Journal of Engineering Mathematics **29**: 11–31, 1995.

Multigrid acceleration of a block structured compressible flow solver

H. KUERTEN and B. GEURTS

Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Received 17 June 1993; accepted in revised form 16 January 1994

Abstract. We study a multiblock method for compressible turbulent flow simulations and present results obtained from calculations on a two-element airfoil. A cell-vertex or vertex-based spatial discretization method and explicit multistage Runge–Kutta time stepping are used. The vertex-based method is found to give better results than the cell-vertex method. In the latter method a larger amount of artificial dissipation is required since different control volumes are used for the discretization of the viscous and convective fluxes. The slow convergence of the time stepping method makes a multigrid acceleration technique indispensable. This technique leads to an acceleration by about a factor of 10. The numerical predictions are in good agreement with experimental results. It is shown that the convergence of the multigrid process depends considerably on the ordering of the various loops. If the block loop is put inside the stage loop the process converges more rapidly than if the block loop is situated outside the stage loop in case a three-stage Runge–Kutta method is used. If a five-stage scheme is adopted the process does not converge in the latter block ordering. Finally, the process based on the five-stage scheme is about 60% more efficient than with the three-stage scheme, if the block loop is inside the stage loop.

1. Introduction

Numerical simulations of turbulent flow in aerodynamic applications are frequently based on the Reynolds-averaged Navier–Stokes equations. One of the main problems in aeronautics is the prediction of flow quantities in complicated geometries, such as the multi-element airfoil (see Fig. 1). Such a configuration is used for the simulation of take-off and landing conditions. The simulation of turbulent flow around such a multi-element airfoil configuration was one of the applications selected for the compressible flow solver which was developed by our group and NLR as a part of the Dutch ISNaS[1] project [1, 2]. For this application the use of a single block, boundary conforming, structured grid is impossible and one may select either an unstructured grid approach or a block structured grid approach. Although the former technique has been successfully applied by others [3], we selected the block structured approach in view of the transparent data structure in the coding, ease of implementation of the turbulence model and a high flexibility with respect to the use of different physical models in different parts of the computational domain. Moreover, a block structure seems to be especially transparent for a possible parallel processing.

In a previous paper [4] the numerical method has been described and the block structured approach was introduced. Application of the resulting multiblock method to compressible

Fig. 1. Geometry of a two-element airfoil, consisting of the NLR-7301 profile plus a flap with a gap width of 2.6% chord length.

flow around a single airfoil enabled a comparison with a monoblock method. It appeared that for both laminar and turbulent flow around a single airfoil the introduction of the multiblock structure has no influence on the results, with respect to both the steady-state solution and the convergence rate. Furthermore, invoking the Euler equations instead of the Navier–Stokes equations in blocks outside the boundary layer and wake appeared to have no significant influence on the results. In this paper we describe the application of the multiblock concept to the multi-element airfoil.

First in Section 2 a short description of the numerical method and the multiblock algorithm will be given. For the simulation of subsonic, turbulent flow around a two-element airfoil two drawbacks of this method become apparent. In the first place the cell-vertex spatial discretization method which was adopted gives rise to a rather high level of artificial dissipation in the boundary layers and leads for the present application to oscillations in the numerical solution. In the second place the rate of convergence of the method is unacceptably low. In order to overcome these problems an alternative discretization method, which allows for a lower level of artificial dissipation, and a multigrid method as a convergence acceleration technique are presented in this paper. As a point of reference the performance of the multiblock solver for turbulent flow around a two-element airfoil is studied in Section 3, illustrating these two problems. Improvements upon the method are introduced and analyzed in Section 4.

The resulting multigrid multiblock method is first applied to the simulation of inviscid flow in Section 5 showing that an acceptable rate of convergence towards the steady-state solution is obtained. It is further demonstrated that the specific implementation of the solid wall boundary condition for the pressure has a significant effect on the solution. We proceed in Section 5 by applying the method to simulate viscous turbulent flow. The Reynolds-averaged Navier–Stokes equations are solved in the blocks situated in shear layers and the Euler equations in the 'outer' blocks. Although the convergence is not as fast as for monoblock simulations of transonic flow, the multigrid technique yields an appreciable acceleration. The results are compared with wind-tunnel measurements, showing a good agreement for both the pressure coefficient on the airfoil and the displacement thickness. Moreover, the oscillations in the numerical solution which were found with the cell-vertex method, are strongly reduced by the alternative discretization. Finally, in Section 5 several investigations with respect to the possibilities of parallel processing are presented. It appears that a rather high amount of data transfer between the blocks is necessary in order to retain the high convergence rate. This result is in contrast with the results of the single grid simulations reported in [4], where the solutions in different blocks could be independently updated over a large number of time steps without affecting the rate of convergence. It illustrates the effectiveness of a multigrid method in transporting signals through the computational domain through the use of coarser grids. We summarize our findings in Section 6.

2. Numerical method

The numerical method has been described in reference [4], but will briefly be recapitulated here for the sake of completeness. The two-dimensional, compressible Navier–Stokes equations can be written in integral form as

$$\frac{\partial}{\partial t}\left[\int\int_{\Omega} U \,\mathrm{d}x\,\mathrm{d}y\right] + \int_{\partial\Omega} (F\,\mathrm{d}y - G\,\mathrm{d}x) = 0\,, \tag{1}$$

where U represents the vector of dependent variables,

$$U = [\rho, \rho u, \rho v, E]^T , \tag{2}$$

with ρ the density, u and v the Cartesian velocity components, and E the total energy density. Further, Ω is an arbitrary part of the two-dimensional space with boundary $\partial\Omega$ and F and G are the Cartesian components of the total flux vector. This flux vector consists of two parts: the non-dissipative or 'convective' part and the dissipative or 'viscous' part, which describes the effects of viscosity and heat conduction, and involves first order spatial derivatives. The Navier–Stokes equations (1) are averaged over a sufficiently large time interval. Due to the nonlinear terms in the convective fluxes, the resulting 'Reynolds-averaged Navier–Stokes' equations involve averages of products of two velocity components. These terms are modeled by a suitable turbulence model. In the present paper the algebraic Baldwin–Lomax turbulence model, in which the unknown terms are modeled by eddy viscosity terms, is adopted [5].

The spatial discretization of the Navier–Stokes equations is based on a cell-vertex finite volume method. The variables are stored in the grid points and as a control volume for the convective fluxes the union of four neighboring grid cells is taken. For the viscous fluxes the control volume is the area between four neighboring cell centers, as indicated in Fig. 2. The integrations over the cell face, as required in the formulation in equation (1), are performed using the trapezoidal rule.

In order to prevent odd-even decoupling and to capture possible shock waves nonlinear artificial dissipation is added to the basic numerical scheme. This artificial dissipation consists of two contributions: fourth order difference terms which prevent odd-even decoupling, and second order difference terms to resolve shock waves. The second order terms are controlled by a shock sensor, which detects discontinuities in the pressure. In the present flow solver the artificial dissipation in the boundary layers, where the effects of the viscous dissipation should be dominant, may be reduced by multiplication with the ratio of the local and free stream Mach number. The role of the artificial dissipation in relation to the viscous dissipation is discussed in more detail in reference [6].

At the solid wall boundaries the no-slip adiabatic wall condition is used for viscous simulations and the impermeability condition for inviscid simulations. For viscous simulations the density and energy density in the grid points on a solid wall are calculated by

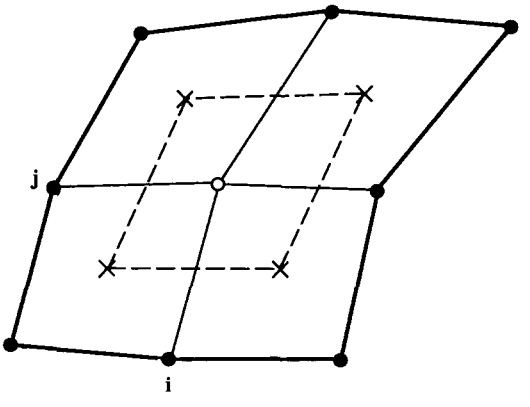

Fig. 2. Control volume for the vertex-based method and for the viscous fluxes in the cell-vertex method (dashed) and for the convective fluxes in the cell-vertex method (solid).

solving the corresponding discrete conservation laws, using the two adjacent cells within the computational domain and their mirror images inside the wall as control volume. The values of the density and energy density in the grid points inside the walls are adjusted such that the adiabatic wall condition is approximated. For inviscid simulations the tangential velocity component, the pressure and the density are extrapolated from inside the computational domain. The boundary conditions at a (subsonic) far field boundary are based on characteristic theory.

The system of ordinary differential equations, which results after spatial discretization, is integrated in time using a time explicit multistage Runge–Kutta method. In the present flow solver a three-stage scheme in which the dissipative fluxes (both viscous and artificial) are calculated once per time step, and a five-stage scheme in which the dissipative terms are calculated only at the odd stages, are implemented. With this treatment both calculation time is saved and the stability region of the method is increased. Extra calculation time is saved by advancing each grid point at the maximum local time step according to its own stability limit. In this way the evolution from the initial solution to the steady-state is no longer time accurate, but the steady-state solution obtained is unaffected.

In the multiblock solver the total computational domain is divided into blocks. We restrict ourselves to grids with continuous grid lines over block interfaces and one boundary condition per edge or vertex. Each block is 'dressed' with two rows of dummies near each of its four edges. In this way the spatial discretization scheme can be applied to each point of a block, including the boundary points. The dummy variables corresponding to block interfaces are updated through copying from the neighboring blocks after all blocks have been updated. Depending on the loop ordering this treatment of the dummy variables is carried out after every stage of the Runge–Kutta scheme, or after a whole time step. During each stage the spatial discretization scheme is applied to all interior points, and subsequently the local boundary conditions are applied to the points on the edges and vertices. These can be solid wall boundary conditions, far field boundary conditions or the application of the spatial discretization scheme in case the point is on a block interface. Hence, the points on block interfaces are treated more than once during one stage. The possible resulting multi-valuedness is removed by averaging. The averaging procedure is executed simultaneously with the update of the dummy variables, i.e. either after every stage of the Runge–Kutta scheme, or after a whole time step.

Due to the topology of the two-element airfoil geometry special points in the computational grid are unavoidable. The computational grids used contain two special points at block boundaries, where five cells meet, in contrast to the usual four. These points can be treated in an elegant way within the same numerical scheme, if the dummy vertices outside the 'current' block are defined appropriately. This is sketched in Fig. 3. Notice that the control volume corresponding to a special point is not unique and depends on which block is currently being treated. The resulting multi-valuedness of the variables at the special point is eliminated by taking the average of the five different values after all blocks have been treated.

Another special feature connected to the two-element airfoil geometry is the definition of the turbulent viscosity. In the standard Baldwin–Lomax turbulence model different formulations are used for boundary layers and wakes, which are matched with a suitable smoother in the trailing edge region of an airfoil. The multi-element airfoil geometry requires a further extension of the turbulence model, since there are regions in the flow which are situated in both the wake of the main airfoil and the boundary layer of the flap. For these regions the Baldwin–Lomax model yields two values for the turbulent viscosity, one stemming from the

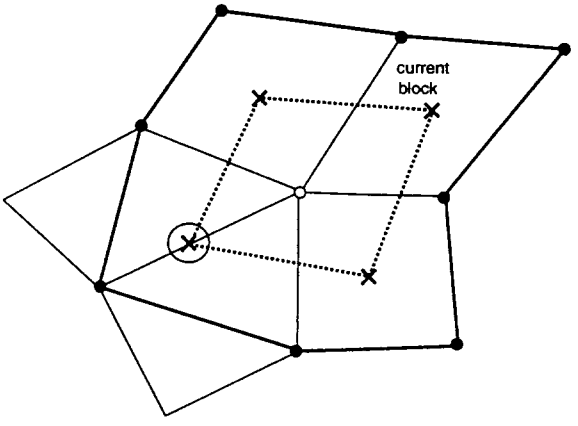

Fig. 3. Control volume for a special point; vertex-based method and viscous flux in cell-vertex method (dashed); convective flux in cell-vertex method (solid).

wake formulation applied to the main airfoil and the other stemming from the boundary layer formulation of the flap. The resulting turbulent viscosity is averaged in the following way [7]

$$\mu_t = \frac{\mu_{t_1} y_2^2 + \mu_{t_2} y_1^2}{y_1^2 + y_2^2}, \tag{3}$$

where μ_{t_i} is the turbulent viscosity according to one of the formulations, and y_i the distance to the wake center line or profile. In regions which are in the boundary layer of both parts of the airfoil, or in the wake of both parts, an analogous treatment is used.

After this description of the numerical method and the multiblock algorithm, we consider its application to the simulation of compressible turbulent flow around a two-element airfoil.

3. Application of the flow solver to the multi-element airfoil

We will present results for a two component airfoil geometry consisting of the NLR-7301 wing section, from which a flap has been cut out at a deflection angle of 20° and with a gap width of 2.6% chord length [8] (see Fig. 1). The combination of a Mach number of 0.185 and an angle of incidence of 6° or 13.1°, of which the latter is close to maximum lift conditions, yields subsonic flow. The Reynolds number based on the chord length of the airfoil is 2.51×10^6. In the viscous calculations the locations of the transition from laminar to turbulent flow are prescribed at positions obtained from experiments.

The C-type computational grids (either for inviscid or viscous flow) were constructed by J.J. Benton from British Aerospace, and are subdivided in 37 blocks (see Fig. 4). The grid lines are continuous over block boundaries. Two grids are used: one 'Euler' grid (inviscid) consisting of 16448 cells, and a 'Navier–Stokes' grid (viscous), which is refined in the boundary layers and wakes and consists of 28288 cells. Fig. 4 shows that the two special points, where five blocks meet, are located well outside the boundary layers.

For both angles of incidence results from wind-tunnel measurement by Van den Berg [8] are available, including velocity profiles in the boundary layers and the pressure coefficient on the profile. Since the flow is attached apart from a small laminar separation bubble near

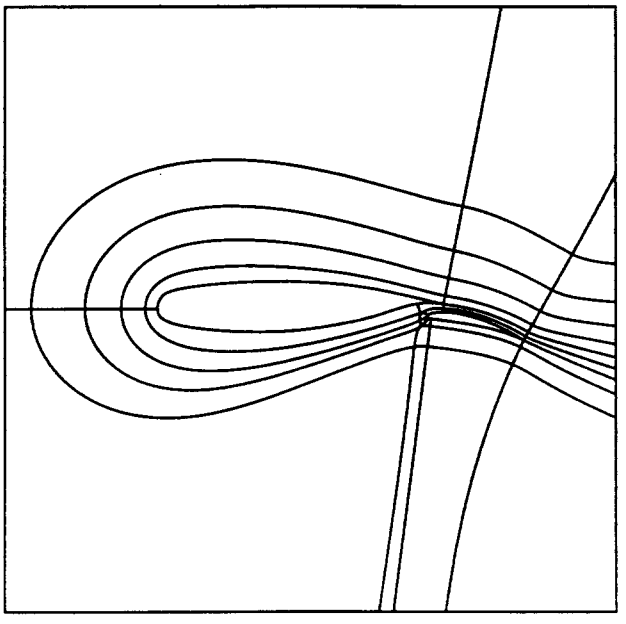

Fig. 4. Block structure of the computational grid for the two-element airfoil, generated by J.J. Benton from British Aerospace.

the leading edge of the airfoil, the adopted turbulence model should be adequate and yield a useful comparison between experiment and calculation.

We consider as an example the steady-state solution for turbulent flow at an angle of incidence of 6°. The above multiblock algorithm is used with a three-stage Runge–Kutta scheme, and with the stage loop inside the block loop, i.e. the same loop ordering as used in [4]. It is checked that the steady-state solution is unchanged if in the outer blocks the inviscid Euler equations are solved instead of the Reynolds-averaged Navier–Stokes equations. This enables a substantial reduction of calculation time. The calculated steady-state pressure coefficient on the airfoil and flap is compared with the measurements in Fig. 5. The overall agreement between the numerical and experimental results is quite reasonable. However, near the trailing edge of the airfoil and flap and on the upper side of the flap oscillations in the numerical solution are visible. In Fig. 6, where an enlargement of the pressure coefficient on the flap is plotted, this phenomenon is shown in more detail.

Apart from this inadequacy in the solution another disadvantage of the present multiblock algorithm is the slow convergence rate. Over 50,000 time steps are needed to decrease the residuals by only four orders of magnitude. In the following sections we will introduce an alternative spatial discretization method, replacing the cell-vertex method, and a multigrid acceleration technique in order to substantially improve the multiblock method with respect to these two points.

4. Improved algorithm

In this section methods are presented which remove the disadvantages mentioned in the previous section, i.e. the occurrence of oscillations in the numerical solution and the low rate of convergence.

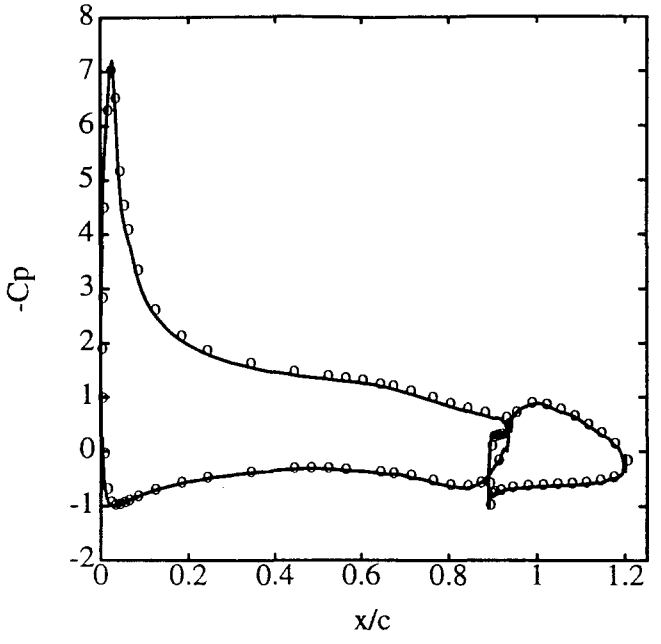

Fig. 5. Comparison between the calculated (solid) and experimental (circles) pressure coefficient on the airfoil and flap for viscous flow at an angle of 6.0°; cell-vertex method.

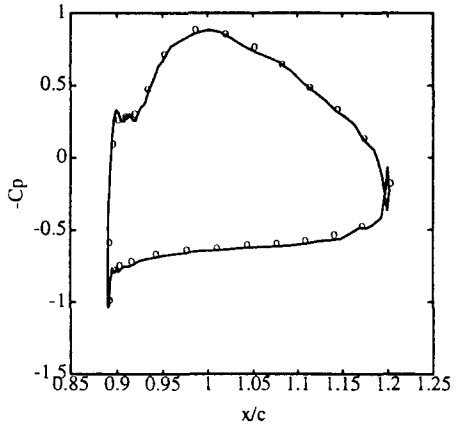

Fig. 6. Comparison between the calculated (solid) and experimental (circles) pressure coefficient on the flap for viscous flow at an angle of 6.0°; cell-vertex method.

4.1. Vertex-based algorithm

It has been reported in the literature that the cell-vertex discretization method with central differencing and nonlinear artificial dissipation leads to a rather high amount of dissipation in boundary layers [6, 9]. After spatial discretization the Navier–Stokes equations can schematically be written as

$$\frac{\mathrm{d}U_{i,j}}{\mathrm{d}t} = f_{i,j}^{(c)} + f_{i,j}^{(v)} + f_{i,j}^{(a)}, \tag{4}$$

where $U_{i,j}$ is the vector of dependent variables in a grid point $\{i, j\}$, $f^{(c)}$ is the convective

flux, $f^{(v)}$ is the convective flux, $f^{(v)}$ the viscous flux and $f^{(a)}$ the artificial dissipation. The latter consists of contributions from both spatial directions, each of which has the form

$$d_{i+1/2} - d_{i-1/2}\,, \tag{5}$$

where

$$d_{i+1/2} = S_{i+1/2}[\varepsilon^{(2)}_{i+1/2}\Delta_{i+1/2}U - \varepsilon^{(4)}_{i+1/2}\Delta^3_{i+1/2}U]\,. \tag{6}$$

Here, $S_{i+1/2}$ is a scaling factor, $\varepsilon^{(2)}_{i+1/2}$ and $\varepsilon^{(4)}_{i+1/2}$ are functions of a shock sensor, and $\Delta_{i+1/2}$ and $\Delta^3_{i+1/2}$ are first and third order difference operators. A natural choice for the scaling factor $S_{i+1/2}$ is [10]

$$S_{i+1/2} = \frac{1}{2}[S_i + S_{i+1}]\,, \tag{7}$$

where

$$S_i = \frac{M_{i,j}}{M_\infty}\lambda^{(i)}_{i,j}\,, \tag{8}$$

and $\lambda^{(i)}_{i,j}$ is the maximum eigenvalue of the flux Jacobian matrix in the i-direction. Furthermore, the factor $M_{i,j}/M_\infty$, where M is the local Mach number reduces the artificial dissipation in the boundary layer, which is necessary for simulations at high Reynolds number to ensure the dominance of the viscous flux effects over the artificial dissipation in the shear layers.

In the cell-vertex scheme the discrete convective flux is based on a control volume which differs from that used for the viscous flux and artificial dissipation. The latter terms both use the area between the dashed lines in Fig. 2 as a control volume. Especially in high aspect ratio grid cells this treatment of the convective flux requires an extra amount of artificial dissipation in order to arrive at a stable method. This can be achieved with an increase of the scaling factor $S_{i+1/2}$ by replacing equation (8) by

$$S_i = \frac{M_{i,j}}{M_\infty}[1 + (\lambda^{(j)}_{i,j}/\lambda^{(i)}_{i,j})^{2/3}]\lambda^{(i)}_{i,j}\,, \tag{9}$$

where $\lambda^{(j)}_{i,j}$ is the maximum eigenvalue of the flux Jacobian matrix in the j-direction. The term between the brackets increases the artificial dissipation in the streamwise direction in the boundary layer (see e.g. [11, 14]).

An alternative to the cell-vertex method is the vertex-based method, where the control volume for the convective flux is the same as for the other two fluxes. The convective flux through an edge of this control volume is then integrated with the midpoint rule, and the value of the flux vector at the midpoint follows as the average over the two neighboring grid points. The better consistency of this discretization method enables a reduction in artificial dissipation through the use of the scaling factor (8) instead of (9) (see e.g. [6, 9]).

4.2. Multigrid acceleration

The second disadvantage of the multiblock algorithm is the low rate of convergence, which is even further decreased by the reduction of the artificial dissipation proposed above, i.e. when considering a vertex-based method instead of the cell-vertex method. This problem of

slow convergence is well-known for Runge–Kutta schemes, which have the property that the high frequency components in the error are damped faster than the low frequency components.

We will show this in the following way. Since we want to calculate the steady state solution of the Navier–Stokesokes equations, it is required that the error, defined as

$$V(\mathbf{x}, t) = U(\mathbf{x}, t) - U_0(\mathbf{x}) , \tag{10}$$

where $U(\mathbf{x}, t)$ is the time-dependent solution and $U_0(\mathbf{x})$ is the desired steady state solution, decreases sufficiently fast. Substitution of the definition of V in the Navier–Stokes equations and linearization leads to an evolution equation for $V(\mathbf{x}, t)$ of the form

$$V_t + AV_x + BV_y = CV_{xx} + DV_{xy} + EV_{yy}, \tag{11}$$

where the matrices A, B, C, D and E depend on U_0. In particular, the dominant part of A and B consists of the flux Jacobian matrices of the Euler equations in the two spatial directions and the other three matrices are proportional to the viscosity. This implies that the eigenvalues of A are approximately equal to the eigenvalues of the inviscid flux Jacobian matrix, i.e. equal to u and $u \pm c$, where u is the velocity component in the x-direction and c the speed of sound.

The effects on the error resulting from an application of Runge–Kutta schemes to equation (11) can be demonstrated using the one-dimensional scalar model equation

$$v_t + av_x = \mu v_{xx} , \tag{12}$$

where a is the constant convection velocity and μ the constant viscosity. The behavior of the error v as a function of t can be studied by a stability analysis. To this end the equation is first discretized in space and the evolution of a particular Fourier mode in time is calculated. The amplification factor is defined as the ratio of the amplitudes of this Fourier mode after and before one time step. It is a function of the wave number of the Fourier mode, and directly shows the decrease of the various components contained in the error during the integration in time of equation (12).

A typical result is presented in Fig. 7. There the amplification factor after 10 Runge–Kutta time steps is plotted as a function of the wave number for the two different time integration schemes used in this paper. These results are obtained with two different values for a. One of them corresponds with the maximum eigenvalue of the flux Jacobian matrix, the other with a value which is smaller by a factor of 10. In the low Mach number simulations presented in this paper the maximum and minimum eigenvalue of the flux Jacobian matrix may well differ by a factor of 10. Hence, it is required that the time integration scheme effectively reduces the error for both values of a.

Fig. 7 shows that the high frequency components, with $\theta > \pi/2$ are sufficiently damped for both time integration schemes and both values of the convection speed. However, the damping of the low frequency modes is considerably less and this behavior is more prominent for the combination of the three-stage Runge–Kutta scheme with the small convection speed.

A solution to this problem, which has often been reported in the literature [6, 9, 11] is the use of a multigrid technique. The idea behind this technique can be explained by the following two-grid model. The high frequency modes are damped on the original, fine grid, as shown above. Hence, after several time steps on the fine grid (so-called pre-relaxations),

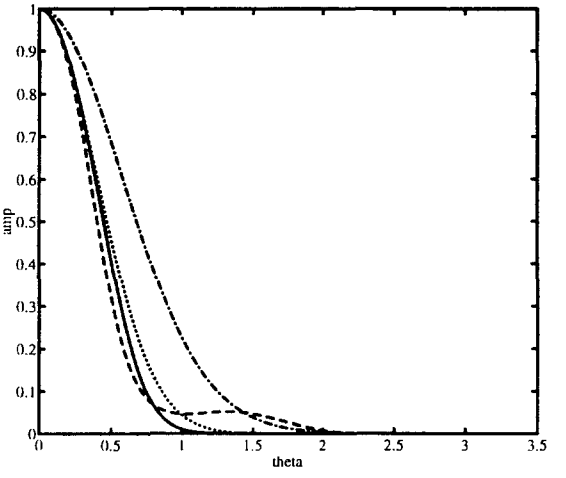

Fig. 7. Amplification factor of Fourier modes after 10 time steps for a one-dimensional model problem; three-stage scheme and large a (solid); five-stage scheme and large a (dashed); three-stage scheme and small a (dash-dotted); five-stage scheme and small a (dotted).

the error mainly consists of low frequency components. Then the solution and error are transferred to a coarser grid, e.g. a grid with half the number of grid points in both directions. Through the transfer to this coarser grid the wave numbers of the error are multiplied by two. Hence, on the coarser grid more Fourier modes of the error can effectively be damped by application of the time integration scheme. After several time steps on this grid the solution is transferred again to the original fine grid. This prolongation introduces some high frequency error, which can be reduced by some additional time steps (so-called post-relaxations) on the fine grid.

The effect of the two-grid model is shown in Fig. 8, where the amplification factors for the same four examples as shown in Fig. 7 are plotted. In the calculation five pre- and post-relaxations and 10 time-steps on the coarse grid have been used. The figure clearly

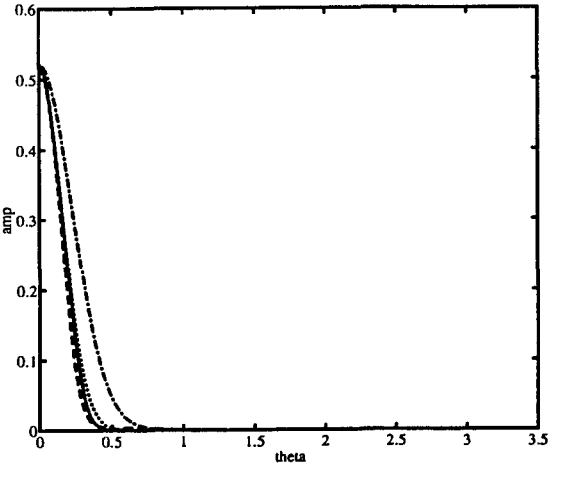

Fig. 8. Amplification factor of Fourier modes after one V-cycle on two grids for a one-dimensional model problem; three-stage scheme and large a (solid); five-stage scheme and large a (dashed); three-stage scheme and small a (dash-dotted); five-stage scheme and small a (dotted).

shows the beneficial effects of the two-grid model, as all components of the error with $\theta > \pi/6$ have almost completely vanished. As the computational effort of a time-step on the coarse grid is less than on a fine grid, a single-grid calculation cannot result in a comparable error reduction within the same calculation time.

In the multigrid technique a sequence of grids is defined and the two-grid model is successively applied till the coarsest grid is reached. Since this grid has a strongly reduced number of grid points, the low frequency components of the error can sufficiently be decreased with a small computational effort.

The multigrid technique applied in this paper can only be used for the calculation of steady state solutions. Consider the general discrete equation to explain the nonlinear multigrid technique employed here

$$N^l(U^l) = R^l \,, \tag{13}$$

where U^l represents the solution vector at the original fine grid l, N^l is the spatial discretization of the second term in equation (1) on the same grid level, and R^l is a possible right hand side, which equals zero on the finest grid. Using the time stepping method as described above a fixed number of pre-relaxations is performed on this grid, resulting in an approximate solution $\tilde{U}^l$ of equation (13). The defect vector, which is a measure for the error, is defined as

$$D^l = N^l(\tilde{U}^l) - R^l \equiv N^l(\tilde{U}^l) - N^l(\tilde{U}^l + V^l) \,, \tag{14}$$

where V^l is the (unknown) error on the fine grid.

As this defect mainly contains low frequency modes, it can accurately be represented on a coarser grid L. Thus we write

$$D^L = N^L(\tilde{U}^L) - N^L(\tilde{U}^L + V^L) \,, \tag{15}$$

where N^L is the spatial discretization of the nonlinear operator in equation (1) on the coarse grid, and D^L, $\tilde{U}^L$ and V^L represent the restrictions of the defect vector, solution vector and error to the coarse grid. We can reformulate this equation as

$$N^L(W^L) = R^L \,, \tag{16}$$

where

$$R^L = N^L(\tilde{U}^L) - D^L \,, \tag{17}$$

which we want to solve for W^L $(= \tilde{U}^L + V^L)$, thus yielding an approximation for the desired error V^L on the coarse grid. As the structure of the equation is the same as equation (13), this solution can be approximated by applying a fixed number of time steps on this coarse grid, using the restriction of the solution $\tilde{U}^l$ as initial solution. This leads to an approximate solution $\tilde{W}^L$. Next, the correction to the solution, $\tilde{W}^L - \tilde{U}^L$, is prolonged to the fine grid and added to the approximate solution $\tilde{U}^l$, and starting from this corrected solution a fixed number of post-relaxations of equation (13) is performed.

In the flow solver described in this paper the solution is restricted to the coarser grid by injection, and the defect vector by full weighting, i.e.

$$U^L_{i,j} = U^l_{2i,2j}\,, \tag{18}$$

and

$$D^L_{i,j} = D^l_{2i,2j} + \frac{1}{2}(D^l_{2i+1,2j} + D^l_{2i,2j+1} + D^l_{2i-1,2j} + D^l_{2i,2j-1}) + \frac{1}{4}(D^l_{2i+1,2j+1} + D^l_{2i-1,2j+1} + D^l_{2i-1,2j-1} + D^l_{2i+1,2j-1})\,. \tag{19}$$

The correction to the solution, C, is prolonged to the finer grid by means of bilinear interpolation, i.e.

$$\begin{aligned} C^l_{2i,2j} &= C^L_{i,j} \\ C^l_{2i+1,j} &= \frac{1}{2}(C^L_{i,j} + C^L_{i+1,j}) \\ C^l_{2i+1,2j+1} &= \frac{1}{4}(C^L_{i,j} + C^L_{i+1,j} + C^L_{i+1,j+1} + C^L_{i,j+1})\,. \end{aligned} \tag{20}$$

In this solver an initial solution on the finest grid is obtained with a full multigrid method. During the nonlinear multigrid phase either V- or W-cycles can be chosen. In order to increase the smoothing properties of the Runge–Kutta time-stepping technique an implicit averaging of the residuals can be applied with frozen residuals at the boundaries of each block.

Details on multigrid methods for nonlinear partial differential equations can be found for example in [12, 13]. For monoblock applications this method has given satisfactory results for both two-dimensional and three-dimensional flows [6, 14].

In the multi-element airfoil application care has to be taken in the definition of the residual vector in the special points. The proposed treatment of a special point implies that the control volume is different in each of the five blocks where such a point is found. In the required averaging the five residual vectors in a special point are weighed with their corresponding time steps. Without this weighing the multigrid process cannot converge to the single grid steady-state solution.

There are various possibilities for intertwining the different loops in this multigrid, multiblock solver with a multistage time stepping method. In the present study the grid loop is chosen as the outer loop and the effect of interchanging the block and the stage loop will be studied. Several 'competing' requirements serve as possible guidance for selecting a specific ordering of these loops. On the one hand an anticipated parallel processing of the different blocks is more efficient if the data transfer between the blocks is kept to a minimum, i.e. with the stage loop inside the block loop. On the other hand the good convergence of the multigrid monoblock solver may be reduced as the dummy variables near the block boundaries are kept frozen during more stages of the time step. This would suggest to put the block loop inside the stage loop. In order to study this dilemma we implemented these two loop orderings in a flexible way: a single parameter determines whether the block loop is situated inside or outside the stage loop.

In the next section the results of the vertex-based method in combination with multigrid acceleration are presented for both inviscid and viscous flow. For inviscid flow the effect of two different implementations of the numerical boundary conditions is studied. For viscous flow the effects of different loop arrangements on the convergence rate are shown.

Furthermore, the speed-up due to the multigrid method is calculated in terms of both number of operations and calculation time.

5. Results

5.1. Inviscid flow

In this section we consider the relatively simple inviscid flow case, where in all blocks the Euler equations are solved. This allows to separate problems related to the turbulence model from possible algorithmic problems.

In Fig. 9 the multigrid convergence behavior of the solver in the 13.1° case is shown. The discrete L_2-norm of the residual of the density is plotted as a function of the number of W-cycles. The calculation is performed with the block loop inside the stage loop, i.e. with the dummy variables updated after every stage, and using the five-stage Runge–Kutta scheme. A converged solution is obtained within a much smaller calculation time when compared to the single grid approach even though only three different grid levels are available. Both for the single grid and the multigrid calculations the same solution converged up to machine accuracy is obtained. This would not be possible without the special weighing procedure of the residual vectors in a special point. The specific block structure nor the treatment of the special points leads to any algorithmic difficulties. For this inviscid test a comparison with experimental results is not meaningful and will not be made. A result which can be checked, however, is the predicted value of the drag coefficient, which should theoretically be equal to zero, due to the fact that no shock waves are present in this subsonic flow. The artificial dissipation and the numerical boundary conditions at the airfoil, however, lead to a numerical boundary layer and hence to a positive drag coefficient. The implementation of the numerical boundary condition for the pressure is found to be especially critical. If the tangential velocity component and the density at the airfoil are linearly extrapolated and a zeroth order implementation of zero normal pressure gradient is used, the drag coefficient is predicted equal to 0.0914, which is a rather high value. If, however, linear extrapolation is also used for the pressure, the drag coefficient is reduced to

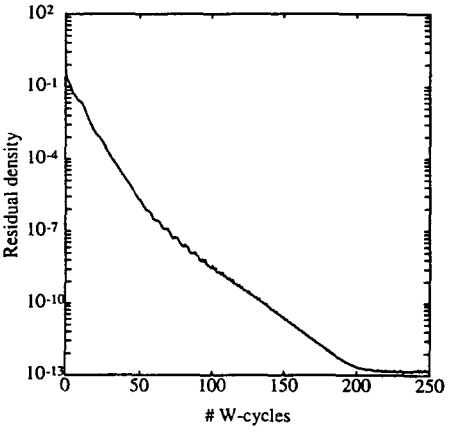

Fig. 9. Convergence behavior for inviscid flow at an angle of incidence of 13.1°; vertex-based method.

the more acceptable value of 0.0423. This change also leads to a slight increase in the convergence rate.

5.2. Viscous flow

We consider the simulations of turbulent, viscous flow and present results for the 6° case, which was already shown in Section 3. In a simulation of turbulent flow at high Reynolds number it is important that the effects related to the physical dissipation are not outweighed by those of the numerical or artificial dissipation. This requirement could give rise to difficulties in the present multigrid method, since the time stepping method used requires a certain minimum amount of dissipation for sufficient smoothing of the large wave number components of the error (see reference [6]). If the artificial dissipation in the boundary layer is reduced by scaling with the ratio of the local and free stream Mach number, i.e. decreasing the smoothing properties of the time stepping method, a converged solution (engineering accuracy) could be obtained by increasing the number of pre- and post-relaxations. The convergence behavior of this calculation during the FAS stage is shown in Fig. 10, where the discrete L_2-norm of the residual of the density is plotted as a function of the number of W-cycles. In the blocks outside the boundary layers and wakes the Euler equations are solved instead of the Navier–Stokes equations. The same loop ordering and Runge–Kutta scheme as in the inviscid calculation described above are applied.

In this calculation implicit residual averaging is applied as well. Although this averaging increases the convergence rate measured in number of W-cycles, the calculation time necessary to obtain the steady-state solution is almost unchanged. There are two reasons for this. Firstly, the computational cost of the inversions of the tridiagonal matrices in the implicit residual averaging is quite high. Secondly, due to the fact that the residuals on the block boundaries are frozen in the averaging process the CFL number cannot be increased as much as in a monoblock simulation.

The good agreement with the wind-tunnel measurements can be inferred from Fig. 11, where the experimental and numerically predicted pressure coefficient on the airfoil and flap are shown. The oscillations near the trailing edges and on the flap, which were found with the cell-vertex discretization method, are strongly reduced in the vertex-based method, as can be seen more clearly in Fig. 12 where an enlargement of Fig. 11 in the flap region is

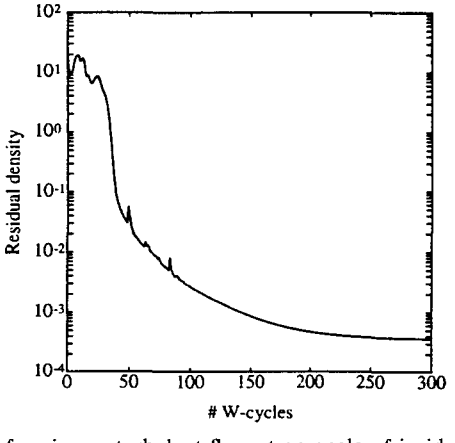

Fig. 10. Convergence behavior for viscous turbulent flow at an angle of incidence of 6.0°; vertex-based method.

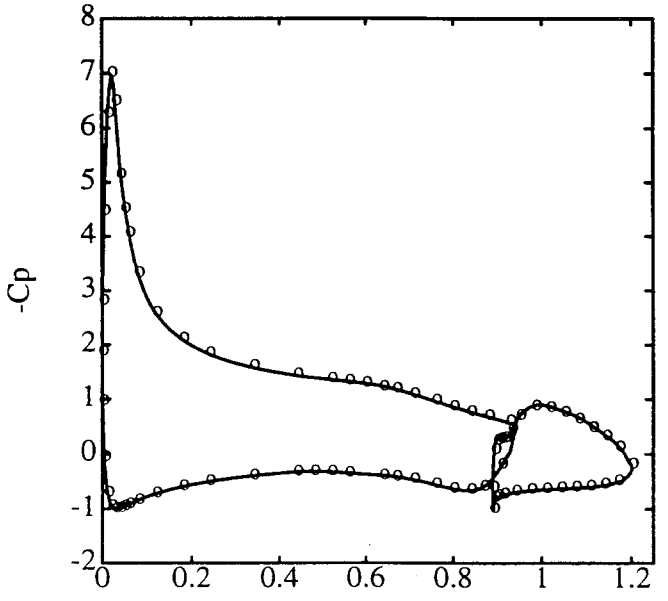

Fig. 11. Comparison between the calculated (solid) and experimental (circles) pressure coefficient on the airfoil and flap for viscous flow at an angle of 6.0°; vertex-based method.

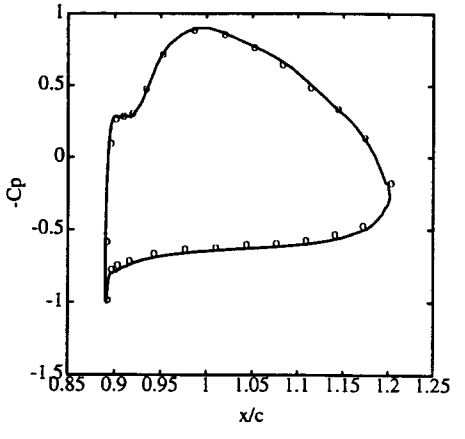

Fig. 12. Comparison between the calculated (solid) and experimental (circles) pressure coefficient on the flap for viscous flow at an angle of 6.0°; vertex-based method.

drawn. The main motivation for the replacement of the cell-vertex method by the vertex-based method was that the latter requires a lower amount of artificial dissipation and hence corresponds better with the physics, as pointed out in references [6] and [9]. The observation that the drag coefficient decreases from 0.0378 in the cell-vertex method to 0.0360 in the vertex-based method shows that the artificial dissipation is indeed lower. However in both methods the drag coefficient is overestimated, the experimental value being 0.0229 [8]. A similar overestimation has been reported by Larsson [15] for the same test case. There are several possible explanations for this discrepancy. Firstly, the experimental drag coefficient is deduced from the total-pressure defect and the static pressure measured in the wake, whereas the numerically predicted value follows from a surface pressure integration. Secondly, the predicted drag coefficient for inviscid flow simulations shows that the method

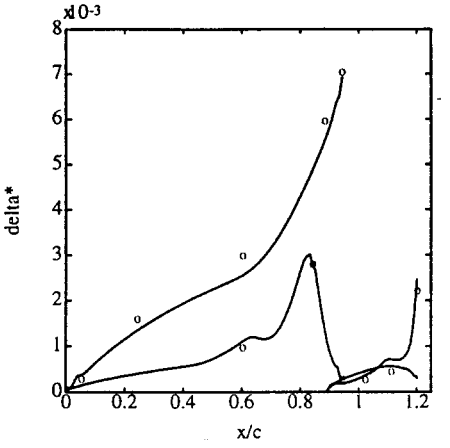

Fig. 13. Comparison between the calculated (solid) and experimental (circles) displacement thickness for viscous flow at an angle of 6.0°; vertex-based method.

generates a numerical boundary layer, which might still be present in the finer Navier–Stokes grid. This effect could be studied in more detail by local grid refinement. The lift coefficient predicted with the vertex-based method, 2.448, corresponds better with the experimental value of 2.416 [8].

The comparison between the measured and numerically predicted displacement thickness (δ^*) is shown in Fig. 13. The displacement thickness is a sensitive measure for the assertion that the artificial dissipation in the boundary layer does not outweigh the physical dissipation. The good agreement shown in Fig. 13 indicates that the scaling of the artificial dissipation with the local Mach number leads to an accurate representation of the flow in the regions close to the airfoil and flap.

5.3. Different loop arrangements

This solution was obtained with the block loop inside the stage loop of the five-stage Runge–Kutta time stepping method. Hence, the variables at the dummy vertices outside a block are updated after every stage, which implies that the effects of the multiblock structure on the convergence are kept to a minimum. The high frequency of data transfer between the blocks makes this method less efficient for parallel processing. However, with the block loop outside the stage loop, i.e. with an update of the dummy variables only after five flux evaluations, a converged solution could not be obtained. Apparently, the interval between two moments of data transfer between the blocks has to be sufficiently small in order to obtain a convergent multigrid method.

Further evidence for this statement is found from calculations with a three-stage instead of a five-stage Runge–Kutta time stepping method. If the block loop is outside the stage loop, the dummy variables are updated after three flux evaluations. Although the rate of convergence is lower than in the case with the block loop inside the stage loop, i.e. maximum data transfer between the blocks (see Fig. 14), the solution has converged to within engineering accuracy after ≈ 200 W-cycles. A comparison of the three-stage and five-stage schemes with the block loop inside the stage loop shows that the five-stage scheme is more efficient: about 60 W-cycles suffice to reduce the residuals to the same engineering level as with the three-stage scheme after 200 W-cycles, which can be inferred from a

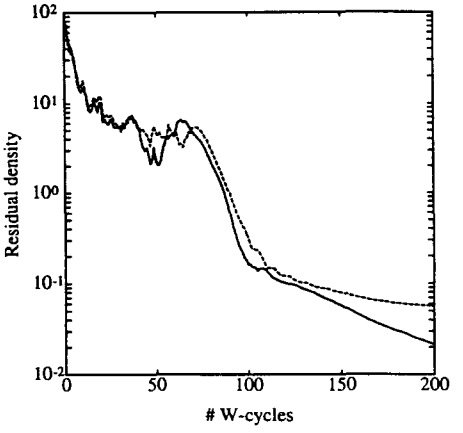

Fig. 14. Convergence behavior of the three-stage Runge–Kutta scheme for turbulent flow; comparison between block loop inside (solid) and outside (dashed) stage loop.

comparison of Figs. 10 and 14. The five-stage scheme leads to a reduction in calculation time of approximately 60% in this instance. The better convergence of the five-stage scheme in a multigrid method could already be inferred from Fig 8.

5.4. Multigrid speed-up

Although both for inviscid and for viscous turbulent flow simulations converged results have been found, the performance of the multigrid multiblock flow solver is not as good as that of the corresponding monoblock solver. Firstly, the required number of pre- and post-relaxations is about 10, which is quite high. Secondly, the speed-up on a vector computer is lower than usual.

For monoblock simulations of transonic turbulent flow around an airfoil or wing-alone a converged solution to within engineering accuracy could be obtained after approximately 50 W-cycles [6, 9, 11], where in each W-cycle a smaller number of pre- and post-relaxations was applied. In the simulations reported here approximately twice as many fine-grid relaxations are necessary to reach the same level of convergence. There are several reasons for this increased computational effort. Firstly, in the present grid only three grid levels are available. This is caused by the relatively small number of grid points per block. Secondly, since the residuals at block boundaries are frozen during residual averaging, its beneficial effect on the convergence is reduced. Thirdly, the present simulations are performed at a Mach number of 0.185, whereas the transonic simulations reported in reference [6] were performed at a Mach number on the order of 0.8. This implies that in the present simulations the ratio of the minimum and maximum propagation speed of the characteristic waves, $u/(u + c)$, where u is the fluid velocity and c the speed of sound, is rather small. Hence, the system of equations is quite stiff, which makes the Runge–Kutta time stepping method less effective. A preconditioning method, like the one described by Turkel [16], could be useful to accelerate convergence in the low Mach number range, and future study will be devoted to this acceleration technique.

Apart from these reasons for an increase in the number of relaxations, the speed-up on a vector computer is not as large as on a serial computer. This is shown in Figs. 15 and 16, where the convergence for both a single grid and a multigrid calculation is shown. In Fig. 15

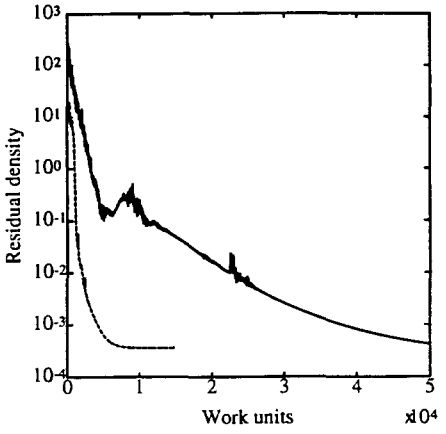

Fig. 15. Convergence behavior of a single grid calculation (solid) and a multigrid calculation (dashed) as a function of the CPU time on a serial computer.

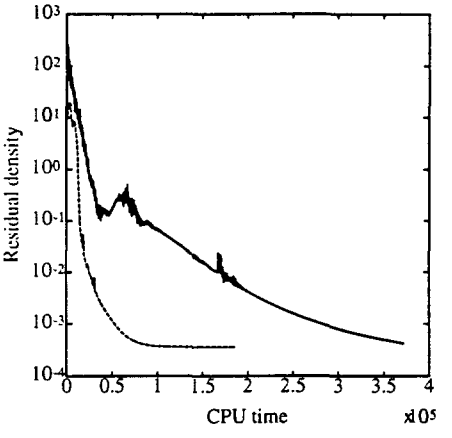

Fig. 16. Convergence behavior of a single grid calculation (solid) and a multigrid calculation (dashed) as a function of the CPU time on a vector computer.

the horizontal axis represents the CPU-time on a serial computer; in Fig. 16 it represents the CPU-time on a vector computer (both in arbitrary units). The speed-up factor on a serial computer is close to 12.5, whereas the speed-up on a vector computer is only approximately 8. This difference can be explained by the fact that the number of grid points in several blocks is quite small on the coarsest mesh. Evidently, this reduces the vector length, and hence the efficiency when using a vector computer. There are several methods to increase the 'vector speed-up'. In the present flow solver the variables on block interfaces are treated after the interior points have been addressed to, even if the usual discretization scheme is applicable. A simultaneous treatment of interior and boundary points, if possible, not only increases the vector lengths but also prevents double calculations of identical fluxes. It has been shown that this simultaneous treatment can lead to a gain in CPU-time by a factor of 3 [17]. Another way to increase the vector lengths is the merging of blocks. The reason for the rather high number of blocks for the two-element airfoil studied here is the requirement that each block interface is bounded by exactly two blocks. In this way only one boundary

condition applies at each block interface. Relaxing this requirement would increase the complexity of the code, but also its efficiency on vector computers.

6. Discussion

We presented simulation results obtained with a multiblock method for a two-element airfoil. For viscous turbulent flow the standard cell-vertex discretization method yields a solution which contains oscillations. Moreover, the rate of convergence to the steady state solution is unacceptably low.

Therefore the discretization method has been changed to the vertex-based method, which is more consistent in the sense that all flux calculations are based on the same control volume. The convergence has been accelerated by the nonlinear multigrid method and implicit residual averaging. Both viscous and inviscid calculations are performed using the same multigrid process and spatial discretization method.

The inviscid calculations have shown that a solution which is converged up to machine accuracy can be obtained with this multigrid method. A comparison with the single grid simulation method shows that a considerable reduction in calculation time is obtained with the multigrid method, although the convergence of the single grid method for inviscid calculations was already quite acceptable. We also investigated two different numerical boundary conditions at the solid walls. It appeared that linear extrapolation of the pressure not only leads to a better convergence than constant extrapolation, but also gives rise to a much smaller entropy layer around the airfoil. The resulting drag coefficient, which theoretically should equal zero in this subsonic flow, is reduced by almost 60%.

In the viscous calculations a total reduction in computational effort with a factor of about 12.5 can be reached with the multigrid method. The calculation time is reduced by a factor of about 8. The difference in these two measures for the speed-up due to the multigrid method can be attributed to the strongly reduced vector lengths associated with the coarser grid calculations reducing the effectiveness of processing the method on a vector computer. The numerical predictions obtained for the lift- and pressure coefficients compare well with experimental results and give confidence in the use of the Baldwin–Lomax model for this application. Moreover, the displacement thickness is found to closely agree with the experiments. This indicates that the scaling of the artificial dissipation with the local Mach number leads to an accurate representation of the flow in the boundary layers.

The convergence of the multigrid process is studied in detail, showing that the ordering of the various loops in the process has a considerable effect. Interchanging the block- and stage loops, keeping the grid loop as the outer loop yields an optimal convergence when the block loop is put inside the stage loop. If the stage loop is put inside the block loop then convergence of the multigrid process is absent when using the five-stage Runge–Kutta scheme as the relaxation method. Apparently, the smoothing of the relaxation method becomes less effective as the number of stages between two 'updates' of the dummy variables increases. This result has some less favorable consequences in view of a possible parallel processing of the multigrid method. On the one hand parallel processing seems more efficient if the frequency of data transfer between the blocks can be reduced. On the other hand the reduction of this frequency results in a reduction of the convergence rate of the multigrid process, and in some instances even to an absence of convergence. This suggests

that in a possible parallel processing of this multigrid method, an optimal rate of data exchange between the blocks should be determined.

Acknowledgements

The authors are greatly indebted to Frans Brandsma for several useful and stimulating discussions and to Bart van Esch for invaluable support with the postprocessing of the results. Part of this research was sponsored by the Stichting Nationale Computerfaciliteiten (National Computing Facilities Foundation, N.C.F.) for the use of supercomputing facilities, with financial support from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Netherlands Organisation for Scientific Research, N.W.O.).

Notes

[1] ISNaS is an abbreviation for Information System for flow simulation based on the Navier–Stokes equations, and is a cooperation of Delft Hydraulics, the National Aerospace Laboratory NLR and the Universities of Delft and Twente, The Netherlands.

References

1. F.J. Brandsma, M.E.S. Vogels, J. van der Vooren, D. Dijkstra and J.G.M. Kuerten, Predesign document of the ISNaS compressible flow simulator. ISNaS 88.04.027 (1988).
2. M.E.S. Vogels, Software design of a multi-block, multi-zone Navier–Stokes solver. In: Proceedings of the First European Computational Fluid Dynamics Conference, Brussels, Belgium, September 7–11 (1992), Ch. Hirsch, J. Périaux and W. Kordulla, eds., pp. 953–960.
3. D.J. Mavriplis, Turbulent flow calculations using unstructured and adaptive meshes, Proceedings of the 12th International Conference on Numerical Methods in Fluid Dynamics, Oxford, England, July 9–13 (1990), K.W. Morton, ed., pp. 228–232.
4. B.J. Geurts and H. Kuerten, Numerical aspects of a block structured compressible flow solver, *J. Engg. Math.* 27 (1993) 293–307.
5. B. Baldwin and H. Lomax, Thin layer approximation and algebraic model for separated turbulent flow, *AIAA*-78-257 (1978).
6. J.G.M. Kuerten, B.J. Geurts, J.W. van der Burg, A.W. Vreman and P.J. Zandbergen, Development and applications of a 3-D compressible Navier–Stokes solver. Proceedings of the 13th International Conference on Numerical Methods in Fluid Dynamics, Rome, Italy, July 6–10 (1992), Springer Verlag, M. Napolitano and F. Sabetta, eds., pp. 529–533.
7. F.J. Brandsma, Mathematical physics aspects of simulations based on the Navier–Stokes equations. *NLR TP* 89069 L (1989).
8. B. van den Berg, Boundary layer measurements on a two-dimensional wing with flap. *NLR TR* 79009 U (1979).
9. J.W. van der Burg, Numerical techniques for transonic flow calculations. Ph.D. Thesis, University of Twente (1993).
10. V.N. Vatsa and B.W. Wedan, Navier–Stokes solutions for transonic flow over a wing mounted in a tunnel. *AIAA* 88-0102 (1988).
11. L. Martinelli, Calculations of viscous flows with a multigrid method. Ph.D. Thesis, Princeton University (1987).
12. P. Wesseling, An Introduction To Multigrid Methods. John Wiley & Sons Ltd., England (1992).
13. A. Brandt, Guide to multigrid development. In: Multigrid methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, pp. 220–312, Springer Verlag, Berlin (1981).
14. R. Radespiel, A cell-vertex multigrid method for the Navier–Stokes equations. NASA Technical Memorandum 101557 (1989).
15. T. Larsson, High-lift calculations using Navier–Stokes methods, in Proceedings of the First European

Computational Fluid Dynamics Conference, Brussels, Belgium, September 7–11 (1992), Ch. Hirsch, J. Périaux and W. Kordulla, eds., pp. 683–689.
16. E. Turkel, Preconditioning methods for solving the incompressible and low speed compressible equations. *J. Comp. Phys.* 72 (1987) 277–298.
17. B.P.M. van Esch, Design document of the module MERSET. *NLR TR* 92067 L (1992).