INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 8, 1217–1227 (1988)

THE APPLICATION OF MULTIGRID TO NAVIER–STOKES SIMULATION OF 3D FLOW IN AXIAL AND RADIAL FLOW TURBOMACHINERY

W. N. DAWES

Whittle Laboratory, University of Cambridge, Cambridge, CB3 0DY, U.K.

SUMMARY

This paper describes the implementation of a 3D Navier–Stokes solver within the framework of a multigrid strategy. The ability of multigrid to improve and sustain code performance over a range of mesh sizes and a variety of difficult flow problems is investigated. The code is applied to the study of the shock-induced boundary layer separation on a channel hump, the flow in a 2D cascade of compressor blades, the secondary flow development in a linear cascade of gas turbine blades and the Eckardt backswept centrifugal impeller. In each case multigrid leads to significantly improved performance.

KEY WORDS Multigrid 3D Navier–Stokes solver Flow in turbomachinery

INTRODUCTION

Stable, efficient algorithms for time-marching the three-dimensional Navier–Stokes equations exist and are much researched. Several codes have been developed to the point at which application to practical flows in turbomachinery can begin. The physical realism of our numerical simulations is now limited by turbulence modelling and by our ability to use a sufficiently fine mesh to resolve the flow field.

Whilst turbulence modelling is likely to remain a pacing item for some time, rapidly developing computer hardware means that computer memory, and hence mesh size, is becoming less of a problem. As Navier–Stokes solvers are increasingly brought into the design environment, the key issue is to maintain the performance of the code, and hence the solution cost, as finer and finer meshes are employed.

The convergence history of a typical code has two phases. Initially, convergence is rapid as the algorithm efficiently eliminates high-frequency components of the solution error. In the second phase the low-frequency errors have become dominant and the convergence rate slows. The spectral radius associated with the second phase has the form $\lambda_\infty = 1 - O(\Delta X^2)$ and it is clear that mesh refinement carries with it the risk of dramatic deterioration in code performance and escalation of solution cost.

One way round this difficulty is to deploy the basic solution algorithm within a multigrid strategy. The basic multigrid idea is to solve the flow simultaneously on a succession of coarser meshes. Solution residues are computed on the finest mesh and restricted to the coarser meshes. On each coarse mesh, corrections to the basic flow variables are computed which are then prolonged to the finest mesh. Each mesh efficiently eliminates its own high-frequency errors and so

0271–2091/88/101217–11$05.50 *Received 2 April 1988*

a sweep through the meshes allows a spectrum of errors to be handled optimally. As finer meshes are employed, more multigrid levels can be implemented and so, in principle, the convergence rate of the code should be independent of mesh size and thus the solution cost should vary only linearly with the number of mesh nodes used.

EQUATIONS OF MOTION

The three-dimensional Reynolds-averaged Navier–Stokes equations are written in finite volume form and cast in the blade-relative frame using cylindrical co-ordinates (r, θ, x)[1].

$$\frac{\partial}{\partial t}\oint_{\mathrm{VOL}} \mathbf{U}\,\mathrm{dVOL} = \oint \mathbf{H}\,\mathrm{d}\overline{\mathrm{AREA}} + \oint \rho \mathbf{S}\,\mathrm{dVOL}, \tag{1}$$

where

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho W_x \\ r\rho W_\theta \\ \rho W_r \\ \rho E \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \rho\mathbf{q} \\ \rho W_x\mathbf{q} + \tau\hat{\mathbf{i}}_x \\ r\rho W_\theta\mathbf{q} + \tau\hat{\mathbf{i}}_\theta \\ \rho W_r\mathbf{q} + \tau\hat{\mathbf{i}}_r \\ \rho I\mathbf{q} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ -2\Omega r W_r \\ W_\theta^2/r + r\Omega^2 + 2\Omega W_\theta \\ 0 \end{bmatrix},$$

with $\mathbf{q} = W_x\hat{\mathbf{i}}_x + W_r\hat{\mathbf{i}}_r + W_\theta\hat{\mathbf{i}}_\theta$ the relative velocity, Ω the rotation speed, τ the stress tensor (containing both the static pressure and the viscous stresses) and $I = c_p T_{0_{\mathrm{rel}}} - \frac{1}{2}(\Omega r)^2$ the rothalpy. The system is closed by an equation of state

$$p = \rho(\gamma - 1)(E - 0{\cdot}5*(\mathbf{q}\cdot\mathbf{q} - (\Omega r)^2))$$

and a mixing length turbulence model patterned after Baldwin and Lomax.[2]

NUMERICAL SOLUTION PROCEDURE

Finite volume formulation

The governing equations (1) are written in integral conservation form and the numerical discretization is designed to mimic this. We divide the computational domain into hexahedral cells and store the variables $\mathbf{U} = (\rho, \rho W_x, r\rho W_\theta, \rho W_r, \rho E)$ at cell centres (ijk). The integrals in the equations are replaced by discrete summation around the faces of the computational cell:

$$\frac{\Delta \mathbf{U}_{ijk}}{\Delta t}\Delta\,\mathrm{VOL}_{ijk} = \sum_{\mathrm{CELL}(ijk)} \mathbf{H}\,\Delta\overline{\mathrm{AREA}} + \rho S_{ijk}\Delta\mathrm{VOL}_{ijk} = \mathbf{F}(\mathbf{U})_{ijk}. \tag{2}$$

Fluxes through cell faces are found by linear interpolation of density, velocity, etc. between cell centres, and so the formal spatial accuracy is second order on smoothly varying meshes and global conservation is ensured. Viscous stresses are computed by defining a local curvilinear co-ordinate system and the chain rule.

Artificial viscosity

An adaptive artificial viscosity term recommended by Jameson and Baker[3] is added to the discretized equations (2) to control odd–even point solution decoupling and to suppress

oscillations in regions with strong pressure gradients. Equation (2) becomes

$$\frac{\Delta \mathbf{U}_{ijk}}{\Delta t}\,\Delta \mathrm{VOL}_{ijk} = \mathbf{F}(\mathbf{U})_{ijk} + \mathbf{D}(\mathbf{U})_{ijk}, \tag{3}$$

where **D** is the dissipative operator.

The artificial viscosity term has the form

$$\mathbf{D}(\mathbf{U}) = \mathbf{D}_I + \mathbf{D}_J + \mathbf{D}_K,$$

where D_I, D_J and D_K represent the contributions from each of the curvilinear co-ordinate directions. Each contribution is written in conservation form as, for example,

$$D_I = d_{i+\frac{1}{2}jk} - d_{i-\frac{1}{2}jk}.$$

The right-hand-side terms have the form

$$d_{i+\frac{1}{2}jk} = \frac{\Delta \mathrm{VOL}}{\Delta t}\left[\varepsilon^{(2)}_{i+\frac{1}{2}jk}(U_{i+1jk} - U_{ijk}) - \varepsilon^{(4)}_{i+\frac{1}{2}jk}(U_{i+2jk} - 3U_{i+1jk} + 3U_{ijk} - U_{i-1jk})\right]. \tag{4}$$

The coefficients $\varepsilon^{(2)}$ and $\varepsilon^{(4)}$ are determined by the flow in a self-adaptive manner:

$$\varepsilon^{(2)}_{i+\frac{1}{2}jk} = K^{(2)}\,\mathrm{AMIN1}\,(0{\cdot}5,\,(\pi_{ijk} + \pi_{i+1jk}) * 0{\cdot}5),$$

$$\varepsilon^{(4)}_{i+\frac{1}{2}jk} = \mathrm{AMAX1}\,(0,\,(K^{(4)} - \alpha\varepsilon^{(2)}_{i+\frac{1}{2}jk})),$$

where

$$\pi_{ijk} = |p_{i+1jk} - 2p_{ijk} + p_{i-1jk}|/(p_{i+1jk} + 2p_{ijk} + p_{i-1jk}).$$

In the current effort we have used values for the constants of $K^{(2)} = 1$, $K^{(4)} = 0{\cdot}01$ and $\alpha = 2$. Numerical experimentation showed that the dissipation in directions normal to solid boundaries should be set to zero to avoid masking the physical viscosity.

Boundary conditions

At inflow the total temperature and pressure are fixed and either the flow angle or the absolute swirl velocity is held constant depending on whether the relative flow is subsonic or supersonic. At outflow the hub static pressure is fixed and the radial variation derived from the simple radial equilibrium equation.

The finite volume mesh is constructed so that cell faces lie on solid surfaces (blades, hub and casing) and along the periodic boundaries upstream and downstream of the blade row. Consequently, cells adjacent to periodic boundaries are updated just as if they were interior cells with the flux across the periodic boundary formed from linear interpolation between variables stored at the centres of the cells on either side of the boundary.

For cells adjacent to solid boundaries, zero fluxes of mass, momentum and energy are imposed through the cell face aligned with the solid boundary. In addition, boundary conditions must be devised for the wall static pressure and wall shear stress; these two, acting on the wall cell face, are used in updating the momentum and energy equations for the cells adjacent to the solid wall. The wall static pressure is found by setting the derivative of pressure normal to the wall equal to zero. This is an accepted high-Reynolds-number approximation to solving the normal momentum equation. To prescribe the wall shear stress, we use the velocities stored at cell centres adjacent to the wall and the known zero value of velocity on the wall to compute the velocity gradients at the wall. These gradients together with the wall viscosity are used with a locally defined curvilinear co-ordinate system to compute the wall shear stresses. If the mesh spacing near the wall is large relative to the sublayer scale, then a logarithmic skin friction correlation is used.

Preprocessed algorithm

The discretized equations are time-marched using an implicit preprocessed algorithm described in References 4 and 5. In outline, we define a residue $\mathbf{R}^*$ by the two steps

$$\begin{aligned}\mathbf{R}_{ijk} &= \frac{\Delta t}{\Delta \mathrm{VOL}_{ijk}}[\mathbf{F}^n_{ijk} + \mathbf{D}^n_{ijk}],\\ \hat{\mathbf{U}}_{ijk} &= \mathbf{U}^n_{ijk} + \mathbf{R}_{ijk},\\ \mathbf{R}^*_{ijk} &= \frac{\Delta t}{\Delta \mathrm{VOL}_{ijk}}[\hat{\mathbf{F}}_{ijk} + \mathbf{D}^n_{ijk}].\end{aligned} \tag{5}$$

We then update the variables by solving the implicit set of equations

$$\left[I - \varepsilon_I \frac{\Delta t^2}{\Delta \mathrm{VOL}^2}\lambda_I^2 \delta_{II}^2\right]\left[I - \varepsilon_J \frac{\Delta t^2}{\Delta \mathrm{VOL}^2}\lambda_J^2 \delta_{JJ}^2\right]\left[I - \varepsilon_K \frac{\Delta t^2}{\Delta \mathrm{VOL}^2}\lambda_K^2 \delta_{KK}^2\right]\Delta \mathbf{U} = \mathbf{R}^*, \tag{6}$$

where ε_I, ε_J and ε_K are free parameters (of order unity), λ_I, λ_J and λ_K are the spectral radii of the Jacobians associated with the convective fluxes in the I, J and K directions, and $\delta_I^2 \phi$, for example, represents $(\phi_{I+1} - 2\phi_I + \phi_{I-1})$. The left-hand side is factored into three tridiagonal matrices for efficient inversion.

Although in principle the algorithm can be made stable for any size of time step by suitable choice of the free parameters ε_I, ε_J and ε_K, in practice it is found that there is an optimum range of time steps which leads to a minimum number of steps to convergence. Extensive numerical experimentation for a range of geometries showed that larger time steps required larger values of ε for stability, and that after a certain point the solutions became too smoothed during the transient, delaying convergence. It was found that CFL numbers in the range 2–3 were optimal with associated values of ε equal to unity. For the results presented in this paper, the CFL number was 2 and $\varepsilon_I = \varepsilon_J = \varepsilon_K = 1$.

MULTIGRID CONVERGENCE ACCELERATION

Rationale

The convergence history of a typical code has two phases. Initially, convergence is rapid as high-frequency components of the solution error are efficiently eliminated. Then, when low-frequency errors have become dominant, the convergence rate slows as the spectral radius associated with this second phase is of the form $1 - O(\Delta X^2)$.

For the model convection equation

$$\frac{\partial u}{\partial t} + A\frac{\partial u}{\partial X} = 0, \tag{7}$$

application of the present preprocessed algorithm, (5) and (6), shows that the spectral radius for short wavelengths is

$$\lambda_\pi = 1 - (11c\varepsilon - 172c^2\varepsilon^2)/(1 + 4c^2) \tag{8}$$

and for long wavelengths is

$$\lambda_\infty = 1 - 0{\cdot}5c^2(\pi \Delta X/L)^2, \tag{9}$$

where c is the courant number $A\Delta t/\Delta X$, ε is the coefficient of the fourth-derivative smoothing and L is the length of the computational domain. Thus the initial convergence rate λ_π is a function of the time step size and smoothing only; but the ultimate convergence rate λ_∞ is a quadratic function of the mesh size and this is why mesh refinement can cause dramatic deterioration of code performance.

The basic principle of the multigrid technique[3, 6] is to take advantage of the fact that the finite volume residue $\mathbf{R}_{ijk}$ (equation (5)) has errors over the whole range of wavelengths which can be supported by the mesh. The time-marching algorithm, equation (6), is efficient at eliminating the short-wavelength components of this error, but much less so for higher wavelengths. So we define a succession of ever coarser meshes, typically by deleting every other mesh line, derive an appropriate representation of the residue on each mesh and use the basic time-marching solver, equation (6), to reduce the level of the error associated with the current mesh. Thus sweeping through the meshes should allow each of the wavelengths in the residue error to be attacked with optimum efficiency.

There have been several successful applications of multigrid to the Euler equations (for example, Jameson[3] for external flow and Ni[7] for cascade flow) and some work on viscous cascade flow (for example, Chima[8]). However, much remains unclear. In particular, not much is known about the application of the standard multigrid methodology[6] to flow with rapidly refined meshes, with severe mesh aspect ratios, with strong shock waves and with boundary layers, boundary layer separation and secondary flow. This paper adopts a standard multigrid approach, like a 'recipe', and attempts solutions for a wide variety of 'difficult' flows.

Multigrid recipe

(i) *Update finest mesh* variables using the basic time-marching solver, equation (6), represented by

$$L^h \Delta U^h = R^h(U^h) + P^h. \tag{10}$$

Here U^h is the variable (ρ, ρw_x, etc.) on the finest mesh, h; R^h is the fine mesh residue; L^h is the fine mesh implicit operator; ΔU^h is the correction to the fine mesh variables,

$$U^h = U^h + \Delta U^h; \tag{11}$$

and P^h is the forcing function (zero on the finest mesh).

(ii) *Collect residues and variables* to the next mesh, the $2h$ mesh, using a suitable collection operator I_h^{2h}, and define a forcing function P^{2h}:

$$U^{2h} = I_h^{2h} U^h, \qquad P^{2h} = I_h^{2h} R^h(U^h) - R^{2h}(U^{2h}). \tag{12}$$

The collection operator performs a volume-weighted average to maintain conservation,

$$I_h^{2h} U^h = \sum U^h \Delta \mathrm{VOL}^h / \sum \mathrm{VOL}^h, \tag{13}$$

with the sum being over the fine mesh cells gathered together to form the next mesh cell. This procedure also eliminates error with twice the wavelength of the fine mesh which cannot be supported on the next mesh.

(iii) *Update current mesh* variables using the basic time-marching solver:

$$L^{2h} \Delta U^{2h} = R^{2h}(U^{2h}) + P^{2h}. \tag{14}$$

The forcing function guarantees fine mesh accuracy by causing the coarse mesh solution to be driven by the fine mesh residues.

(iv) *Prolong the coarse mesh corrections* to the finest mesh using a prolongation operator I^h_{2h}, and update the finest mesh variables:

$$U^h = U^h + I^h_{2h} \Delta U^{2h}. \tag{15}$$

This prolongation must not reintroduce errors on the finest mesh eliminated in step (i).

Outline of implementation

The guiding principle of the current implementation is that only the *finest* mesh should solve the Navier–Stokes equations themselves; the role of the coarser meshes is to propagate information (mainly blockage) concerning the fine mesh solution to the flow field at large. On the finest mesh the viscous no-slip condition on blade surfaces is applied in terms of a shear stress on the cell face aligned with the blade surface (see section on 'boundary conditions'); this drives a viscous solution. On the coarser meshes this wall shear stress is set to zero and the solution is instead driven by the blockage distribution represented by the 'collected' finest mesh variables. This idea is similar to the standard method of coupling a boundary layer solution to an inviscid core–flow solver via the boundary layer displacement thickness and surface transpiration. In the current context the collected fine mesh variables are thought of as containing generalized 'displacement thickness' type information. In other words, in the current implementation the finest mesh is Navier–Stokes, the coarser meshes are Euler plus blockage.

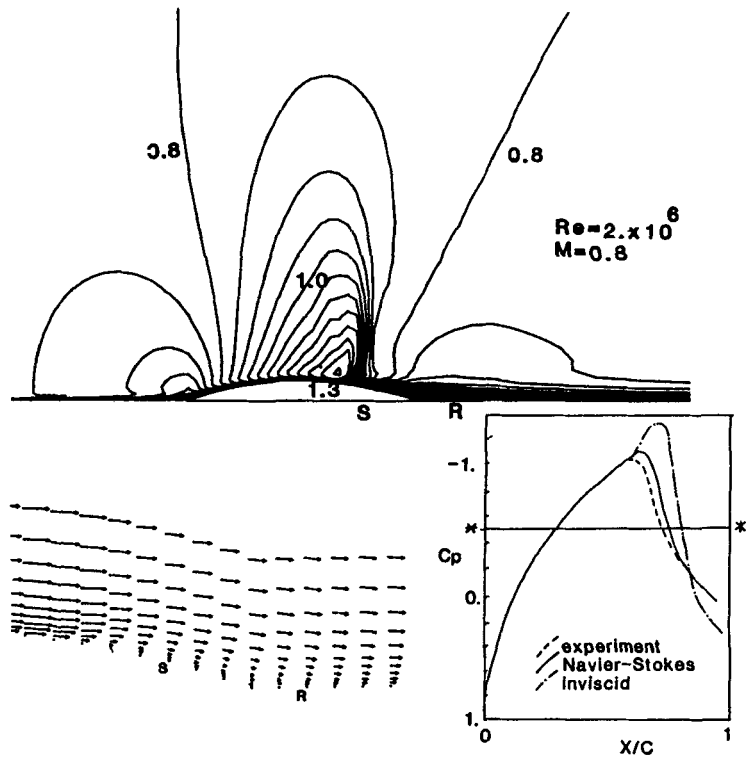

Figure 1. Transonic flow over an 18% channel hump

Table I

	Mesh	Steps to convergence	CPU minutes
81×25	No multigrid, $1h$	1800	569
	1 level, $1h$, $2h$	750	—
	2 levels, $1h$, $2h$, $4h$	500	276
81×49	2 levels, $1h$, $2h$, $4h$	600	690

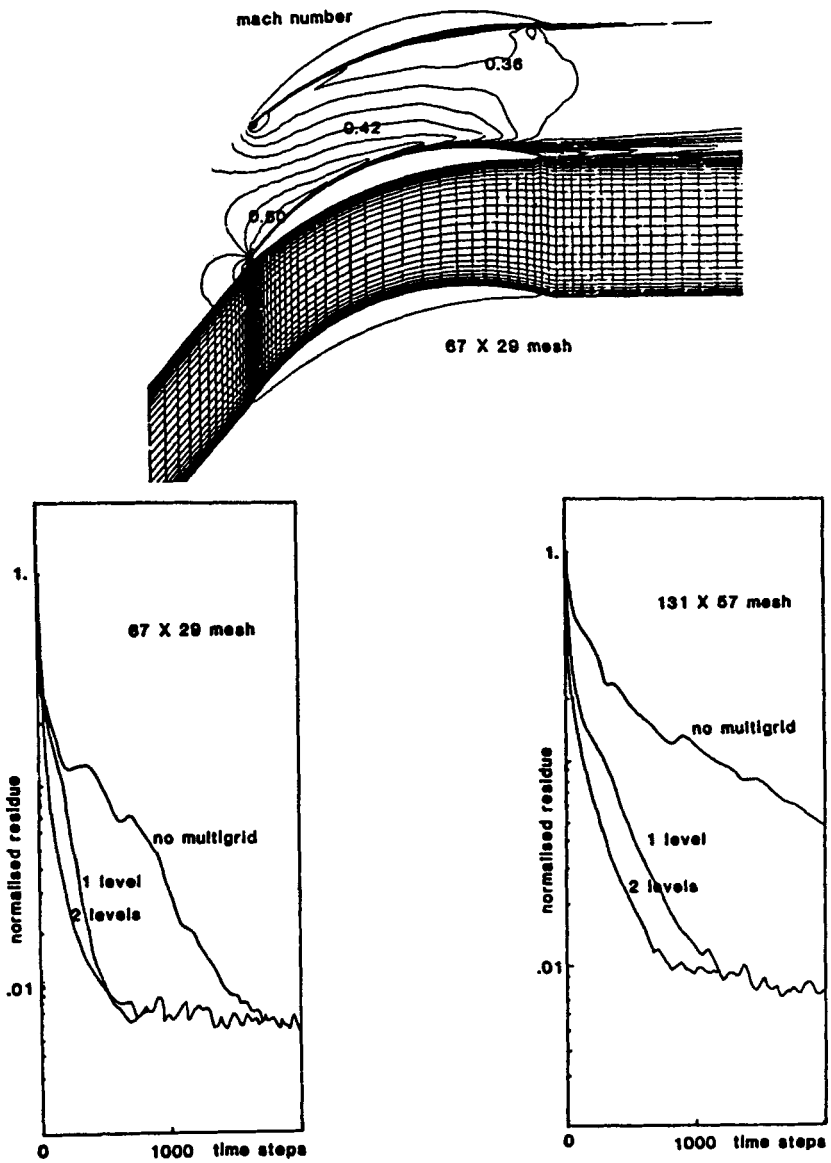

Figure 2. Cascade of compressor blades

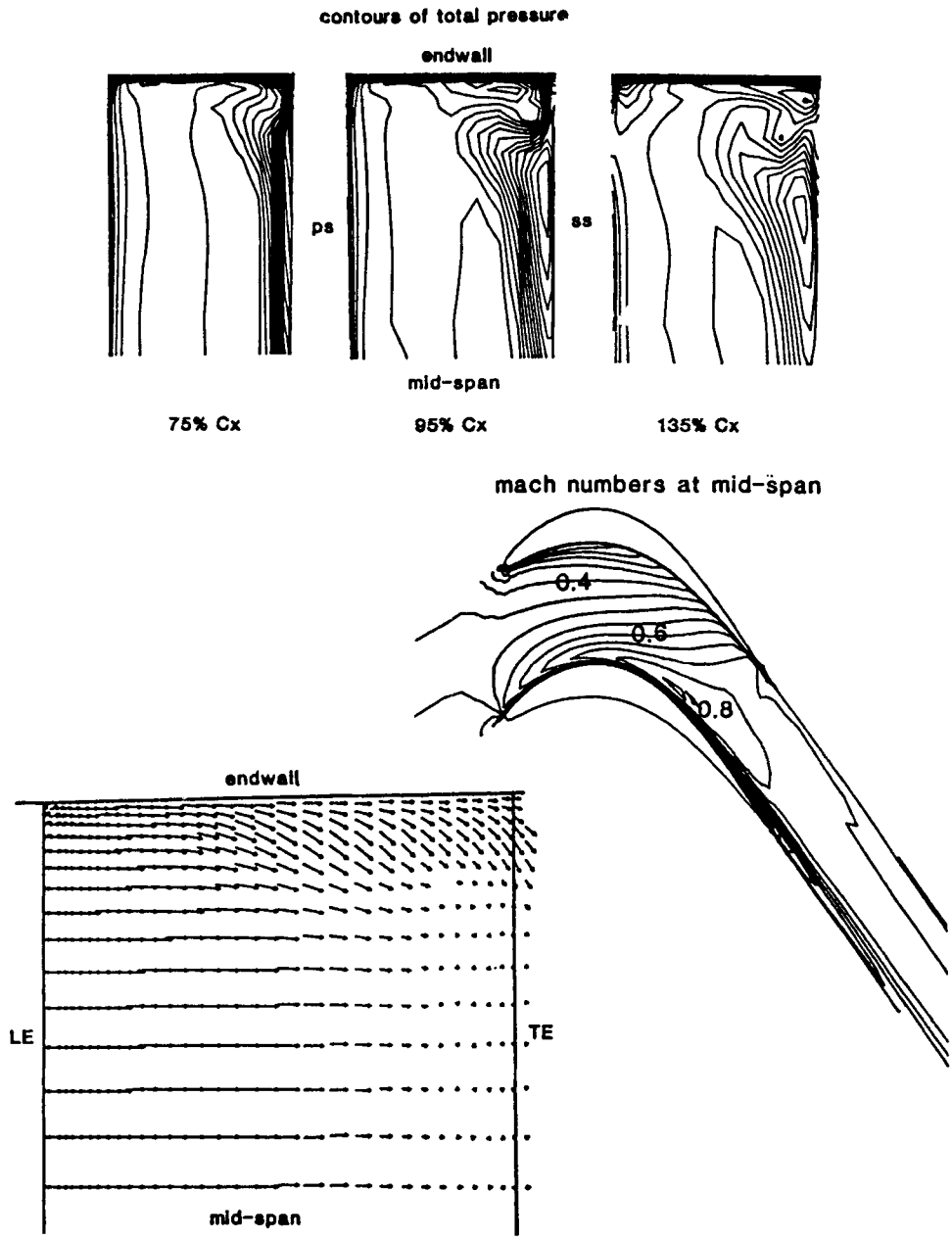

Figure 3. Secondary flow in a linear cascade of gas turbine rotor blades

PRESENTATION AND DISCUSSION OF RESULTS

The problems addressed range from a simple channel hump to a centrifugal impeller. All computations were performed on a Perkin Elmer 3230 mini-system using 32-bit arithmetic. In each case, over ranges of mesh size, multigrid allowed significantly improved code performance. A solution cost varying only linearly with mesh size is demonstrated.

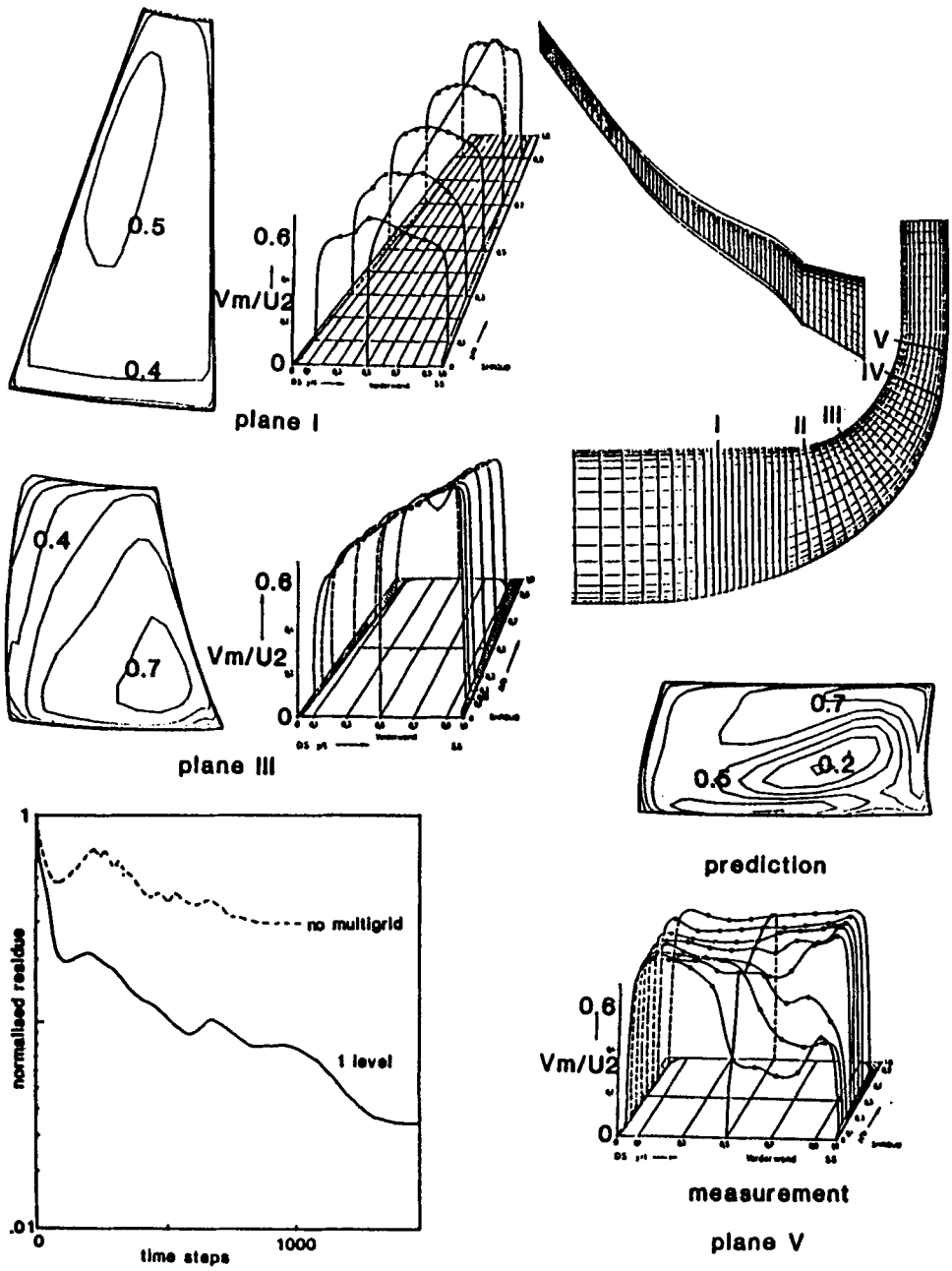

Figure 4. High-speed centrifugal impeller

Transonic flow over an 18% *channel hump*

Figure 1 shows Mach number contours predicted by the present multigrid Navier–Stokes solver together with predicted velocity vectors (showing the TE separation bubble) and a comparison of predicted hump surface pressure coefficient with experiment. The application of the

multigrid methodology is summarized in Table I. The convergence criterion was reduction of the RMS ROVX flux residue to 2.E−06. Use of levels beyond $4h$ gave little extra improvement. Comparing the two mesh densities shows that multigrid allows the convergence rate to be maintained and a solution to be obtained in approximately the same number of steps and with a solution cost nearly linear with mesh size (the aim of multigrid).

Cascade of compressor blades

Figure 2 shows the 67×29 mesh and predicted Mach numbers for this simple generic compressor blade. The convergence history shows dramatic improvement in rate with one level, but little extra improvement with the second level; the mesh becomes too coarse to carry enough meaningful information. The first-level computations demonstrate the theoretical performance: the number of time steps is reduced by a factor of three ($\Delta t + 2\Delta t = 3\Delta t$) with similar reductions in CPU cost.

The second convergence history shows the effect of doubling the mesh density to 131×57. Without multigrid the code performance deteriorates: one level of multigrid improves the rate; two levels improve the rate still further to the same rate achieved on the 67×29 mesh. This confirms that multigrid should allow solutions always to be achieved in the same number of time steps so that the solution cost is simply linear with mesh density.

Secondary flow in a linear cascade of gas turbine rotor blades

Figure 3 shows the predicted development of the secondary flow in a linear cascade. The contours of total pressure show the endwall fluid swept towards the suction side of the passage and rolling up into a passage vortex. The velocity vectors show the extent of the spanwise flow. Predictions were performed using a $17 \times 49 \times 17$ mesh and without multigrid converged in 900 time steps. Using one level of multigrid reduced the number of time steps required to 500.

High-speed centrifugal impeller

The flow in the backswept impeller[9] is complex, being dominated by strong secondary flows leading to the formation of a pronounced jet-wake structure. The predictions shown compared with measurement in Figure 4 were performed on a $17 \times 59 \times 17$ mesh. Without multigrid the convergence rate is very poor. Just one level of multigrid produces a dramatic improvement, making, in fact, a solution feasible. Further levels of multigrid give little additional improvement as the meshes become too coarse to retain any useful flow information. A large part of the success of multigrid here is due to the accelerated establishment of the mass flow in the impeller; this prevents the 'slopping' which severely retards (or even prevents) convergence when not using multigrid.

REFERENCES

1. K. P. Sarathy, 'Computation of three-dimensional flowfields through rotating blade rows and comparison with experiment', *ASME J. Eng. Power*, **104**, 394–402 (1982).
2. B. Baldwin and H. Lomax, 'Thin layer approximation and algebraic model for separated turbulent flows', *AIAA Paper 78-257*, 1978.
3. A. Jameson and T. J. Baker, 'Multigrid solution of the Euler equations for aircraft configurations', *AIAA Paper 84-0093*, 1984.
4. W. N. Dawes, 'A pre-processed implicit algorithm for 3D viscous compressible flow', *Notes on Numerical Fluid Mechanics, Vol. 12*, Vieweg, 1986.
5. W. N. Dawes, 'Application of full Navier–Stokes solvers to turbomachinery flow problems', *VKI Lectures Series 2. Numerical Techniques for Viscous Flow Calculations in Turbomachinery Bladings*, January 1986.

6. A. Brandt, 'Multi-level adaptive solutions to boundary-value problems', *Math. Comput.*, **31**(138), 333–390 (1977).
7. R. H. Ni, 'A multiple-grid scheme for solving the Euler equations', *AIAA Paper 81-1025*, June 1981.
8. R. V. Chima, 'Analysis of inviscid and viscous flows in cascades with an explicit multiple-grid algorithm', *NASA Technical Memorandum 83636*, 1984.
9. D. Eckardt, 'Detailed flow investigations within a high speed centrifugal compressor impeller', *ASME Paper 76-FE-3*, 1976.