-1}) & h_{M-1} \\ & & & & h_{M-1} & 2h_{M-1} \end{bmatrix},$$

and $r_{m,n}$ is the right-hand side of (15).

(ii) Solve the set of one-dimensional tridiagonal problems

$$P^y \mathbf{v}_m = \mathbf{s}_m; \quad m = 1,2,\ldots,M, \tag{17}$$

for $\mathbf{v}_m$ along lines of constant x (i.e. m fixed), again using Gaussian elimination,where

$$\mathbf{v}_m = \begin{bmatrix} v_{m,1} \\ v_{m,2} \\ \cdot \\ \cdot \\ v_{m,N-1} \\ v_{m,N} \end{bmatrix}, \quad \mathbf{s}_m = \begin{bmatrix} s_{m,1} \\ s_{m,2} \\ \cdot \\ \cdot \\ s_{m,N-1} \\ s_{m,N} \end{bmatrix}, \quad P^y = \frac{1}{6}\begin{bmatrix} 2k_1 & k_1 & & & & \\ k_1 & 2(k_1+k_2) & k_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & k_{N-2} & 2(k_{N-2}+k_{N-1}) & k_{N-1} \\ & & & & k_{N-1} & 2k_{N-1} \end{bmatrix}.$$

The above algorithm requires $O(MN)$ operations and $O(MN)$ storage (i.e. it is of optimum order), is stable to round-off error, gives the exact solution (to within round-off error) to the set of linear equations defined by (15), easily generalises to three dimensions, and includes the equations appropriate to boundary points.

To put this in the context of an application code, let $M = N = 100$, and then the number of arithmetic operations and words of storage is $O(10^4)$ which poses no problem for today's computers. On the other hand, a 'banded solver' would require $O(10^6)$arithmetic operations and words of storage, which is two orders of magnitude more voracious. Of these two drawbacks (namely the larger number of arithmetic operations and the larger storage requirements) perhaps the most severe is the storage requirement, since the problem is unlikely to fit in (computer) memory. This then results in the need to perform extensive I/O to and from mass storage and the code becomes I/O bound (i.e. the CPU will spend most of its time waiting for operands to arrive from mass storage).

Regarding vectorisation, the algorithm appears, at first sight not to vectorise, because Gaussian elimination is a set of recursive operations. However, by performing each operation of Gaussian elimination in parallel for all members of the set of tridiagonal problems in x or y (as the case may be), a vectorisable operation is established in the transverse direction, and the algorithm is thus fully vectorisable.

It is interesting to note that even though the above algorithm is of optimal order, it can nevertheless still be improved upon in the context of taking a derivative [e.g. when computing U and V from χ and ψ using (8) and (9)], by exploiting the particular form of the right-hand side of (15). It can be shown that (15) may be rewritten as

$$\frac{1}{6}\left[h_{m-1}v_{m-1,n}+2\left(h_{m-1}+h_m\right)v_{m,n}+h_m v_{m+1,n}\right]=\frac{1}{2}\left[u_{m+1,n}-u_{m-1,n}\right]. \qquad (18)$$

The algorithm thus reduces to solving (18) for $v_{m,n}$ along lines of constant y (ie constant n). This is equivalent to solving a set of one-dimensional FE problems, holding y fixed, and is consistent with the underlying mathematics where the x-derivative at a point is obtained by a limiting process holding y fixed. This consistency does not obtain with triangular elements.

For *triangular* elements the best we can usually do for the solution of the mass-matrix problem is to either use a 'banded-solver' or an iterative method, both of which are considerably (typically at least an order of magnitude) more expensive than the algorithm for rectangular elements defined by (16) and (17).

2.3 Comparison with FD evaluation of first derivatives

Let us compare the efficiency and accuracy of evaluating a derivative using linear rectangular elements with those using second and fourth-order finite differences. First, we note that on any *uniform* subdomain (i.e . $h_{m-1}=k_{n-1}=h=\text{constant}$), equation (18) gives an $O\left(h^4\right)$accurate estimate for the derivative at the nodes, a result often referred to in the literature as super-convergence at nodes (Strang and Fix, 1973). Triangular elements generally give $O\left(h^2\right)$ accuracy and furthermore the solution of (14) is far more costly, so that we pay considerably more to get considerably less. The second-order centred finite-difference solution amounts to replacing the left-hand site of (18) by $hv_{m,n}$ (mass-lumping), and in comparison with rectangular FEs we find there is less computational effort (approximately a factor of 2.5) but considerably less accuracy [$O\left(h^2\right)$ compared to $O\left(h^4\right)$]. On the other hand, fourth-order finite-differences are somewhat more expensive and somewhat less accurate than rectangular FEs.

2.4 Products

Consider the problem of evaluating a product

$$v=uw, \qquad (19)$$

where u and w are given at nodal points and we require values of v at these nodal points. Expanding as before in terms of the basis functions $e^i(x,y)$, multiplying by an arbitrary basis function $e^k(x,y)$ and orthogonalising the error to the basis we obtain

$$P\mathbf{v}=\mathbf{u}N\mathbf{w}, \qquad (20)$$

where P is as before,

$$(\mathbf{u}N\mathbf{w})_k=\sum_{i,j}u_i w_j\int_D e^i(x,y)e^j(x,y)e^k(x,y)dxdy, \qquad (21)$$

the summations over i and j are performed over all nodal points, and D is the domain.

The double sum in (21) is not as formidable as it appears since $e^k(x,y)$ is only non-zero in a relatively small neighbourhood of the kth node and therefore only nearest neighbours of u and w are involved in the calculation. Its efficient evaluation for rectangular (two dimensional) and box (three dimensional) elements is discussed in detail by Staniforth and Beaudoin (1986) who show that it is advantageous to use Simpson quadrature rather than Gaussian. The evaluation of the right-hand side of (20) takes $O(MN)$ operations on an $M \times N$ grid for both rectangular and triangular elements (but triangular elements are more costly), and the prime difference in algorithmic efficiency is again the computational cost of solving a set of linear equations (the 'mass-matrix problem' $P\mathbf{v} = \mathbf{r}$). For rectangular elements the solution of this problem is approximately a factor of two less expensive than the evaluation of the RHS but, as mentioned above, for triangular elements it is *considerably* more costly, both in terms of arithmetic operations and storage.

2.5 Additional economies for linear rectangular (and box) elements, and a useful notation

As a further illustration of efficiency 'tricks' that may be employed when using linear rectangular elements, consider the evaluation of

$$D = u_x + v_y, \tag{22}$$

where u, v and D are all functions of the three dimensions x, y and z. The Galerkin finite-element approximation to this equation may be written formally as

$$P^x P^y P^z \mathbf{D} = P_x P^y P^z \mathbf{u} + P_y P^x P^z \mathbf{v}, \tag{23}$$

where P^x and P_x are tridiagonal one-dimensional operators having weights $\left[h_{m-1}/6, (h_{m-1} + h_m)/3, h_m/6\right]$ and $[-1/2, 0, 1/2]$ respectively, and with similar definitions obtaining for P^y, P_y, P^z and P_z.

The solution of (23) is usually found by explicitly applying the six one-dimensional operators on the right-hand side and then successively 'inverting' the three one-dimensional operators on the left-hand side resulting in the successive application of nine operators of approximately equal cost. However, we can achieve exactly the same result with half the work (i.e. by applying four operators instead of nine). To see this, we formally multiply (23) by $(P^x)^{-1}(P^y)^{-1}(P^z)^{-1}$ and use commutativity of operators to obtain

$$\mathbf{D} = (P^x)^{-1} P_x \mathbf{u} + (P^y)^{-1} P_y \mathbf{v}.$$

These formal operations may be justified rigorously, and the end result is equivalent to using one-dimensional FEs along lines of constant y to calculate u_x,one-dimensional FEs along lines of constant x to calculate v_y and then summing the result. Such a simplification is not possible with triangular (or tetrahedral) elements.

This is one example of the usefulness of the 'subscript/superscript' notation introduced above for rectangular (and box) elements. Another (similar) example is to consider the solution of the three-dimensional mass matrix problem $P\mathbf{v} = \mathbf{r}$, where $\mathbf{r}$ is given. Rewriting P as the product ($P^x P^y P^z$) of one-dimensional operators, this problem reduces to the successive solution of three sets of one-dimensional problems, namely

(i) solve $P^z \mathbf{f} = \mathbf{r}$ for columns (fixed x and y);

(ii) solve $P^y \mathbf{g} = \mathbf{f}$ for fixed x and z;

(iii) solve $P^x \mathbf{v} = \mathbf{g}$ for fixed y and z.

Again, the algorithm is of optimal order (i.e . $O(1)$ operations per node) and does not require any I/O to mass storage.

2.6 Rectangular vs. triangular elements

Why are rectangular elements (and box elements in three-dimensions) in general more efficient? Because the underlying geometrical partitioning of the domain, together with the form of the basis functions, leads to matrices whose structure may be exploited. Although linear rectangular elements at first glance appear more expensive than triangular elements (they need four degrees of freedom to define them, rather than three), the fact that they are expressible as the *product* of one-dimensional elements leads to considerable economies in their manipulation. Not only are fewer operations and less memory required in general for rectangular elements than for triangular (tetrahedral) elements, they are also inherently more vectorisable (because they are well-ordered in memory and easily accessed), further enhancing efficiency.

3. ELEMENT ORDER, SPATIAL EVOLUTIONARY ERROR AND CODE MODULARITY

Having indicated in the previous section the high cost for even simple problems of linear triangular elements when compared to both linear rectangular elements and finite differences, we now restrict our attention to rectangular elements. The next question to address is 'among rectangular elements, which ones are most suitable for fluid flow problems?' Put another way, is there any virtue to using higher-order elements (ie piecewise polynomials of higher order) such as quadratic or cubic elements, rather than linear elements? The answer appears to be no for several reasons, many of which may be found in Cullen and Morton (1980). They analysed the error associated with calculating an advection term such as $u\partial v/\partial x$ directly or calculating it as a two-stage process (compute the derivative, then the product) when using linear elements in the context of an evolutionary problem. They concluded that both methods asymptotically give an $O(h^4)$estimate for the spatial evolutionary error but that the two-stage method has a smaller coefficient. For a coding point of view, the two-stage method is probably to be preferred, since all terms may be computed in a modular way using a set of 'kernel' subroutines that handle the fundamental operations of differences, products and the solution of linear equations involving the mass matrix P.

It has been found that quadratic elements are generally (there may be some exceptions) less accurate and more costly than linear elements not to mention more complicated. Why are they less accurate? Because they have no super-convergence properties at nodes. What about cubic elements? They can have super-convergence properties but are much more expensive to work with per degree of freedom, boundary conditions are difficult to implement, program complexity is increased and, because of their higher order, there are more computational modes to worry about. The law of diminishing returns seems to apply.

4. STAGGERED, NON-STAGGERED AND MIXED-ORDER ELEMENTS AND THE FORM OF THE GOVERNING EQUATIONS

When formulating a fluid flow model, one often has to choose among several different forms of the governing equations. Furthermore, in the finite-element framework one is not restricted to using the same elements for all variables. Two possibilities come immediately to mind. First it is possible to use equal-order elements on a staggered grid (by analogy with staggered FD formulations). Secondly it is possible to use the same grid for all variables but to use mixed-order elements (ie expand some of the dependent variables in terms of one set of elements, and the others in terms of another set of elements of different order). In an ideal world one would expect that the choice of the form of the governing equations would be independent of the choice of element. In fact, we do not live in an ideal world and the two choices are intimately linked.

As a first example of how these two choices are linked, consider the shallow-water equations (1)-(3). Williams (1981) analysed various formulations for the FE solution of the linearised one-dimensional form of these equations using linear elements on both staggered and unstaggered meshes, where either velocity components or vorticity and divergence (as in the Introduction of the present study) were used as the momentum variables. His analysis indicates that to obtain good results with linear elements it is necessary to use either

(i) velocity components as momentum variables, and stagger the nodal points for the free surface height; or

(ii) vorticity and divergence as momentum variables and no staggering;

and furthermore that using velocity components as momentum variables and no staggering propagates energy in the wrong direction and is likely to cause noise problems in a non- linear model. That the conclusions of this analysis also apply to higher dimensions and the non-linear equations is well supported by:

(i) the absence of noise problems reported by Staniforth and Mitchell (1977, 1978), Cullen and Hall (1979) and Staniforth and Daley (1979) when using vorticity/divergence formulations and unstaggered elements;

(ii) the importance of introducing artificial smoothing to eliminate noise problems when using velocity components and unstaggered elements, reported by Cullen (1976); and

(iii) the noise problems reported by Walters (1983) for some of the formulations examined.

Williams' analysis is applicable to equal-order (namely linear) elements. It is also possible to use mixed-order elements (e.g. linear elements for velocity components and constant elements for free-surface height and vice versa), and it was concluded by Williams and Zienkiewicz (1981) and Walters (1983) that such schemes are viable.

The above discussion is centred on horizontal considerations, but similar considerations should also be expected to apply in the vertical. An analysis of a linearised version of the vertical FE discretisation scheme of Staniforth and Daley (1977) for the hydrostatic primitive equations is given by Béland et al. (1983). The analysis is similar in concept to that of Williams (1981), but focuses on the vertical instead of the horizontal. For the particular scheme analysed (unstaggered linear elements) it was shown that there is no spurious vertical propagation of energy in the wrong direction and that the scheme is basically sound. This analysis and that of Côté et al. (1983) did however lead to the diagnosis of a weakness, namely the existence of a vertical computational mode.

Although this mode may in principle be forced by the parametrised terms of the model, in practice it was found that a small amount of vertical diffusion adequately controls it. The source of the weakness was traced (see equation 10.20 of Côté et al., 1983) to the form chosen for the governing equations, and in particular to the treatment of the hydrostatic equation. A revised formulation was successfully implemented by Béland and Beaudoin (1985).

Cliffe (1981) presented an interesting analysis (based on the earlier analysis of Lee et al. (1980)) of the conservation properties of GFEM approximations to the Boussinesq equations. The basic idea is to expand each variable in terms of a finite-element space of unspecified order (or degree of continuity), and then to examine the conservation consequences of specific choices. An interesting feature of this framework is that it is easy to determine the *minimum* degree of continuity required of a certain variable after that of another has been set. The key point here is that if the finite-element space is too large (ie of too high an order) then computational modes will appear ('spurious pressure modes' in the terminology of Cliffe, 1981). It appears therefore to be desirable to choose the element order for one of the dependent variables of the problem (the higher the order, the higher the degree of complexity of the final algorithm), and to choose the minimum order for the other variables that is required to satisfy the desired conservation laws. Although the present author is not convinced that the best schemes are necessarily those which exactly conserve certain quantities (rather than those that almost conserve a larger number) the above- mentioned framework is nevertheless a valuable analysis tool for choosing elements appropriate to a given problem.

5. STABILITY

The stability properties of FE schemes turn out to be very similar to those of FD schemes. For pure advection (ie equation(3) with a non-divergent velocity field) a leap-frog scheme is conditionally stable whereas a forward (Euler) time scheme is unconditionally unstable (Cullen, 1973). For pure diffusion, a leap-frog scheme is unconditionally unstable, whereas a forward scheme is conditionally stable. These conclusions hold true for both FD and FE schemes, and the principal differences in stability are that the coefficients appearing in the stability conditions are slightly different. For one-dimensional advection, a leap-frog scheme using linear FEs has a stability condition $C = U\Delta t / \Delta x < 0.58$ whereas for centred second-order FDs $C < 1$, and for centred fourth-order FDs $C < 0.73$. Thus the most restrictive of these schemes from the point of view of stability is the FE one; however it is also the most accurate, and the price for increased accuracy is thus increased cost (ie more time steps).

A stability analysis of three different schemes (FD, FE and spectral) for solving the linearised shallow-water equations (cf.(1-3)) using a semi-implicit time scheme is given by Staniforth and Mitchell (1977). This analysis is performed in terms of response functions, and the conditions for stability for each of the three schemes is obtained by substituting the response functions appropriate to the method into the final result. The analysis was performed for a formulation using vorticity and divergence as momentum variables (cf. (1)-(3)) and also for two different formulations using velocity components as momentum variables. For a FE discretisation it was concluded that one of the velocity component formulations was not viable because it was overdamped, the other was not viable because it was computationally too expensive, and that this is a direct consequence of using a same-implicit time discretisation. This choice of time discretisation turned out to be quite fortunate, because only the vorticity/divergence formulation was left, which Williams (1981) later demonstrated was the only FE scheme on an unstaggered grid using linear elements that does not suffer from propagation of small scales in the wrong direction!

A second example of a stability analysis is that given by Côté et al. (1983) for the hydrostatic primitive equations. This analysis, although applied principally to the vertical discretisation FE scheme of Staniforth and

Daley (1977), is also applicable to FD discretisation schemes. It is similar to that described by Simmons et al. (1978) its principal virtues being that it is somewhat more general and less empirical. An explicit stability criterion (that the static stability of the reference temperature profile be greater than the explicit one) is given in the limit of small Δt for a case examined numerically by Simmons et al. (1978) and there is good agreement between the results. It was also shown that the first modes to go unstable if the criterion is violated are the computational modes due to the use of a (three time-level) semi-implicit time scheme. All of these conclusions apply equally well to FD schemes as they do to FE schemes, except for minor details.

6. SOME FURTHER CONSIDERATIONS

Although finite-element Galerkin schemes are optimal in the sense that for a given choice of finite-element space they orthogonalise the error to the basis, this does not necessarily mean that in the context of a fluid flow model they are the optimum choice among algorithms of a given complexity. An illustration of this point may be found in Staniforth and Mitchell (1977) who showed that a minor change in the approximation of second derivative terms, and no changes elsewhere, led to a striking improvement in the accuracy of the result. To see why this is so, we examine the problem of evaluating second derivatives.

6.1 **Second derivatives**

Consider the problem of evaluating

$$v = u_{xx}, \tag{24}$$

where u is known at the set of nodal points, and we require values of v at these same points. Expanding u and v in terms of linear FEs (cf. equation 13) and orthogonalising the error to the basis (by multiplying by an arbitrary basis function $e^k(x)$ and integrating over the domain) we obtain

$$P^x \mathbf{v} = P_{xx} \mathbf{u}, \tag{25}$$

where P^x and P_{xx} are tridiagonal matrices having weights $\left[h_{m-1}/6, (h_{m-1}+h_m)/3, h_m/6\right]$ and $\left[1/h_{m-1}, -(1/h_{m-1}+1/h_m), 1/h_m\right]$ respectively. It is easily shown by Taylor series that this gives an $O(h^2)$ approximation to the second derivative on any uniform subdomain, which is no better than that obtained for half the computational effort using centred second-order FDs. However, by rewriting (25) as

$$\tilde{P}^x \mathbf{v} = P_{xx} \mathbf{u}, \tag{26}$$

where P^x has been modified to be a tridiagonal matrix having weights $\left[h_{m-1}/12, 5(h_{m-1}+h_m)/12, h_m/12\right]$ we obtain an $O(h^4)$ approximation at no extra computational cost as our reward for venturing outside the Galerkin FE framework.

Staniforth and Mitchell (1977) demonstrated that this idea does not adversely affect the computational stability of a FE fluid flow model (the shallow-water equations of section 1), and can be used to good advantage for problems in higher dimensions such as the solution of the two-dimensional Poisson problem

$$f_{xx} + f_{yy} = g. \tag{27}$$

The approximation used for this problem was

$$\left(\tilde{P}^y P_{xx} + \tilde{P}^x P_{yy}\right)\mathbf{f} = \tilde{P}^x \tilde{P}^y \mathbf{g} \tag{28}$$

where P_{yy} and $\tilde{P}^y$ are the analogues of P_{xx} and $\tilde{P}^x$ as redefined. The resulting set of difference equations (involving a 9-point operator) was solved using discrete Fourier transforms, which are economical in their memory requirements ($O(MN)$ on a $M \times N$ mesh) and computational effort ($O(MN \log N)$) operations).

6.2 Aliasing

Let us now turn our attention towards aliasing and compare the FE treatment with the FD treatment. It is well known that evaluating the pointwise product of the two terms involved in an advection term (such terms are implicitly contained in the RH sides of (1)- (3)) generally leads to non-linear computational instability in the context of an evolution problem. The cause of this instability is the aliasing of that part of the spectrum generated by the product that cannot be resolved by the mesh. The cure in all Eulerian methods (FD, FE, spectral, etc.) is to either implicitly or explicitly control this aliasing by smoothing (filtering) the result. The spectral method is the most direct and simply ignores the least significant half of the spectrum, whereas FD methods smooth by averaging various quantities.

The optimum treatment of aliasing is therefore a trade-off between accuracy and stability; too little smoothing leads to computational instability, whereas too much degrades accuracy. Where does the FE method stand in all this? An illuminating example is given by comparing Arakawa's approximation for two-dimensional incompressible flow on a uniform grid with that of the FE method using linear elements. It was shown by Jespersen (1974) that the treatments of the advection terms are identical. The only difference between the two methods, therefore, is that the time derivative term in the FE approximation is multiplied by the projection (or mass) matrix P of section 2, and the FE method is consequently a little more expensive. However the FE approximation leads to an $O(h^4)$ estimate for the spatial evolutionary error rather than the $O(h^2)$ estimate for Arakawa's (1966) method, and the increased accuracy more than compensates for the added work.

Noting that the application to one side of an equation of a 'smoothing' operator (such as the projection matrix P) is equivalent to the application to the other side of an 'unsmoothing' operator, the above result may be interpreted as follows: both methods control stability by using the same smoothing operator, but the FE method 'sharpens the response' to increase accuracy without adversely affecting stability, and is consequently a more efficient scheme.

6.3 Boundary conditions

An often-overlooked aspect of the FE method when using linear elements as compared to higher-order FD schemes, is the incorporation of boundary conditions. Higher-order FD schemes in the literature usually achieve better accuracy by involving more neighbouring points in the calculations, thus increasing the bandwidth of the matrices involved. For example, fourth order FD derivative approximations in one dimension involve five adjacent points rather than the three adjacent points of second-order FDs and the FE method using linear elements. In three-point schemes, the boundary conditions are used to obtain an equation associated with a boundary point, and the discretised governing equation is applied directly at all internal points. With a five-point FD scheme it is necessary to impose an extra computational boundary condition at all the internal points immediately adjacent to the boundary in order to obtain as many equations as there are unknowns. This can be a delicate procedure and if not done properly can result in the forcing of the computational modes associated with the use of a higher-order difference scheme. The linear FE scheme on the other hand can achieve fourth-order accuracy (for a first derivative, for example) without the need for additional (artificial) boundary conditions for the points immediately adjacent to boundaries. This is particularly advantageous when solving Poisson problems such as (10) and (11).

7. TIME SCHEMES

The choice of time discretisation and how it interacts with the space discretisation can have an important impact on the efficiency of a fluid dynamics code. The simplest time schemes are explicit. In these schemes the partial time derivative of a variable is isolated on the left-hand side of an equation and approximated in terms of a time difference involving the new and previous time steps, whereas the right-hand side is evaluated explicitly using known values of the dependent variables at previous time steps. The right-hand side may be evaluated using the traditional Galerkin FE method, by breaking it down into several steps or by grouping them together as fluxes. Terms on the right-hand side are evaluated using the methods described in the preceding sections, and the mass matrix problem (associated with the time differencing of the left-hand side) is solved using the efficiency algorithm of section 2.

Explicit time schemes are very efficient and appropriate for problems where the time step is restricted by the time truncation error rather than by stability considerations. However, they are not particularly efficient for stiff sets of equations (such as the shallow-water equations (1)-(3)), and it is often advantageous to treat some or all, of the terms implicitly in time. This can be done in several ways. For example, Baker and Soliman (1983) approximate all terms as time averages or differences over times $n\Delta t$ and $(n+1)\Delta t$. This results in a set of coupled non-linear equations at each time step which are solved iteratively. The advantage of such an approach is that fewer time steps are required for stiff systems of equations because of the enhanced stability. The disadvantage is that each time step is more costly than that of an explicit time scheme because of the need to iterate. Nevertheless this approach can be cost effective, provided that the time step may be increased significantly without loss of accuracy, and the iterative technique is efficient.

A further alternative, particularly effective for stiff sets of equations, is to identify those terms that are responsible for restricting the time step because of stability. The linear contributions of these terms are then treated implicitly in time, whereas perturbations about them, and the remaining terms, are treated explicitly. This idea was first applied to a finite-difference discretisation of the shallow-water equations by Kwizak and Robert (1971), who found that such a scheme (which they termed semi-implicit) is five times more efficient than an explicit leap-frog scheme. Later, Staniforth and Mitchell (1977) applied it to an FE discretisation of the same equations with a similar improvement. The extension to the three-dimensional hydrostatic primitive equations was first demonstrated for an FD discretisation by Robert et al. (1972) and subsequently by Staniforth and Daley (1979) for a FE formulation.

Semi-implicit schemes offer a good compromise for stiff systems of equations. They require fewer time steps than explicit schemes (because of their enhanced stability) and yet do not require significantly more computations per time step.

8. CONCLUSIONS

For fluid flow problems in regular domains, triangular (and tetrahedral) elements cannot in general compete with rectangular (and box) elements, because they do not permit an efficient solution of the mass-matrix problem. An optimal (or close to optimal) scheme for given computational effort in this context is achieved by a judicious mix of techniques.

First derivative and product terms are well handled by GFEM schemes using linear rectangular (and box) elements. They have the attributes of simplicity, accuracy stability and straightforward incorporation of boundary conditions, and compete favourably with fourth-order FD schemes. However for these elements it is often advantageous to go outside the Galerkin framework when approximating second derivatives, as described in section 6. Higher-order rectangular elements are a possibility, but the introduction of more computational modes and added programming complexity are decided disadvantages, and the law of diminishing returns applies.

It is important to analyse carefully the properties of linearised versions of the discrete models in order to obtain maximum accuracy and efficiency. Of particular importance is the form of the governing equations, vis-à-vis the choice of staggered, non-staggered and/or mixed-order elements, and the choice of an appropriate time scheme.

ACKNOWLEDGEMENTS

Thanks are due to Denise Smith and Margaret Andrews for their help preparing the manuscript, and to Chris Smith for reviewing it.

REFERENCES

Ahlberg, J.H., E.N. Wilson and J.L. Walsh, 1967. *The Theory of Splines and their Applications*, Academic Press, NY, pp. 14-15.

Arakawa, A., 1966: Computational design for long-term numerical integration of the equations of fluid motion: two-dimensional incompressible flow - Part I. *J. Comp. Phys.*, **1**, 119-143.

Baker, A.J. and M.O. Soliman, 1983: A finite element algorithm for computational fluid dynamics. *AIAA J.*, **21**, 816-827.

Béland, M., J. Côté and A. Staniforth, 1983: The accuracy of a finite-element vertical discretization scheme for primitive equation models: comparison with a finite-difference scheme. *Mon. Wea. Rev.*, *111*, 2298-2318.

Béland, M. and C. Beaudoin, 1985: A global spectral model with a finite element formulation for the vertical discretization: adiabatic formulation. *Mon. Wea. Rev.*, **113**, 1910-1919.

Cliffe, K.A., 1981: On conservative finite element formulations of the invscid Boussinesq equations. *Int. J. Numer Methods Fluids*, **1**, 117-127.

Côté, J., M. Béland and A. Staniforth, 1983: Stability of vertical discretization schemes for semi-implicit primitive equation models; theory and application. *Mon. Wea. Rev.*, **111**, 1189-1207.

Cullen, M.J.P., 1973: A simple finite-element method for meteorology problems. *J. Inst. Math.Applications*, **11**, 15-21.

Cullen, M.J.P., 1976: On the use of artificial smoothing in Galerkin and finite difference solutions of the primitive equations. *Quart. J. R. Met. Soc.*, **102**, 77-93.

Cullen, M.J.P. and C.D. Hall, 1979: Forecasting and general circulation results from finite-element models. *Quart. J. R. Met. Soc.*, **105**, 571-592 .

Cullen, M.J.P. and K.W. Morton, 1980: Analysis of evolutionary error in finite element and other methods. *J. Comput. Phys.*, **34**, 245-267.

Jespersen, D.C., 1974: Arakawa's method is a finite-element method. *J. Comput. Phys.*, **16**, 383-390.

Kwizak, M. and A.J. Robert, 1971: A semi-implicit scheme for grid point atmospheric models of the primitive equations. *Mon. Wea. Rev.*, **99**, 32-36.

Lee, R.L., P.M. Gresho, S.T. Chan and R.L. Sani, 1980: A comparison of several conservative forms for finite-element formulations of the incompressible Navier-Stokes or Boussinesq equations. *Proc. 3rd Int. Conf. on Finite-elements in Flow Problems*, Banff, Canada, DH Norrie (ed.), pp 216-227.

Robert, A.J., J. Henderson and C. Turnbull, 1972: An implicit time integration scheme for baroclinic models of the atmosphere. *Mon. Wea. Rev.*, **100**, 329-335.

Simmons, A.J., B.J. Hoskins and D.M. Burridge, 1978: Stability of the semi-implicit method of time integration. *Mon. Wea. Rev.*, **106**, 405-412.

Staniforth, A.N. and R.W. Daley, 1977: A finite-element formulation for the vertical discretization of sigma-coordinate primitive-equation models. *Mon. Wea. Rev.*, **105**, 1108-1118.

Staniforth , A.N. and H.L. Mitchell, 1977: A semi-implicit finite-element barotropic model. *Mon. Wea. Rev.*, **105**, 154-169 .

Staniforth, A.N. and H.L. Mitchell, 1978: A variable-resolution finite-element technique for regional forecasting with the primitive equations. *Mon. Wea. Rev.*, **106**, 439-447.

Staniforth, A.N. and R.W. Daley, 1979: A baroclinic finite-element model for regional forecasting with the primitive equations. *Mon. Wea. Rev.*, **107**, 107-121.

Staniforth , A.N. and C. Beaudoin, 1986: On the efficient evaluation of certain integrals in the Galerkin FE method. *Int. J. Numer. Methods Fluids*, **6**, 317-324 .

Strang, G. and G.J. Fix, 1973: An Analysis of the Finite Element Method, Prentice Hall, Englewood Cliffs, NJ.

Walters, R.A., 1983: Numerically induced oscillations in finite-element approximations to the shallow water equations. *Int. J. Numer. Methods Fluids*, **3**, 591-604 .

Williams, R.T., 1981: On the formulation of finite-element prediction models. *Mon. Wea. Rev.*, **109**, 463-466.

Williams, R.T. and O.C Zienkiewicz, 1981: Improved finite-element forms for the shallow-water wave equations. *Int. J. Numer. Methods Fluids*, **1**, 81-97 .