INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 19, 259–268 (1994)

A PARTIALLY IMPLICIT METHOD FOR SIMULATING VISCOUS AEROFOIL FLOWS

K. J. BADCOCK

Aerospace Engineering Department, University of Glasgow, Glasgow G12 8QQ, U.K.

SUMMARY

A partially implicit method for the unsteady compressible Navier–Stokes equations is developed. The method is based on an explicit treatment of streamwise fluxes and an implicit treatment of normal fluxes. This leads to a linear system which is generated by an efficient finite difference procedure and which is block pentadiagonal. The method is tested on a shock-induced oscillatory flow over an aerofoil. Parallel implementations of an explicit, fully implicit and partially implicit method are investigated.

KEY WORDS Navier–Stokes Parallel Partially implicit

1. INTRODUCTION

An important part of the aircraft design process is the demonstration of aeroelastic stability under flight conditions. Computational tools have an important role to play owing to the expense of wind tunnel programmes for aeroelastic studies.[1] For freestream Mach numbers which are outside the transonic range, linear aerodynamics can provide reasonable predictions of the flutter boundary. However, for transonic cases there are significant non-linear effects which in general can only be satisfactorally described by the Euler or Navier–Stokes equations.

Implicit methods are generally preferred for viscous flow simulations owing to the time step restrictions for explicit methods introduced by the small mesh spacings needed to resolve boundary layers. The most popular method for solving the unsteady compressible Navier–Stokes equations was introduced by Beam and Warming[2] in 1978 and is based on an alternating direction implicit (ADI) approximate factorisation. Published applications include moving aerofoils, shock-induced oscillations (SIOs) and aeroelastic studies. The application of explicit methods to unsteady viscous applications has been limited. One example is the multiple-grid/Runge–Kutta method used in Reference 3 to study the SIO problem and high-angle-of-attack flows.

Parallel computers are becoming increasingly important in computational fluid dynamics. The potential of multiple high-performance processors has provided opportunities for tackling problems which have hitherto been intractable for serial computers. For a code to fully exploit the theoretical performance of a parallel machine, the algorithm must divide up the work evenly amongst the processors while minimizing communication and the idleness which results when one processor waits for results from another.

In the present paper we shall examine various time-stepping strategies for solving flows over aerofoils from the point of view of both efficiency of the algorithm and parallel implementation. The methods that are studied are the commonly used implicit and explicit methods and a hybrid

CCC 0271–2091/94/150259–10

Received July 1993
Revised February 1994

method which treats some terms explicitly and some implicitly. The details of the methods are given in the next section.

The division of the work amongst the processors is particularly simple for the aerofoil problem considered owing to the small number of processors available and the simple geometry and structured mesh used. However, flows over wings are of practical interest and the aerofoil problem represents a suitable first model problem for its three- dimensional counterpart. With the advent of software such as PVM it is now possible to run codes in parallel on workstations. Manufacturers are bringing out machines that use a small number (around four or eight) of processors of the type currently used in high-performance workstations. Examples include the IBM SP1 and the DEC Alpha farm. This form of processing is attractive owing to the flexibility of such a system and the high performance that can be attained at a small cost relative to some forms of supercomputing. There are also machines such as the Intel iPSC i860 Hypercube used in the present work, which feature a small number (around 64) of fast processors. Hence the simple mapping used in this work might not be appropriate for massively parallel systems but it is practical for this increasingly important form of parallel processing.

We shall therefore concentrate on the communication aspects of the parallel implementation. Explicit methods typically require a small amount of communication. The need to solve a linear system at each step of an implicit method results in increased communication and the inherently sequential nature of back substitution limits the efficiency of the parallel implementation. Algorithms can be designed to have good parallel properties at the expense of performance on a single processor. An example of this is the use of a special preconditioning technique for a conjugate gradient solver in Reference 4, which results in slower convergence than some other techniques on a serial computer but which has a far higher parallel efficiency. If enough processors are available to compensate for the poorer serial performance, then the good parallel properties allow a rapid solution. In this paper we seek to exploit this philosophy by combining the best features of explicit and implicit methods to provide an algorithm which requires the same communications as an explicit method but retains some of the advantages of an implicit method.

The rest of this paper is organized as follows. In the next section the various options for the time discretization are discussed and the model and spatial discretization are described. An efficient method for the calculation of the Jacobians of the flux approximations is formulated. The unsteady aerofoil test problem is briefly discussed and results to demonstrate the relative efficiencies of the three approaches are given.

2. A PARTIALLY IMPLICIT METHOD

In this section we develop a method for solving the unsteady, thin layer, Navier–Stokes equations given by

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \frac{\partial \mathbf{S}}{\partial y}, \tag{1}$$

where

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \varepsilon \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(\varepsilon + p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho u \\ \rho uv \\ \rho v^2 + p \\ v(\varepsilon + p) \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 0 \\ \sigma_{xy} \\ \sigma_{yy} \\ u\sigma_{xy} + v\sigma_{yy} - q_y \end{bmatrix}.$$

Here

$$\sigma_{yy} = 2\mu v_y - \tfrac{2}{3}\mu(u_x + v_y), \qquad \sigma_{xy} = \sigma_{yx} = \mu(u_y + v_x),$$

$$q_y = -\kappa\, \partial T/\partial y, \qquad p = (\gamma - 1)[\varepsilon - \tfrac{1}{2}\rho(u^2 + v^2)], \qquad T = c_S[\varepsilon/\rho - \tfrac{1}{2}(u^2 + v^2)].$$

The symbols $\rho, u, v, \varepsilon, p, \mu, \kappa$ and T represent the fluid density, the two components of velocity, energy, pressure, viscosity, heat conductivity and temperature respectively. The constants γ and c_S stand for the ratio of the specific heats and the specific heat at constant volume respectively. The viscosity is assumed to vary with temperature by Sutherland's law. The Baldwin–Lomax model is used to provide a contribution to the viscosity from turbulence. The terms arising from the turbulence model are all treated explicitly in the present work and this was found not to influence the stability of the method.

We anticipate that shock waves will be present in the flow and so the spatial discretization we use is Osher's flux function with a MUSCL-type interpolation and a limiter to prevent spurious oscillations. The differentiability of Osher's flux function is a property which is important for the implicit formulation given below. The other main feature of the flows that we wish to simulate is the presence of a boundary layer. This requires a fine mesh in one direction which introduces severe stability limitations on the time step. It is this problem that is addressed in this section.

In the following we shall adopt the notation

$$\partial F/\partial x \approx R_x, \qquad \partial(G - S)/\partial y \approx R_y.$$

Here R_x and R_y are obtained from the Osher scheme approximation to the convection terms and a central difference approximation to the viscous fluxes.

The first option for the time discretization is to use an explicit method of the form

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^n}{\Delta t} = -\mathbf{R}_x(\mathbf{q}^n) - \mathbf{R}_y(\mathbf{q}^n). \tag{2}$$

An example shown below will illustrate that the stability limitations on Δt can be several orders of magnitude smaller than the value required for accuracy reasons. This leads to a very inefficient method, although time-accurate multiple-grid methods are being developed[3,5] which allow this problem to be partly overcome. The explicit method has the advantage that it requires very little memory when compared with the implicit method below. The second possibility is to use a fully implicit method of the form

$$\left(\frac{\mathbf{I}}{\Delta t} + \frac{\partial \mathbf{R}_x}{\partial \mathbf{q}} + \frac{\partial \mathbf{R}_y}{\partial \mathbf{q}}\right)(\mathbf{q}^{n+1} - \mathbf{q}^n) = -\mathbf{R}_x(\mathbf{q}^n) - \mathbf{R}_y(\mathbf{q}^n). \tag{3}$$

The time step can now be chosen on the basis of accuracy rather than stability. However, there are several major drawbacks, including the large memory requirement to store the matrix on the left-hand side of (3) and the difficulty of adapting the method to run efficiently on a parallel computer as discussed in Section 1.

We shall therefore consider a partially implicit scheme. This method removes the stability limitation arising from the fine mesh required to refine the boundary layer around an aerofoil. In the following the derivation of the method is made in Cartesian co-ordinates for simplicity. However, implementation has been made in curvilinear co-ordinates with the implicit and explicit

directions corresponding to curvilinear directions. If a solid boundary lies on the plane $x = \text{const.}$, the equation for the updates can be written as

$$\left(\frac{\mathbf{I}}{\Delta t} + \frac{\partial \mathbf{R}_y}{\partial \mathbf{q}}\right)(\mathbf{q}^{n+1} - \mathbf{q}^n) = -\mathbf{R}_x(\mathbf{q}^n) - \mathbf{R}_y(\mathbf{q}^n). \tag{4}$$

The matrix on the left-hand side of (4) decouples into a block pentadiagonal system for each y-line in the mesh and therefore the solution of the linear system is much simpler than for the fully implicit method. The calculation of the Jacobians is discussed in the next section.

The cost of each partially implicit step is roughly three times more expensive than the corresponding explicit method and is relatively cheap compared with the fully implicit method. In addition, the method is very parallelizable, since it has the same communications as the explicit method if the mapping on to processors is done by mesh lines in the y- direction. A further benefit is that since the linear system can be solved one y-line at a time, the storage requirements are little more than those of the explicit method. For the present spatial formulation the storage required relative to the explicit method for the flow variables and the matrix is $1 + 40/j_n$ for the partially implicit method and 61 for the fully implicit method, where j_n is the number of grid points normal to the aerofoil.

On the debit side there is still the stability restriction arising from the x-direction, but this is usually much less restrictive than the limit arising from the fine mesh used to resolve a boundary layer. Also, it should be noted that the uncoupling of pentadiagonal systems for each mesh line is lost if the full Navier–Stokes equations are required. Fortunately, the thin layer equations provide a good description for many flows of interest.

3. EFFICIENT CALCULATION OF THE JACOBEAN MATRIX

The use of a finite difference approximation to the Jacobian of the discretization is attractive because of the simplicity of the calculation and implementation. However, the process is computationally intensive and so efficient methods must be used if the resulting computer code is not to be quick to develop but too slow to use.

In Reference 6 the problem of computing a finite difference approximation to a sparse Jacobian is addressed. To fix ideas, assume that the function whose Jacobian $\mathbf{A}$ we wish to approximate is denoted by $\mathbf{R}$. The following algorithm can be used to minimize the number of function evaluations required.

1. Select a set of the indices S of the components of $\mathbf{q}$ such that $\mathbf{A}_{i,j} = 0\ \forall i, j \in S, i \neq j$.
2. Perturb $q_i\ \forall i \in S$.
3. Evaluate $\mathbf{R}$ for the perturbed argument and evaluate the finite difference approximation to the terms $\mathbf{A}_{i,j}$ for $1 < i < n$ and $\forall j \in S$, where n is the dimension of $\mathbf{q}$.
4. Repeat steps 2 and 3 for a different S until all the components of $\mathbf{A}$ have been evaluated.

Note that in each cell there are four unknowns, only one of which may be pertrbed for each function evaluation. The total number of evaluations required is $3 \times 3 \times 4$ for the first-order method and $5 \times 5 \times 4$ for the second-order method. It can be seen that the evaluation is indeed expensive. This method considers the cell-based residuals and was used in Reference 7.

However, a more efficient procedure arises from considering the residual flux-by-flux. We shall consider the numerical fluxes $\mathbf{G}_{i,j+1/2}$ and $\mathbf{S}_{i,j+1/2}$. The method for $\mathbf{F}_{i+1/2,j}$ is similar to that used for $\mathbf{G}_{i,j+1/2}$.

Table I. Comparison of CPU times in seconds for the residual-based and flux-based methods for calculating the Jacobian on different meshes

Mesh	Residual-based	Flux-based	Speed-up
64 × 16	33·4	3·3	10·1
128 × 32	137·6	15·5	8·8

The viscous flux $\mathbf{S}_{i,j+1/2}$ depends on the values in cells (i,j) and $(i,j+1)$. It is fairly straightforward to calculate and encode the contributions to the Jacobian owing to the relatively simple central difference approximations used.

For the convective terms the complicated nature of the numerical fluxes arising from Osher's scheme makes analytical expressions unattractive. If MUSCL interpolation is used, then the flux $\mathbf{G}_{i,j+1/2}$ depends on values in the cells $(i,j-1)$, (i,j), $(i,j+1)$ and $(i,j+2)$. We can denote this by

$$\mathbf{G}_{i,j+1/2} = \mathbf{G}_{i,j+1/2}(\mathbf{q}_{i,j-1}, \mathbf{q}_{i,j}, \mathbf{q}_{i,j+1}, \mathbf{q}_{i,j+2}), \tag{5}$$

which in turn can be rewritten as

$$\Delta_{i,j+1/2} = \mathbf{G}_{i,j+1/2}(\mathbf{q}_L, \mathbf{q}_R), \tag{6}$$

where

$$\mathbf{q}_L = \mathbf{q}_L(\mathbf{q}_{i,j-1}\mathbf{q}_{i,j}, \mathbf{q}_{i,j+1}), \qquad \mathbf{q}_R = \mathbf{q}_R(\mathbf{q}_{i,j}\mathbf{q}_{i,j+1}, \mathbf{q}_{i,j+2}).$$

To calculate the derivatives of $\mathbf{G}_{i,j+1/2}$, the chain rule is used to yield

$$\frac{\partial \mathbf{G}_{i,j+1/2}}{\mathbf{q}_{i,k}} = \frac{\partial \mathbf{G}_{i,j+1/2}}{\mathbf{q}_{\mathrm{L}}} \frac{\partial \mathbf{q}_{\mathrm{L}}}{\mathbf{q}_{i,k}} + \frac{\partial \mathbf{G}_{i,j+1/2}}{\mathbf{q}_{\mathrm{R}}} \frac{\partial \mathbf{q}_{\mathrm{R}}}{\mathbf{q}_{i,k}}. \tag{7}$$

The derivatives of the MUSCL interpolation can be evaluated analytically and a symbolic algebra package called REDUCE has been used to quickly generate efficient code. The calculation of the terms in $\delta\mathbf{G}_{i,j+1/2}/\delta\mathbf{q}_L$ requires four evaluations of $\mathbf{G}_{i,j+1/2}$ and similarly for $\delta\mathbf{G}_{i,j+1}/\delta\mathbf{q}_R$, leading to eight evaluations in total.

The derivatives for each flux can be added into the Jacobian as they are calculated. The flux-based method requires an equivalent of eight function evaluations instead of the 100 evaluations required for the cell-based method. This neglects the computation of the MUSCL derivatives, which turns out to be fast compared with one function evaluation. In theory the new method should be over 12 times faster. The actual comparison for two mesh sizes is given in Table I. The theoretical speed-up is not observed owing to the cost of additional multiplications in composing the chain rule and searching for the correct place in the matrix to place the derivative approximations. However, the speed-up is still by a factor of around 10.

4. RESULTS

The test problem considered is turbulent flow over an 18% thick circular arc aerofoil at zero incidence, a Reynolds number of 11×10^6 and a freestream Mach number of 0.771. The mesh used has 71 points around the aerofoil and 31 points normal. The far field is located at 10

Table II. Frequencies for the SIO problem computed by various methods

Method	Reference	Frequency
1-D-Imp	Present	0·41
AF-CGS	8	0·41
MacCormack	9	0·40
MG/explicit	3	0·40
Beam–Warming	10	0·41
Experiment	11	0·49

chords. For these conditions the shock waves on the upper and lower surfaces of the aerofoil oscillate out of phase. The period of the flow is roughly eight non-dimensional time units.

A comparison of computed and experimentally determined frequencies that are available in the literature for this case is shown in Table II. The present results and those in Reference 8 compare favourably with published experimental and computational results based on central differencing with artificial dissipation. A fuller investigation and verification of the solution is made in Reference 8.

The degree of approximation in the partially implicit method is less than that involved in the fully implicit method, since no linearization is required for the streamwise fluxes. Stability restrictions also mean that small time steps must be taken relative to those which can be used for the fully implicit method. It is therefore not surprising that the solution obtained by the partially implicit method is close to the solution obtained on the same grid by the fully implicit method which has converged with respect to the time step. This is illustrated in Figure 1, where one cycle of the oscillation in the lift coefficient is shown. The only difference between the results is for the fully implicit case with 200 steps per cycle. This indicates that good time resolution can be obtained by the fully implicit method with 400 steps per cycle. The partially implicit method requires 40,000 steps per cycle because of stability restrictions. Around 80×10^6 steps would be required for the explicit method and this case was not tried owing to the excessive computer time which it would have involved. It is clear from these results that the time step for the partially implicit method is determined by stability rather than accuracy.

The increase in the allowable time step for the implicit method over the partially implicit method is by two orders of magnitude, resulting in a substantial increase in efficiency even though one fully implicit step takes about 2.5 times longer than one partially implicit step. However, the fully implicit method requires over 60 times the memory for this case. This can prove very restrictive even for medium-size aerofoil problems and is a major problem for three-dimensional flows. Discussion of possible applications where the partially implicit method might prove to be more efficient in terms of run time than the fully implicit method is postponed until Section 5. We continue this section with an examination of the algorithms from the point of view of communication times to evaluate their parallel performance.

A parallel version of the code implementing the partially implicit method was written for the Intel Hypercube at the SERC Daresbury Laboratory. The ease of parallelization is one attractive feature of the algorithm. The mapping from the computational mesh to the processors is done by lines and is shown for an aerofoil flow in Figure 2. The fluxes in the streamwse direction are calculated first and the values at the boundaries between processors are communicated at this stage. After these communications and the communication of the global time step the calculation on each processor can proceed independently.

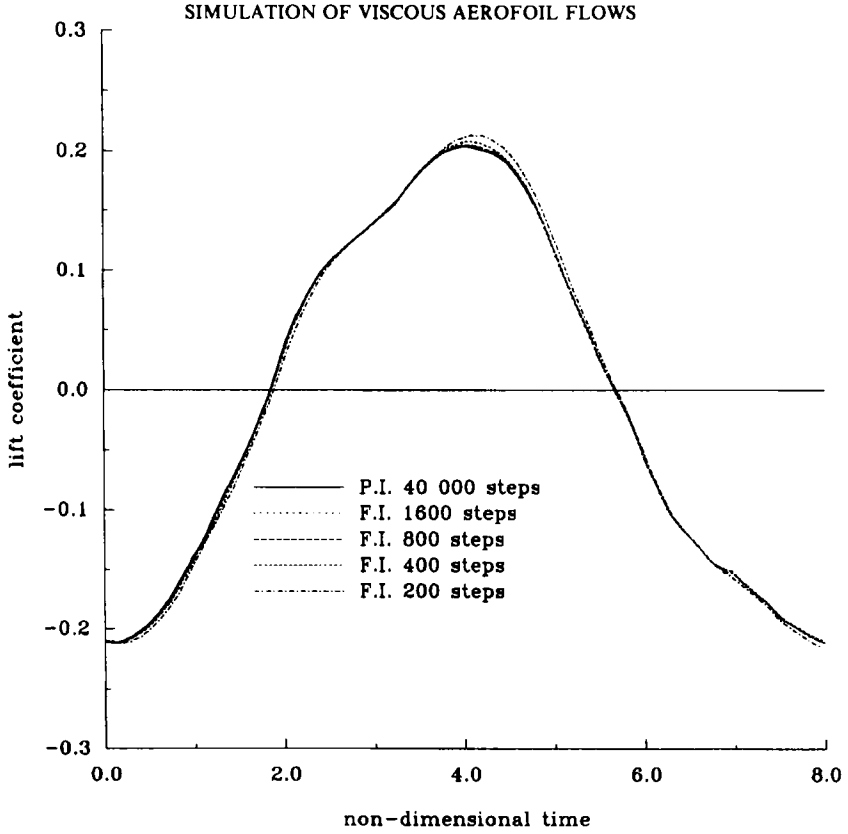

Figure 1. Lift coefficient versus non-dimensional time for one period of the flow over an 18% circular arc aerofoil at zero incidence, a Reynolds number of 11×10^6 and a freestream Mach number of 0.771. The results shown correspond to 200, 400, 800 and 1600 steps per cycle for the fully implicit (F.I.) method and 40,000 steps per cycle for the partially implicit (P.I.) method

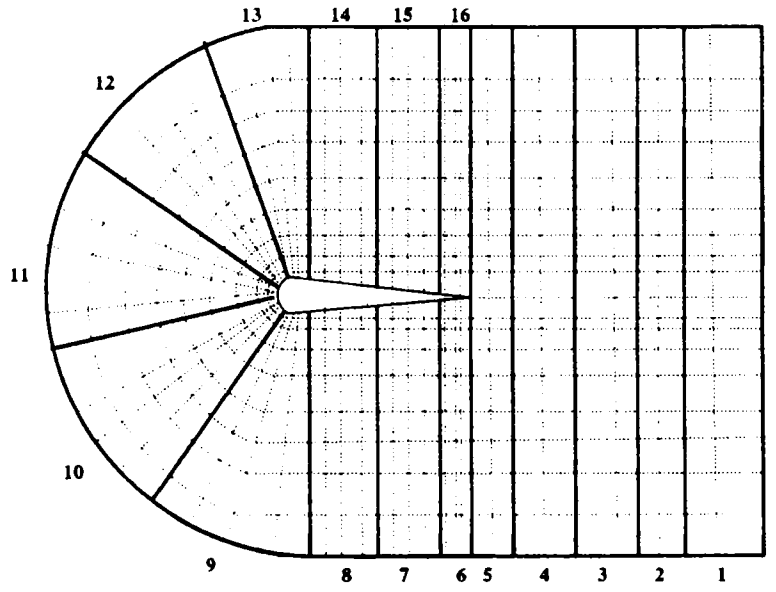

Figure 2. Mapping from computational space to the processors for the aerofoil problem for 16 nodes.

Table III. Parallel efficiencies defined as $100CPU_1/(n \times CPU_n)$, where n is the number of processors, CPU_n is the CPU time taken on n processors and CPU_1 is the time taken on one processor

Processors	Explicit method	Partially implicit method	Fully implicit method[12]
1	100	100	100
4	90	97	76
8	79	95	70
16	63	90	59

Table IV. Computation times t_{calc} and communication times t_{comm} in milliseconds and their ratio for the updating phase of the calculations for the explicit, partially implicit and fully implicit methods

	Explicit method			Partially implicit method			Fully implicit method
Processors	t_{calc}	t_{comm}	Ratio	t_{calc}	t_{comm}	Ratio	Ratio
1	1810	0	—	10140	0	—	—
4	464	6	83	2560	8	311	8
8	241	10	24	1290	11	117	4
16	129	10	13	659	10	67	2

The parallel efficiencies are shown in Table III. The relative communication to computation is examined in Table IV for the three methods as the number of nodes is increased. The fact that the partially implicit method involves more work than the explicit method on each processor at each step but the communications are the same explains the improved efficiency of the partially implicit method over the explicit method. The results for the fully implicit method of Reference 12 are included for reference and the reader is referred therein for details of the method and its parallel implementation. It is clear that the efficiency of the fully implicit method is dropping more rapidly than that of the partially implicit method as the number of nodes is increased. This is due to the large amount of communication required by the implicit method.

5. CONCLUSIONS

In this paper a partially implicit method for the unsteady compressible Navier–Stokes equations was proposed. This method uses implicit approximations normal to solid boundaries in order to remove stability restrictions which arise from the fine meshes needed to resolve boundary layers. The linear system decouples to a block pentadiagonal system for each row of the mesh. An efficient way of calculating the finite difference approximations to generate the linear system was given. The decoupling of the linear system allowed an efficient parallelization of the algorithm which yielded high efficiencies when implemented on the Intel Hypercube. An investigation of the ratio of communication time to computation time for each method was presented and this showed that the partially implicit method was most efficient from this point of view.

For the aerofoil results presented above it was found that stability limits arising from the streamwise direction mean that the partially implicit method is much less efficient overall

than the fully implicit method despite the better parallel properties of the partially implicit method.

However, there are two possible cases where the partially implicit method could prove practical. First, the partially implicit method does have potential to improve the performance of codes that use explicit time stepping, such as the multiple-grid/Runge–Kutta algorithm used in Reference 3. For the results on a single grid the increase in efficiency resulting from the increased allowable time step is of the order of 1000. One of the main attractions of an explicit method is the low memory requirement and using the partially implicit method would not substantially increase the memory use.

Secondly, for some three-dimensional flows over wings a method which uses an implicit treatment for the streamwise and normal fluxes and an explicit treatment of the spanwise terms is under consideration and has considerable potential. The relatively coarse grid often used in the spanwise direction does not lead to restrictive stability limits in the same way that the streamwise terms did in the present paper. If this applies, then considerable improvements in efficiency and memory use are achieved over the fully implicit method. The reduced memory requirement in particular is very important for application to three-dimensional flows. In addition, this method has the potential to make very efficient use of parallel machines of the type used for the calculations of this paper.

ACKNOWLEDGEMENT

This work was carried out under SERC/MOD grant GR H 47371.

REFERENCES

1. G. P. Guruswamy, 'Unsteady aerodynamic and aeroelastic calculations for wings using Euler equations', *AIAA J.*, **28**; 461–469 (1990).
2. R. M. Beam and R. F. Warming, 'An implicit factored scheme for the compressible Navier–Stokes equations', *AIAA J.*, **16**; 393–402 (1978).
3. M. Gillan, 'A computational analysis of viscous flows over porous aerofoils,' *Ph.D. Thesis*, Aeronautical Engineering, Queen's University, Belfast, 1993.
4. X. Xu, N. Qin and B. E. Richards, 'α-GMRES: a new parallelizable iterative solver for large sparse non-symmetric linear systems arising from CFD', *Int. j. numer. methods fluids*, **15**; 613–623 (1992).
5. D. C. Jesperson, 'A time-accurate multiple-grid algorithm', *AIAA Tech. Rep. 85–1493*, 1985.
6. M. J. D. Powell, A. Curtis and J. K. Reid, 'On the estimation of sparse Jacobian matrices', *J. Inst. Math. Appl.*, **13**; 117–120 (1974).
7. N. Qin and B. E. Richards, 'Sparse quasi-Newton method for Navier–Stokes solution', *Notes Numer. Fluid Dyn.*, **29**; 474–483 (1990).
8. K. J. Badcock, 'An efficient unfactored implicit method for unsteady aerofoil flows', *GU Aero Tech. Rep. 9313*, 1993.
9. L. L. Levy, 'Experimental and computational steady and unsteady transonic flows about a thick airfoil', *AIAA J.*, **16**; 564–572 (1978).
10. J. L. Steger, 'Implicit finite-difference simulation of flow about arbitrary two-dimensional geometries', *AIAA J.*, **16**; 679–686 (1978).
11. L. L. Levy, J. B. McDevitt and G. S. Deiwert, 'Transonic flow about a thick circular-arc airfoil', *AIAA J.*, **14**; 606–613 (1976).
12. K. J. Badcock, 'AF-CGS-P: an unfactored method in parallel for turbulent pitching aerofoil flows', *GU Aero Tech. Rep. 9323*, 1993.