PERGAMON

Computers & Fluids 30 (2001) 137–159

computers & fluids

www.elsevier.com/locate/compfluid

An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids

Hong Luo[a,*], Joseph D. Baum[a], Rainald Löhner[b]

[a]*Science Applications International Corporation, Applied Physics Operation, 1710 Goodridge Drive, MS 2-6-9, McLean, VA 22102, USA*

[b]*Institute for Computational Sciences and Informatics, George Mason University, Fairfax, VA 22030,USA*

Received 1 November 1999; accepted 27 March 2000

Abstract

An accurate, fast, matrix-free implicit method has been developed to solve the three-dimensional compressible unsteady flows on unstructured grids. A nonlinear system of equations as a result of a fully implicit temporal discretization is solved at each time step using a pseudo-time marching approach. A newly developed fast, matrix-free implicit method is then used to obtain the steady-state solution to the pseudo-time system. The developed method is applied to compute a variety of unsteady flow problems involving moving boundaries. The numerical results obtained indicate that the use of the present implicit method leads to a significant increase in performance over its explicit counterpart, while maintaining a similar memory requirement. © 2000 Elsevier Science Ltd. All rights reserved.

1. Introduction

The use of unstructured meshes for computational fluid dynamics problems has become widespread due to their ability to discretize arbitrarily complex geometries and due to the ease of adaptation in enhancing the solution accuracy and efficiency through the use of adaptive refinement techniques.

* Corresponding author. Tel.: +1-703-676-5957; fax: +1-703-676-5323.
E-mail address: luo@apo.said.com (H. Luo).

In recent years, significant progress has been made in developing numerical algorithms for computing steady flows on unstructured grids. Usually, explicit temporal discretizations such as multi-stage Runge–Kutta schemes are used to drive the solution to steady-state. Acceleration techniques such as local time-stepping and implicit residual smoothing have also been combined in this context. In general, explicit schemes and their boundary conditions are easy to implement, vectorize and parallelize, and require only limited memory storage. However, for large-scale problems and especially for the solution of the Navier–Stokes equations, the rate of convergence slows down dramatically, resulting in inefficient solution techniques. In order to speed up convergence, a multigrid strategy or an implicit temporal discretization is required.

On the other hand, numerical algorithms for computing unsteady flows have lagged behind. Explicit methods, deemed to be too slow for obtaining steady solutions, may be the only choice for certain unsteady applications such as shock wave simulations, when the time scales of interest are small, or more precisely, when they are comparable to the spatial scales. However, when dealing with many low frequency phenomena such as wing flutter, periodic pitching of an airplane, and store separation from flight vehicles, explicit methods are extremely time-consuming, since the allowable time step is much more restrictive than that needed for an acceptable level of time accuracy. Therefore, it is desirable to develop a fully implicit method, where the time step is solely determined by the flow physics and is not limited by numerical stability consideration.

When an implicit scheme is used to compute unsteady flows, one has to drive the unsteady residual to zero (or at least to truncation error) at each time step. In the context of factored implicit schemes or linearized implicit schemes, this is usually done by employing inner iterations [1–3]. It is the role of these inner iterations to eliminate errors, if any, due to factorization and linearization, and sometimes also errors arising from employing a lower-order approximation on the implicit side. Alternatively, a pseudo-time marching can be used to drive the unsteady residual to zero. In this context, multigrid techniques are typically used to accelerate this pseudo-time stepping evolution to reduce the computational cost of each implicit time step [4–7].

The objective of the effort discussed in this paper is to develop an accurate, fast implicit method for solving unsteady compressible flow problems around 3D complex, realistic aerodynamic configurations on unstructured grids. Typically, hundreds of thousands of mesh points are necessary to represent such engineering type configurations accurately. Any implicit methods requiring the storage of the Jacobian matrix would be impractical, if not impossible to use to solve such large scale problems, where the storage requirement can easily exceed the memory limitation of present computers. In the present work, a nonlinear system of equations arising from a fully implicit temporal discretization of the unsteady Euler equations is solved at each time step using a pseudo-time marching approach. A newly developed fast, matrix-free implicit method [8], which has proven to be computationally efficient and robust for steady flows around complex geometries, has been extended to obtain the steady-state solution to the pseudo-time system. The developed method is applied to compute a variety of unsteady flow problems involving moving boundaries. The numerical results obtained demonstrate that the use of the present implicit method leads to a significant increase in performance over its explicit counterpart, while maintaining a similar memory requirement.

2. Governing equations

The unsteady compressible Euler equations for a moving control volume can be expressed in integral form as

$$\frac{\partial}{\partial t}\int_{V(t)} \mathbf{U}\,\mathrm{d}V + \int_{S(t)} (\mathbf{F} - \mathbf{U}\dot{\mathbf{x}})\cdot \mathbf{n}\,\mathrm{d}S = 0, \tag{2.1}$$

where $V(t)$ is the moving control volume, $S(t)$ its boundary, $\mathbf{n}$ the unit outward normal vector to the boundary, and $\dot{\mathbf{x}}$ the velocity of the moving boundary. The flow variable vector $\mathbf{U}$ and the inviscid flux vector $\mathbf{F}$ are defined by

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix}, \tag{2.2}$$

where the summation convention has been used and ρ, p, and e denote the density, pressure, and specific total energy of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. This set of equations is completed by the addition of the equation of state

$$p = (\gamma - 1)\rho\left(e - \frac{1}{2}u_j u_j\right), \tag{2.3}$$

which is valid for perfect gas, where γ is the ratio of the specific heats.

3. Numerical method

The governing equation (2.1) is discretized using a finite volume cell-vertex formulation, where the cell-averaged variables are stored at the vertices of the grid. The control volumes are non-overlapping dual cells constructed by the median planes of the tetrahedra. In the present study, the inviscid flux terms at the dual mesh cell interface are computed using the Advection Upwind Splitting Method (AUSM+) scheme [9]. A MUSCL [10] approach is used to achieve high-order accuracy. The Van Albada limiter based on primitive variables is used to suppress the spurious oscillation in the vicinity of the discontinuities.

This finite volume approximation yields the following semi-discrete system of nonlinear equations:

$$\frac{\partial(\mathbf{U}V)}{\partial t} + \mathbf{R} = 0, \tag{3.1}$$

where $\mathbf{U}$ is the solution vector, V is the volume of the dual mesh cell (equivalent to the lumped mass matrix in the finite element), and $\mathbf{R}$ is the residual vector approximating the second integral in Eq. (2.1).

An explicit time-accurate advance of Eq. (3.1) in time can be accomplished using a multi-stage Runge–Kutta scheme. However, when the maximum allowable time step imposed by an explicit stability requirement is much less than that imposed by the acceptable level of time accuracy, implicit schemes are to be preferred. A family of implicit schemes for Eq. (3.1) is

$$\frac{V^{n+1}\mathbf{U}^{n+1} - V^n\mathbf{U}^n}{\Delta t} + (1-\alpha)\mathbf{R}^{n+1} + \alpha\mathbf{R}^n = 0. \tag{3.2}$$

Here n denotes the time level. If $\alpha = 0$, the scheme is the backward Euler method. If $\alpha = 1/2$, the resulting scheme known as Crank–Nicholson method is second-order accurate in time. A more popular three-point backward-difference method approximation is not used in the present work because (a) it requires more memory, as it involves three levels, and (b) it has to reduce to a two-level scheme, when a remeshing occurs.

Eq. (3.2) represents a nonlinear system of coupled equations, that has to be solved at each time step. It can be solved by treating it as a steady-state problem by introducing a pseudo-time variable t_*

$$\frac{\partial V\mathbf{U}}{\partial t_*} + R_*(\mathbf{U}) = 0, \tag{3.3}$$

and 'time-marching' the solution using local pseudo-time steps Δt_*, until $\mathbf{U}$ converges to $\mathbf{U}^{n+1}$. Here $\mathbf{U}$ is the approximation to $\mathbf{U}^{n+1}$ and the unsteady residual $R_*(\mathbf{U})$ is defined as

$$R_*(\mathbf{U}) = \frac{V\mathbf{U}}{\Delta t} + (1-\alpha)R(\mathbf{U}) + S(\mathbf{U}^n, V^n), \tag{3.4}$$

with the source term

$$S(\mathbf{U}^n, V^n) = \alpha\mathbf{R}^n - \frac{V^n\mathbf{U}^n}{\Delta t}, \tag{3.5}$$

which remains fixed during the pseudo-time marching procedure.

In the literature, multigrid techniques are typically used to accelerate this pseudo-time stepping evolution to reduce the computational cost of each implicit time step [4–7]. They require good smoothers, as well as efficient intergrid transfer operators. Particularly the construction of good smoothers tends to be difficult for the highly stretched and distorted grids encountered in high Reynolds-number applications. Furthermore, for a production environment, the construction and handling of the multiple grid systems must be made automatic. It was felt that one-grid method would be preferable for the non-expert user community, due to its simplicity and 'black-box' character. The main contribution of this work is to show that a matrix-free implicit method, which has proven to be computationally efficient and robust for steady flows around complex geometries, can be extended and used to accelerate convergence to steady-state of this pseudo-time unsteady problem. Using backward Euler time-integration, Eq. (3.3) can be written as

$$V\frac{\Delta\mathbf{U}^m}{\Delta t_*} + R_*^{m+1} = 0, \tag{3.6}$$

where Δt_* is the pseudo-time increment, and $\Delta\mathbf{U}^m$ is the difference of unknown vector between pseudo-time levels m and $m+1$, i.e.,

$$\Delta\mathbf{U}^m = \mathbf{U}^{m+1} - \mathbf{U}^m. \tag{3.7}$$

Eq. (3.6) can be linearized in time by setting

$$\mathbf{R}_*^{m+1} = \mathbf{R}_*^m + \frac{\partial \mathbf{R}_*^m}{\partial \mathbf{U}}\Delta\mathbf{U}. \tag{3.8}$$

This yields the following system of linear equations

$$\left[\frac{V^{n+1}}{1-\alpha}\left(\frac{1}{\Delta t_*}+\frac{1}{\Delta t}\right)\mathbf{I}+\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}}\right]\Delta\mathbf{U}^m = -\mathbf{R}^m - \frac{V^{n+1}\mathbf{U}^m}{(1-\alpha)\Delta t} + \frac{1}{1-\alpha}\left(\frac{V^n\mathbf{U}^n}{\Delta t} - \alpha\mathbf{R}^n\right), \tag{3.9}$$

that needs to be solved at each time step during the pseudo-time advancement. Starting with $m=1$, $\mathbf{U}^1=\mathbf{U}^n$, the sequence of iterates $\mathbf{U}^m$, $m=1,2,3,\ldots$ converges to $\mathbf{U}^{n+1}$, when the right-hand-side unsteady residual $\mathbf{R}_*$ equals to zero. Note that when $\Delta t_* = \infty$ and one time step ($m=1$) is used to get the steady state solution to the pseudo-time system, the above scheme becomes the well-known linearized implicit scheme:

$$\left[\frac{V^{n+1}}{(1-\alpha)\Delta t}\mathbf{I}+\frac{\partial \mathbf{R}^n}{\partial \mathbf{U}}\right](\mathbf{U}^{n+1}-\mathbf{U}^n) = -\mathbf{R}^n - \frac{V^{n+1}\mathbf{U}^n}{(1-\alpha)\Delta t} + \frac{1}{1-\alpha}\left(\frac{V^n\mathbf{U}^n}{\Delta t} - \alpha\mathbf{R}^n\right). \tag{3.10}$$

Therefore, it is apparent that the fully implicit scheme is more accurate than its linearized counterpart, as the latter only uses one time step (one iteration) to drive the unsteady residual to zero, and the former can be thought of as using inner iterations to achieve this goal. It should be noted that the linearizations itself would not degrade the time accuracy, as the linearizations are carried out to the $O(\Delta t^2)$ terms. It is the approximations to the left-hand side of Eq. (3.10) that degrade the time accuracy. These approximations are usually used to improve the efficiency, reduce the memory requirements, and enhance the stability in the linearized implicit schemes.

The system of linear equations (3.9) is solved iteratively by a GMRES algorithm with a LU-SGS preconditioner [8]. The idea behind this recently developed GMRES + LU-SGS method is to combine the efficiency of the iterative methods and low memory requirement of approximate factorization methods in an effort to develop a fast, low storage implicit method. A crucial approximation in this approach is to use the following simplified flux function

$$\mathbf{R}_i = \sum_j \frac{1}{2}\left[\mathbf{F}(\mathbf{U}_i,\mathbf{n}_{ij}) + \mathbf{F}(\mathbf{U}_j,\mathbf{n}_{ij}) - |\lambda_{ij}|(\mathbf{U}_j-\mathbf{U}_i)\right]|\mathbf{s}_{ij}|, \tag{3.11}$$

where

$$|\lambda_{ij}| = |\mathbf{V}_{ij}\cdot\mathbf{n}_{ij}| + C_{ij}, \tag{3.12}$$

where $\mathbf{s}_{ij}$ is the area vector normal to the control-volume interface associated with the edge ij, $\mathbf{n}_{ij}=\mathbf{s}_{ij}/|\mathbf{s}_{ij}|$ its unit vector in the direction $\mathbf{s}_{ij}$, $\mathbf{V}_{ij}$ the velocity vector, C_{ij} the speed of sound, and the summation is over all neighboring vertices j of vertex i to derive the left-hand-side

Jacobian matrix. In addition, only a first-order representation of the numerical fluxes is linearized in order to reduce the number of non-zero entries in the matrix and to simplify the linearization process. This results in the graph of the sparse matrix $\partial \boldsymbol{R}/\partial \boldsymbol{U}$ being identical to the graph of the supporting unstructured mesh. The linearization of flux function (3.11) yields

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_i} = \sum_j \frac{1}{2}\left[J(\mathbf{U}_i, \mathbf{n}_{ij}) + |\lambda_{ij}|\mathbf{I}\right]|\mathbf{s}_{ij}| \tag{3.13}$$

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{U}_j} = \sum_j \frac{1}{2}\left[J(\mathbf{U}_j, \mathbf{n}_{ij}) - |\lambda_{ij}|\mathbf{I}\right]|\mathbf{s}_{ij}| \tag{3.14}$$

where $J = \partial F/\partial U$ represents the Jacobian of the flux vector. Using an edge-based data structure, the left-hand-side matrix in Eq. (3.9) is stored in upper, lower, and diagonal forms, which can be expressed as

$$U_{ij} = \frac{1}{2}\left[J(\mathbf{U}_j, \mathbf{n}_{ij}) - |\lambda_{ij}|\mathbf{I}\right]|\mathbf{s}_{ij}|, \tag{3.15}$$

$$L_{ij} = \frac{1}{2}\left[-J(\mathbf{U}_i, \mathbf{n}_{ij}) - |\lambda_{ij}|\mathbf{I}\right]|\mathbf{s}_{ij}|, \tag{3.16}$$

$$D_{ii} = \frac{V}{1-\alpha}\left(\frac{1}{\Delta t_*} + \frac{1}{\Delta t}\right)\mathbf{I} + \sum_j \frac{1}{2}\left[J(\mathbf{U}_i, \mathbf{n}_{ij}) + |\lambda_{ij}|\mathbf{I}\right]|\mathbf{s}_{ij}|. \tag{3.17}$$

Note that U, L, and D represent strict upper, strict lower, and diagonal matrices, respectively. Both upper and lower matrices require a storage of *nedge* $\times$ *neqns* $\times$ *neqns* and the diagonal matrix needs a storage of *npoin* $\times$ *neqns* $\times$ *neqns*, where *npoin* is the number of grid points, *neqns* ($=5$ in 3D) the number of unknown variables, and *nedge* the number of edges. Note that in 3D, *nedge* ≈ 7*npoin*. Clearly, the upper and lower matrices consume substantial amounts of memory, taking 93% of the storage required for the left-hand-side Jacobian matrix. When GMRES is used to solve the linear system (3.9), all it needs is the matrix–vector products, not the left-hand-side matrix itself explicitly. Such matrix–vector products can be approximated by a finite difference as:

$$J\Delta\mathbf{U} \approx \Delta\mathbf{F} = \mathbf{F}(\mathbf{U} + \Delta\mathbf{U}) - \mathbf{F}(\mathbf{U}). \tag{3.18}$$

to eliminate the storage of the upper and lower matrices. This idea of the matrix-free approach, in which the product of Jacobian matrix and incremental vector is approximated by the increment of the flux vector, was first introduced in the work of Men'shov and Nakamura [11] on structured meshes, and later extended to unstructured meshes by Sharov and Nakahashi [12]. Note that the evaluation of the flux vector is not an expensive operation, as it only involves convective fluxes, not the full right-hand-side residual.

It is well known that the speed of convergence of an iterative algorithm for a linear system depends on the condition number of the left-hand-side matrix. GMRES works best when the eigenvalues of the matrix are clustered. The easiest and the most common way to improve the efficiency and robustness of GMRES is to use preconditioning to attempt to cluster the eigenvalues at a single value. In the present work, the following matrix

$$P = (D + L)D^{-1}(D + U), \tag{3.19}$$

derived in the LU-SGS method [11,12] is used as the preconditioner matrix. An apparent advantage of the LU-SGS preconditioner is that it uses the Jacobian matrix of the linearized scheme as a preconditioner matrix, as compared with ILU preconditioner, and consequently does not require any additional memory storage and computational effort to store and compute the preconditioner matrix. Furthermore, the storage of the approximate Jacobian matrix can be completely eliminated by approximating the Jacobian with numerical fluxes [8,11,12] which will lead to a fast, low-storage implicit algorithm.

The present GMRES + LU-SGS method only requires the storage of the diagonal matrix. In addition, a storage corresponding to $2 \times \mathit{nedge}$ is required for the two index arrays, which are necessary to achieve the vectorization of the LU-SGS method. The need for additional storage associated with the GMRES algorithm is an array of size $(k + 2) \times \mathit{neqns} \times \mathit{npoin}$, where k is the number of search directions. Since the GMRES + LU-SGS method is completely separated from the flux computation procedure, memory, which is used to compute fluxes can be used by the GMRES + LU-SGS. Overall, the extra storage of the GMRES + LU-SGS method is approximately 10% of the total memory requirements.

4. Numerical results

For all the computations, the accuracy level of 0.01, i.e., two orders of magnitude drop in residual, is used to derive the unsteady residual at each time step for the pseudo-time system. The relative L_2 norm of the density residual is taken as a criterion to test convergence history. The solution tolerance for GMRES is set to 0.1 with 10 search directions and 20 iterations. However, it takes a few iterations (typically about five) to satisfy the stopping criterion.

4.1. Sod shock tube problem

The shock tube problem constitutes a particularly interesting and difficult test case, since it presents an exact solution to the full system of one-dimensional Euler equations containing simultaneously a shock wave, a contact discontinuity, and an expansion fan. This problem is chosen to validate the numerical schemes and assess the temporal accuracy of the numerical solution obtained by the present method, since an analytical solution exists. The numerical solution obtained by a linearized implicit method will also be presented for comparison. The linearized implicit scheme uses a three-point backward-difference approximation for the temporal discretization. The left implicit Jacobian operator is approximated using a first-order approximation. There is no attempt to drive the unsteady residual to zero using inner iterations. The initial conditions in the present computation are the following:

$$\rho = 1.000, \quad u = 0, \quad p = 1.0, \qquad 0.0 \leq x \leq 50.0$$

$$\rho = 0.125, \quad u = 0, \quad p = 0.1, \qquad 50.0 < x \leq 100.$$

Fig. 1a shows the mesh used in the computation. It contains 101 points in the X-direction and three points in the Y-direction. This is a 2D simulation of a 1D problem. Different time steps have been used to compute this problem to study the temporal accuracy of the implicit methods. Fig. 1b displays the computed density distribution at $t = 200$ obtained by the present fully implicit method using time steps $\Delta t = 0.2$, 0.4, and 0.8, while the density distribution obtained using the linearized implicit method is shown in Fig. 1c. As the time step is quite small, only a few time steps (typically less than three) are needed to achieve a steady-state solution to the pseudo-time system at each time step. When a small time step $\Delta t = 0.2$ is used in the computation, both methods produce an accurate solution. However, when the time step is doubled to $\Delta t = 0.4$, the result obtained using the linearized method is getting distorted, while the fully implicit method is still able to yield a reasonably accurate result. When the time step is further increased to $\Delta t = 0.8$, both methods yield an erroneous solution, though the results obtained with the linearized implicit method are much worse than those obtained with the fully implicit method. This example verifies that the fully implicit method yields a more accurate temporal accuracy than its linearized counterpart. The reason for this is due to the fact that no inner iterations have been used to drive the unsteady residual to zero in the linearized implicit method. The formal second-order accuracy in time is lost due to the linearization process, especially when the left-hand-side matrix is carried out using the first-order upwind scheme.

4.2. *Transonic flow past a pitching NACA0012 airfoil*

The second case studied is an unsteady transonic flow around a NACA0012 airfoil which performed pitching oscillations with a harmonically varying angle of attack

$$\alpha(t) = \alpha_m + \alpha_0 \sin(\omega t)$$

around the quarter-chord location. α_m is the mean value of the angle of attack and α_0 the oscillation amplitude. The reduced frequency, which is the important similarity parameter for flows with unsteady boundary conditions, is defined as

$$\kappa = \frac{\omega c}{2U_\infty},$$

where U_∞ is the freestream velocity and c either the chord length of the airfoil, the wing root c_r, or the mean aerodynamic chord length c_{ac}. For the test case chosen, $\alpha_m = 0.016^\circ$, $\alpha_0 = 2.51^\circ$, the reduced frequency $\kappa = 0.0814$. The freestream Mach number is 0.755. This problem is chosen to assess the efficiency of the numerical methods. The mesh containing 157,650 elements, 30,633 points, and 7917 boundary points is shown in Fig. 2a. This is a 3D simulation of a 2D problem. The unsteady calculation was started from the steady-state solution at a Mach number of 0.755 and an angle of attack of 0.016°. The computed pressure contours in the flow fields for this steady state is displayed in Fig. 2b. The time histories of the lift and moment coefficients vs. the time dependent angle of attack and the time using 16, 32, and 64 time steps per cycle are presented in Fig. 2c–f. Also shown is a comparison with the

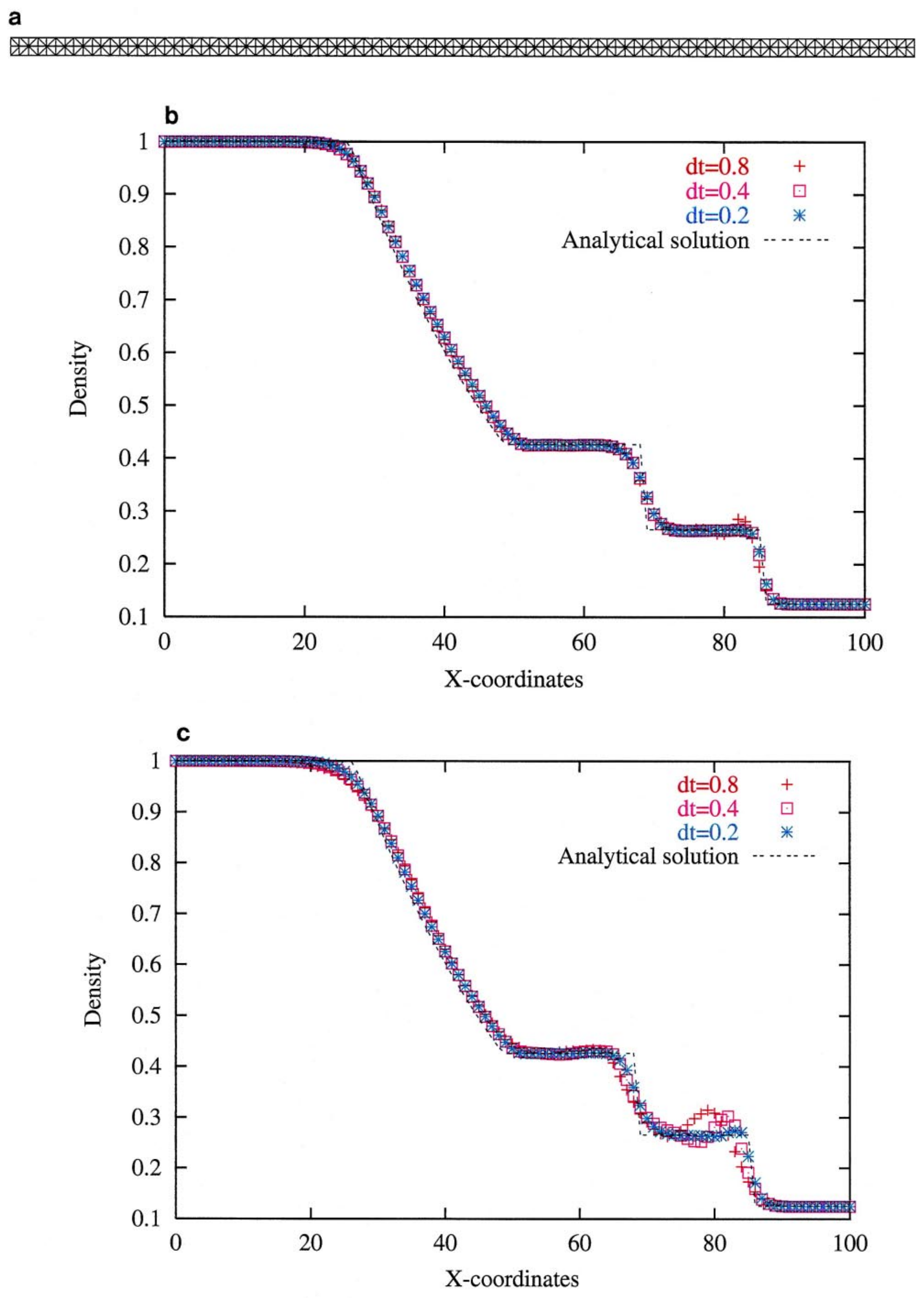


Fig. 1. (a) Mesh used for computing Riemann shock tube problem (*nelem* = 400, *npoin* = 303, *nboun* = 240); (b) density profiles obtained using different time steps by the fully implicit method; (c) density profiles obtained using different time steps by the linearized implicit method.

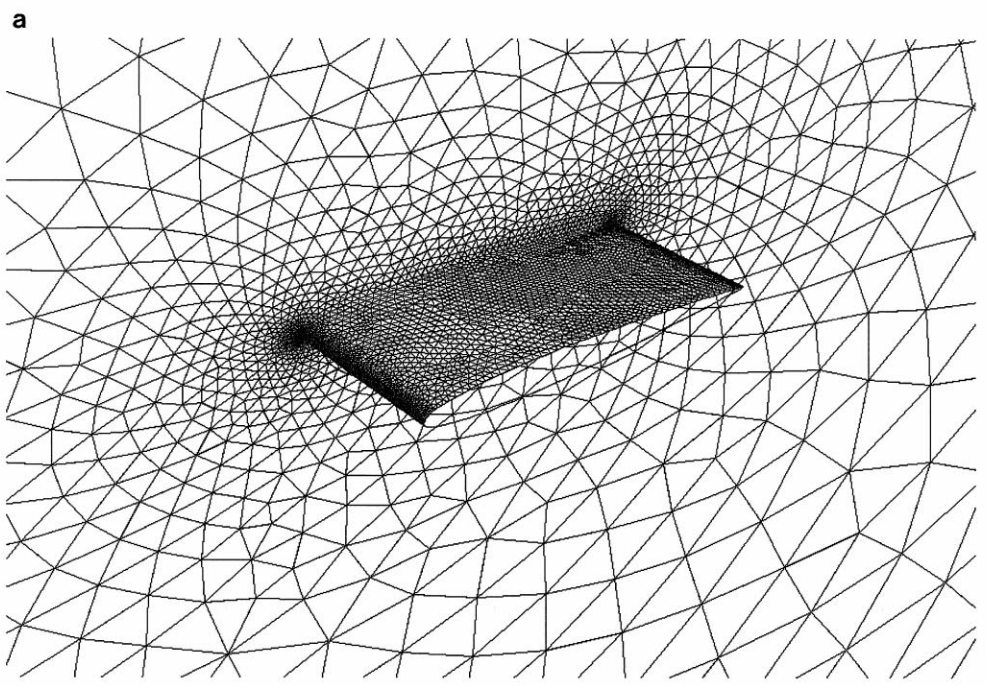
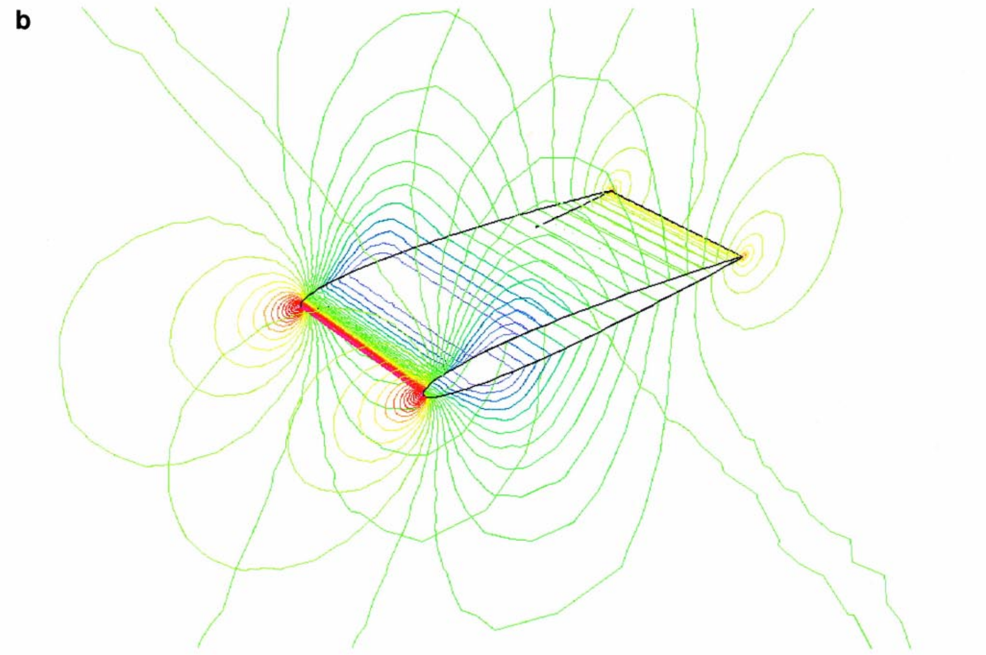
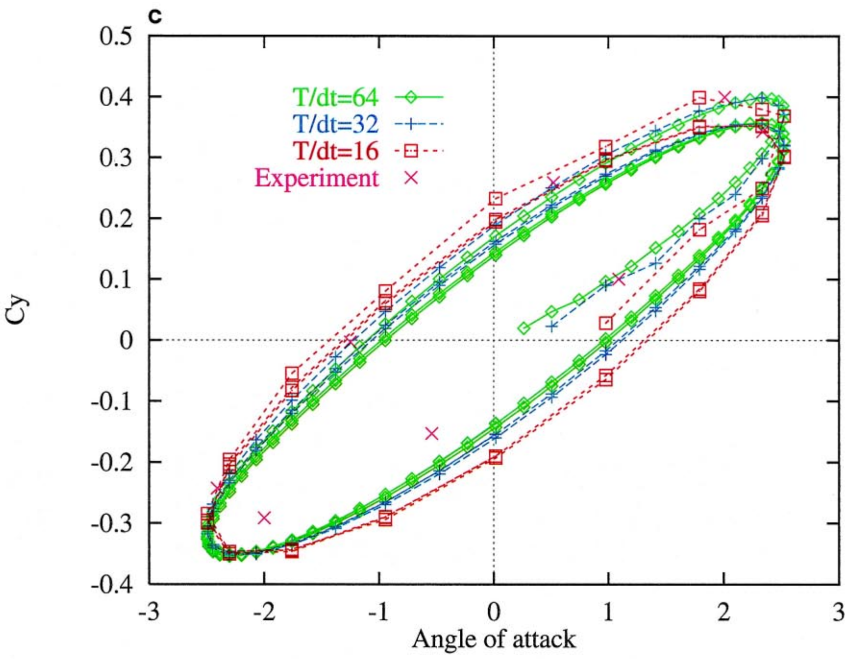



Fig. 2. (a) Surface mesh used for computing transonic flow around NACA0012 pitching airfoil configuration (*nelem* = 157,650, *npoin* = 30,633, *nboun* = 7917); (b) computed pressure contours on the NACA0012 wing at $M_\infty = 0.755$ and $\alpha = 0.016^\circ$; (c) lift coefficient history vs. angle of attack for NACA0012 pitching airfoil using different time steps per cycle; (d) moment coefficient history vs. angle of attack for NACA0012 pitching airfoil using different time steps per cycle; (e) lift coefficient history vs. time steps for NACA0012 pitching airfoil using different time steps per cycle; (f) moment coefficient history vs. time steps for NACA0012 pitching airfoil using different time steps per cycle; (g) lift coefficient history vs. time steps for NACA0012 pitching airfoil using different solution accuracy level for pseudo-time system; (h) moment coefficient history vs. time steps for NACA0012 pitching airfoil using different solution accuracy level for pseudo-time system.



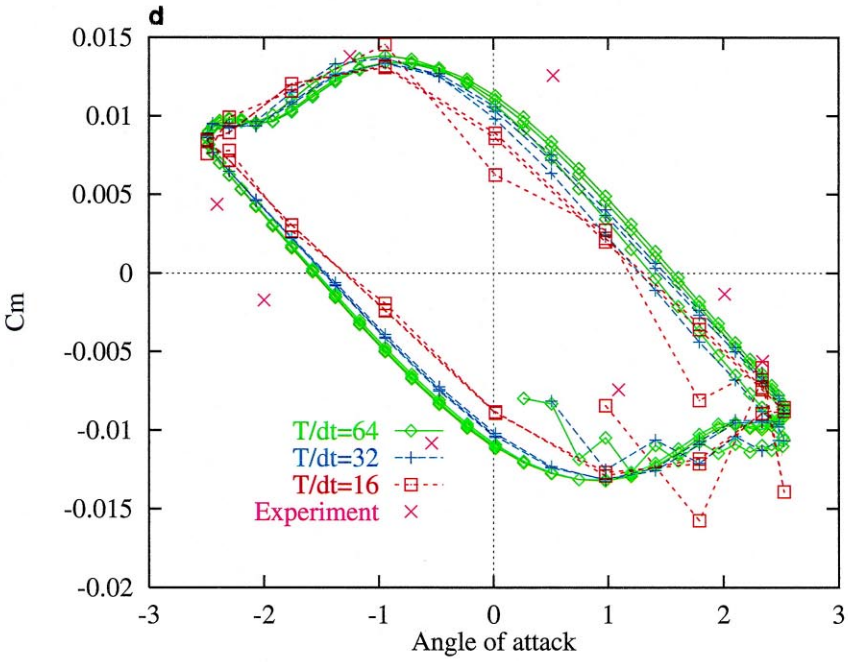
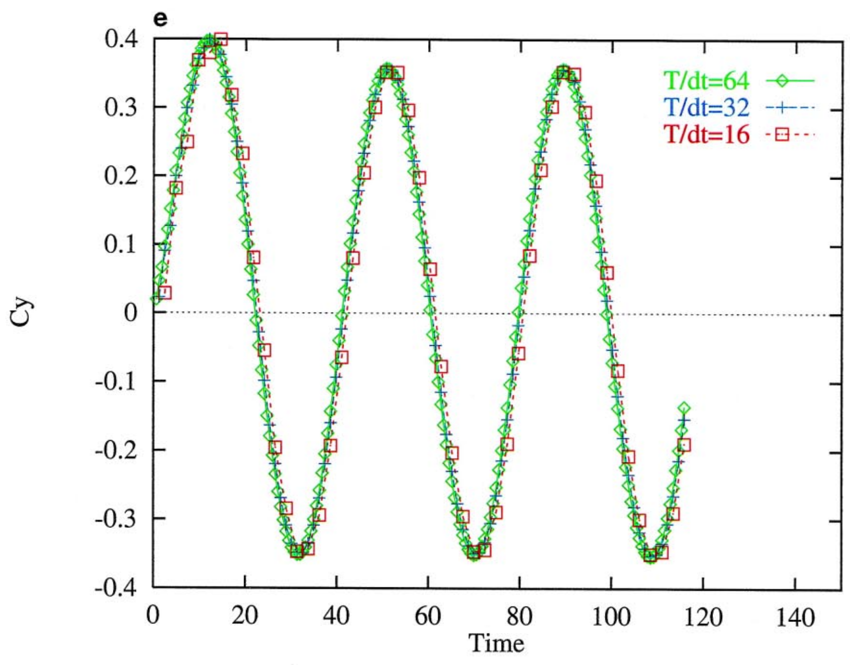


Fig. 2 (*continued*)



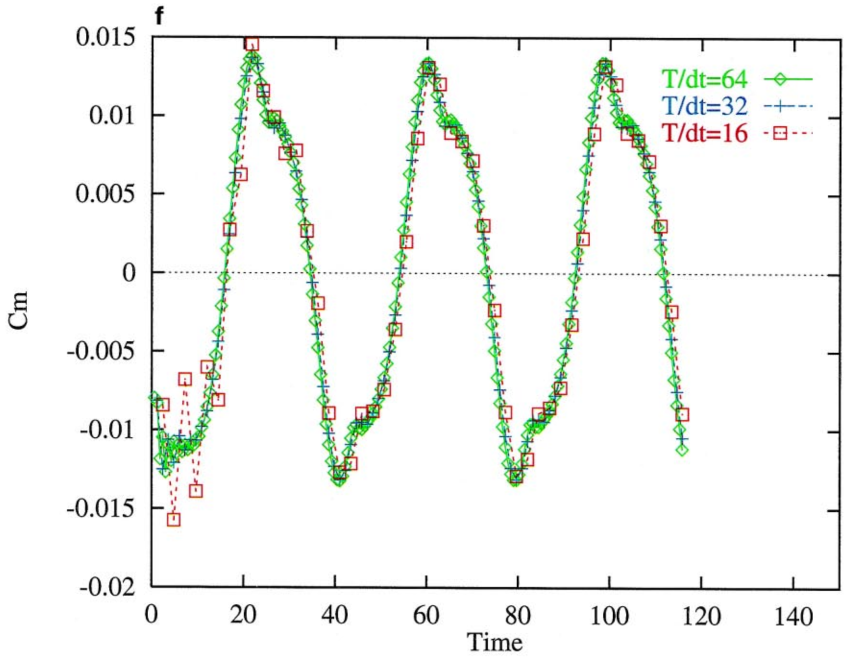
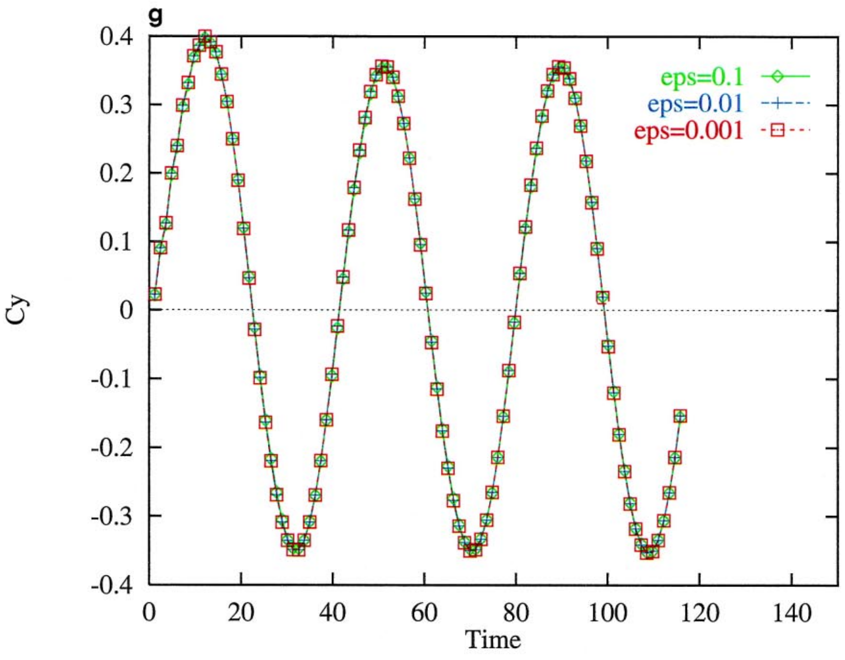


Fig. 2 (*continued*)



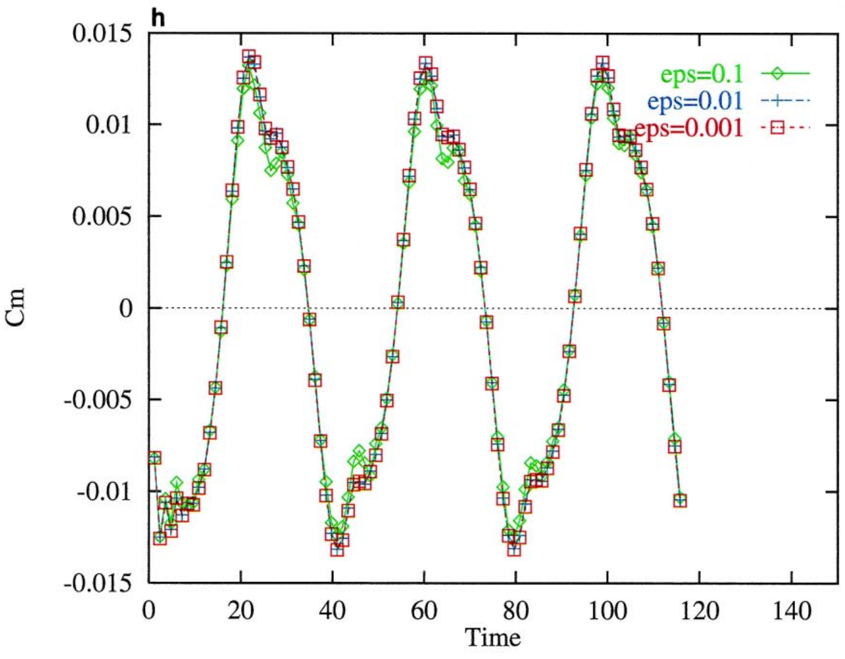


Fig. 2 (*continued*)

experimental data of Landon [13]. With 16 time steps per pitch cycle, some discrepancy is observed. This level of agreement is typical for an inviscid calculation. To study how accurately one has to solve the pseudo-time system, different accuracy levels were used to drive the unsteady residual at each time step. Fig. 2g–h show the lift coefficient and moment coefficient vs. time steps, where the different accuracy level has been used to drive the unsteady residual at each time step. The results are indistinguishable when the relative unsteady residual has dropped two or three orders of magnitude. As the CPU time depends primarily on the number of time steps per cycle, computations were done to show the influence of CFL variations on the CPU time. The computational efforts for the implicit solutions with various Δt per cycle and the explicit solution are listed in Table 1. The implicit method offers about a two-order-of-magnitude improvement over its explicit counterpart, clearly demonstrating the efficiency of the present implicit method.

4.3. Transonic flow past a pitching LANN wing

The third, well-documented case is an unsteady transonic flow over a pitching LANN wing configuration. The LANN wing is a typical transport-type wing with a high aspect ratio, leading- and trailing-edge sweep and supercritical airfoils. The results presented here refer to the design cruise condition: $M_\infty = 0.82$, $\alpha_0 = 0.25^\circ$, $\alpha_m = 0.6^\circ$. The reduced frequency based on the mean aerodynamic chord length is 0.076. The pitching axis is located at 62.1% of the root chord from the wing apex. The mesh used in the computation consists of 1,476,459 elements, 259,848 grid points, and 16,830 boundary points. The upper and lower surface meshes are shown in Fig. 3a. The computed pressure contours on the upper and lower surfaces at the steady-state are displayed in Fig. 3b. The time variation of the computed and experimentally measured surface pressure distributions can be expressed in terms of Fourier components in-phase and 90° out-of-phase with the unsteady angle of attack. The Fourier series representation of the surface pressure coefficient is

$$C_p\left(\frac{x}{c}, t\right) = C_{p_m}\left(\frac{x}{c}\right) + \sum_{n=1}^{\infty}\mathrm{Re}\left\{\left[C_{p,\,\alpha_0}\left(\frac{x}{c}\right)\right]\alpha_0 e^{in\omega t}\right\}$$

Table 1
Comparison of the CPU time between explicit and implicit methods for NACA0012 airfoil

Performance comparison				
Methods	Explicit	Implicit	Implicit	Implicit
Δt	$0.288E-2$	2.41216	1.20608	0.60304
CFL	2	1675	838	419
Time steps/cycle	13400	16	32	64
3T	40200	48	96	192
CPU (s)	548562	3484	5583	9023
Speed-up	1	157	98	61

where $C_{p_m}(\frac{x}{c})$ is the mean value of the local surface pressure coefficient and $[C^n_{p,\alpha_0}(\frac{x}{c})]$ the *n*th complex component of the local unsteady pressure coefficient, per radian. The real and imaginary value of $C^n_{p_m}$ can be expressed as

$$\mathrm{Re}\left[C^n_{p,\alpha_0}\left(\frac{x}{c}\right)\right] = (C^n_p/\alpha_0)\cos(\phi_n)$$

and

$$\mathrm{Im}\left[C^n_{p,\alpha_0}\left(\frac{x}{c}\right)\right] = (C^n_p/\alpha_0)\sin(\phi_n)$$

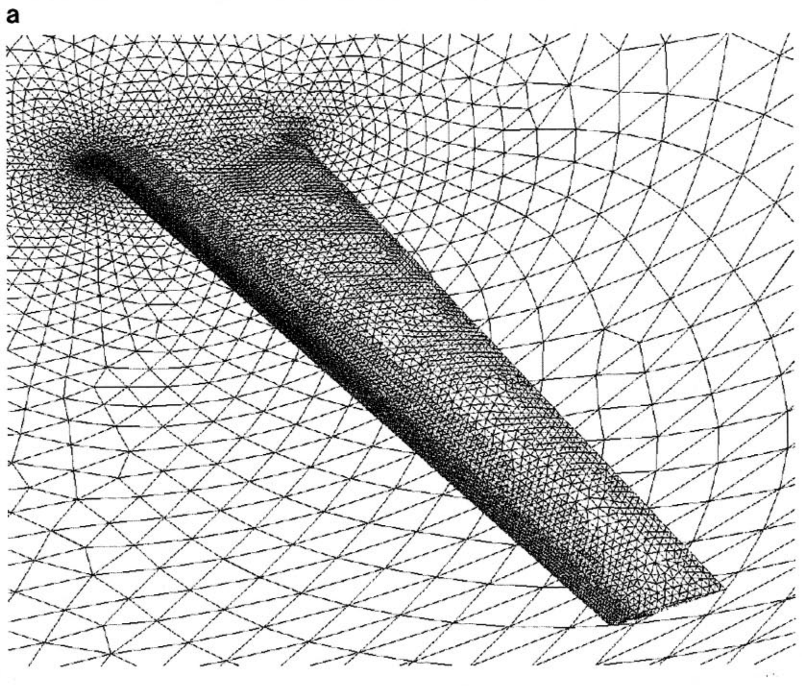
where $\mathrm{C}^{\mathrm{n}}_{\mathrm{p}}$ is the *n*th harmonic unsteady pressure (real) and ϕ_n is the *n*th harmonic phase shift between the angle of attack and the pressure response. The mean pressure distribution in Fig. 3c, the real in Fig. 3d and the imaginary component parts in Fig. 3e for the first harmonic ($n = 1$) of the Fourier series are in good agreement with experimental data on the lower surface [13]. On the upper surface the leading-edge suction peak of the mean pressure distribution is represented exactly at all six spanwise sections. The shift of the shock location aft of the measured shock position and the overprediction of the shock strength indicates that noticeable viscous effects are present in the real flow. The peaks in the in-phase and out-of-phase components are shifted downwards and overpredicted, too. With the exception of these, the unsteady pressures are reproduced fairly well by the present implicit method. The computational efforts for both implicit and explicit solutions are listed in Table 2. The implicit method offers almost two orders of magnitude improvement over its explicit counterpart.

4.4. Canopy ejection from an F/A18-C/D fighter

As an example of application, the developed implicit method was used to compute an F/A18-C/D fighter canopy ejection (Fig. 4). During the initial opening of the canopy, a number of topological changes occur in the geometry. Most computational efforts were spent on local

Table 2
Comparison of the CPU time between explicit and implicit methods for LANN wing

Performance comparison		
Methods	Explicit	Implicit
Δt	$0.946E-03$	0.326
CFL	1	344
Time steps/cycle	11008	32
3T	33024	96
CPU (s)	1783296	22459
Speed-up	1	80



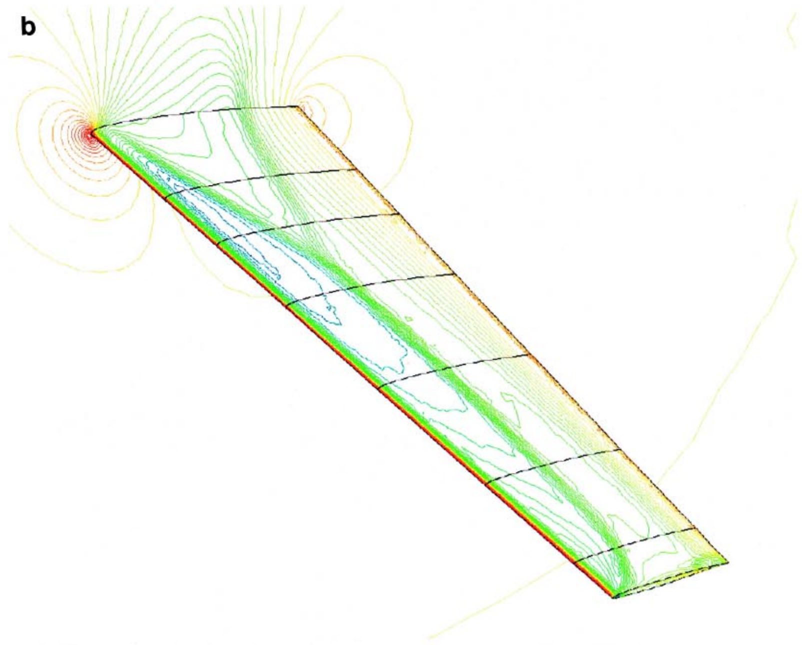


Fig. 3. (a) Surface mesh used for computing transonic flow around pitching LANN wing configuration (*nelem* = 954,508, *npoin* = 169,156, *nboun* = 13,500); (b) computed steady-state pressure contours on the LANN wing at $M_\infty = 0.82$ and $\alpha = 0.6°$; (c) spanwise comparison of the measured and computed mean pressure distribution over the LANN wing, $M_\infty = 0.82$, $\alpha_m = 0.6°$, $k_{AC} = 0.076$, $\alpha_0 = 0.25°$; (d) spanwise comparison of the measured and computed in-phase component on the LANN wing, $M_\infty = 0.82$, $\alpha_m = 0.6°$, $k_{AC} = 0.076$, $\alpha_0 = 0.25°$; (e) spanwise comparison of the measured and computed out-of-phase component on the LANN wing, $M_\infty = 0.82$, $\alpha_m = 0.6°$, $k_{AC} = 0.076$, $\alpha_0 = 0.25°$.

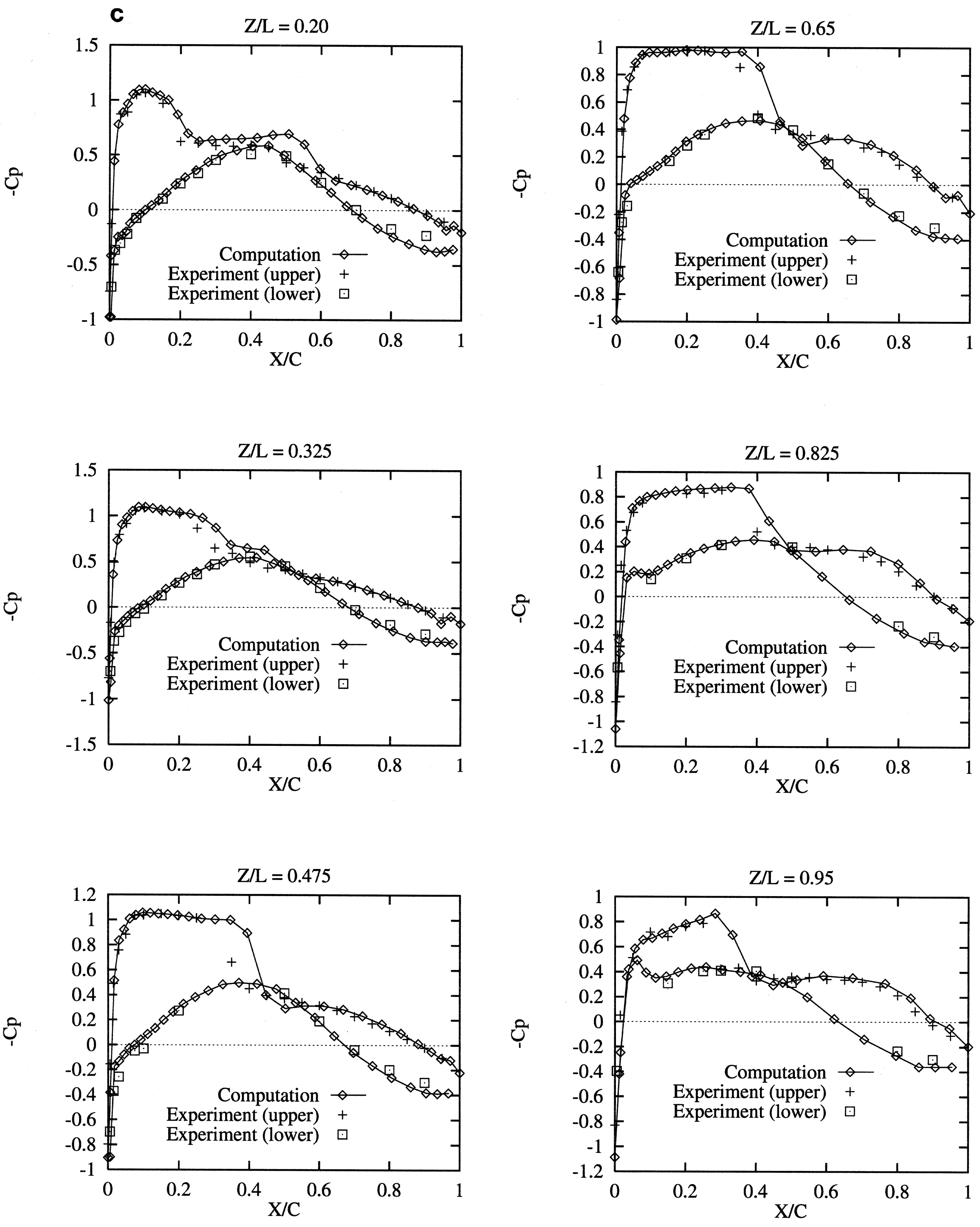


Fig. 3 (*continued*)

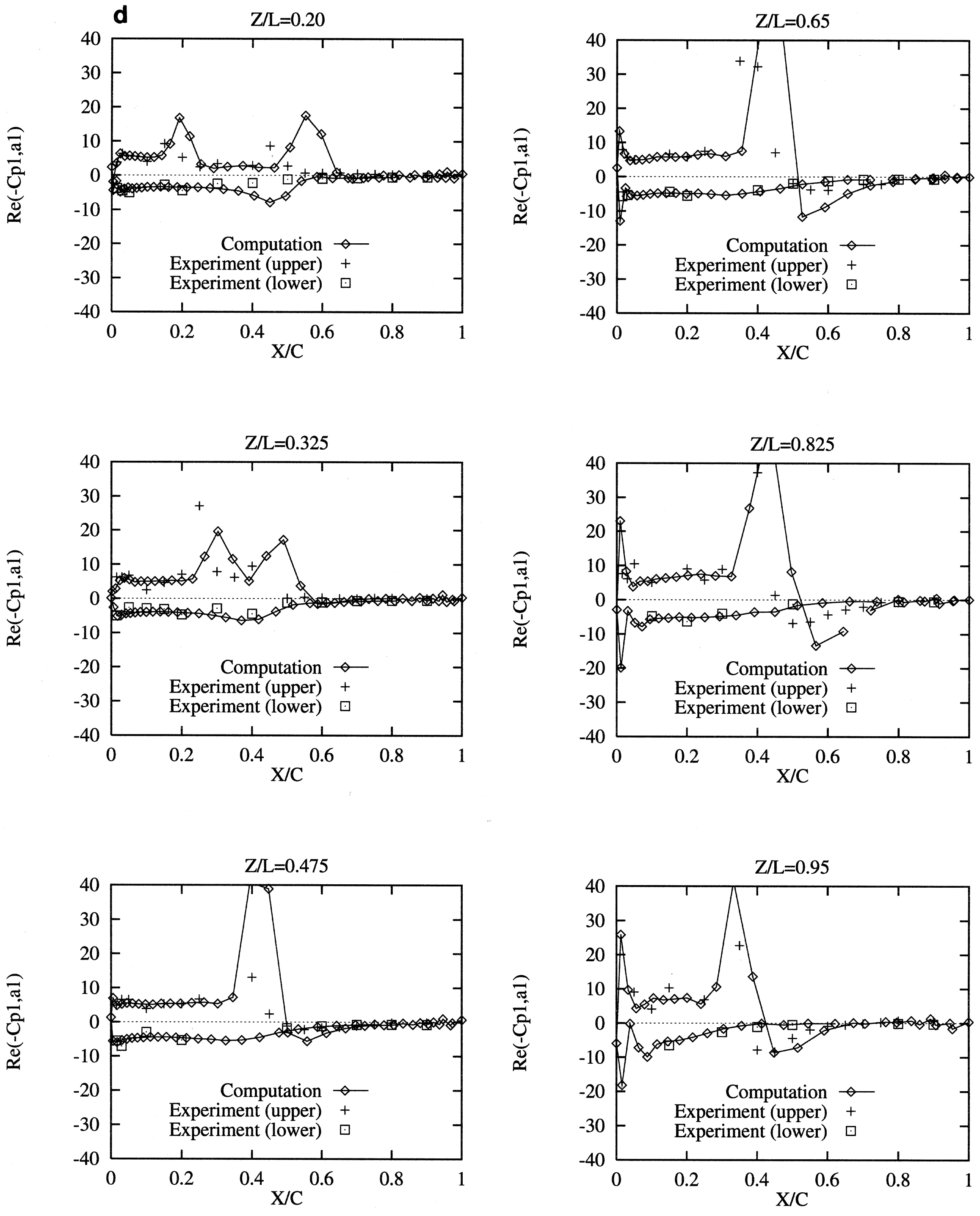


Fig. 3 (*continued*)

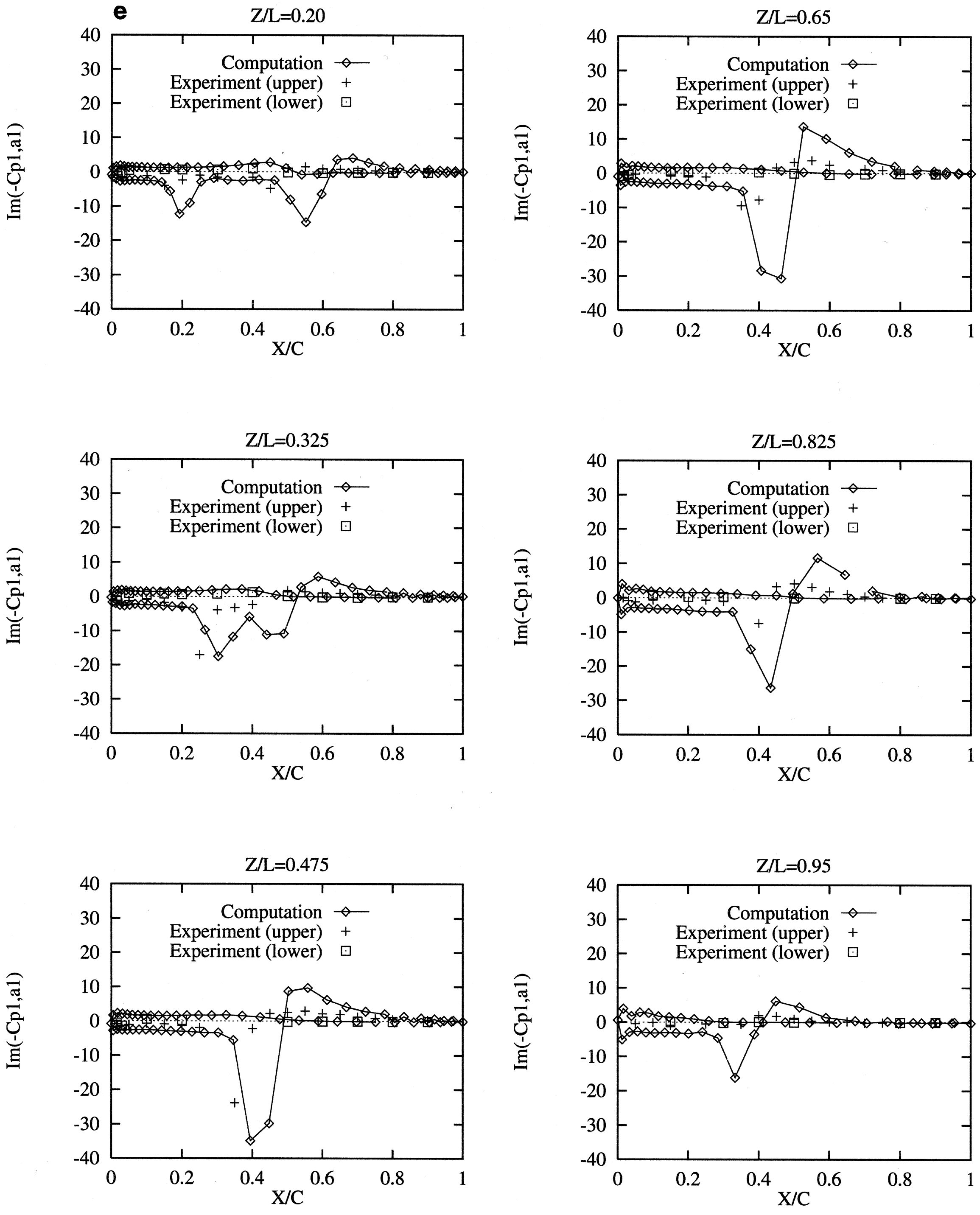


Fig. 3 (*continued*)

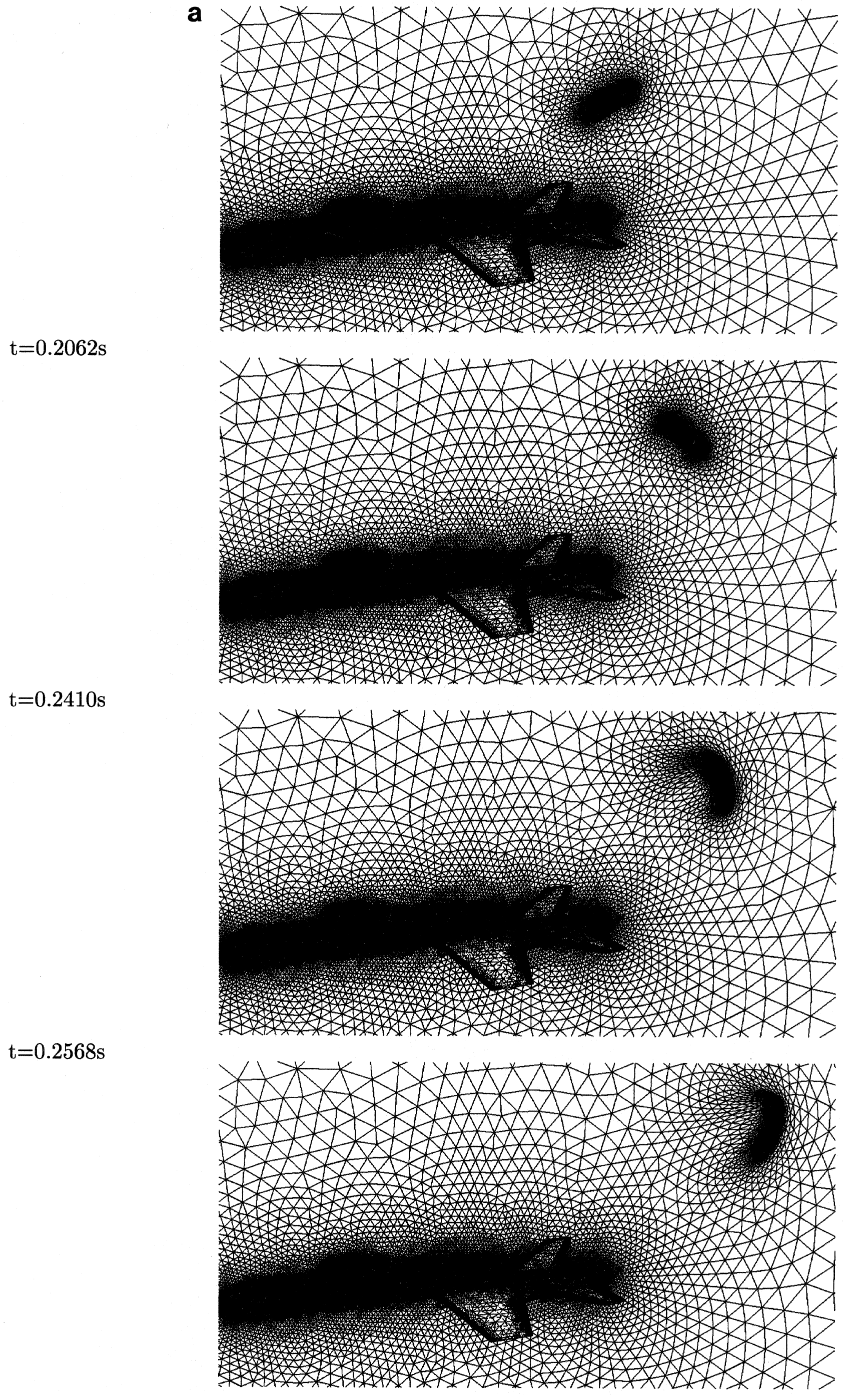


Fig. 4. (a) Unstructured triangular surface mesh for canopy ejection problem at different times; (b) Mach number contours for canopy ejection from an F/A-18C/D fighter at different times.

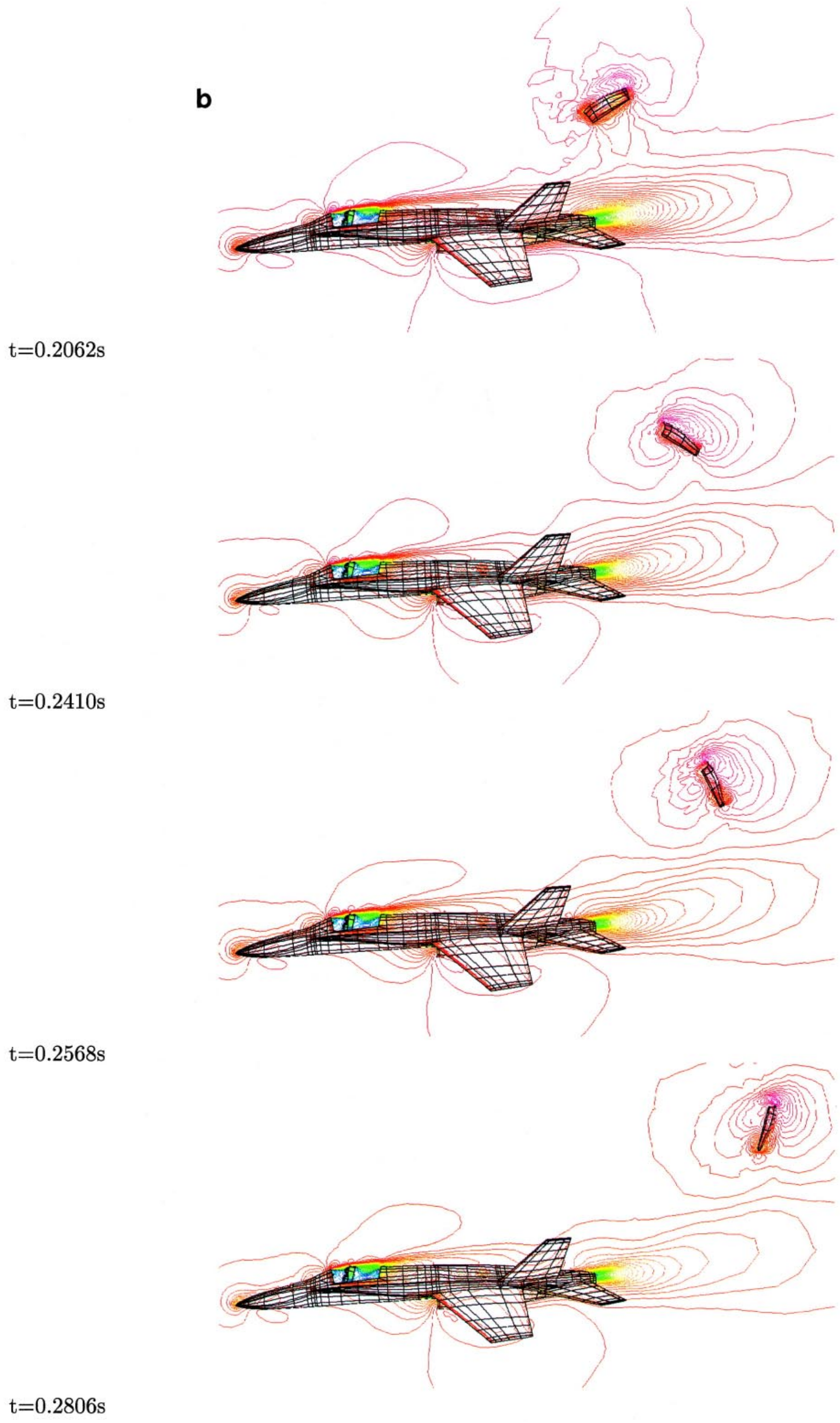


Fig. 4 (*continued*)

and global remeshing. To show the effectiveness of the implicit method, the simulation was started at $t = 0.2038$ s and ended at $t = 0.2808$ s. During this time, the canopy has moved away from the fighter. A time step of 0.001 s, corresponding to CFL number of 200, is used in the computation. The average size of the mesh was about 250,000 points, and 1,300,000 elements. The size of grid varied a little during the computation, due to several local and global remeshings. The simulation required approximately 10 CPU hours on a Cray C-90, single processor, while the explicit method would require about 25 CPU hours. Although the saving does not seem impressive, keep in mind that a significant portion of the CPU time is spent on local and global remeshing, which is the same for both explicit and implicit methods and requires about 8 CPU hours. For the flow solution phase alone, the implicit method is about nine times faster than its explicit counterpart for this computation.

5. Conclusions

An accurate, fast, matrix-free implicit method has been developed to solve three-dimensional compressible unsteady flows on unstructured grids. The developed method has been applied to compute a variety of unsteady flow problems involving moving boundaries. The numerical results obtained indicate that the use of the present implicit method leads to a significant increase in performance over its explicit counterpart, while maintaining competitive memory requirements.

Acknowledgements

This research was sponsored by the Defense Special Weapon Agency. Dr. Michael E. Giltrud served as the technical program monitor. Partial funding was also provided by NAWCAD, with Dr. Thomas J. Marquette serving as a technical monitor.

References

[1] Pulliam TH. Time accuracy and the use of implicit schemes. AIAA Paper 93-3360, 1993.
[2] Rai MM. Navier–Stokes simulations of blade–vortex interaction using high-order accurate upwind schemes. AIAA Paper 87-0543, January 1987.
[3] Brennis A, Eberle A. Application of an implicit relaxation method solving the Euler equations for time-accurate unsteady problems. Transaction of AMSE J Fluids Engrg 1990;112:510–20.
[4] Jameson A. Time-dependent calculations using multigrid, with application to unsteady flows past airfoils and wings. AIAA Paper 91-1569, July 1991.
[5] Vassberg JC. A fast, implicit unstructured-mesh Euler method. AIAA Paper 92-2693, 1992.
[6] Crumpton PI, Giles MB. Implicit time accurate solutions on unstructured dynamic grids. AIAA Paper 95-1671, 1995.
[7] Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. AIAA Paper 95-1705, 1995.
[8] Luo H, Baum JD, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grid. Journal of Computational Physics 1998;146:664–90.

[9] Liou MS. Progress towards an improved CFD method: AUSM+. AIAA Paper 95-1701, June 1995.
[10] Van Leer B. Towards the ultimate conservative difference scheme. Part II: Monotonicity and conservation combined in a second-order scheme. Journal of Computational Physics 1974;14:361–70.
[11] Men'shov I, Nakamura Y. An implicit advection upwind splitting scheme for hypersonic air flows in thermochemical nonequilibrium. In: 6th Int. Symp. CFD. 1995.
[12] Sharov D, Nakahashi K. Reordering of 3D hybrid unstructured grids for vectorized LU-SGS Navier–Stokes computations. AIAA Paper 97-2102, 1997.
[13] Landon H. Compendium of unsteady aerodynamic measurements. AGARD Report No. 702, 1983.