COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING 78 (1990) 165–179
NORTH-HOLLAND

ADAPTIVE IMPLICIT–EXPLICIT FINITE ELEMENT ALGORITHMS FOR FLUID MECHANICS PROBLEMS[†]

T.E. TEZDUYAR and J. LIOU

Department of Aerospace Engineering and Mechanics and Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Received 10 November 1988
Revised manuscript received 17 April 1989

The adaptive implicit–explicit (AIE) approach is presented for the finite element solution of various problems in computational fluid mechanics. In the AIE approach the elements are dynamically (adaptively) arranged into differently treated groups. The differences in treatment could be based on considerations such as the cost efficiency, the type of spatial or temporal discretization employed, the choice of field equations, etc. Several numerical tests are performed to demonstrate that with this approach substantial savings in the CPU time and memory can be achieved.

1. Introduction

Large-scale problems in computational fluid mechanics may involve linear equation systems with massive global matrices. Matrices of this size usually arise from the implicit time-integration of spatially discretized time-dependent equations or from a direct solution approach to spatially discretized time-independent equations. For example, the incompressible Navier–Stokes equations in the vorticity–stream function formulation involve a time-dependent convection–diffusion equation for the vorticity and a Poisson equation for the stream function. For most problems of practical interest, especially in three dimensions, handling of large global matrices places a very heavy demand on the computational resources in terms of the CPU time and memory. This demand is too heavy to accommodate even for the largest and fastest computers available today. In this paper we describe, in the context of the vorticity–stream function formulation, alternative solution techniques which substantially reduce the need for the storage and factorization of large global matrices.

The implicit–explicit algorithm proposed by Hughes and Liu [1, 2] (for solid mechanics and heat conduction problems) involves static allocation of the implicit and explicit elements based on the stability limit of the explicit algorithm. In [3] the same static allocation approach was applied to fluid mechanics problems. The adaptive implicit–explicit (AIE) scheme was first presented, in its preliminary stage, in [3]. It is based on dynamic grouping of the elements into the implicit and explicit subsets as dictated by the element level stability and accuracy considerations. For this purpose the algorithm monitors the element level Courant number

[†]This research was sponsored by NASA-Johnson Space Center under contract NAS-9-17892 and by NSF under grant MSM-8796352.

and some measure of the dimensionless wave number. The element level Courant number is based on the local convection and diffusion transport rates whereas the dimensionless wave number reflects the local smoothness of the 'previous' solution and the spatial discretization. The 'previous' solution can come from the previous time step, nonlinear iteration step, or pseudo-time iteration step, whichever is appropriate. In this approach we can have 'implicit refinement' where it is needed. Elsewhere in the domain, computations are performed explicitly thus resulting in savings in memory and CPU time. The savings can be further increased by performing, as often as desired, an equation renumbering at the implicit zones to obtain optimal bandwidths. Compared to the adaptive schemes based on grid-moving or element-subdividing, the AIE method involves minimal bookkeeping and no geometric constraints.

It is important to note that the dynamic grouping of the AIE scheme does not necessarily have to be with respect to implicit and explicit elements. The letters 'I' and 'E' in 'AIE' could be referring to any two procedures. In fact, there is no reason for the number of possible choices to be limited to two. For a given element, the rationale for favoring one procedure over another one could be based on any factor, such as the cost efficiency, the type of spatial or temporal discretization, the differential equations used for modelling, etc. In this paper, as an example, we implement the AIE scheme also in conjunction with the flux corrected transport (FCT) method [4, 5].

2. Problem statement and the finite element formulation

We consider problems governed by the convection–diffusion equation and the two-dimensional incompressible Navier–Stokes equations. For the Navier–Stokes equations we use the vorticity–stream function formulation; this formulation consists of a time-dependent transport equation for the vorticity ω and a Poisson equation relating the stream function ψ to the vorticity. Since the procedures for the spatial and temporal discretizations for the convection–diffusion equation are similar to those for the transport equation of the vorticity–stream function formulation, only the latter will be discussed in this section.

The field equations for the vorticity–stream function formulation of the two-dimensional incompressible Navier–Stokes equations are given as

$$\frac{\partial \omega}{\partial t} + \boldsymbol{u} \cdot \nabla \omega = \nu \nabla^2 \omega \,, \tag{2.1}$$

$$\nabla^2 \psi = -\omega \,, \tag{2.2}$$

where $\boldsymbol{u}$ is the velocity and ν is the kinematic viscosity. The incompressibility condition is automatically satisfied by the following definition of the stream function:

$$\frac{\partial \psi}{\partial x_2} = u_1 \,, \tag{2.3a}$$

$$\frac{\partial \psi}{\partial x_1} = -u_2 \,. \tag{2.3b}$$

The flow domain can have several internal boundaries corresponding to possible obstacles (holes) in the flow field. In such cases the unknowns of the problem are: the value of the vorticity at all interior points (ω_*), the value of the vorticity at all boundary points where both components of the velocity are specified (ω_{Gq}) (this includes all the external and internal solid boundaries), and the value of the stream function at all interior points and on the internal boundaries (ψ_{*q}).

The differential equations given by (2.1) and (2.2), together with the initial condition for the vorticity and the suitable boundary conditions for the vorticity and the stream function, can be translated, via a proper variational formulation [6], into a set of ordinary differential equations. That is

$$\tilde{M}(d_{*q})a_* + \tilde{C}(d_{*q})v_* = \tilde{F}(d_{*q}, v_{Gq}, a_{Gq}), \tag{2.4a}$$

$$v_*(0) = (v_*)_0, \tag{2.4b}$$

$$Kd_{*q} = F_{II}(v_*, v_{Gq}), \tag{2.5a}$$

$$Mv_{Gq} = F_{III}(d_{*q}, v_*), \tag{2.5b}$$

where d_{*q}, v_*, a_*, v_{Gq} and a_{Gq} are the vectors of nodal values of ψ_{*q}, ω_*, $\frac{\partial \omega_*}{\partial t}$, ω_{Gq} and $\frac{\partial \omega_{Gq}}{\partial t}$, respectively.

REMARK 2.1. Particular attention must be paid to the boundary conditions for the vorticity on the (external and internal) solid surfaces and for the stream function on the internal boundaries. Discussion of these boundary conditions and the derivation of the proper variational formulations can be found in [6].

REMARK 2.2. Equation systems (2.4a) and (2.5) are derived from the variational formulations corresponding to (2.1) and (2.2), respectively. The initial condition for (2.1) is represented by (2.4b).

REMARK 2.3. The matrices K and M are symmetric and positive-definite. The matrix M is topologically one-dimensional because it is associated with the vector v_{Gq} corresponding to the (unknown) value of the vorticity on the boundaries. The matrices $\tilde{M}$ and $\tilde{C}$ are functions of the stream function because in the variational formulation of (2.1) we employ a Petrov–Galerkin method [7] with weighting functions dependent upon the velocity field.

REMARK 2.4. Equations (2.4) and (2.5) are solved by employing a predictor/multicorrector algorithm in conjunction with a block-iteration scheme [6]. In fact the way we have written these equations reflects the way the block-iteration scheme works. In the blocks corresponding to (2.4a), (2.5a) and (2.5b), we update v_*, d_{*q} and v_{Gq}, respectively. To move from time step n to $n+1$ we start with the predictions

$$(v_*)^0_{n+1} = (v_*)_n + (1-\alpha)\,\Delta t(a_*)_n, \tag{2.6a}$$

$$(a_*)^0_{n+1} = 0, \tag{2.6b}$$

and perform the following iterations:

block 1:

$$\bar{M}^i_{n+1}(\Delta a_*)^i_{n+1} = R^i_{n+1}\,, \tag{2.7a}$$

where

$$\bar{M}^i_{n+1} = \tilde{M}((d_{*_q})^i_{n+1}) + \alpha\,\Delta t\,\tilde{C}((d_{*_q})^i_{n+1}) \tag{2.7b}$$

and

$$\begin{aligned} R^i_{n+1} = {} & \tilde{F}((d_{*_q})^i_{n+1}, (v_{G_q})^i_{n+1}, (a_{G_q})^i_{n+1}) \\ & - (\tilde{M}((d_{*_q})^i_{n+1})(a_*)^i_{n+1} + \tilde{C}((d_{*_q})^i_{n+1})(v_*)^i_{n+1})\,, \end{aligned} \tag{2.7c}$$

with updates

$$(v_*)^{i+1}_{n+1} = (v_*)^i_{n+1} + \alpha\,\Delta t(\Delta a_*)^i_{n+1}\,, \tag{2.7d}$$

$$(a_*)^{i+1}_{n+1} = (a_*)^i_{n+1} + (\Delta a_*)^i_{n+1}\,. \tag{2.7e}$$

Here i is the iteration count, Δt is the time step and α is a parameter which controls the stability and accuracy of the time integration.

block 2:

$$K(d_{*_q})^{i+1}_{n+1} = F_{\mathrm{II}}((v_*)^{i+1}_{n+1}, (v_{G_q})^i_{n+1})\,. \tag{2.8}$$

block 3:

$$M(v_{G_q})^{i+1}_{n+1} = F_{\mathrm{III}}((d_{*_q})^{i+1}_{n+1}, (v_*)^{i+1}_{n+1})\,. \tag{2.9}$$

The iterations continue until a predetermined convergence criterion is met.

3. The adaptive implicit–explicit (AIE) scheme

In our block-iterative procedure, at every iteration we need to solve three equation systems: (2.7a), (2.8) and (2.9). The cost involved in (2.9) is not major (see Remark 2.3) and therefore we solve it with a direct method. We are mainly interested in minimizing the CPU time and memory demands of equations (2.7a) and (2.8) which, after dropping all the subscripts and superscripts, can be rewritten as follows:

$$\bar{M}\,\Delta a = R\,, \tag{3.1}$$

$$Kd = F\,. \tag{3.2}$$

We need to remember that K is symmetric and positive-definite but $\bar{M}$, in general, is not. We propose to employ the AIE scheme for both of these equations.

The AIE scheme in the context of the vorticity transport equation

Let $\mathscr{E}$ be the set of all elements, $e = 1, 2, \ldots, n_{el}$. The assembly of the global matrix $\bar{M}$ in (3.1) can be expressed as follows:

$$\bar{M} = \sum_{e \in \mathscr{E}} \bar{M}^e , \tag{3.3}$$

where $\bar{M}^e$ is the element contribution matrix which is obtained by permuting the element level matrix $\bar{m}^e$. It should be noted that $\bar{M}^e$ has the same dimension as the global matrix $\bar{M}$ but only as many nonzero entries as $\bar{m}^e$ (e.g. 4×4 for a two-dimensional quadrilateral element with a scalar unknown).

Let $\mathscr{E}_I$ and $\mathscr{E}_E$ be the subsets of $\mathscr{E}$ corresponding to the implicit and explicit elements, respectively, such that

$$\mathscr{E} = \mathscr{E}_I \cup \mathscr{E}_E , \tag{3.4a}$$

$$\emptyset = \mathscr{E}_I \cap \mathscr{E}_E . \tag{3.4b}$$

Consequently, from (3.3) we get

$$\bar{M} = \sum_{e \in \mathscr{E}_I} \bar{M}^e + \sum_{e \in \mathscr{E}_E} \bar{M}^e . \tag{3.5}$$

The AIE scheme is based on modifying $\bar{M}$ by replacing $\bar{M}^e$ ($\forall e \in \mathscr{E}_E$) with its lumped mass matrix part, that is

$$\bar{M} = \sum_{e \in \mathscr{E}_I} \bar{M}^e + \sum_{e \in \mathscr{E}_E} M_L^e . \tag{3.6}$$

The grouping given by (3.4) is done dynamically (adaptively) based on the stability and accuracy considerations.

The stability criterion is in terms of the element Courant number $C_{\Delta t}$ which is defined as

$$C_{\Delta t} = \|u\| \, \Delta t / h , \tag{3.7}$$

where h is the 'element length' [7]. Any element with its Courant number greater than the stability limit of the explicit method should belong to the implicit group $\mathscr{E}_I$.

For accuracy considerations we use a quantity which is meant to be a measure of the dimensionless wave number [7]. For this purpose, in [3] the 'jump' in the solution across an element was defined as

$$j^e(\omega) = (\max_a (\omega_a^e) - \min_a (\omega_a^e)) / (\omega_{max} - \omega_{min}) , \tag{3.8}$$

where a is the element node number and ω_a^e is the value of the dependent variable ω at node a of the element e. The instantaneous global maximum and minimum values of ω are denoted by $\omega_{\max}$ and $\omega_{\min}$. This definition reflects the magnitude of the variation of the solution across an element. It was proposed in [3] that the elements with jumps greater than a predetermined value should belong to the group $\mathscr{E}_{\mathrm{I}}$.

In our numerical tests involving one-dimensional propagation of various triangular profiles we observed that the accuracy of the solution is affected not only by the jump in the solution but also by the derivative of the flux of the dependent variable. Therefore, as an alternative criterion we propose to employ the product of the jump and the derivatives of the flux; that is

$$\sigma^e = j^e(\omega)\left(\left|\frac{\partial(u_1\omega)}{\partial x_1}\right| + \left|\frac{\partial(u_2\omega)}{\partial x_2}\right|\right)^e . \tag{3.9}$$

A global scaling is needed for this quantity and we propose the following one:

$$\sigma_{\mathscr{E}}^e = \sigma^e/\sigma_{\max}^e , \tag{3.10}$$

where

$$\sigma_{\max}^e = \max_{e\in\mathscr{E}}\left(\left|\frac{\partial(u_1\omega)}{\partial x_1}\right| + \left|\frac{\partial(u_2\omega)}{\partial x_2}\right|\right)^e . \tag{3.11}$$

In this case the elements with $\sigma_{\mathscr{E}}^e$ greater than a predetermined value belong to group $\mathscr{E}_{\mathrm{I}}$.

Implementation of the AIE scheme is quite straightforward. A global search is performed on the entire set of elements. With this search, based on the two criteria given by (3.7) and (3.10), the elements are grouped into the subsets $\mathscr{E}_{\mathrm{I}}$ and $\mathscr{E}_{\mathrm{E}}$.

REMARK 3.1. In the AIE approach one can have a high degree of refinement throughout the mesh and raise the implicit flag only for those elements which need to be treated implicitly.

REMARK 3.2. The savings in the CPU time and memory can be maximized by performing, as often as desired, an equation renumbering at the 'implicit zones' to obtain optimal bandwidths. Bandwidth optimizers are already available for finite element applications, especially in the area of structural mechanics.

REMARK 3.3. Time-dependent convection–diffusion of a passive scalar can be treated as a special case of (2.1) with the velocity field $\boldsymbol{u}$ given. The only equation system that needs to be solved is (3.1).

The AIE scheme in the context of the flux-corrected transport (FCT) method

The AIE scheme does not have to be based only on implicit–explicit grouping. It can be based on any element level, dynamic, and rational selection between an 'I-procedure' and an 'E-procedure'. Here we give an example by employing the AIE scheme in conjunction with the FCT method described in [4, 5]. We apply this method to the time-dependent convection–diffusion of a passive scalar (see Remark 3.3).

The FCT method given in [4, 5] is based on correction of the flux obtained with a lower-order scheme with the flux obtained with a higher-order scheme, while limiting the maximum allowable correction. The balance law is satisfied at the element level. In our case, as the lower-order scheme we use the one-pass explicit SUPG method and as the higher-order scheme the multi-pass explicit SUPG method. With a sufficient number of iterations (typically three) the multi-pass explicit method produces solutions indistinguishable from those obtained by the implicit method. In the utilization of our AIE scheme we define the 'E-procedure' to be the one-pass explicit SUPG method alone and the 'I-procedure' to be the FCT method based on the lower- and higher-order schemes described above.

REMARK 3.4. Variations of the Taylor–Galerkin method have been used in [5] as lower- and higher-order discretization techniques. We consider the Taylor–Galerkin method to be a close kin (under some circumstances a special case) of the SUPG method proposed in [8–10]. One can easily see the close connection between the Taylor–Galerkin method and the one-pass explicit 'temporal criterion'-based 'A^2-form' of the SUPG method described in [8–10]. It is not our intention here to make any statement about what lower- and higher-order schemes should be used with the FCT method or about whether the FCT method should be preferred over some other method. We just want to give an example.

For the AIE scheme we employ the explicit 'v-form' of the predictor/multicorrector algorithm described by (2.7). That is

$$\boldsymbol{M}_{\mathrm{L}}(\Delta \boldsymbol{v}_*)^i_{n+1} = \bar{\boldsymbol{R}}^i_{n+1}\,, \tag{3.12a}$$

where $\boldsymbol{M}_{\mathrm{L}}$ is the (diagonal) lumped mass matrix version of $\tilde{\boldsymbol{M}}$ and

$$\bar{\boldsymbol{R}}^i_{n+1} = \Delta t\,\tilde{\boldsymbol{F}}_{n+\alpha} - \tilde{\boldsymbol{M}}((\boldsymbol{v}_*)^i_{n+1} - (\boldsymbol{v}_*)_n) - \Delta t\,\tilde{\boldsymbol{C}}(\alpha(\boldsymbol{v}_*)^i_{n+1} + (1-\alpha)(\boldsymbol{v}_*)_n)\,. \tag{3.12b}$$

Note that these expressions are quite simple due to the fact that this is a linear problem and that no stream function is involved.

The nodal value correction vector $(\Delta \boldsymbol{v}_*)^i_{n+1}$ and the residual vector $\bar{\boldsymbol{R}}^i_{n+1}$ can be written as sums of their element level contribution vectors:

$$(\Delta \boldsymbol{v}_*)^i_{n+1} = \sum_{e \in \mathscr{E}} (\Delta \boldsymbol{v}^e_*)^i_{n+1}\,, \tag{3.13}$$

$$\bar{\boldsymbol{R}}^i_{n+1} = \sum_{e \in \mathscr{E}} (\bar{\boldsymbol{R}}^e)^i_{n+1}\,. \tag{3.14}$$

The AIE/FCT scheme employed is outlined below:

1. Given $(\boldsymbol{v}_*)_n$, calculate a predicted value for $(\boldsymbol{v}_*)_{n+1}$:

$$(\boldsymbol{v}_*)^0_{n+1} = (\boldsymbol{v}_*)_n + (1-\alpha)\,\Delta t(\boldsymbol{a}_*)_n\,. \tag{3.15}$$

2. Based on this predicted value, by using a lower-order scheme calculate a lower-order residual $(\bar{\boldsymbol{R}}^e)^{\mathrm{low}}_{n+1}$ $\forall e \in \mathscr{E}$; compute the lower-order correction corresponding to this

residual and determine the lower-order solution:

$$(\Delta \boldsymbol{v}_*^e)_{n+1}^{\text{low}} = \boldsymbol{M}_{\text{L}}^{-1}(\bar{\boldsymbol{R}}^e)_{n+1}^{\text{low}} \quad \forall e \in \mathscr{E} , \tag{3.16}$$

$$(\boldsymbol{v}_*)_{n+1}^{\text{low}} = (\boldsymbol{v}_*)_{n+1}^0 + \sum_{e \in \mathscr{E}} (\Delta \boldsymbol{v}_*^e)_{n+1}^{\text{low}} . \tag{3.17}$$

3. Based on the predicted value $(\boldsymbol{v}_*)_{n+1}^0$ calculate the residual $(\bar{\boldsymbol{R}}^e)_{n+1}^0$ $\forall e \in \mathscr{E}_{\text{I}}$ and compute the correction corresponding to this residual:

$$(\Delta \boldsymbol{v}_*^e)_{n+1}^0 = \boldsymbol{M}_{\text{L}}^{-1}(\bar{\boldsymbol{R}}^e)_{n+1}^0 \quad \forall e \in \mathscr{E}_{\text{I}} . \tag{3.18}$$

4. Shift the correction $(\Delta \boldsymbol{v}_*^e)_{n+1}^0$ $\forall e \in \mathscr{E}_{\text{I}}$ by subtracting the lower-order corrections and determine the first iterative solution:

$$i \leftarrow 0 ,$$

$$(\Delta\Delta \boldsymbol{v}_*^e)_{n+1}^0 = (c^e)_{n+1}^0((\Delta \boldsymbol{v}_*^e)_{n+1}^0 - (\Delta \boldsymbol{v}_*^e)_{n+1}^{\text{low}}) \quad \forall e \in \mathscr{E}_{\text{I}} , \tag{3.19}$$

$$(\boldsymbol{v}_*)_{n+1}^1 = (\boldsymbol{v}_*)_{n+1}^{\text{low}} + \sum_{e \in \mathscr{E}_{\text{I}}} (\Delta\Delta \boldsymbol{v}_*^e)_{n+1}^0 . \tag{3.20}$$

Here $(c^e)_{n+1}^i$ is a parameter [4, 5] determined by the maximum correction allowed for element e.

5. Perform a desired number of iterations (n_{FCT}); in each iteration, compute the higher-order correction and determine the FCT solution (based on the higher-order correction and the lower-order solution):

$$i \leftarrow i + 1 ,$$

$$(\Delta \boldsymbol{v}_*^e)_{n+1}^i = \boldsymbol{M}_{\text{L}}^{-1}(\bar{\boldsymbol{R}}^e)_{n+1}^i \quad \forall e \in \mathscr{E}_{\text{I}} , \tag{3.21}$$

$$(\Delta\Delta \boldsymbol{v}_*^e)_{n+1}^i = (c^e)_{n+1}^i((\Delta\Delta \boldsymbol{v}_*^e)_{n+1}^{i-1} + (\Delta \boldsymbol{v}_*^e)_{n+1}^i) \quad \forall e \in \mathscr{E}_{\text{I}} , \tag{3.22}$$

$$(\boldsymbol{v}_*)_{n+1}^{i+1} = (\boldsymbol{v}_*)_{n+1}^{\text{low}} + \sum_{e \in \mathscr{E}_{\text{I}}} (\Delta\Delta \boldsymbol{v}_*^e)_{n+1}^i , \tag{3.23}$$

if $i < n_{\text{FCT}}$ then goto 5.

6. Pick up the value of the last iterative solution:

$$(\boldsymbol{v}_*)_{n+1} = (\boldsymbol{v}_*)_{n+1}^{i+1} \tag{3.24}$$

and goto the next time step:

$$n \leftarrow n + 1$$

goto 1.

Limiting the steps 3–6 to the group $\mathscr{E}_{\text{I}}$ results in savings in the computational cost involved.

Solution of the discrete Poisson equation

We use a preconditioned conjugate gradient method [11] to solve the equation system given by (3.2). In conjunction with the AIE scheme employed for (3.1) we define our preconditioner matrix P to be

$$P = \sum_{e \in \mathscr{E}_I} K^e + \sum_{e \in \mathscr{E}_E} \operatorname{diag}(K^e), \tag{3.25}$$

where K^e is the element contribution matrix of K. If $\mathscr{E}_E = \emptyset$ then $P = K$ and the solution technique becomes a direct one. If $\mathscr{E}_I = \emptyset$ then the method becomes a Jacobi iteration [11]. Other definitions for P are of course possible; the one given by (3.25) is quite simple to implement.

REMARK 3.5. The incompressible Navier–Stokes equations in the velocity–pressure formulation, upon spatial and temporal discretizations, can also translate to a set of equations given by (3.1) and (3.2). In this case (3.1) and (3.2) would correspond, respectively, to the momentum equation and the incompressibility constraint. The latter would consist of a discrete Poisson equation for pressure.

REMARK 3.6. Another example for the type of problems which can translate to (3.1) and (3.2) is the electrophoresis separation process (see [12]). In the context of a block-iteration scheme, the time-dependent transport equation for the chemical species and the equation for the electric potential would transform to (3.1) and (3.2), respectively.

4. Numerical tests

We have tested the AIE algorithm on problems governed by the convection–diffusion and the two-dimensional incompressible Navier–Stokes equations.

One-dimensional pure advection of a cosine wave

The purpose of this simple test is to provide a conceptual illustration of the way the AIE scheme works. A cosine wave of unit amplitude is being advected from the left to the right with an advection velocity of 1.0. A uniform mesh is being used and the element Courant number is 0.8.

The AIE scheme is based on selection between the implicit and the one-pass explicit versions of the streamline–upwind/Petrov–Galerkin (SUPG) procedure. The threshold value for the test parameter given by (3.10) is 10^{-5}. Figure 1 shows the solution and the corresponding distribution of the implicit elements at time steps 0, 50 and 100. Compared to the implicit method, the AIE scheme results in a reduction of 80% in the CPU time and 81% in the memory needed for the global matrix; the solutions obtained by the implicit and the AIE schemes are indistinguishable.

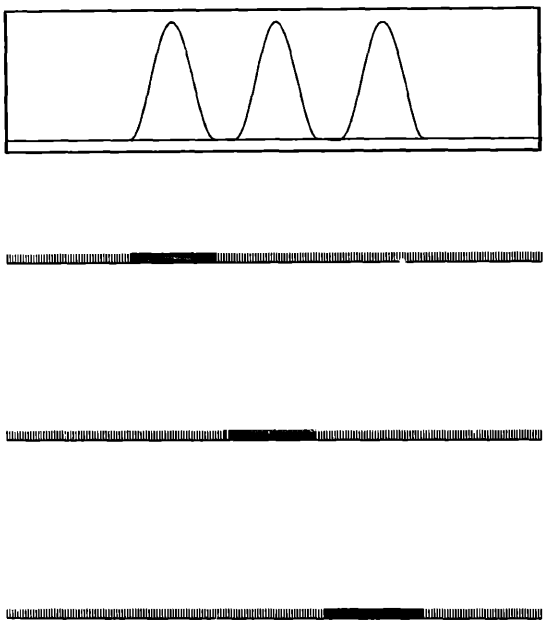

Fig. 1. One-dimensional pure advection of a cosine wave: solution obtained by the AIE SUPG scheme and the corresponding distribution of the implicit elements at $t = 0.0, 0.2, 0.4$.

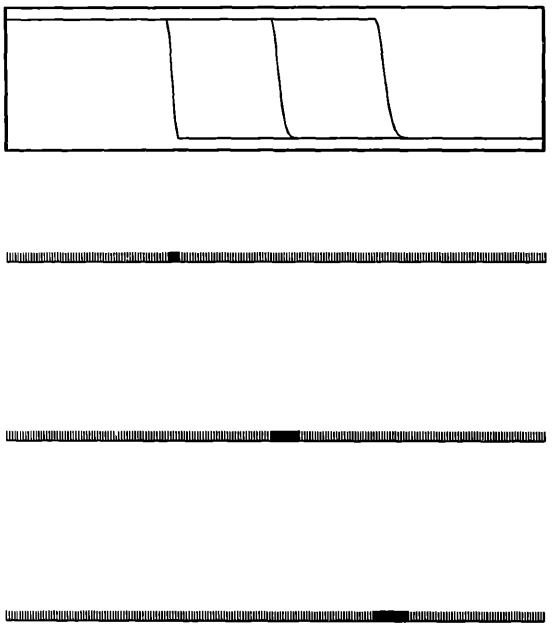

Fig. 2. One-dimensional pure advection of a discontinuity: solution obtained by the AIE/FCT scheme and the corresponding distribution of the FCT elements at $t = 0.0, 0.2, 0.4$.

One-dimensional pure advection of a discontinuity

The conditions in this case are nearly identical to those in the previous case. This time the profile which is being advected is a discontinuity which initially spans four elements. The AIE scheme is based on selection between the one-pass explicit SUPG method and the SUPG/FCT method implemented as described in Section 3. The threshold value for the parameter given by (3.10) is 10^{-7}. Figure 2 shows the solution and the corresponding distribution of the FCT elements, at time steps 0, 50 and 100. Compared to the full FCT approach the savings in the CPU time is 45%.

Two-dimensional pure advection of a plateau

This test case is for evaluating the performance of the AIE approach for two-dimensional problems with moving sharp fronts. A 40×40 mesh is chosen in a 1.0×1.0 computational domain. All boundary conditions are Dirichlet type. The advection velocity is in the diagonal direction and has unit magnitude. The time step is 0.01.

The first AIE scheme is based on selection between the explicit and discontinuity-capturing [13]/implicit versions of the SUPG procedure. The threshold value for the parameter given by (3.10) is 10^{-4}. Figure 3 shows the solution and the distribution of the discontinuity-capturing/implicit elements at time steps 0, 16 and 32. Compared to the implicit method, the AIE scheme results in a reduction of 45% in the CPU time and 69% in the memory needed for the global matrix.

The second AIE scheme is based on selection between the one-pass explicit and FCT

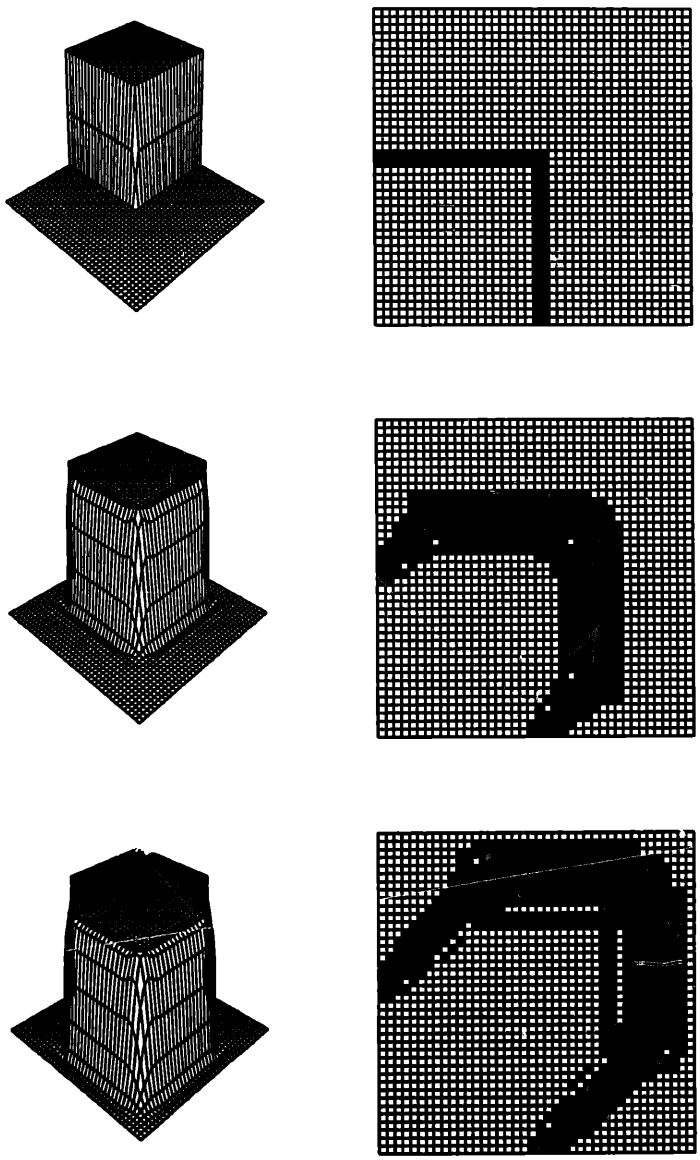

Fig. 3. Two-dimensional pure advection of a plateau: solution obtained by the AIE SUPG/discontinuity-capturing scheme and the corresponding distribution of the discontinuity-capturing/implicit elements at $t = 0.0, 0.24, 0.48$.

versions of the SUPG procedure. The threshold value for the parameter given by (3.10) is 10^{-4}. Figure 4 shows the solution and the distribution of the FCT elements at time steps 0, 16 and 32. Compared to full FCT approach the AIE scheme results in a 35% reduction in CPU time.

Flow past a circular cylinder at Reynolds number 100

In this problem we use a mesh (see Fig. 5) with 1940 elements and 2037 nodal points. The dimensions of the computational domain, normalized by the cylinder diameter, are 16 and 8 in the flow and the cross flow directions, respectively. The free-stream velocity is 0.125 and the initial value of the vorticity is zero everywhere in the domain. The time step is 1.0.

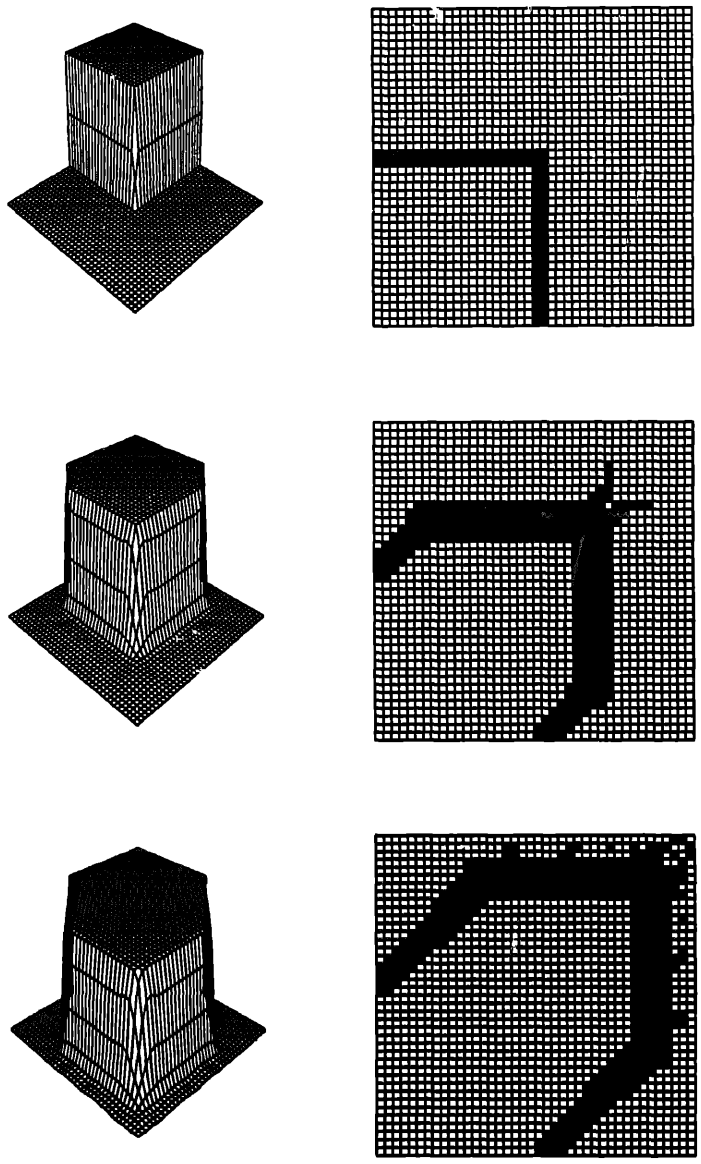

Fig. 4. Two-dimensional pure advection of a plateau: solution obtained by the AIE SUPG/FCT scheme and the corresponding distribution of the FCT elements at $t = 0.0, 0.24, 0.48$.

The AIE scheme employed to solve the vorticity transport equation and the Poisson equation is exactly as described in Section 3. For the vorticity transport equation, both the implicit and the explicit methods are SUPG based and involve three iterations per time step. In this problem we estimate the maximum element Courant number to be somewhere around 1.5–1.6; therefore the stability criterion given by (3.7) also becomes active in the implicit–explicit grouping. In fact our numerical experiments show that the computations do not converge when the entire set of elements belong to the explicit group. The threshold value of the parameter given by (3.10) is 10^{-5}. For the solution of the discrete Poisson equation, the iterations continue until the residual norm falls below 10^{-6} or its ratio to the initial residual norm reaches 10^{-4}.

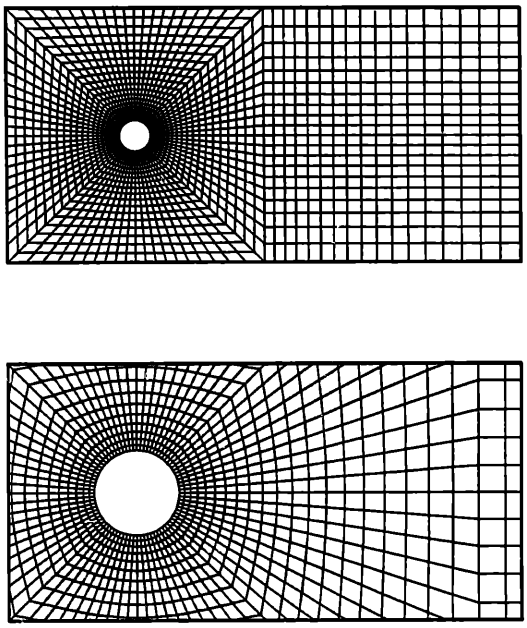

Fig. 5. Flow past a circular cylinder at Reynolds number 100: the finite element mesh (1940 elements and 2037 nodes).

The AIE method gives the expected solution for this problem: initially a symmetric flow pattern with two attached eddies growing in the wake of the cylinder, then the symmetry breaks, and as time goes by the vortices are formed alternately at the upper and lower downstream vicinity of the cylinder and carried along the Karman vortex street. Figures 6–8 show the solution obtained by the AIE scheme and the corresponding distributions of the implicit elements at time steps 800, 1200 and 1600.

The AIE method reduces the memory needed for the global matrices by 44% and the CPU time by 7%.

5. Concluding remarks

In this paper we have presented the AIE technique for large equation systems emanating from the finite element formulation of fluid mechanics problems.

In the adaptive implicit–explicit (AIE) method the elements are dynamically grouped into implicit and explicit subsets based on the element level stability and accuracy criteria. This dynamic grouping idea can also be applied in any context where there are some reasons for preferring a given procedure over another one. As an example we implemented the AIE concept also in conjunction with the flux-corrected transport method.

We applied this method to various test problems and demonstrated that substantial savings in the CPU time and memory can be achieved.

In our future studies we plan to experiment with the combination of the AIE approach and the element-by-element (EBE) preconditioned iterative technique. For example we can use

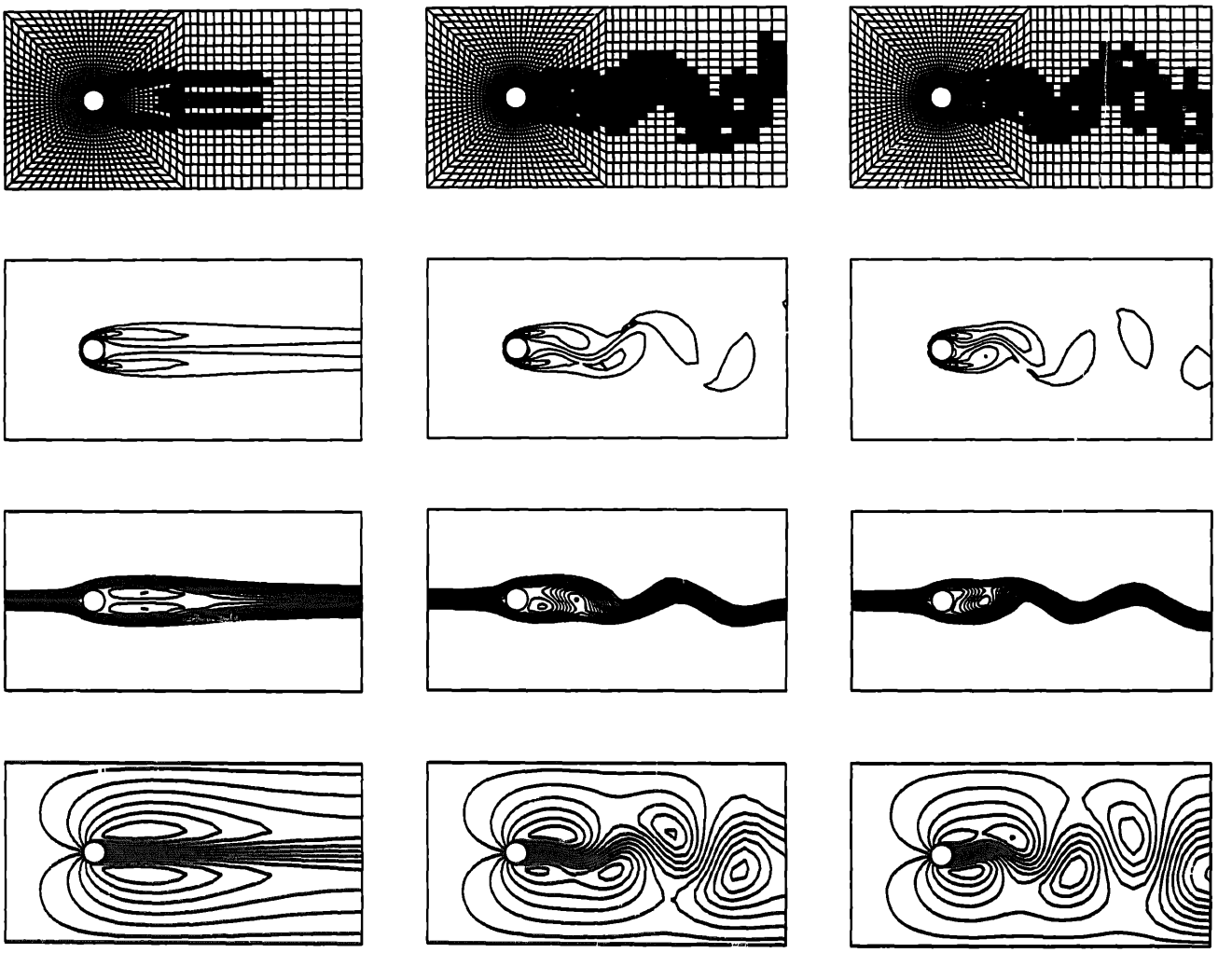

Fig. 6. Flow past a circular cylinder at Reynolds number 100: solution obtained by the AIE SUPG scheme at $t = 800$; from top to bottom: distribution of the implicit elements, vorticity, streamlines and relative streamlines.

Fig. 7. Flow past a circular cylinder at Reynolds number 100: solution obtained by the AIE SUPG scheme at $t = 1200$; from top to bottom: distribution of the implicit elements, vorticity, streamlines, and relative streamlines.

Fig. 8. Flow past a circular cylinder at Reynolds number 100: solution obtained by the AIE SUPG scheme at $t = 1600$; from top to bottom: distribution of the implicit elements, vorticity, streamlines and relative streamlines.

the EBE scheme to solve the discrete Poisson equation for the stream function, employ the AIE scheme to solve the time-dependent transport equation for the vorticity, but use, again, the EBE scheme at the implicit zones of the implicit–explicit distribution.

References

[1] T.J.R. Hughes and W.K. Liu, Implicit–explicit finite elements in transient analysis: Stability theory, J. Appl. Mech. 45 (1978) 371–374.

[2] T.J.R. Hughes and W.K. Liu, Implicit–explicit finite elements in transient analysis: Implementation and numerical examples, J. Appl. Mech. 45 (1978) 375–278.

[3] T.E. Tezduyar and J. Liou, Element-by-element and implicit–explicit finite element formulations for computational fluid dynamics, in: R. Glowinski, G.H. Golub, G.A. Meurant and J. Periaux, eds., First Internat. Symp. Domain Decomposition Methods for Partial Differential Equations, (SIAM, Philadelphia, PA, 1988) 281–300.

[4] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.

[5] R. Lohner, K. Morgan, J. Peraire and M. Vahdati, Finite element flux-corrected transport (FEM-FCT) for the Euler and Navier–Stokes Equations, in: R.H. Gallagher, R. Glowinski, P.M. Gresho, J.T. Oden and O.C. Zienkiewicz, eds., Finite Elements in Fluids 7 (Wiley, New York, 1987) 105–121.

[6] T.E. Tezduyar, R. Glowinski and J. Liou, Petrov–Galerkin methods on multiply-connected domains for the vorticity–stream function formulation of the incompressible Navier–Stokes equations, Internat. J. Numer. Methods Fluids 8 (1988) 1269–1290.

[7] T.E. Tezduyar and D.K. Ganjoo, Petrov–Galerkin formulations with weighting functions dependent upon spatial and temporal discretization: Application to transient convection–diffusion problems, Comput. Methods Appl. Mech. Engrg. 59 (1986) 47–71.

[8] T.E. Tezduyar and T.J.R. Hughes, Development of time-accurate finite element techniques for first-order hyperbolic systems with particular emphasis on the compressible Euler equations, Report prepared for NASA-Ames University Consortium Interchange No. NCA2-OR745-104, May 1982.

[9] T.E. Tezduyar and T.J.R. Hughes, Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations, Proc. AIAA 21st Aerospace Sciences Meeting, AIAA Paper 83-0125 (Reno, Nevada, January 1983).

[10] T.J.R. Hughes and T.E. Tezduyar, Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations, Comput. Methods Appl. Mech. Engrg. 45 (1984) 217–284.

[11] R. Glowinski, Numerical Methods for Nonlinear Variational Problems (Springer, New York, 1984).

[12] D.K. Ganjoo, W.D. Goodrich and T.E. Tezduyar, A new formulation for numerical simulation of electrophoresis separation processes, Comput. Methods Appl. Mech. Engrg. 75 (1989) 515–530.

[13] T.E. Tezduyar and Y.J. Park, Discontinuity-capturing finite element formulations for nonlinear convection–diffusion–reaction equations, Comput. Methods Appl. Mech. Eng. 59 (1986) 307–325.