Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 226 (2007) 1784–1807
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Multigrid solution method for the steady RANS equations

Jeroen Wackers [a,*], Barry Koren [a,b]

[a] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*
[b] *Delft University of Technology, Faculty of Aerospace Engineering, P.O. Box 5058, 2600 GB Delft, The Netherlands*

Received 11 December 2006; accepted 10 June 2007
Available online 15 June 2007

Abstract

A novel multigrid method for the solution of the steady Reynolds-averaged Navier–Stokes equations is presented, that gives convergence speeds similar to laminar flow multigrid solvers. The method is applied to Menter's one-equation turbulence model. New aspects of the method are the combination of nonlinear Gauss–Seidel smoothing on the finest grid with linear coarse-grid corrections, and local damping in the initial stages of the computation, to keep the solution stable; the damping needed is estimated with the nonlinear smoother. Efficiency on the finest grid is increased with full multigrid, second-order accuracy is obtained with defect correction. Tests on boundary layers and airfoil flows show the efficiency of the method.

MSC: 65N55; 76D05; 76F99

Keywords: Turbulence; Reynolds-averaged Navier–Stokes; Menter model; Linear multigrid; Line Gauss–Seidel

1. Introduction

Fast solution techniques for turbulent flow equations are of great importance in today's engineering practice. Most real-life flows are turbulent, be they air or water flows around vehicles or flow in pipes and channels. So in design processes for these devices, turbulent flow simulation is indispensable and for efficient design, the flow solvers must be fast. Quasi-steady turbulent flows are usually modelled with the steady Reynolds-averaged Navier–Stokes (RANS) equations; an efficient way to solve a variety of flow problems is multigrid [1–3].

To obtain the RANS equations, the unsteady turbulent flow field is averaged. The result is a set of flow equations that may have steady solutions. These equations contain unknown closure terms for the effect of turbulence, which are modelled approximately with a turbulence model: one or more differential equations

[*] Corresponding author. Tel.: +31 20 592 4176; fax: +31 20 592 4199.
E-mail addresses: J.Wackers@cwi.nl (J. Wackers), B.Koren@cwi.nl (B. Koren).

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.06.007

for the turbulent stress. As opposed to the laminar Navier–Stokes equations, these turbulence models contain source terms, that represent the production and dissipation of turbulence.

Multigrid solution of the RANS equations is not straightforward. For the laminar Navier–Stokes equations, efficient multigrid techniques have been developed where the steady flow equations are solved directly with a combination of nonlinear multigrid and Gauss–Seidel smoothing; examples are found in the work of Hemker et al. [4,5], of Dick et al. [6,7], and of Trottenberg et al. [8,2]. But due to the source terms in the turbulence model, the RANS equations cannot be solved with these techniques. Instead, multigrid is usually combined with a time stepping approach. Either time integration is used as a smoother in the multigrid algorithm or multigrid is used for the individual time steps in an implicit time integrator, that time-marches the unsteady RANS equations to convergence. In the first category, Mavriplis [9,10] uses Jameson's multigrid method with a Runge–Kutta time integration smoother, on unstructured grids. Liu and Zheng [11] present a finite-volume method on structured meshes. Carré [12] uses linear multigrid to solve implicit time steps and increases these time steps as the solution converges. Steelant et al. [7] form an exception: they use damped multigrid with a line smoother.

In the solution process, the link between the Navier–Stokes equations and the turbulence model is important. In many flow solvers, the turbulence model is considered loosely coupled to the other flow equations, therefore it is solved separately: alternately, the flow field is updated with the turbulence fixed and the turbulence is updated with the flow fixed. For the flow field step, an existing laminar flow solver can be used. However, Liu and Zheng [11] claim that this technique is inefficient; they report improved convergence when all equations are solved together. On the other hand, Steelant et al. [7] get the best results with a loosely coupled approach.

In this paper, we show that the steady Reynolds-averaged Navier–Stokes equations with Menter's turbulence model [13] can be solved with multigrid and Gauss–Seidel smoothing, without the need for time stepping, and that convergence rates can be obtained that are similar to the most efficient multigrid solvers for laminar flow. We also explain why a fully coupled solution of the flow field and the turbulence is necessary to obtain this convergence. Our novel multigrid technique is a combination of nonlinear line Gauss–Seidel smoothing on the finest grid and linearised coarse grid corrections. Local damping is applied in the initial part of the solution process; this does not reduce the convergence rate because it is not needed in most of the domain. The nonlinear smoothing is used to estimate the need for damping.

To get an efficient solution process on the finest grid, a full multigrid method (i.e., the initial solution on a grid comes from solutions on coarser grids) is essential. Second-order accuracy is obtained with defect-correction iteration [14,15,2]. Defect correction iteration converges slowly in terms of the residual, but it improves a first-order accurate solution to second-order accuracy in only a few steps.

The solution technique is applied to a finite-volume discretisation of the incompressible RANS equations on structured, curved grids. The discretisation is cell-centred and combines artificial compressibility convective fluxes with central diffusive fluxes and finite-difference stencils for the source terms. It is based on the discretisation presented in [16].

The structure of the paper is as follows. It starts with two introductory sections: in Section 2, a brief overview is given of the RANS equations and of Menter's one-equation turbulence model. And Section 3 introduces the multigrid method for laminar flows [16], on which the present multigrid method is based. Section 4 forms the heart of the paper: it shows why the RANS equations cannot be solved with classical multigrid, it presents a suitable Gauss–Seidel smoother and it introduces the linear multigrid algorithm. Furthermore, it explains why the coupled solution is essential for fast convergence. This discussion is valid for general spatial discretisations; Section 5 gives the finite-volume discretisation used in our numerical tests. In Section 6, results from four test problems show the efficiency of the method for different flows and the accuracy of the solutions obtained. The paper ends with a conclusion.

2. Turbulent flow equations

This section gives a brief overview of the flow equations used here, the Reynolds-averaged Navier–Stokes (RANS) equations and Menter's turbulence model [13].

2.1. RANS equations

For the RANS equations, the turbulent flow field is ensemble-averaged. In the equations for the averaged flow quantities, the only contribution from the turbulence that remains is a turbulent stress term. Under the Boussinesq hypothesis, this term can be modelled by simply adding a non-constant turbulent viscosity to the laminar viscosity in the standard Navier–Stokes equations.

In two dimensions, the steady RANS equations for incompressible flows are:

$$\begin{aligned}
&\frac{\partial}{\partial x}(p+u^2)+\frac{\partial}{\partial y}(uv)=\frac{\partial}{\partial x}((v+v_{\mathrm{T}})2u_x)+\frac{\partial}{\partial y}((v+v_{\mathrm{T}})(u_y+v_x)),\\
&\frac{\partial}{\partial x}(uv)+\frac{\partial}{\partial y}(p+v^2)=\frac{\partial}{\partial x}((v+v_{\mathrm{T}})(u_y+v_x))+\frac{\partial}{\partial y}((v+v_{\mathrm{T}})2v_y),\\
&\frac{\partial}{\partial x}(u)+\frac{\partial}{\partial y}(v)=0.
\end{aligned}\tag{1}$$

Here p is the pressure, divided by the (constant) density, v is the laminar dynamic viscosity and v_{T} is the turbulent viscosity. As v_{T} is not known exactly, it is modelled with an approximate turbulence model.

2.2. Menter's turbulence model

To approximate v_{T}, one or more equations are added to the system (1). Menter's turbulence model is a robust and accurate one-equation model, that computes the turbulent viscosity directly. It is similar to the Spalart–Allmaras model [17]. In two dimensions, it is given by:

$$\frac{\partial(\tilde{v}_{\mathrm{T}}u)}{\partial x}+\frac{\partial(\tilde{v}_{\mathrm{T}}v)}{\partial y}=\frac{\partial}{\partial x}\left(\left(v+\frac{\tilde{v}_{\mathrm{T}}}{\sigma_m}\right)\frac{\partial\tilde{v}_{\mathrm{T}}}{\partial x}\right)+\frac{\partial}{\partial y}\left(\left(v+\frac{\tilde{v}_{\mathrm{T}}}{\sigma_m}\right)\frac{\partial\tilde{v}_{\mathrm{T}}}{\partial y}\right)+P-D.\tag{2}$$

This is a convection–diffusion-reaction equation: P and D are source terms modelling the production and dissipation of turbulence. To get correct behaviour of the model near walls, the actual v_{T} to be used in (1) is scaled:

$$v_{\mathrm{T}}=\left(1-\mathrm{e}^{-\left(\frac{\tilde{v}_{\mathrm{T}}}{A^+\kappa v}\right)^2}\right)\tilde{v}_{\mathrm{T}}.\tag{3}$$

A^+ and κ are constants. The boundary conditions for $\tilde{v}_{\mathrm{T}}$ are straightforward; this is one of the advantages of Menter's model. On a wall, $\tilde{v}_{\mathrm{T}}=0$ as turbulence dies out near walls. And on inflow boundaries, a small positive value for $\tilde{v}_{\mathrm{T}}$ is set, usually about $0.01v$; if $\tilde{v}_{\mathrm{T}}$ at the inflow is zero, it remains zero throughout the domain because no turbulence is produced. The solution is not sensitive to the inflow $\tilde{v}_{\mathrm{T}}$, as long as it is significantly smaller than the maximum $\tilde{v}_{\mathrm{T}}$ [13].

The production and dissipation terms are the heart of the model. The production term is:

$$P=c_1\frac{v+v_{\mathrm{T}}}{v+\tilde{v}_{\mathrm{T}}}\tilde{v}_{\mathrm{T}}\sqrt{2(u_x^2+v_y^2)+(u_y+v_x)^2},\tag{4}$$

and the dissipation is:

$$D=c_2c_3D_{BB}\tanh\left(\frac{D_{k\text{–}\varepsilon}}{c_3D_{BB}}\right),\tag{5}$$

with

$$D_{k\text{–}\varepsilon}=\tilde{v}_{\mathrm{T}}^2\frac{(u_{xx}+u_{yy})^2+(v_{xx}+v_{yy})^2}{u_x^2+u_y^2+v_x^2+v_y^2},\quad D_{BB}=(\tilde{v}_{\mathrm{T}_x})^2+(\tilde{v}_{\mathrm{T}_y})^2.\tag{6}$$

$D_{k\text{–}\varepsilon}$ is the main dissipation term. Eq. (5) reduces to $D\approx c_2D_{k\text{–}\varepsilon}$ when $D_{k\text{–}\varepsilon}$ is small, the limiting with c_3D_{BB} is only needed for regions where $D_{k\text{–}\varepsilon}$ is large due to small velocity derivatives. The $D_{k\text{–}\varepsilon}$ in (6) is actually an alternative form that is suggested, but not used, by Menter. It is slightly more complex than Menter's original form, but it can be discretised on a five-point stencil (see Section 5). Both the production and the dissipation term are invariant under rotation and the production term P is always positive, while the dissipation $-D$ is always negative.

The model constants have the values $c_1 = 0.144$, $c_2 = 1.86$ and $A^+ = 13.0$. The von Karman constant $\kappa = 0.41$. Furthermore, $c_3 = 7$ and $\sigma_m = 1$.

3. Multigrid

For laminar flows, multigrid is a well-known and mature solution technique [8,1–3]. Many different varieties exist. Here, we briefly introduce the nonlinear multigrid technique and smoother that we have used before [16], a classical technique similar to the one described in [6]. This section serves as a basis for the following Section 4, where we discuss the changes to this nonlinear multigrid technique that are needed to solve the RANS equations.

3.1. Grids and discretisation

The method is constructed for a cell-centred finite-volume discretisation on curvilinear structured meshes. So the grids consist of quadrilateral cells, that may be non-rectangular, and the states in the middles of the cells are stored. A typical cell is shown in Fig. 1. The finest grid is called Ω_K. A set of underlying coarse grids Ω_k with $0 \leqslant k \leqslant K-1$ is made by merging 2×2 blocks of cells in the next finer grid into single cells, so each cell $(\Omega_k)_{i,j}$ in grid Ω_k corresponds to four cells $(\Omega_{k+1})_{2i(+1),2j(+1)}$ in the next finer grid Ω_{k+1}. The RANS operator (1) plus the Menter model (2) is denoted by $\mathcal{F}(\boldsymbol{q})$, $\boldsymbol{q} = [u, v, p, \tilde{\nu}_{\mathrm{T}}]^{\mathrm{T}}$. For the following discussion, we limit ourselves to five-point stencils: for each cell, the state in that cell and in its four neighbours is used. The discretisation on grid Ω_k is $\mathcal{F}_k$, the state on that grid is $\boldsymbol{q}_k$.

3.2. Multigrid algorithm

In the multigrid technique, the high-frequency errors in the initial solution are removed on the fine grid Ω_K and the lower-frequency errors on the underlying coarser grids $\Omega_0 \cdots \Omega_{K-1}$. The final problem to be solved is $F_K \boldsymbol{q}_K = 0$, the general problem on each grid is $\mathcal{F}_k \boldsymbol{q}_k = \boldsymbol{s}_k$, for some source term $\boldsymbol{s}_k$. We call the line Gauss–Seidel smoothing operator M_k and introduce a finite-volume prolongation operator P^k_{k-1} that moves a solution from one cell on grid $k-1$ to the four cells on grid k that lie in the same location:

$$(\boldsymbol{q}_k)_{2i(+1),2j(+1)} = P^k_{k-1}(\boldsymbol{q}_{k-1})_{i,j} = (\boldsymbol{q}_{k-1})_{i,j}. \tag{7}$$

In the same way, a restriction operator R^{k-1}_k is defined for defects $\boldsymbol{d}$:

$$(\boldsymbol{d}_{k-1})_{i,j} = R^{k-1}_k(\boldsymbol{d}_k)_{2i(+1),2j(+1)} = (\boldsymbol{d}_k)_{2i,2j} + (\boldsymbol{d}_k)_{2i+1,2j} + (\boldsymbol{d}_k)_{2i,2j+1} + (\boldsymbol{d}_k)_{2i+1,2j+1}. \tag{8}$$

Then the multigrid procedure (for iteration n) is defined recursively as follows. It is started on the finest grid, with $\boldsymbol{s}_K = 0$.

```
$\boldsymbol{q}_k^{n+1}$ = recursive function $NMG(k, \boldsymbol{q}_k^n, \boldsymbol{s}_k^n)$
$\tilde{\boldsymbol{q}}_k^n = (M_k)^{q_1}\boldsymbol{q}_k^n$                                   $q_1$ pre-relaxation steps,
if $k \neq 0$ then
    $\boldsymbol{d}_{k-1}^n = R_k^{k-1}(\mathcal{F}_k\tilde{\boldsymbol{q}}_k^n - \boldsymbol{s}_k^n)$                 defect on coarse grid,
    $\boldsymbol{s}_{k-1}^n = \mathcal{F}_{k-1}\boldsymbol{q}_{k-1}^n - w_{k-1}^n\boldsymbol{d}_{k-1}^n$                source term,
    $\boldsymbol{q}_{k-1}^{n+1} = NMG^\sigma(k-1, \boldsymbol{q}_{k-1}^n, \boldsymbol{s}_{k-1}^n)$           $\sigma$ MG steps on coarser grid,
    $\tilde{\tilde{\boldsymbol{q}}}_k^n = \tilde{\boldsymbol{q}}_k^n + \frac{1}{w_{k-1}^n}P_{k-1}^k(\boldsymbol{q}_{k-1}^{n+1} - \boldsymbol{q}_{k-1}^n)$     prolongation of correction,
end if
$\boldsymbol{q}_k^{n+1} = (M_k)^{q_2}\tilde{\tilde{\boldsymbol{q}}}_k^n$                                $q_2$ post-relaxation steps.
```

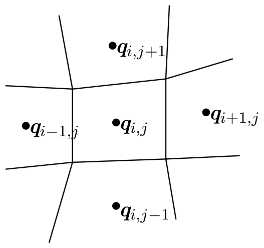

Fig. 1. A typical cell, with four neighbours.

Our experiments in [16] show that, for the type of flows we consider, a W-cycle ($\sigma = 2$) is the most efficient. The scaling parameter w_{k-1}^n is usually 1. But if the defect $\boldsymbol{d}_{k-1}^n$ is very large, it may be impossible to find a solution to $\mathcal{F}_{k-1}\boldsymbol{q}_{k-1}^n = \boldsymbol{s}_{k-1}^n$, because $\mathcal{F}$ is nonlinear; then $\boldsymbol{s}_{k-1}^n$ can be reduced by making w_{k-1}^n small. The prolongated correction is unscaled through division by w_{k-1}^n.

3.3. Gauss–Seidel smoothing

The multigrid algorithm is combined with alternating line Gauss–Seidel relaxation; this is a good smoothing technique for convection-dominated flows. In each iteration step n, the solution is updated row by row or column by column. The state $\boldsymbol{q}_k^{n+1}$ in each line of cells is changed such, that the residual in those cells becomes zero, given the current state in all other cells. These states are the old states $\boldsymbol{q}_k^n$ in cells that have not been updated yet and the new, updated states $\boldsymbol{q}_k^{n+1}$ in all other cells. So we solve (for row smoothing, marching in positive j-direction):

$$\mathcal{F}_k((\boldsymbol{q}_k^{n+1})_{i,j}, (\boldsymbol{q}_k^{n+1})_{i-1,j}, (\boldsymbol{q}_k^{n+1})_{i,j-1}, (\boldsymbol{q}_k^{n+1})_{i+1,j}, (\boldsymbol{q}_k^{n})_{i,j+1}) = 0, \quad \forall i \in \text{row } j. \tag{10}$$

A similar expression is found for column smoothing. Eq. (10) is solved with a numerical root finder, the matrix-vector version of the Newton–Raphson method. Start with an initial guess $\boldsymbol{q}_{\text{NR}}^0 = (\boldsymbol{q}_k^n)_{i,j\in\text{line}}$, then define each new step l as:

$$\boldsymbol{q}_{\text{NR}}^{l+1} = \boldsymbol{q}_{\text{NR}}^{l} - \left(\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k}\right)_{\text{line}}^{-1} (\mathcal{F}_k)_{\text{line}}(\boldsymbol{q}_{\text{NR}}^{l}). \tag{11}$$

On convergence, we set $(\boldsymbol{q}_k^{n+1})_{i,j\in\text{line}} = \boldsymbol{q}_{\text{NR}}^{l}$. The matrix $\partial\mathcal{F}_k/\partial\boldsymbol{q}_k$ is block-tridiagonal (a cell has four neighbours that influence it, but only two are in the line), so Eq. (11) can be solved with a block Thomas algorithm.

The process can be changed by under/overrelaxation, i.e. introducing a parameter $\omega > 0$ and choosing:

$$(\boldsymbol{q}_k^{n+1})_{i,j\in\text{line}} = (\boldsymbol{q}_k^{n})_{i,j\in\text{line}} + \omega(\boldsymbol{q}_{\text{NR}}^{l} - (\boldsymbol{q}_k^{n})_{i,j\in\text{line}}). \tag{12}$$

For nonlinear Gauss–Seidel, underrelaxation ($\omega < 1$) is used to keep low-frequency errors stable. We use $\omega = 0.9$ [16].

3.4. Full multigrid

The multigrid solution process is used in a full multigrid framework. We actually start the solution process on the coarsest grid Ω_0, with Gauss–Seidel smoothing only. When a converged solution on this grid is found, it is prolongated as an initial solution to the next finer grid Ω_1, where a converged solution is produced with multigrid on two grids. This process is continued until the finest grid is reached. Full multigrid reduces the number of iterations needed on the finest grid: these are replaced by cheaper iterations on coarser grids.

3.5. Defect correction

Multigrid with standard smoothing is ineffective for the solution of second-order accurate discretisations. These can be solved with defect correction: a first-order discretisation $\mathcal{F}_K$ with a multigrid solver is used as an approximate inverse for the second-order discretisation $\mathcal{F}_{K,2}$. Before each multigrid cycle, a source term is computed on grid K:

$$\boldsymbol{d}_K^n = \mathcal{F}_K(\boldsymbol{q}_K^n) - \mathcal{F}_{K,2}(\boldsymbol{q}_K^n). \tag{13}$$

Then a multigrid cycle is applied to $\mathcal{F}_K(\boldsymbol{q}_K^{n+1}) = \boldsymbol{d}_K^n$. After this cycle, a new source term is set, then a new multigrid cycle is started, etc. On convergence, the solution satisfies $\mathcal{F}_{K,2}(\boldsymbol{q}_K^n) = 0$.

For defect correction, the residual converges slowly. But when the defect correction is started from a converged first-order accurate solution, then a few defect correction cycles are enough to make the solution second-order accurate, even if it is not converged. (Theoretically, second-order accuracy is reached after one defect correction cycle [15].) Thus, defect correction can be seen as an iterative improvement for a first-order accurate solution.

4. Solution of Menter's model

Standard multigrid, as applied to the laminar Navier–Stokes equations, cannot be used directly for the RANS equations. This has two reasons. One: because of the source terms in the turbulence equation, the flow equations are no longer positive definite, so the line smoothing does not converge. And two: the solutions on the coarse grids do not resemble the solutions on the fine grid enough.

This section gives the solution to these problems. In Section 4.1, the problem with the flow equations is analysed. An improved smoother is presented in Section 4.2. And Section 4.3 gives a working coarse grid correction algorithm.

4.1. Source term and negative eigenvalues

Classical relaxation techniques, like line smoothing, only work when the discretised system $\mathcal{F}_k$ is of vector-positive type [18]. This means that the system has no unstable eigenmodes: when $\mathcal{F}_k$ is linearised, all eigenvalues λ of the resulting linear system must have $\Re(\lambda) > 0$. Discretisations of convection–diffusion equations, like the laminar Navier–Stokes equations, may be vector-positive; this depends on the discretisation. For example, the artificial compressibility discretisations by Dick and his co-workers [18,6] are always vector-positive, but their later AUSM+ based discretisations [19,20] are not.

In the RANS equations, a more fundamental problem appears. There, the *continuous* system makes it impossible to find a discretisation that is always vector-positive. The occurrence of eigenvalues with $\Re(\lambda) \leqslant 0$ is caused by the source term in the turbulence equation.

4.1.1. Linearised flow equations

To study the effect of the source term, we construct a linearised version of the continuous system $\mathcal{F}$. We choose a function $\boldsymbol{Q} = [U,V,P,N_\mathrm{T}]^\mathrm{T}$ and write functions close to $\boldsymbol{Q}$ as $\boldsymbol{q} = \boldsymbol{Q} + \boldsymbol{q}'$. Substituting this in (1) and (2) gives a linear operator L for the small disturbances $\boldsymbol{q}'$, such that:

$$\mathcal{F}(\boldsymbol{Q}+\boldsymbol{q}') \approx \mathcal{F}(\boldsymbol{Q}) + L\boldsymbol{q}'. \tag{14}$$

We find:

$$L = \begin{bmatrix} \begin{array}{c} 2U\partial_x + V\partial_y \\ -N(2\partial_{xx}+\partial_{yy}) \end{array} & \begin{array}{c} U\partial_y \\ -N\partial_{xy} \end{array} & \partial_x & \begin{array}{c} -2U_x\partial_x \\ -(U_y+V_x)\partial_y \end{array} \\[2ex] \begin{array}{c} V\partial_x \\ -N\partial_{xy} \end{array} & \begin{array}{c} U\partial_x + 2V\partial_y \\ -N(\partial_{xx}+2\partial_{yy}) \end{array} & \partial_y & \begin{array}{c} -(U_y+V_x)\partial_x \\ -2V_y\partial_y \end{array} \\[2ex] \partial_x & \partial_y & 0 & 0 \\[2ex] \begin{array}{c} N_\mathrm{T}\partial_x \\ -P_2(2U_x\partial_x + (U_y+V_x)\partial_y) \\ +D_2(U_{xx}+U_{yy})(\partial_{xx}+\partial_{yy}) \\ +D_3(U_x\partial_x + U_y\partial_y) \end{array} & \begin{array}{c} N_\mathrm{T}\partial_y \\ -P_2((U_y+V_x)\partial_x + 2V_y\partial_y) \\ +D_2(V_{xx}+V_{yy})(\partial_{xx}+\partial_{yy}) \\ +D_3(V_x\partial_x + V_y\partial_y) \end{array} & 0 & \begin{array}{c} U\partial_x + V\partial_y - N(\partial_{xx}+\partial_{yy}) \\ -N_{\mathrm{T}x}\partial_x - N_{\mathrm{T}y}\partial_y \\ -P_1 + D_1 \end{array} \end{bmatrix} \tag{15a}$$

For simplicity, we use $D = c_2 D_{k-\varepsilon}$ instead of Eq. (5) and the low-Reynolds correction (Eq. (3)) is not used, so $\nu_{\mathrm{T}} = \tilde{\nu}_{\mathrm{T}}$. This makes no large difference to the system. The symbols like ∂_x and ∂_{xx} denote differentiation operators, the abbreviations are:

$$N = \nu + N_{\mathrm{T}}, \tag{15b}$$

representing the total viscosity, and:

$$P_1 = c_1\sqrt{2(U_x^2 + V_y^2) + (U_y + V_x)^2}, \quad P_2 = c_1 \frac{1}{\sqrt{2(U_x^2 + V_y^2) + (U_y + V_x)^2}}, \tag{15c}$$

$$D_1 = 2c_2 N_{\mathrm{T}} \frac{(U_{xx} + U_{yy})^2 + (V_{xx} + V_{yy})^2}{U_x^2 + U_y^2 + V_x^2 + V_y^2}, \quad D_2 = 2c_2 N_{\mathrm{T}}^2 \frac{1}{U_x^2 + U_y^2 + V_x^2 + V_y^2}, \tag{15d}$$

$$D_3 = -2c_2 N_{\mathrm{T}}^2 \frac{1}{(U_x^2 + U_y^2 + V_x^2 + V_y^2)^2}, \tag{15e}$$

representing parts of the production and dissipation terms.

The operator in the first three rows, first three columns is the laminar Navier–Stokes operator. We see the continuity equation $\partial_x u + \partial_y v = 0$ in the third row and the momentum equations with convection and viscous diffusion in the first two rows. The fourth row and column add the effect of turbulence: in the fourth column, we see terms due to the non-constant viscosity; the fourth row contains the turbulence equation, where the terms are combinations of convection–diffusion effects and source term contributions.

These combined terms cause zero eigenvalues in the system (the limiting case of eigenvalues with $\Re(\lambda) \leqslant 0$), when they cancel each other out. For example, in the (4,4) term, the convection and the non-constant diffusion term are comparable, but they have different constants. The source term also has an important effect on the (4,4) term. Altogether, situations can arise where the fourth equation does not change when the local $\tilde{\nu}'_{\mathrm{T}}$ changes. In that case, the operator has a zero eigenvalue. In a realistic flow solution, any combination of values for U, V, N_{T} and the source term parameters P_1, P_2, D_1, D_2, and D_3 may appear, so zero eigenvalues are bound to arise. They appear either by this cancelling of the (4,4) term or by interaction of the (1–2,4) terms with the (4,1–2) terms.

One could think that the use of stable discretisations, e.g. upwinding, can make a discretised version of this system vector-positive. Unfortunately, that is not true: the terms in (15) that may cancel each other are part of totally different parts of the operator. Those parts, like the convection operator and the source term, cannot be combined, so they cannot be discretised with one stable technique. Therefore, even in a discretised system, situations where the system is not vector-positive cannot be prevented.

4.1.2. Determinant

To understand the nature of the system, it is useful to study the determinant of (15a). This determinant is:

$$\begin{aligned}\det(L) = &-\Delta[N\Delta - (U\partial_x + V\partial_y)][N\Delta - ((U - N_{\mathrm{T}_x})\partial_x + (V - N_{\mathrm{T}_y})\partial_y) + P_1 - D_1] \\ &+ (U_y + V_x)(\partial_{yy} - \partial_{xx})[P_2(U_y + V_x)(\partial_{yy} - \partial_{xx}) + D_2((V_{xx} + V_{yy})\partial_x - (U_{xx} + U_{yy})\partial_y)\Delta \\ &+ D_3(V_x\partial_{xx} - U_y\partial_{yy})],\end{aligned} \tag{16}$$

where Δ is the Laplace operator $\partial_{xx} + \partial_{yy}$. This determinant is much more complex than the one for the laminar Navier–Stokes operator, given here for comparison:

$$\det(L_{\mathrm{lam}}) = -\Delta(N\Delta - (U\partial_x + V\partial_y)). \tag{17}$$

The most noticeable difference is, that the turbulent determinant has two separate terms, while the laminar determinant has only one product of terms. The second term is a combination of the turbulent viscosity effect on the momentum equations and the velocity effects in the turbulent source term. The operator has zero eigenvalues when det $(L) = 0$. In the laminar case, this happens only when one of the terms of the product is zero. In the turbulent case, it happens whenever the two terms cancel each other out.

In multigrid, one usually determines the type of the system by looking at the highest derivatives in the determinant. But since this system is singularly perturbed (ν is small), we study the terms with the highest two derivatives, i.e. those of fifth and sixth order. These are:

$$\det(L) \approx -N^2\Delta^3 + N\Delta^2((2U - N_{\mathrm{T}_x})\partial_x + (2V - N_{\mathrm{T}_y})\partial_y) + D_2(U_y + V_x)(\partial_{yyyy} - \partial_{xxxx})((V_{xx} + V_{yy})\partial_x - (U_{xx} + U_{yy})\partial_y). \tag{18}$$

Even here, a part of the second term remains. And even though the term D_2 contains N_{T}^2, it is $\mathcal{O}(N_{\mathrm{T}})$ in a boundary layer, where u_y is $\mathcal{O}(N_{\mathrm{T}}^{-1/2})$. Therefore, its contribution is non-trivial: the source term really has an effect on the occurrence of zero eigenvalues.

4.1.3. Stability near a solution

An interesting property of the system saves us: if the turbulence model is stable, then the operator has no eigenvalues with $\Re(\lambda) \leqslant 0$ near a solution of the steady RANS equations. This can be seen by considering the time-dependent flow equations and substituting a steady solution with a small disturbance. If any of the eigenvalues of L would have a negative real part, then the disturbance would grow in time, so the flow would be unstable. Thus, the existence of a stable steady solution guarantees that all λ have $\Re(\lambda) > 0$ near that solution. This is good news, as a properly designed turbulence model must have stable steady solutions. It is the task of the turbulence model designer to guarantee this. Therefore, we may assume that near a solution, L always has $\Re(\lambda) > 0$ for all λ.

Fig. 2 gives an example. It shows a part of the eigenvalue spectrum for a discretised version of (1) and (2). In the first figure, the spectrum is given for a state that is far away from the converged solution. We see a large number of eigenvalues with negative real parts. When the flow is converged (Fig. 2b), all these λ have moved to the right and crossed the imaginary axis. No eigenvalues with $\Re(\lambda) \leqslant 0$ remain.

Concluding, we have found:

(1) that the linearised operator L may have eigenvalues with $\Re(\lambda) \leqslant 0$, that prevent the convergence of line Gauss–Seidel,
(2) that this is *not* the case for converged solutions,
(3) that the interaction between the turbulent viscosity in the momentum equations and the velocity contributions to the source term plays an important role in the behaviour of the system.

In the following, these findings are used to construct an efficient multigrid algorithm.

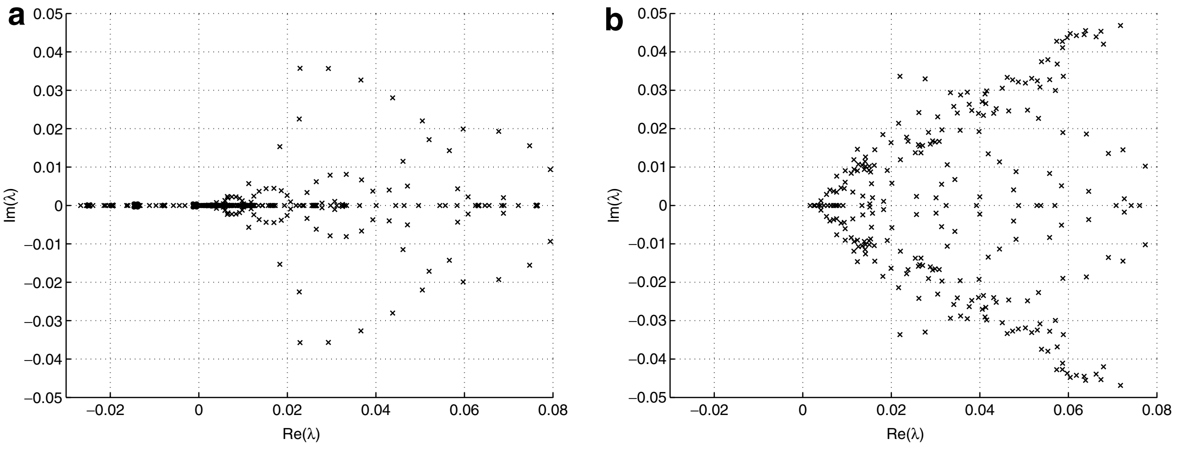

Fig. 2. Detail of the eigenvalue spectrum (leftmost eigenvalues) for the discretised RANS equations (the discretisation from Section 5, applied to the zero pressure gradient boundary layer problem from Section 6.2, on a 32×32 cell grid). Graph (a) state after one line GS sweep, starting with a uniform flow. Graph (b) converged solution.

4.2. Improved line smoothing

The previous section showed why normal line smoothing cannot be used to solve the RANS equations. Since line smoothing works well for the laminar Navier–Stokes equations, it is tempting to consider separate smoothing of the laminar state variables (u, v, and p) and the turbulence $\tilde{\nu}_T$, alternately updating $\tilde{\nu}_T$ or the laminar state variables while the other(s) are kept fixed. This technique works for time steppers, but not for the more powerful Gauss–Seidel smoothing. The coupling between velocity and $\tilde{\nu}_T$ in the source term is so strong that smoothing u and v while keeping $\tilde{\nu}_T$ constant may send the state in a wrong direction, increasing the error. In some of our experiments, errors increased with a factor 100 or more in a single smoothing step! Therefore, it is essential that all four state variables are smoothed together. This section describes how such a smoother is made stable.

4.2.1. Damped Gauss–Seidel

The alternating line Gauss–Seidel smoothing is stabilised by adding local damping. Instead of applying the line smoother to $\mathcal{F}_k(\boldsymbol{q}_k) = 0$, we apply it, in each smoothing step n, to the system:

$$\mathcal{F}_k(\boldsymbol{q}_k^{n+1}) + \alpha I_4 \boldsymbol{q}_k^{n+1} = \alpha I_4 \boldsymbol{q}_k^{n}. \tag{19}$$

Applying one Gauss–Seidel step gives an approximate solution of this system. The positive function α may be non-constant in space and different for different n. I_4, for each cell, is a 4×4 zero matrix with 1 in the $(4,4)$ position. Thus, we damp only the corrections of the turbulent viscosity; Eq. (19) is similar to implicit time stepping for $\tilde{\nu}_T$ only. Linearised, it reads:

$$(L_k + \alpha I_4)(\boldsymbol{q}'_k)^{n+1} = \alpha I_4 (\boldsymbol{q}'_k)^{n}, \tag{20}$$

where L_k is the linearisation of $\mathcal{F}_k$. We see:

(1) that, if the damped Gauss–Seidel process converges, it converges to $\mathcal{F}_k(\boldsymbol{q}_k) = 0$,
(2) that line smoothing is a good solver for the system (19) if the linear operator $(L_k + \alpha I_4)$ has $\Re(\lambda) > 0, \forall \lambda$,
(3) that, since (19) resembles an implicit time stepping procedure for $\tilde{\nu}_T$, it is expected to converge to $\mathcal{F}_k(\boldsymbol{q}_k) = 0$ if a stable steady solution exists.

As seen in Section 4.1, the first three rows and columns of L_k form the laminar Navier–Stokes operator, which has no eigenvalues with $\Re(\lambda) \leqslant 0$. Therefore, L_k can be made diagonally dominant by increasing the $(4,4)$ term only; $L_k + \alpha I_4$ has no $\Re(\lambda) \leqslant 0$ eigenvalues when α is chosen sufficiently large.

However, it is not necessary—rather, it is wasteful—to set α large everywhere, all the time. As we have seen, the $\Re(\lambda) \leqslant 0$ eigenvalues disappear when the solution process converges, so damping is only needed in the early stages of the solution. And in most of the flow field, certainly outside boundary layers and turbulent wakes, no $\Re(\lambda) \leqslant 0$ eigenvalues appear anyway, so no damping is needed there either.

Therefore, we take α constant in a line. We set α for each individual line, for every smoothing step, to the smallest value that gives positive eigenvalues. Thus, the α in a certain cell may be different for successive horizontal and vertical sweeps. The choice of α is explained below.

4.2.2. Estimating α

The smallest possible α can be estimated, very elegantly, with the Newton–Raphson algorithm that is used to solve the nonlinear flow equations in the individual lines. Newton–Raphson (NR) relies on linearisations $\partial\mathcal{F}_k/\partial\boldsymbol{q}_k$ of the flow equations $\mathcal{F}_k$ in each line to find the roots of these sets of nonlinear equations. But if one or more of the eigenvalues of the linearised flow equations are zero, then the corresponding part of the nonlinear system is dominated by higher-order effects. In that case, a linearisation is no longer a good approximation of the nonlinear system, so NR loses its quadratic convergence. It converges slowly, or not at all.

Therefore, we choose α locally by monitoring the convergence rate of NR for each line. Each smoothing step is started with a low value for α, constant in the whole domain. When the NR iteration in a line does not converge fast enough, it is restarted with a larger α for that line. This is repeated, if necessary, until a sufficient convergence rate for NR is obtained. Then we can be sure that $L_k + \alpha I_4$ has no eigenvalues close to zero, that are associated with that line. And, as the eigenvalues near 0 lie close together (Fig. 2), this indicates that all eigenvalues have $\Re(\lambda) > 0$. The advantage of this procedure is, that the convergence of NR has to be monitored anyway, to determine when to stop the iteration; no costly, complicated estimation of eigenvalues is needed.

The basis for a definition of sufficiently fast convergence for NR is the residual of the flow equations in the line, after l Newton–Raphson steps:

$$r^l = \sum_{\text{line}} |\mathcal{F}(\boldsymbol{q}^l_{\text{NR}}) + \alpha I_4(\boldsymbol{q}^l_{\text{NR}} - (\boldsymbol{q}^n_k)_{i,j\in\text{line}})|. \tag{21}$$

When applied to a system without zero eigenvalues, NR ideally shows quadratic convergence: $r^l \sim (r^{l-1})^2$. But in practice, this is seldom obtained, even with α sufficiently large. For high residuals, nonlinear effects dominate the system behaviour and for very small residuals, little inaccuracies in the computer code often prevent convergence below a certain threshold. Therefore, basing α on a test for quadratic convergence is too restrictive. Instead, we require a specified reduction in r within a fixed number of steps:

$$r^l/r^0 < \epsilon \text{ for some } l \leqslant l_{\max}. \tag{22}$$

If this criterion is not met for a line, then α is increased. If it is, we consider the iteration converged.

The convergence criterion (22) is different from the usual NR convergence criterion (which ends the iteration when $r^l < \epsilon$), for a very good reason. It is expensive to find the smallest α exactly by trial and error, so each time when the criterion (22) is not met, we increase α significantly (by a factor 10, in our experiments). But in situations where the criterion is *almost* met without increasing α, a very small reduction in the initial residual r^0 could mean that the criterion $r < \epsilon$ is suddenly met without a higher α. This leads to many sudden changes in α for different sweeps in the same line; such 'jittery' behaviour can prevent convergence of the Gauss–Seidel smoothing. When the relative convergence criterion (22) is used, α is more consistent between successive sweeps of the same line.

4.2.3. Smoothing algorithm

Summarizing, the algorithm for the Gauss–Seidel smoothing in one line is:

```
$\alpha = \alpha_0$                                                        initial $\alpha$
**while** $\alpha \leqslant \alpha_{\max}$
    $\boldsymbol{q}^0_{\text{NR}} = (\boldsymbol{q}^n_k)_{i,j\in\text{line}}$                               initial state
    **for** $l = 1, l_{\max}$
        $\boldsymbol{q}^l_{\text{NR}} = \boldsymbol{q}^{l-1}_{\text{NR}} - \left(\frac{\partial \mathcal{F}_k}{\partial \boldsymbol{q}_k} + \alpha I_4\right)^{-1}_{\text{line}} (\mathcal{F}_k + \alpha I_4)_{\text{line}}(\boldsymbol{q}^{l-1}_{\text{NR}})$     NR step
        **if** $r^l/r^0 < \epsilon$ **end**
  **end for**
    $\alpha := \alpha_{\text{factor}} \cdot \alpha$                                          new $\alpha$
**end while**
```

As stated above, this is an estimator for the smallest α, it is not exact. But practical experience shows that this algorithm functions well. We found that suitable values for the parameters (for flows with $u, v, p = \mathcal{O}(1)$) are $\alpha_0 = 10^{-2}$, $\alpha_{\max} = 10^7$, $\alpha_{\text{factor}} = 10$, $\epsilon = 10^{-5}$, and $l_{\max} = 10$.

We remark that in many solution methods for the RANS equations, e.g. [11,21], the source term is split in a positive and a negative part. For each smoothing step or time step, the positive part (the production) is kept constant, while the negative part (the dissipation) is updated together with the convective and diffusive fluxes. This is a kind of damping too, it increases the diagonal dominance of L_k. The difference with our algorithm is, that this damping by source-term splitting is not switched off, even when it is no longer needed because the solution approaches convergence. Therefore, our damping algorithm is more efficient.

4.3. Linear multigrid

Nonlinear multigrid (NMG), as described in Section 3.2, does not work for the RANS equations. In tests, application of NMG usually gave worse convergence than Gauss–Seidel smoothing alone, or even caused divergence of the solution process. The $\Re(\lambda) \leqslant 0$ eigenvalues described in the previous two sections are *not* the cause of this problem: multigrid has been applied successfully to problems with both positive and negative eigenvalues (see e.g. [2], section A8.5.3 for the solution of the Helmholtz problem). Given a good smoother, which we have, multigrid ought to work. But it does not.

4.3.1. NMG operators fail

The problem is, that the coarse grid operators $\mathcal{F}_k$ do not resemble the fine grid operator $\mathcal{F}_K$ sufficiently. The source term in Menter's model is the small difference of two large variables (production and dissipation), that contain products of both first and second spatial derivatives. Thus, small errors in these derivatives may cause large differences in the source term; the turbulence model only makes sense when the grid is fine enough to resolve the interior structure of a boundary layer well. Therefore, the model needs a minimum grid resolution to be accurate, typically about 20 cells over the thickness of a boundary layer. For coarser grids, the solution becomes highly grid-dependent. An example for a boundary layer flow is given in Fig. 3. The solution on the 32×32 cell grid is accurate (compare with the fine grid solution in Fig. 7), but the solutions on the coarser grids are a lot worse. Note that the maximum value for $\tilde{\nu}_T$ is off by about 30% on the 16×16 grid and that this maximum is actually lower than the one on the 8×8 grid.

If the solutions on the coarse grids differ from the solutions on the fine grid, then the operators that produce these solutions differ too. Therefore, the coarse grid operators $\mathcal{F}_k$ are not suitable for constructing approximate inverses for $\mathcal{F}_K$ and the NMG coarse grid correction does not work. This is confirmed by our tests, which showed that acceptable coarse grid corrections could be obtained with NMG, but only when the coarsest grid used was fine enough to accurately resolve the boundary layers. If coarser grids are used, the convergence deteriorates. However, for good multigrid convergence, proper smoothing on these coarse grids is essential.

4.3.2. Galerkin operators

So, to get proper coarse grid corrections, we use Galerkin coarse grid operators. Instead of restricting the state on the fine grid to the coarser grids and constructing coarse grid operators with these states, we directly restrict the fine grid operator to the coarse grids. Thus, we can be sure that the coarse grid operators resemble the fine grid operator reasonably well. The Galerkin operators are found by first constructing a linearised version of the fine grid operator and then restricting this linear operator to the coarse grids. So we switch from nonlinear to linearised coarse grid corrections. Therefore, we no longer compute states $\boldsymbol{q}$ on the coarse grids, but small corrections $\boldsymbol{u}$.

The linearised version of the fine grid operator $\mathcal{F}_K$ is the discretised equivalent of the operator L from Eq. (14), i.e. the Jacobian of $\mathcal{F}_K$:

$$L_K = \frac{\partial \mathcal{F}_K(\boldsymbol{q}_K)}{\partial \boldsymbol{q}_K}. \tag{23}$$

For a finite-volume discretisation with a five-point stencil, as introduced in Section 3, the linear operator in one cell is:

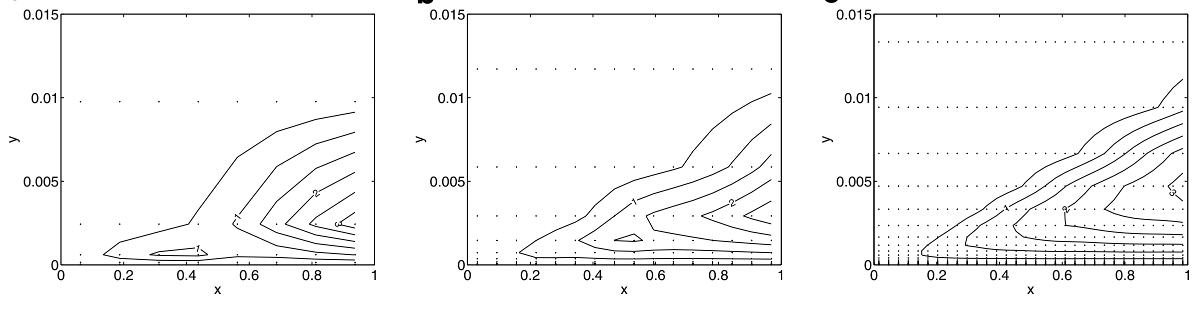

Fig. 3. Grid convergence study for a zero pressure gradient boundary layer flow at $Re = 10^7$. Solutions for $\tilde{\nu}_T$ on an 8×8 grid (a), 16×16 grid (b), and 32×32 grid (c). The cell centres are marked by $+$. (The test case is also described in Section 6.2.)

$$(L_K \boldsymbol{u}_K)_{i,j} = L^{0,0}_{i,j}(\boldsymbol{u}_K)_{i,j} + L^{1,0}_{i,j}(\boldsymbol{u}_K)_{i+1,j} + L^{0,1}_{i,j}(\boldsymbol{u}_K)_{i,j+1} + L^{-1,0}_{i,j}(\boldsymbol{u}_K)_{i-1,j} + L^{0,-1}_{i,j}(\boldsymbol{u}_K)_{i,j-1}, \tag{24a}$$

where

$$L^{0,0}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i,j}}, \quad L^{1,0}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i+1,j}}, \quad L^{0,1}_{i,j} = \frac{\partial(\mathcal{F}_K)_{i,j}}{\partial(\boldsymbol{q}_K)_{i,j+1}}, \quad \text{etc.} \tag{24b}$$

So for each cell $(\Omega_K)_{i,j}$, the operator L_K consists of five 4×4 matrices.

The coarse grid operators then follow from Galerkin's principle:

$$L_{k-1}\boldsymbol{u}_{k-1} = R^{k-1}_k L_k P^k_{k-1} \boldsymbol{u}_{k-1}. \tag{25}$$

For the prolongation (7) and the restriction (8), this means that the coarse grid operator L_{k-1} can be evaluated by copying the correction $\boldsymbol{u}_{k-1}$ in all coarse cells $(\Omega_{k-1})_{i,j}$ to the four fine cells $(\Omega_k)_{2i(+1),2j(+1)}$ that lie in the same location as each coarse cell, then evaluating the fine grid operator L_k for this $\boldsymbol{u}_k$ and finally summing the fluxes from each group of four cells. This, in turn, means that the matrices for L_{k-1} are found by a summation of the matrices in L_k: the 20 matrices of the four operators in the fine cells corresponding to coarse cell $(\Omega_{k-1})_{i,j}$ are summed to form the five matrices in $(L_K)_{i,j}$, such that a matrix for a fine cell is added to the matrix for that coarse cell in which the fine cell lies. Fig. 4 demonstrates this.

4.3.3. LMG algorithm

With these coarse grid operators, the linear multigrid algorithm is constructed. It is different for the finest level and the coarser levels as only the operator on the finest grid is nonlinear:

```
$\boldsymbol{q}^{n+1}_K$ = function $LMG(\boldsymbol{q}^n_K)$ :
$\tilde{\boldsymbol{q}}^n_K = (M_K)^{q_1}\boldsymbol{q}^n_K$                                   $q_1$ pre-relaxation steps,
$L^n_K = \partial\mathcal{F}_K(\tilde{\boldsymbol{q}}^n_K)/\partial\boldsymbol{q}_K$                          linearisation,
$\boldsymbol{s}^n_{K-1} = R^{K-1}_K\mathcal{F}_K(\tilde{\boldsymbol{q}}^n_K)$                          source term,
$L^n_{K-1} = R^{K-1}_K L^n_K P^K_{K-1}$                               linearised system,
$\boldsymbol{u}^n_{K-1} = 0$                                            initial solution,
$\boldsymbol{u}^{n+1}_{K-1} = LMGC^{\sigma}(K-1, \boldsymbol{u}^n_{K-1}, \boldsymbol{s}^n_{K-1}, L^n_{K-1})$       MG on coarser grid,
$\tilde{\tilde{\boldsymbol{q}}}^n_K = \tilde{\boldsymbol{q}}^n_K + P^K_{K-1}\boldsymbol{u}^{n+1}_{K-1}$                         correction,
$\boldsymbol{q}^{n+1}_K = (M_K)^{q_2}\tilde{\tilde{\boldsymbol{q}}}^n_K$                                 $q_2$ post-relaxation steps.

$\boldsymbol{u}^{n+1}_k$ = recursive function $LMGC(k, \boldsymbol{u}^n_k, \boldsymbol{s}^n_k, L^n_k)$ :
$\tilde{\boldsymbol{u}}^n_k = (M^L_k)^{q_1}\boldsymbol{u}^n_k$                                   $q_1$ pre-relaxation steps,
if $k \neq 0$ then
    $\boldsymbol{s}^n_{k-1} = R^{k-1}_k L_k \tilde{\boldsymbol{u}}^n_k$                             source term,
    $L^n_{k-1} = R^{k-1}_k L^n_k P^k_{k-1}$                             linearised system,
    $\boldsymbol{u}^n_{k-1} = 0$                                        initial solution,
    $\boldsymbol{u}^{n+1}_{k-1} = LMGC(k-1, \boldsymbol{u}^n_{k-1}, \boldsymbol{s}^n_{k-1}, L^n_{k-1})$       MG on coarser grid,
    $\tilde{\tilde{\boldsymbol{u}}}^n_k = \tilde{\boldsymbol{u}}^n_k + P^k_{k-1}\boldsymbol{u}^{n+1}_{k-1}$                       correction,
end if
$\boldsymbol{u}^{n+1}_k = (M^L_k)^{q_2}\tilde{\tilde{\boldsymbol{u}}}^n_k$                                 $q_2$ post-relaxation steps.
```

For the smoothing on the coarser grids, we use line Gauss–Seidel, just like on the fine grid. Only, the Newton–Raphson iteration is not needed anymore, as the linear flow equations in a line can be solved exactly in one step. As for the NMG algorithm, each smoothing step M_k consists of both a horizontal and a vertical sweep. This smoothing is relatively expensive, but it increases the robustness of the procedure.

4.3.4. Comments on the algorithm

The change from nonlinear to linear coarse grid corrections is not that big. For small residuals, the NMG algorithm from Section 3.2 is more or less linear anyway: the computation of $\mathcal{F}_{k-1}(\boldsymbol{q}^{n+1}_{k-1}) = \mathcal{F}_{k-1}(\boldsymbol{q}^n_{k-1}) + \boldsymbol{d}^n_{k-1}$

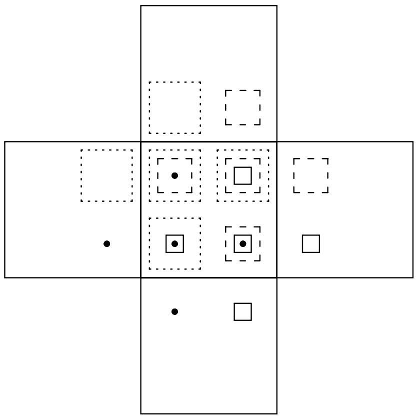

Fig. 4. Construction of a coarse grid linear operator. The matrices in the coarse five-point stencil (—) are made by summing four fine grid stencils (· · ·, --, –, •).

is a matrix-free evaluation of the Jacobian of $\mathcal{F}_{k-1}$ when $\boldsymbol{d}^n_{k-1}$ is small, i.e. when $\mathcal{F}_{k-1}(\boldsymbol{q}^{n+1}_{k-1}) \approx \mathcal{F}_{k-1}(\boldsymbol{q}^n_{k-1}) + L_{k-1}(\boldsymbol{q}^{n+1}_{k-1} - \boldsymbol{q}^n_{k-1})$. And as complex systems like the RANS equations cannot handle large source terms, the defect $\boldsymbol{d}$ is *always* made small by reducing the parameter w when $\boldsymbol{d}$ is large (see e.g. [16]); reducing w is equivalent to linearising the coarse grid operators.

The only real difference between the NMG and LMG algorithms is that the LMG coarse grid operators are Galerkin operators. According to the literature, Galerkin operators may cause problems (see e.g. [2], section 5.4). In particular, Galerkin coarse grid operators for convection problems are less diagonally dominant than the fine grid operator. As a consequence, the convergence rate for multigrid becomes worse when more coarse grids are added. In our experiments (Section 6) this effect is noted, but it causes no problems.

The unusual combination of a linear coarse grid correction with nonlinear smoothing on the finest grid is chosen for two reasons. First, the nonlinear smoother is robust, it acts like a ‘safety net’ for the solution process. It can correct small unphysical solutions ($\tilde{\nu}_{\mathrm{T}} < 0$, for example) that arise from the coarse grid correction. Furthermore, it performs well in extreme situations, like the start of the solution process from a uniform flow, where the linear multigrid method is not (yet) effective. And second, we use the damping factor α from the line smoothing (Eq. (19)) in the coarse grid correction too, i.e. we replace (23) with:

$$L_K = \frac{\partial(\mathcal{F}_K(\boldsymbol{q}_K) + \alpha I_4 \boldsymbol{q}_K)}{\partial \boldsymbol{q}_K}. \tag{26}$$

As explained in the introduction to this section, this damping is not really needed for the convergence of multigrid, but it makes sure that the smoothing on all coarse grids is stable (we cannot set α with the aid of Newton–Raphson on the coarse grids, because the coarse grid operators are linear). The nonlinear smoothing on the finest grid is needed to find the α in Eq. (26): for each cell, we choose α as the minimum of the α’s for that cell in the horizontal and vertical line sweep.

Concerning the computational costs of the linear multigrid algorithm, a single coarse grid correction step takes less CPU time than a nonlinear one. Computing L_K, the linearisation of $\mathcal{F}_K$, is cheap because the Jacobian of $\mathcal{F}_K$ in each cell is already being computed for the nonlinear line smoothing. The restriction of the operators L_k does not take much time either, as it consists of additions only. And finally, the coarse grid smoothing becomes faster, because it does not need Newton–Raphson anymore and because the evaluation of the linear operators is faster than the computation of nonlinear fluxes. If the Galerkin operators require more iterations to reach convergence, then the total CPU time may go up a little; this depends on the individual problem.

The main increase is in the memory usage. Storing the linearised operators takes $5 \cdot (4 \cdot 4) = 80$ reals per cell, much more than the storage needed for nonlinear multigrid. There are two ways to reduce these costs: one is to restrict the fine grid operator L_K to the next coarser grid while it is being computed. This means that storing L_K itself is not needed, a significant gain as more than 75% of all cells lie in the finest grid. The other is to store the linearised operators with a low precision. Their inverse is not computed exactly anyway, so double precision accuracy is not needed for the L_k. This may save another 50–75% in memory.

4.3.5. Full multigrid

As initial condition for the multigrid solution, we usually choose $\tilde{\nu}_T$ very close to zero. Then we can see two distinct 'stages' in the solution process. During the first iterations the boundary layers develop and the turbulence intensity grows, often with a factor $\mathcal{O}(10^4)$ or more. Typically, we see the residual in the turbulence equation increase in this first stage, because the turbulence intensity increases. Then, when the boundary layers have more or less developed, the residuals start to decrease: this is the second stage. Experiments show that multigrid is only effective in this second stage.

Therefore, full multigrid (FMG) is essential for our method. We do not start the solution on the finest grid, but on the coarsest grid on which the boundary layers can be accurately resolved. When the solution on this grid is computed, it is prolongated as initial solution to the next finer grid where the solution is computed too. This is continued until the finest grid is reached. Thus, all but the first computations start at the second stage, with developed boundary layers. The time-consuming development of turbulence is only needed on the coarsest grid.

5. Finite-volume discretisation

The multigrid solution technique presented in the previous sections can, in principle, be applied to different types of discretisations. Therefore, the description of the discretisation for the system (1), (2) was kept general. In this section, we discuss the finite-volume discretisation that is used to run the tests in Section 6; fluxes, source terms and the implementation of boundary conditions are described. The discussion focuses on those aspects that are specific for the turbulence modelling. Each part describes both the first-order accurate discretisation used with multigrid and the second-order discretisation that is solved with defect correction.

As already mentioned in Section 3.1, the RANS equations are discretised on structured curvilinear grids, with a technique similar to e.g. those used in [6,5]. The compact, five-point stencil from Fig. 1 is enough to discretise most of the first-order accurate operator. For the second-order accurate operator, more cells are needed. The fluxes across the cell faces are discretised in two parts: the convective and the diffusive fluxes are computed separately. The convective flux is discretised with an approximate Riemann solver based on artificial compressibility; the diffusive flux is computed with central differences. The turbulent source term forms a third part, it is discretised with finite differences.

5.1. Convective fluxes

Most turbulent flows have a high Reynolds number. Thus, the influence of diffusion in the flow equations is low in most places. From a numerical perspective, this means that the diffusive part cannot be counted upon to stabilise the solution: we need a discretisation of the *convective* fluxes that is stable in itself. This is the main reason for discretising the convective and diffusive fluxes separately: thus, we can directly control the stability of the convective part.

The convective flux function is based on artificial compressibility [22,23] and comparable to the flux used in [18,24,6]. In the time-dependent RANS equations, an artificial time derivative is added to the continuity equation. The resulting hyperbolic system is then discretised with an approximate Riemann solver. The time derivatives are only used to derive this flux.

Our solver is a linearised Osher-type flux function, it couples the normal velocity and the pressure on the two sides. The Riemann solution consists of three waves: two pressure waves (one running left, one right) and a contact discontinuity (running left or right):

$$\lambda^- = \frac{1}{2}u - \sqrt{c^2 + \left(\frac{1}{2}u\right)^2}, \quad \lambda^0 = u_{\frac{1}{2}}, \quad \lambda^+ = \frac{1}{2}u + \sqrt{c^2 + \left(\frac{1}{2}u\right)^2}, \tag{27}$$

with c a constant. The pressure waves give the state $\boldsymbol{q}_{\frac{1}{2}}$ at the cell face, as a function of the left state $\boldsymbol{q}_0$ and the right state $\boldsymbol{q}_1$ (with normal velocity u and tangential velocity v):

$$\begin{aligned} u_{\frac{1}{2}} &= u_0 + \frac{p_1 - p_0 + \rho_1\lambda_1^-(u_1 - u_0)}{\rho_1\lambda_1^- - \rho_0\lambda_0^+}, \\ p_{\frac{1}{2}} &= p_0 - \rho_0\lambda_0^+\frac{p_1 - p_0 + \rho_1\lambda_1^-(u_1 - u_0)}{\rho_1\lambda_1^- - \rho_0\lambda_0^+}, \end{aligned} \tag{28}$$

which also defines the wave speed λ^0. Then the other two state variables v and $\tilde{v}_{\mathrm{T}}$ are chosen upwind:

$$\begin{aligned} v_{\frac{1}{2}} &= v_0, \quad \tilde{v}_{T\frac{1}{2}} = \tilde{v}_{T0} \quad \text{if } u_{\frac{1}{2}} \geqslant 0, \\ v_{\frac{1}{2}} &= v_1, \quad \tilde{v}_{T\frac{1}{2}} = \tilde{v}_{T1} \quad \text{if } u_{\frac{1}{2}} < 0. \end{aligned} \tag{29}$$

We construct convective fluxes with these state variables.

The states $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ at the cell faces are reconstructed from the states in the cell centres. For the second-order accurate fluxes, a limited upwind scheme is used [4,5]. It is known that the Minmod limiter is unsuitable for use with defect correction [15]. Instead, we use the $\kappa = \frac{1}{3}$ limiter proposed by Koren [25]. For the first-order fluxes, $\boldsymbol{q}_0$ and $\boldsymbol{q}_1$ are the states in the adjacent cells.

5.2. Diffusive fluxes

On rectangular grids, the first derivatives in the diffusive fluxes are discretised with central differences; this gives a stable discretisation. For our non-rectangular grids, Peyret control volumes are used. The integral of a derivative over a control volume around a cell face, as in Fig. 5, can be transformed into a boundary integral. Approximating with an average derivative and average states on the control volume faces, it is found that:

$$\begin{aligned} u_x &\approx \frac{1}{A_d}\oint_{\partial\Omega_d} u\,\mathrm{d}y \\ &\approx \frac{1}{A_d}\left(\frac{u_1 + u_2}{2}(y_2 - y_1) + u_3(y_4 - y_2) + \frac{u_4 + u_5}{2}(y_5 - y_4) + u_i(y_1 - y_5)\right), \end{aligned} \tag{30a}$$

$$\begin{aligned} u_y &\approx -\frac{1}{A_d}\oint_{\partial\Omega_d} u\,\mathrm{d}x \\ &\approx \frac{1}{A_d}\left(\frac{u_1 + u_2}{2}(x_1 - x_2) + u_3(x_2 - x_4) + \frac{u_4 + u_5}{2}(x_4 - x_5) + u_i(x_5 - x_1)\right), \end{aligned} \tag{30b}$$

with

$$A_d \approx \frac{1}{4}(A_1 + A_2 + A_4 + A_5 + 2(A_3 + A_i)). \tag{30c}$$

The same equation is used for v and $\tilde{v}_{\mathrm{T}}$. This discretisation is second-order accurate for sufficiently smooth grids, so it is used in both the first-order and second-order scheme.

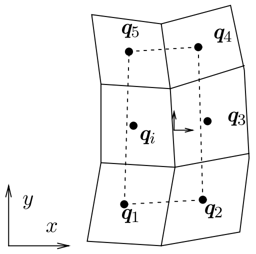

Fig. 5. Control volume for diffusive flux discretisation.

Compared with the diffusive fluxes in the laminar incompressible Navier–Stokes equations, the RANS equations cause two complications. First, the cross-derivative terms like u_{xy} (u_y in the x-direction flux) do not cancel, because the viscosity is not constant: the diffusive operator is similar to the one for the laminar compressible Navier–Stokes equations. For the discretisation, this means that the diffusion operator requires a nine-point stencil, even on rectangular grids, as each cell needs the states in its four neighbours and in the four cells on its diagonals. We see that Eq. (30a) reduces to a central difference equation in u_3 and u_i, but that Eq. (30b) does not. In the linear multigrid algorithm (Section 4.3), this is currently ignored: only the five-point part of the stencil is restricted. But as the convection and the source term are discretised on the five-point stencil only, no convergence problems appear in practice.

A second point is the choice of $\tilde{\nu}_T$ at the cell faces. In the convective fluxes, $\tilde{\nu}_T$ is chosen upwind (Eq. (29)), but using this same viscosity in the diffusion operators causes severe instability in the Gauss–Seidel smoothing. This is caused by the cell faces parallel to the flow, where a small change in velocity, from positive to negative or vice versa, causes a discontinuous change in the face viscosity. Using a central approximation, the average of the two cells next to the cell face, is a possibility, but tests showed that this choice sometimes causes instability too. At this moment, we use the $\tilde{\nu}_T$ from one of the two cells. Which cell, is determined in advance and not changed during the computation. We pick the upwind cell for faces normal to the flow direction and the cell closest to the nearest wall for parallel faces.

5.3. Turbulent source term

The source term in (2) contains first- and second-order derivatives. Furthermore, being a source term, it cannot be converted to a boundary integral over the cell faces. And computing it with the cell face states is not a good idea: these upwind states can change discontinuously with small changes in the velocity (see Section 5.2). Therefore, the source term is approximated with finite differences based on the cell centre states. On our curved, non-uniform grids, two adaptations of the standard finite-difference stencils are needed.

First, when the grid is curved, the centres of a cell and its neighbours are not in line and the line between the upper and lower neighbour is not orthogonal to the line between the left and right neighbour. Therefore, we fit a local orthogonal axis system to each cell (Fig. 6) and project the cell centre locations on these axes. Then the derivatives are computed with the cell states in these projected cell centres. This projection step causes some errors, but these are of second-order in the grid size if the grid is sufficiently smooth. The rotation of the local axes has no influence, as the source term is invariant under rotation.

On the orthogonal axes, finite-difference stencils for non-uniform grids are used. In x-direction, these stencils are:

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}, \quad u_{xx} \approx \frac{\frac{u_{i+1}-u_i}{x_{i+1}-x_i} - \frac{u_i-u_{i-1}}{x_i-x_{i-1}}}{\frac{1}{2}(x_{i+1} - x_{i-1})}. \tag{31}$$

With these derivatives known in each cell centre, the source terms (4) and (5) can be computed in the cell centres. The schemes (31) are in principle first-order accurate, but if the grids are smooth, the accuracy increases

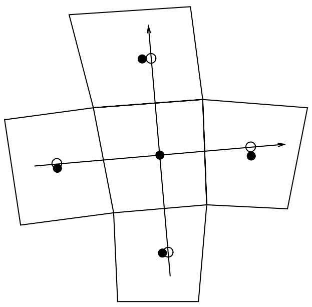
Fig. 6. Local axes and projected cell centre locations for source term computation.

to second order. Therefore, this source term discretisation is used in both the first-order and the second-order accurate schemes.

5.4. Boundary conditions

The convective and diffusive operators require different boundary conditions. The number of convective boundary conditions depends on the type of boundary, while the diffusion and the source term require u, v, and $\tilde{v}_T$ on all boundaries.

On an inflow boundary, Dirichlet conditions are specified for u, v, and $\tilde{v}_T$. These are the same for the convection and the diffusion. p is not specified, it follows from the convective flux function. $\tilde{v}_T$ is set to a small positive value, typically 0.01ν (see Section 2.2). A no-slip wall has Dirichlet conditions too, there u, v, and $\tilde{v}_T$ are all zero.

At a symmetry wall, the normal velocity is zero, while the tangential velocity and $\tilde{v}_T$ have zero normal derivatives. Once again, these conditions are the same for convection and diffusion.

Finally, at an outflow boundary, a Dirichlet condition is specified for the pressure. Convection requires no boundary conditions for the other three state variables. For the diffusion and the source term, weak conditions are imposed: homogeneous Neumann conditions.

6. Test cases

The performance of the multigrid algorithm is assessed with four test cases. The first is a laminar flow, that is computed with linear and nonlinear multigrid. After that, turbulent boundary layer flows are computed. The last two test cases are more complex flows over airfoils.

6.1. Laminar flow

To compare the current linear multigrid algorithm with nonlinear multigrid, a laminar flow is computed with both methods. The test case is the flow in a laminar flat-plate boundary layer with zero pressure gradient, the Reynolds number is $Re = 2000$. Our grid has 128×128 cells and is highly stretched in y-direction. Multigrid computations are done on six grids, the coarsest grid has 4×4 cells. For both algorithms, the same Gauss–Seidel smoothing is used on the fine grid and in both cases, the discretisation is the same first-order accurate one. The only difference is in the coarse-grid corrections.

Table 1 gives a comparison of the results for the FMG solution process. For each grid, from the coarsest to the finest, the table gives the number of iterations needed to reach convergence on that grid (sum of the residuals $<10^{-6}$). It also lists the computation time needed to get the solution on that particular grid. The computations were performed on a 2.2 GHz PC.

The table shows an adverse effect of the Galerkin operators (see Section 4.3): on the finer grids, more iterations are needed for the LMG than for the NMG solution process. The LMG computation times are higher too, but they do not increase as fast as the number of iterations. Therefore, the LMG coarse grid corrections are indeed cheaper than the NMG coarse grid corrections. All in all, this laminar test case shows that the LMG gives similar efficiency as the standard NMG.

Table 1
Laminar boundary layer, iterations, and computation time per grid

Grid	Iterations		t (s)	
	NMG	LMG	NMG	LMG
4×4	4	4	0.1	0.1
8×8	5	5	0.1	0.1
16×16	5	5	0.5	0.4
32×32	5	7	1.5	1.9
64×64	6	9	7.7	9.4
128×128	7	10	37.3	39.7

6.2. Boundary layers

The first turbulent test cases are two boundary layer flows over flat plates: simple flows that are dominated by the development of turbulence in the boundary layer. These cases illustrate the efficiency of linear multigrid for turbulent flows and compare the performance on coarse and fine grids.

Boundary layer flows. The first flow is a boundary layer with no pressure gradient, at a Reynolds number of 10^7 based on the length of the plate. This test case is used in Menter's paper [13]. The grid is the same as for the laminar boundary layer and multigrid is used with six grids. Results are given in Fig. 7. The first figure gives the velocity profile and the second figure the turbulent viscosity $\tilde{\nu}_T$. We see that this viscosity is (almost) zero in the far field, then increases in the boundary layer and returns to zero at the wall. Compared with the velocity profile, the highest turbulence intensity occurs high in the boundary, where the velocity is close to 1. The last figure gives the velocity profile in wall coordinates; good agreement is found with the theoretical profiles in the viscous sublayer and the logarithmic layer.

Similar results are found for a more difficult flow, a boundary layer with an increasingly adverse pressure gradient over the plate. This flow was investigated experimentally by Samuel and Joubert [26], it has a Reynolds number of 7.055×10^6 based on the plate length. The grid is the same as for the previous test case. Results in Fig. 8 show how the adverse pressure gradient slows down the flow. The boundary layer is thicker than in the previous case and the turbulence intensity is higher. A comparison of a velocity profile with experimental measurements shows excellent agreement (Fig. 8c).

Multigrid convergence. Fig. 9a and b give the convergence of the residual during the multigrid computation. The finest grid is very fine for a boundary layer grid (128 cells in vertical direction), the flow can be resolved on most coarse grids too. Therefore, we start the FMG computation on the second (8×8) grid. Here, the devel-

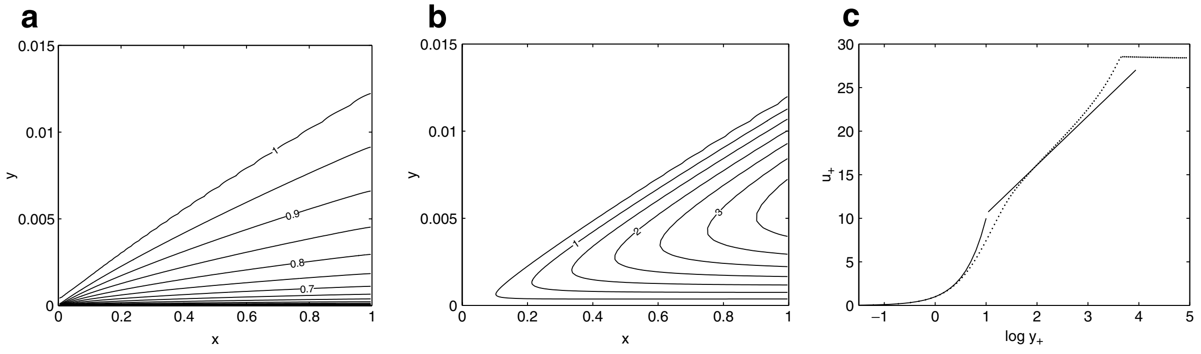

Fig. 7. Turbulent flat-plate flow at $Re = 10^7$, zero pressure gradient. Isoline plots of velocity $|\boldsymbol{u}|$ (a), turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (b), and the velocity profile at $x = 1$ in wall coordinates (···) compared with analytical solution (—) (c).

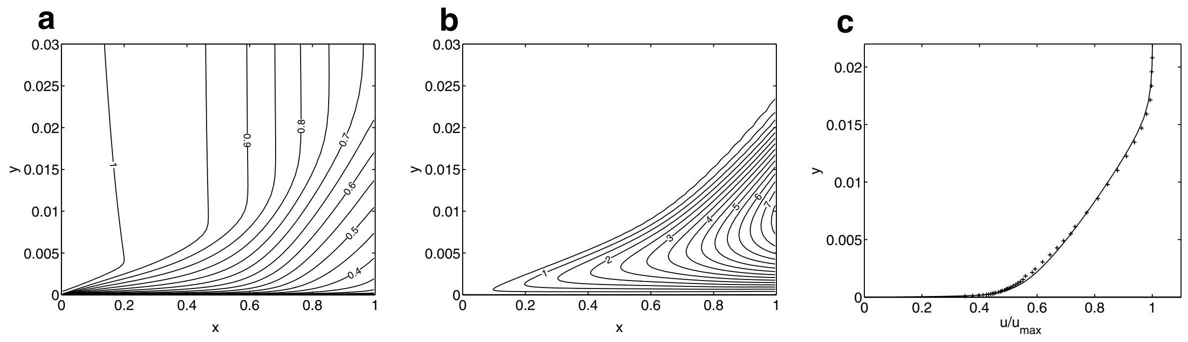

Fig. 8. Turbulent flat-plate flow at $Re = 7.055 \times 10^6$, adverse pressure gradient. Isoline plots of velocity $|\boldsymbol{u}|$ (a), turbulent viscosity $\tilde{\nu}_T \cdot 10^5$ (b), and the velocity profile at $x = 0.82$ (—) compared with measurements (+) from [26] (c).

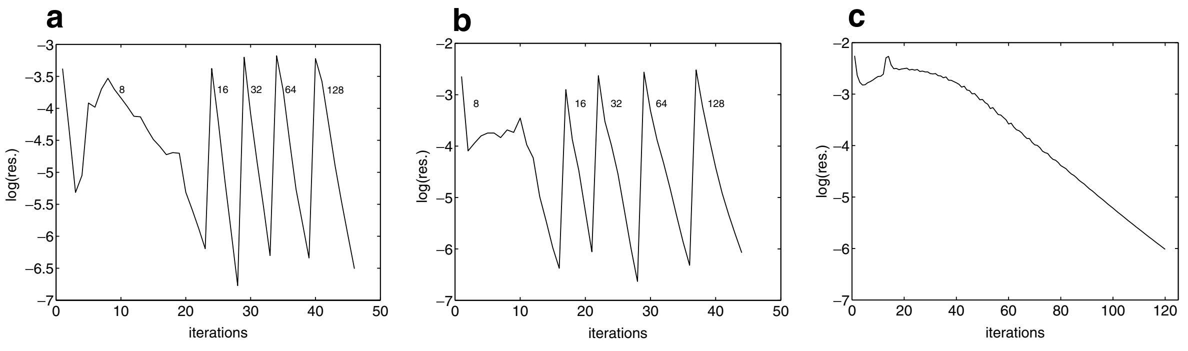

Fig. 9. FMG convergence for boundary layer flows under zero (a) and adverse (b) pressure gradients. Graph (c) gives single-grid convergence for the same case as (a).

opment of the boundary layer, when the residual rises and falls, is clearly seen. On all subsequent grids, the convergence is excellent. The convergence rate does not deteriorate much on the finer grids, which means that the Galerkin operators cause few problems. In fact, convergence on the fine grids is faster than in the preceding laminar case. Also, the convergence is similar for both boundary layers; multigrid performance does not get worse in adverse pressure conditions.

For comparison, Fig. 9c shows the convergence when the zero pressure gradient flow is solved with line Gauss–Seidel alone, on a single grid. This takes 120 iterations on the fine grid, compared to 7 for the multigrid solution. 40 iterations are needed for the development of the boundary layer. The total CPU time is about 5 times higher than for the multigrid solution.

6.3. NACA 0012 airfoil

A more complex test case than the flat-plate boundary layer is the flow over a NACA 0012 airfoil. It features stagnation points, curved walls and the transition from a boundary layer to a turbulent wake. The angle of attack $\alpha = 0$ and the Reynolds number is $Re = 10^6$. Because the NACA 0012 airfoil is symmetric, the flow is computed in the upper half of the flow domain only. An H-type grid is used with 512 cells in x-direction and 256 cells in y-direction, the grid is stretched near the airfoil surface and near the leading and trailing edges. The problem is used to test multigrid convergence for a more difficult flow and to study the convergence of defect correction for the second-order accurate discretisation.

Flow field. The flow around the airfoil is given in Fig. 10. The velocity plot 10a shows the stagnation point at the leading edge, the suction area above the airfoil, the growing boundary layer and the transition from the boundary layer to the wake. The second figure shows the pressure and the last figure the turbulent viscosity. We see here that the turbulence intensity is very low near the leading edge, it starts to grow where the pressure

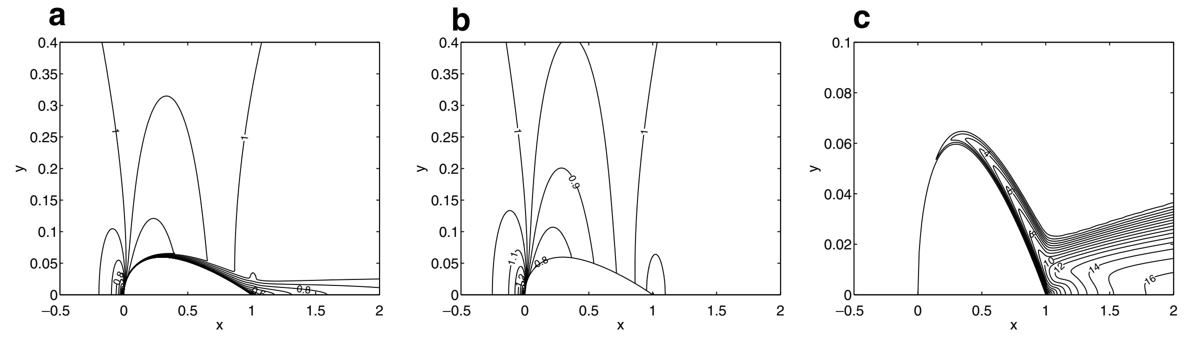

Fig. 10. Flow over a NACA 0012 airfoil (second-order accurate solution, after five DC steps). These plots show the velocity $|\boldsymbol{u}|$ (a), the pressure p (b), and the turbulent viscosity $\tilde{\nu}_T \times 10^5$ (c). For clarity, all plots are stretched in y-direction.

gradient becomes adverse. Behind the airfoil, the location of the maximum value for the turbulence gradually shifts from the centre of the boundary layer to the centre of the wake.

Multigrid convergence. The flow is solved with multigrid on six grids; the coarsest grid has 8×16 cells. As in the boundary layer case, the FMG computation is started on the second grid. But the NACA 0012 flow has more different features than the simple boundary layer flows, so it cannot be resolved accurately on the 16×32 cell grid. Therefore, multigrid does not work on this grid. To retain the full advantage of the FMG algorithm, the flow is solved with single-grid smoothing on the 16×32 grid. From the 32×64 grid on, multigrid is used.

The convergence of the residual is shown in Fig. 11a. The number of iterations per grid is a little higher than for the preceding boundary layer flows, but the initial residual on each grid is higher too (because the flow is more complex), so that is to be expected. Convergence does not deteriorate much on the finer grids, which shows once again that the Galerkin operators work correctly. There is a little 'bump' in the convergence on the 32×64 grid, probably caused by the development of the boundary layer that is not sufficiently resolved on the 16×32 grid.

For comparison, the single-grid convergence is shown in Fig. 11b. This convergence is odd: the residual stays constant for a long time and then suddenly decreases. This behaviour is abnormal for these types of flows, the usual single-grid convergence plots look like Fig. 9c. But it does show that there are cases where the FMG solver can compute flows that cause problems for single-grid smoothing.

Finally, Fig. 11c shows the residual for the first 25 defect correction steps, starting from the first-order accurate solution on the 256×512 grid. In the first five iterations, the residual decreases a little at a relatively high speed. The asymptotic convergence rate is slower; iterating to convergence is expensive. Luckily, this is not needed: a grid convergence study (not shown) proves that the solution after five defect correction steps is converged and much more accurate than the first-order solution.

6.4. Supercritical airfoil

The low-Mach flow over a supercritical airfoil was measured by Nakayama [27]. This airfoil is placed at an angle of attack $\alpha = 4°$, the Reynolds number is 1.2×10^6. Computation of this flow is very challenging, as the flow field has high curvature and strong pressure gradients near the trailing edge. In the same location, two boundary layers of different strength merge to form the turbulent wake. The flow is computed on a 256×256 cell H-type grid.

Flow field. The most typical feature of the Nakayama wing is the concave lower side near the trailing edge. The strong curvature in the flow field there can be seen in the results (Fig. 12). The plots show the stagnation point and a strong suction peak near the leading edge, and a moderate high pressure region in the hollow below the trailing edge. The turbulence intensity near the trailing edge is interesting: because of the adverse pressure gradient above the airfoil, the turbulence intensity is higher above than below the airfoil. The two boundary layers merge, so the turbulence levels have to adapt. For the upper boundary layer, the reduction in turbulence intensity happens in a thin layer, clearly visible in Fig. 12c. This layer is *not* aligned with the flow, it runs upwards into the flow.

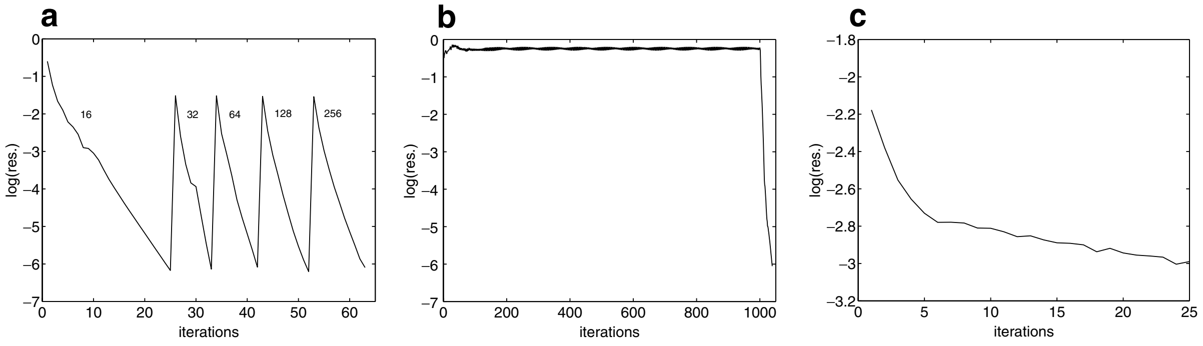

Fig. 11. NACA 0012, convergence of the residual on the 256×512 grid, for multigrid (a), single-grid Gauss–Seidel (b), and second-order accurate defect correction (c).

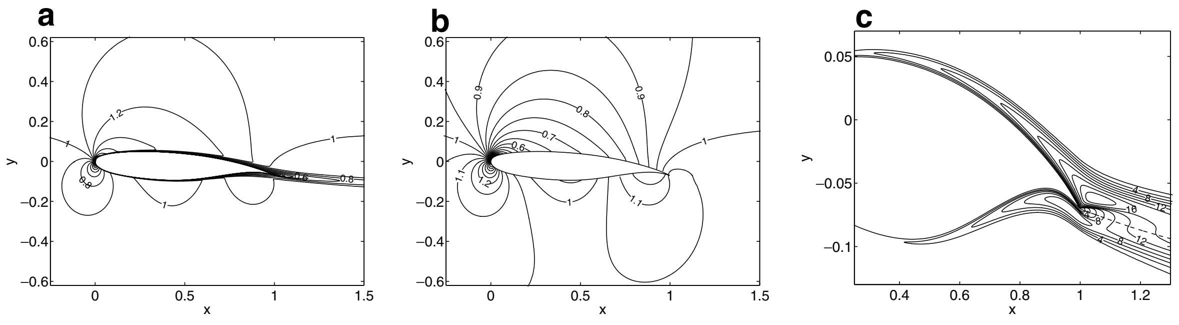

Fig. 12. Flow over a supercritical airfoil (second-order accurate solution, after five DC steps); velocity $|\boldsymbol{u}|$ (a), pressure p (b), and turbulent viscosity $\tilde{\nu}_T \times 10^5$ (c). Plot (c) is stretched in y-direction, - - -: trailing edge streamline.

Grid convergence, comparison with experiment. A grid convergence study is done for the supercritical airfoil. Solutions on three grids are compared with Nakayama's experimental data [27]. Fig. 13 shows results for the first-order accurate discretisation: velocity profiles in three chordwise positions and the pressure coefficient $c_p = (p - p_\infty)/\left(\frac{1}{2}u_\infty^2\right)$, where the reference pressure $p_\infty = 1$ and the reference velocity $u_\infty = 1$ too. The errors in the c_p plot are large: the pressure in the stagnation point is too high and the suction peak is not strong enough. As a result, the pressure on the upper side of the airfoil is not predicted well. The same is seen in the velocity contours: especially above the airfoil, the solution is far from converged, it does not even exhibit asymptotic first-order convergence yet. A good solution with the first-order discretisation would require much finer grids.

The second-order solution, after five DC steps, is much better. The solution on the coarse 64×64 grid is bad, but the other two lie almost on top of each other. The c_p solution is excellent: the suction peak is resolved correctly and c_p in the stagnation point is very close to 1. Also, the velocity profiles show good agreement with the experiment. Discrepancies are probably due to the approximations made in the turbulence model. The only place where the solution is not completely converged is in the wake, but even there the agreement with the experiment is good.

Multigrid. The Nakayama flow is solved with multigrid on six levels, the coarsest level is 8×8 cells. Due to the difficulty of the flow, the coarsest grid on which a solution could be obtained is the fourth grid of 64×64 cells. There, six iterations are done with single-grid Gauss–Seidel to develop the boundary layers, then the multigrid method is started; this can be seen in the multigrid convergence graph 15a. The solution on the fifth and sixth grid is found with multigrid from the start.

The multigrid convergence on the two finest grids is slower than in the previous problems and it is a bit 'jittery'. There are two possible causes for this. One, the airfoil has both an upper and a lower side; to get good smoothing, it was necessary to start the horizontal line smoothing alternately from the lowest line going up and from the highest line going down. This sometimes causes large residuals near the outer boundaries of the domain, that have nothing to do with the turbulence modelling: they appear in laminar flow too.

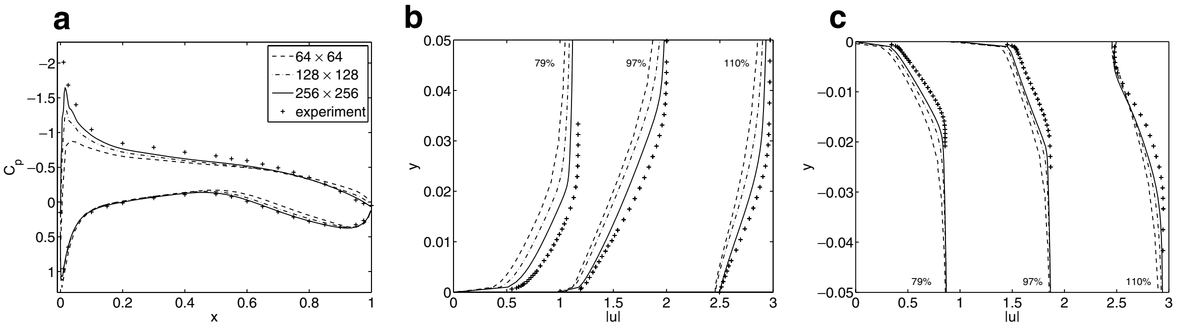

Fig. 13. Supercritical airfoil, first-order accurate grid convergence study. Solutions for c_p (a) and for the velocity profiles above (b) and below the airfoil (c) are compared with measurements (+) from [27].

And two, to get stability near the trailing edge, a small region there (a circle with radius 0.01) needs a higher damping in the coarse-grid corrections. In that circle, the α for the coarse grid corrections is set at a constant high value. This stabilises the computation, but it decreases the convergence a little. This problem can probably be solved by performing coarse grid corrections with the full nine-point stencils instead of the five-point stencils, as diffusion is dominant in the high gradients near the trailing edge. This is an area for further study.

At this moment, the multigrid convergence is acceptable. Compared with a single-grid solution (Fig. 15b), the computation time is reduced by a factor of about 11.5. Defect correction convergence is slow (Fig. 15c). The residual is reduced significantly in the first 10 steps, but then the convergence rate decreases. Still, as Fig. 14 shows, five defect correction steps is enough to get an accurate solution.

6.5. Parameter settings

The linear multigrid algorithm from Sections 4.2 and 4.3 has some free parameters, that can be tuned if desired. However, for our numerical experiments, we found that this was not necessary for most parameters: the values given in Section 4 are satisfactory for all problems. The parameter ϵ, the required relative convergence for the line smoothing residual, has the strongest influence. A lower ϵ means, that a higher convergence rate is required for NR in the lines, which leads to higher values for the damping α. This means better stability, but slower convergence for the multigrid method. Still, $\epsilon = 10^{-5}$ is used for all our problems, with good results.

The user's most important choice is, on which grid the FMG solution is started. For a linear coarse grid correction we can use as many grids as desired, but FMG requires nonlinear solutions on coarse grids. The problems above show, that the coarsest possible starting grid depends on the problem: for more complex flows, we need finer starting grids. We found, as a good rule of thumb, that the starting grid must not have less than about 8–10 cells over the thickness of the expected boundary layers.

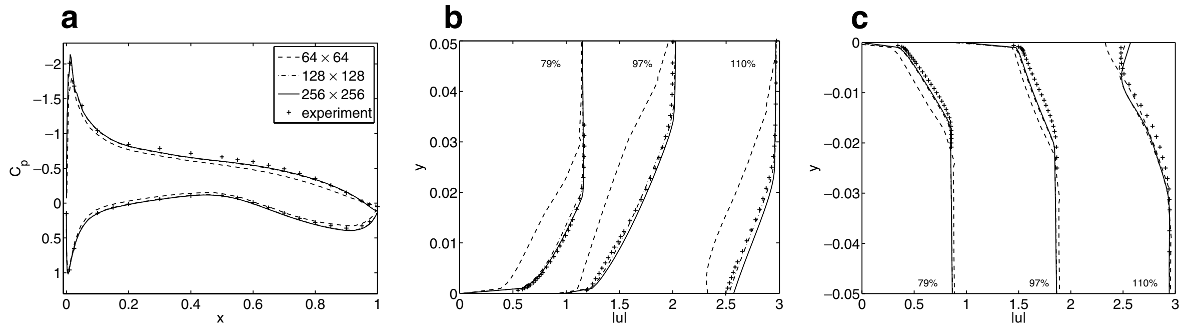

Fig. 14. Supercritical airfoil, second-order accurate grid convergence study (five defect correction steps). The plots show the same variables as in Fig. 13.

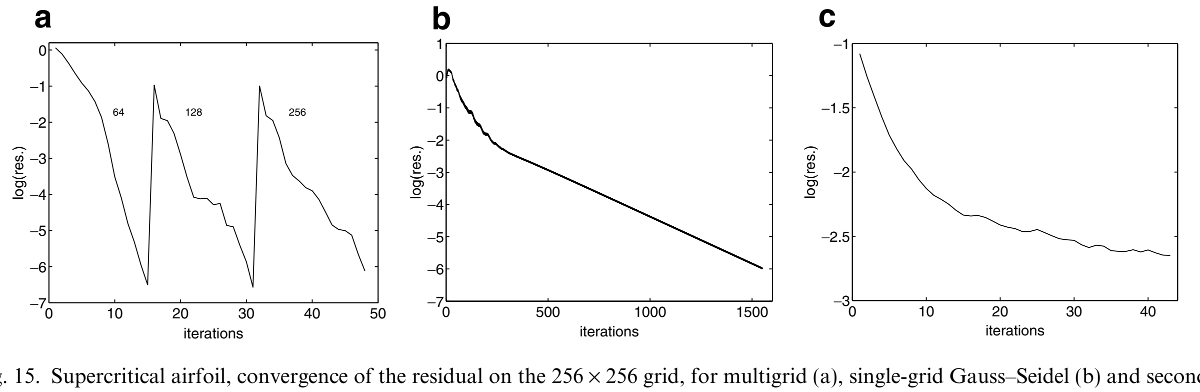

Fig. 15. Supercritical airfoil, convergence of the residual on the 256 × 256 grid, for multigrid (a), single-grid Gauss–Seidel (b) and second-order accurate defect correction (c).

7. Conclusions

We have presented a novel multigrid method for the Reynolds-averaged Navier–Stokes equations with Menter's turbulence model. The method has two new aspects: a linear coarse grid correction with nonlinear smoothing on the finest grid and an efficient locally damped line Gauss–Seidel smoothing. The method is combined with defect correction to obtain second-order accurate solutions.

Standard nonlinear multigrid does not work for the RANS equations. This has two reasons. The first is, that the combination of the non-constant viscosity and the source term in the turbulence equation makes it possible for the system to have unstable eigenmodes, that make Gauss–Seidel smoothing impossible. However, in the neighbourhood of a converged solution, these eigenmodes become stable. Our line Gauss–Seidel smoothing is stabilised by adding local damping to the turbulence equation in each smoothing step, enough to remove the unstable eigenmodes. We showed that the amount of damping necessary can be estimated with the Newton–Raphson root finder in the individual lines: if NR converges quickly, then enough damping is used.

The second problem is, that the nonlinear RANS operators on coarse grids do not resemble the operator on fine grids sufficiently well. This problem is caused by the sensitivity of the turbulence source terms to the grid size on too coarse grids. The problem is solved by switching to a linearised coarse grid correction with Galerkin operators. These resemble the fine grid operator reasonably well and their construction is simple for a finite-volume discretisation. A disadvantage is the higher memory requirement. The combination with nonlinear smoothing on the finest grid adds robustness and allows the computation of the Gauss–Seidel damping factor, that is used to stabilise the smoothing on the coarse grids too.

The multigrid method is combined with a cell-centred finite-volume discretisation. Convective fluxes are constructed with artificial compressibility, diffusive fluxes with central differences. The source terms are computed with finite differences on local orthogonal grids.

Numerical results show that the multigrid convergence is very good. The convergence rates for a laminar problem are compared with nonlinear multigrid: they are a little lower due to the use of Galerkin operators, but the effect on the computation times is small, because the linear method requires less time per multigrid cycle. Turbulent flow problems show comparable convergence. Defect correction converges very slowly. However, this is no problem as only a few defect correction steps bring a solution from first- to second-order accuracy. Results for two airfoil computations confirm this.

Full multigrid is needed for good convergence, because turbulent boundary layers have to develop first when the initial condition is a uniform flow. In this initial stage, multigrid is of little use. By using full multigrid, this first stage is only encountered on the coarsest grid.

Finally, comparisons with analytical and experimental data and grid convergence studies showed that the accuracy of the solutions is good and that the defect-correction iteration is efficient for the solution of second-order accurate problems.

Outlook. The present method works well. However, there is room for improvements to make the method even more robust and accurate. For instance, a larger study of the parameter settings for the multigrid algorithm can lead to better guidelines for the choice of the parameters. Also, it is worthwhile to study discretisation effects, like the choice of the turbulent viscosity at the cell faces that influences the smoothing efficiency. And finally, it is useful to compare coarse-grid corrections using nine-point stencils with the present method, to see if there are any differences.

The present method can probably be extended to other turbulence models. The current analysis is done for Menter's turbulence model, but most one- and two-equation turbulence models consist of the same type of equations: convection–diffusion equations with similar production and dissipation terms. Therefore, we expect that the present method can be used for other RANS turbulence models too and that it is a useful tool for the fast solution of these equations.

Acknowledgments

This research was supported by the Dutch government through the national program BSIK: knowledge and research capacity, in the ICT project BRICKS (http://www.bsik-bricks.nl).

References

[1] W. Hackbusch, Multi-Grid Methods and Applications, Springer-Verlag, Berlin, 1985.
[2] U. Trottenberg, C.W. Oosterlee, A. Schüller, Multigrid, Academic Press, London, 2001.
[3] P. Wesseling, An Introduction to Multigrid Methods, John Wiley & Sons Ltd., Chichester, 1992.
[4] P.W. Hemker, S.P. Spekreijse, Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations, Appl. Numer. Math. 2 (1986) 475–493.
[5] B. Koren, Multigrid and defect correction for the steady Navier–Stokes equations, Application to aerodynamics, CWI Tracts CWI, vol. 74, Centre for Mathematics and Computer Science, Amsterdam, 1985.
[6] E. Dick, J. Linden, A multigrid method for steady incompressible Navier–Stokes equations based on flux difference splitting, Int. J. Numer. Methods Fluids 14 (1992) 1311–1323.
[7] J. Steelant, E. Dick, S. Pattijn, Analysis of robust multigrid methods for steady viscous low Mach number flows, J. Comput. Phys. 136 (1997) 603–628.
[8] C. Frohn-Schauf, Flux-Splitting-Methoden und Mehrgitterverfahren für hyperbolische Systeme mit Beispielen aus der Strömungsmechanik, Berichte der Gesellschaft für Mathematik und Datenverarbeitung, 211, R. Oldenbourg Verlag, München, Wien, 1993.
[9] D.J. Mavriplis, Algebraic turbulence modeling for unstructured and adaptive meshes, AIAA J. 29 (12) (1991) 2086–2093.
[10] D.J. Mavriplis, V. Venkatakrishnan, Agglomeration multigrid for two-dimensional viscous flows, J. Comput. Phys. 24 (1995) 553–570.
[11] F. Liu, X. Zheng, A strongly coupled time-marching method for solving the Navier–Stokes and k–ω turbulence model equations with multigrid, J. Comput. Phys. 128 (1996) 289–300.
[12] G. Carré, An implicit multigrid method by agglomeration applied to turbulent flows, Comput. Fluids 26 (3) (1997) 299–320.
[13] F.R. Menter, Eddy viscosity transport equations and their relation to the k–ϵ model, J. Fluids Engineering 119 (1997) 876–884.
[14] J.-A. Désidéri, P.W. Hemker, Convergence analysis of the defect-correction iteration for hyperbolic problems, SIAM J. Sci. Comput. 16 (1) (1995) 88–118.
[15] P.W. Hemker, J.-A. Désidéri, Convergence behaviour of defect correction for hyperbolic equations, J. Comput. Appl. Math. 45 (1993) 357–365.
[16] J. Wackers, B. Koren, A surface capturing method for the efficient computation of steady water waves, J. Comput. Appl. Math., in press.
[17] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, La Recherche Aérospatiale 1 (1994) 5–21.
[18] E. Dick, A flux-vector splitting method for steady Navier–Stokes equations, Int. J. Num. Meth. Fluids 8 (1988) 317–326.
[19] K. Nerinckx, J. Vierendeels, E. Dick, Mach-uniformity through the coupled pressure and temperature correction algorithm, J. Comput. Phys. 206 (2005) 597–623.
[20] J. Vierendeels, K. Riemslagh, E. Dick, A multigrid semi-implicit line-method for viscous incompressible and low-Mach-number flows on high aspect ratio grids, J. Comput. Phys. 154 (1999) 310–341.
[21] F.R. Menter, Zonal two-equation k–ω turbulence models for aerodynamic flows, AIAA Paper 93-2906, American Institute of Aeronautics and Astronautics (July 1993).
[22] A.J. Chorin, A numerical method for solving incompressible viscous flow problems, J. Comput. Phys. 2 (1967) 12–26.
[23] E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations, J. Comput. Phys. 72 (1987) 277–298.
[24] E. Dick, A multigrid method for steady incompressible Navier–Stokes equations based on partial flux splitting, Int. J. Num. Meth. Fluids 9 (1989) 113–120.
[25] B. Koren, A robust upwind discretization method for advection, diffusion and source terms, in: C.B. Vreugdenhil, B. Koren (Eds.), Numerical Methods for Advection–Diffusion Problems, Notes on Numerical Fluid Mechanics, vol. 45, Vieweg, Braunschweig, 1993, pp. 117–138.
[26] A.E. Samuel, P.N. Joubert, A boundary layer developing in an increasingly adverse pressure gradient, J. Fluid Mech. 66 (1974) 481–505.
[27] A. Nakayama, Characteristics of the flow around conventional and supercritical airfoils, J. Fluid Mech. 160 (1985) 155–179.