Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 208 (2005) 368–383

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Conditional semicoarsening multigrid algorithm for the Poisson equation on anisotropic grids

J. Larsson [a,*], F.S. Lien [a], E. Yee [b]

[a] *Department of Mechanical Engineering, University of Waterloo, University Avenue West, Waterloo, Ont., Canada N2L 3G1*
[b] *Defence R&D Canada – Suffield, P.O. Box 4000, Medicine Hat, Alta., Canada T1A 8K6*

Received 17 August 2004; received in revised form 26 January 2005; accepted 24 February 2005
Available online 14 April 2005

Abstract

A conditional semicoarsening multigrid algorithm is developed for the Poisson equation on stretched grids. The smoothing behaviour of the pointwise smoother, as predicted by local Fourier analysis, is used to determine how to coarsen the grid. This process ensures that the smoothed error can be adequately resolved on the coarse grid, which results in an efficient multigrid algorithm that is simple to implement. The use of a pointwise smoother increases the degree of parallelism of the algorithm.

The resulting algorithm is tested on several different grid types, some with very high degrees of stretching. The computational cost is found to scale almost linearly with the number of grid points, and parallel efficiencies of over 90% are achieved even for very coarse grids.

© 2005 Elsevier Inc. All rights reserved.

1. Introduction

Multigrid techniques are commonly used to accelerate the convergence when solving elliptical problems iteratively. The basic idea behind multigrid algorithms is that a smooth error can be approximated on a coarser grid (see, e.g., Trottenberg et al. [1]). For simple, isotropic, problems, even simple smoothers (e.g., pointwise Gauss–Seidel) efficiently yield a smooth error in all coordinate directions, and hence a grid that is coarser in all coordinate directions can be used. In three dimensions, this yields a coarser grid with 8 times fewer grid points if standard 2:1 coarsening is used. For anisotropic problems, however, the situation becomes more complex. Pointwise smoothers typically only yield a smooth error in some direction(s). A

* Corresponding author. Tel.: +1 519 8884567x3859.
E-mail address: njlarsso@engmail.uwaterloo.ca (J. Larsson).

0021-9991/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2005.02.020

grid that is coarser in all directions can then not represent the error adequately, which results in either poor convergence or divergence of the algorithm. Anisotropic problems arise either due to anisotropy of the underlying physics, or perhaps more commonly due to the use of non-uniform grids, which makes the discrete problem anisotropic.

There are essentially two different approaches to ensure good multigrid convergence on anisotropic problems [1]: either to adapt the smoother such that the error becomes smooth in all directions, or to adapt the coarsening of the grid such that the grid is only coarsened in directions where the error is smooth. One common method in the first category is the use of block-implicit smoothers, e.g., plane-smoothers, which update all grid points in a plane simultaneously. If the planes are swept in all directions (e.g., in the *xy*-, *xz*-, and *yz*-planes) sequentially, the resulting error is smooth in all directions and standard coarsening can be used. The increased cost of this method lies, obviously, in the increased cost of the smoother. The second category of methods is typically called *semicoarsening*, and is the focus of the present study. In these methods a simple (e.g., pointwise) smoother is used, and the increased cost lies in the fact that the coarser grids have relatively many grid points.

In some special situations, the error after the smoothing step is smooth in the same direction(s) in the whole domain, and hence the application of semicoarsening is trivial and yields coarser grids that are structured (assuming that the original grid is structured). For most problems, however, the error is smooth in different directions in different parts of the domain. Application of semicoarsening on a local level, i.e., for each grid point separately, would then yield coarse grids that are unstructured, which increases the computational overhead of the solver substantially.

The present study uses semicoarsening in a somewhat intermediate sense: it is applied locally, while requiring the resulting coarse grids to maintain their structured topology. This is possible by not considering independent grid points, but rather planes of grid points in all coordinate directions. The algorithm yields coarse grids that are not optimal locally, but for which good convergence can be achieved using pointwise smoothers. The use of a pointwise smoother results in an algorithm that is highly parallel and simple to implement.

In the following, the conditional semicoarsening algorithm is presented and tested. It is then applied to the Poisson equation for pressure in a fractional step method for the incompressible Navier–Stokes equations. Fractional step methods (see, e.g., Ferziger and Peric [2]) are quite popular for unsteady fluid flow calculations, such as direct numerical simulations (DNS) and large eddy simulations (LES). In such methods, the solution of the Poisson equation is typically the most time consuming part of the method.

2. Governing equations

The Poisson equation for an unknown function p can be written as

$$\frac{\partial^2 p}{\partial x_i^2} = q, \tag{1}$$

where q is a known source term. Discretization of Eq. (1) on a structured grid using a 7-point stencil (in three dimensions) yields

$$a_c p_c + \sum_{nb} a_{nb} p_{nb} = q_c, \tag{2}$$

where a_c and a_{nb} are the diagonal and off-diagonal matrix coefficients, respectively. The subscript nb here refers to the neighbouring grid points, i.e., $nb \in \{E, W, \dots\}$ (east, west, ... neighbours). The matrix coefficients are completely determined by the geometry of the grid.

3. Multigrid algorithm

A multigrid algorithm has several components to it that need specification, such as the smoother, the intergrid operators, and the cycle type. Since the semicoarsening algorithm can be used on any structured grid, the different directions are referred to as the *i*-, *j*-, and *k*-directions, respectively. Some examples are given for the special case of Cartesian grids, where the (x, y, z) directions are equivalent to the (i,j,k) directions.

3.1. Smoother

Since one goal of the present study is to develop a highly parallel solver, the red-black point Gauss–Seidel (RBPGS) smoother [1] is used here. The implementation pays careful attention to the parallelism by minimizing the communication overhead. Each smoothing sweep consists of the following steps, where red points are those where the sum of the indices (in the *i*-, *j*-, and *k*-directions) is odd, and black points are those where the sum is even:

(1) smooth red grid points adjacent to processor boundaries;
(2) initiate communication of those points;
(3) smooth red grid points away from processor boundaries (i.e., in the interior of each processor's domain);
(4) finalize communication of red grid points;
(5) repeat for black grid points.

All communication is handled by message passing interface (MPI) and is of the 'non-blocking' kind.

When used in multigrid, the performance of a smoother is best quantified in terms of the smoothing factor μ. The smoothing factor is essentially the worst reduction in magnitude of those Fourier modes that *cannot* be represented on the coarse grid (see, e.g., Trottenberg et al. [1]). Yavneh [3] used local Fourier analysis to derive analytical formulas for the smoothing factor when using the RBPGS smoother combined with semicoarsening in different directions. In order to make use of these formulas, the coefficient strength in a certain grid direction is defined here (arbitrarily) as

$$a_i \equiv \sqrt{\frac{a_{\mathrm{E}}^2 + a_{\mathrm{W}}^2}{2}}, \tag{3}$$

and similarly in the other directions, and the normalized coefficient strength becomes

$$\hat{a}_d = \frac{a_d}{a_i + a_j + a_k}, \quad d \in \{i, j, k\}. \tag{4}$$

If I_c is the set of all directions that are coarsened locally, then Yavneh [3] showed that the smoothing factor predicted by local Fourier analysis can be written as

$$\mu = \max\left\{(1 - \hat{a}_{\min})^2 \phi(\hat{a}_{\min}, v, \#I_c), \zeta(v)\right\}, \tag{5}$$

where

$$\hat{a}_{\min} \equiv \min_{d \in I_c} \{\hat{a}_d\}, \tag{6}$$

where v is the total number of smoothing sweeps, $\#I_c$ is the number of directions being coarsened, $\zeta(v)$ is a lower bound on μ, and

$$\phi(\hat{a}_{\min}, v, \#I_c) = \begin{cases} [1 + \hat{a}_{\min}/(2(1 - \hat{a}_{\min}))]^{1/v} & \text{if } \#I_c = 1, \\ 1 & \text{if } \#I_c > 1. \end{cases} \tag{7}$$

Eq. (5) can hence be used to calculate the smoothing factor once the coarse grid has been constructed, i.e., when I_c is known. The lower bound $\zeta(v)$ is only dominant when a large number of smoothing sweeps are performed. $\phi(\hat{a}_{\min}, v, 1)$ is a decreasing function of v and an increasing function of $\hat{a}_{\min}$, but is always greater than or equal to 1.

In another contribution, Yavneh [4] found that the smoothing behaviour of the RBPGS can be improved by the use of over-relaxation (see, e.g., Trottenberg et al. [1]). Assuming standard coarsening, the optimum over-relaxation parameter ω was found to be

$$\omega_{\text{opt}} \approx \frac{2}{1 + \sqrt{1 - (1 - \hat{a}_{\min})^2}}. \tag{8}$$

For uniform grids in three dimensions, $\omega_{\text{opt}} \approx 1.15$.

3.2. Semicoarsening

In light of the underlying ideas of multigrid, the goals or requirements of the semicoarsening procedure can be stated as to, given a fine grid, construct a coarse grid that:

(1) is such that the error on the fine grid is sufficiently smooth for all grid points/directions that are coarsened, i.e., that

$$\mu \leqslant \mu_{\text{lim}}, \tag{9}$$

where μ_{lim} is some maximum allowable smoothing factor;
(2) has as few grid points as possible;
(3) has a structured topology.

The first two requirements are met by attempting to coarsen the grid as much as possible, while still satisfying condition (9). The third requirement imposes limitations on the coarsening process. In the following, a finite volume discretization with colocated (cell-centred) storage of variables is assumed.

It is quite natural to attempt to use Eq. (5) to predict μ such that it satisfies condition (9), but this requires the knowledge of I_c for every cell. The goal of the semicoarsening is to determine I_c, and hence it is clear that I_c is not available until after all cells have been processed. Therefore, an assumption has to be made. Here, the correction factor $\phi(\hat{a}_{\min}, v, \#I_c)$ is neglected, and then (while also neglecting the lower bound $\zeta(v)$), Eq. (5) becomes

$$\mu = (1 - \hat{a}_{\min})^2. \tag{10}$$

From the definition of $\hat{a}_{\min}$, condition (9) then becomes

$$\left(1 - \min_{d \in I_c}\{\hat{a}_d\}\right)^2 = \max_{d \in I_c}\left\{(1 - \hat{a}_d)^2\right\} \leqslant \mu_{\text{lim}}, \tag{11}$$

which is equivalent to saying that only if $(1 - \hat{a}_d)^2 \leqslant \mu_{\text{lim}}$ can the cell be coarsened in direction d without violating condition (9).

To ensure a structured topology of the coarse grid, whole planes of cells need to be treated consistently in every direction. Consider the grid in Fig. 1. To maintain a structured topology for the coarse grid, cells A and B cannot be coarsened in the i-direction without *every* cell in between them being coarsened as well (by

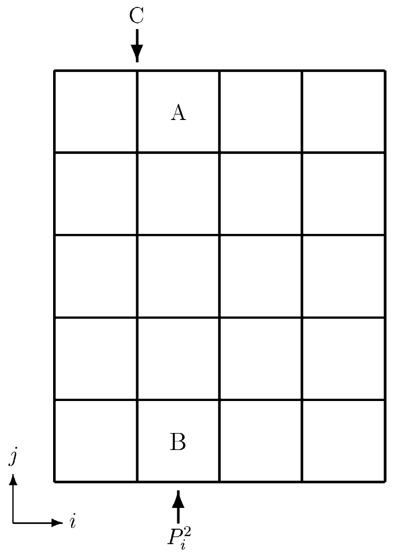

Fig. 1. Sample grid.

removing grid line C). Denoting the lth plane of cells with constant index $d \in \{i,j,k\}$ by P_d^l, the requirements of the semicoarsening listed above are satisfied if the cells in P_d^l are coarsened in direction d only if

$$\max_{P_d^l}\left\{(1-\hat{a}_d)^2\right\} \leqslant \mu_{\text{lim}}. \tag{12}$$

The algorithm for deciding which planes P_d^l to leave uncoarsened due to coefficient anisotropy can then be written as:

Algorithm 1

```
for all cells do
    compute the normalized matrix coefficients ($\hat{a}_d$) by Eq. (4)
    for all directions $d$ do
        if $(1-\hat{a}_d)^2 > \mu_{\text{lim}}$ then
            leave plane normal to the $d$-direction uncoarsened
        end if
    end do
end do
```

When coarsening grids with an even number of cells in every direction, it is natural to remove the even-numbered grid lines (such as C in Fig. 1). When a grid has an odd number of cells in some direction, one must either merge three cells into one somewhere, or leave some plane uncoarsened. In the present application, which by nature involves leaving planes of cells uncoarsened, the latter approach is used.

To decide which plane P_d^l to leave uncoarsened due to an odd number of cells, it is first noted that only those planes with l odd should be considered. To see this, consider the j-direction in Fig. 1. If, for example, P_j^2 is left uncoarsened, then P_j^1 cannot be coarsened either. If, however, the planes with

$l = 1, 3, 5$ are chosen, this situation is avoided. To satisfy requirement 2 above, that the coarse grid has as few cells as possible, one should not leave more planes than necessary uncoarsened. This is achieved by the following algorithm for choosing which plane to leave uncoarsened due to an odd number of cells:

Algorithm 2

```
for all directions $d$ do
  if the number of cells in $d$ is odd then
    for $l = 1, 3, 5, \ldots$ do
      $M_l = \max_{P_d^l} \{(1 - \hat{a}_d)^2\}$
    end do
    leave plane $P_d^k$ with $k = \arg\max_l \{M_l\}$ uncoarsened
  end if
end do
```

The algorithm above finds the odd plane with the worst smoothing factor and leaves this uncoarsened. The full implementation involves applying both algorithms to flag planes as unsuitable for coarsening, followed by straightforward construction of the coarse grid.

To get a sense for what these algorithms do, consider, for example, Eq. (1) discretized on the grid in Fig. 2(a), which shows an *ij*-slice of the grid. The anisotropy of the discretized problem is then completely related to the grid geometry. The grid has a maximum aspect ratio AR of 10, where

$$\mathrm{AR} = \max_{\text{all cells}} \left\{ \frac{\max\{\Delta x, \Delta y, \Delta z\}}{\min\{\Delta x, \Delta y, \Delta z\}} \right\}. \tag{13}$$

The cells in the upper right corner are close to isotropic, and hence $(1 - \hat{a}_i)^2 \approx (1 - \hat{a}_j)^2 \approx (1 - \hat{a}_k)^2 \approx 1/3$. The cells in the lower right corner, on the other hand, are highly stretched, and hence the error will not be smooth in the *i*-direction ($(1 - \hat{a}_i)^2 \approx 0.98$). The planes of cells in the *i*-direction flagged as being left uncoarsened by the two algorithms are shown above the grids in Fig. 2. Note how the coarser grids have relatively many cells, but also how the cells do become more and more isotropic.

The only parameter of the algorithm is the value of μ_{lim}. The optimum value of this parameter will be determined by numerical experiments on a three-dimensional unit cube in Section 3.4. Note that $\mu_{\mathrm{lim}} = 1$ corresponds to standard, uniform, coarsening.

The number of cells in the coarse grids is, of course, dependent on μ_{lim}. This is illustrated in Fig. 3, where the number of cells N on each grid level is plotted for some values of μ_{lim}. When the parameter is less than 1, the coarsening ratio for the first coarse grids in the sequence is quite low as condition (12) severely limits the coarsening process. At some point in the process, the grids are close enough to being isotropic that the coarsening proceeds almost without limitation, and hence the coarsening ratio approaches the standard 8:1.

The algorithms outlined above sweep over all cells, with a few operations at each cell. The work of deciding how to coarsen is hence of the order of N (where N is the number of cells), which is consistent with multigrid. The work of constructing the coarser grid, and the work of intergrid transfers, is equivalent to the work required for uniformly coarsened grids. For problems where the 'decision' process has to be done for every multigrid cycle, the cost of the set-up phase may be quite large compared to the actual solution phase. For the pressure equation in a fractional step method, however, the matrix coefficients (and hence the coarsening 'decisions') are unchanged during the simulation, and hence the cost of the set-up phase is completely negligible.

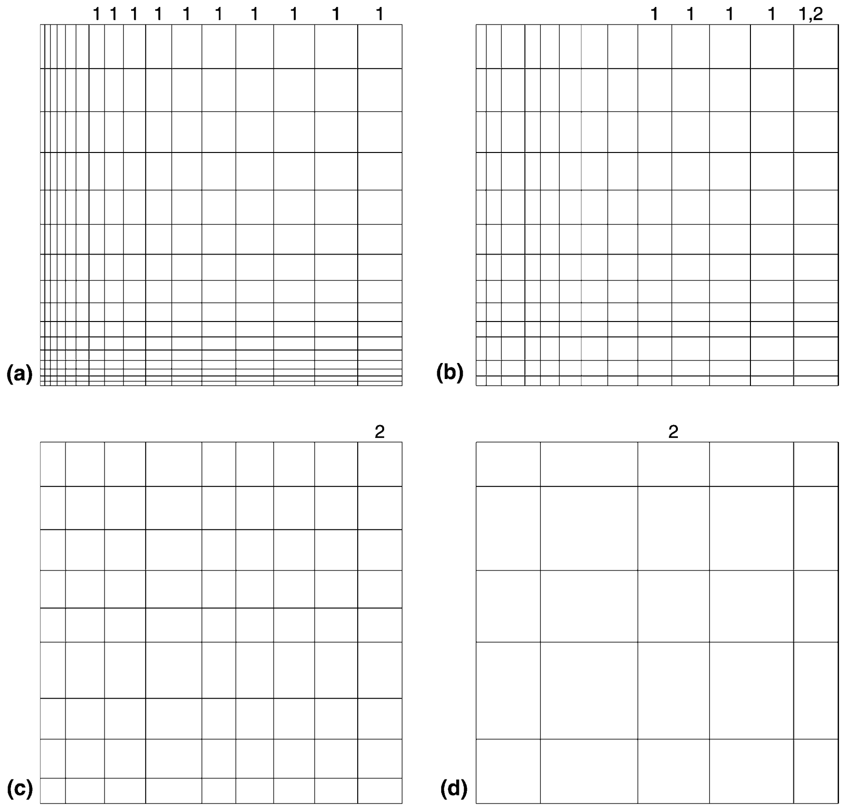

Fig. 2. Sequence of grids created by $\mu_{\text{lim}} = 0.85$. Numbers above the grids indicate planes being left uncoarsened in the *i*-direction by each algorithm (1 due to coefficient anisotropy, 2 due to an odd number of cells). (a) Finest grid level, 16^2 cells, AR = 10. (b) Coarse grid level 1, 13^2 cells, AR = 4.5. (c) Coarse grid level 2, 9^2 cells, AR = 2.2. (d) Coarse grid level 3, 5^2 cells, AR = 2.2.

3.3. Intergrid and coarse grid operators

For problems with smoothly varying coefficients, standard tri-linear prolongation and coarse grid operators defined by direct discretization on the coarse grids are commonly used. For problems with coefficient discontinuities, however, operator-dependent prolongation and Galerkin coarse grid operators have been shown to improve the convergence rate [5]. In the present work, coefficient discontinuities occur whenever there is a large difference in cell size between adjacent cells. For purposes of numerical accuracy, it is standard practice in DNS and LES to use grids with smooth stretching, and hence the discontinuities are expected to be relatively small on the finest grid. Due to the fact that the semicoarsening algorithm typically coarsens small cells and leaves larger ones, the difference in cell size between adjacent cells is normally kept within a factor of 2. For these reasons, standard tri-linear prolongation and coarse grid operators defined by direct discretization on the coarser grids are used in this work. These choices also ensure that the coarse grid stencils consist of only seven points, whereas a Galerkin operator would consist of 27. The parallelization of the smoother is better with a 7-point stencil, since the two-colour RBPGS completely decouples the grid points. The restriction operator is residual summation, which makes physical sense in the context of a finite volume method [1].

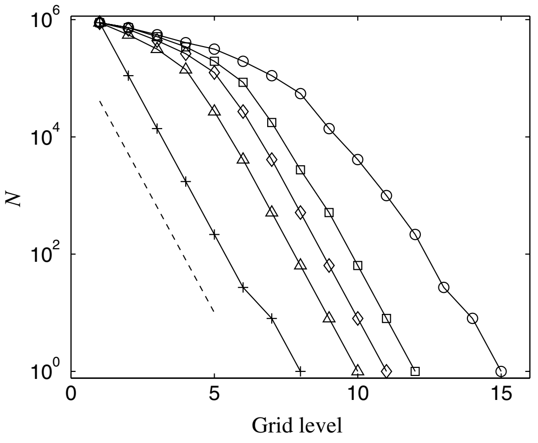

Fig. 3. Number of cells at each grid level for unit cube with 96^3 cells and AR = 100 on the finest grid. ○: $\mu_{\text{lim}} = 0.72$; □: $\mu_{\text{lim}} = 0.78$; ◇: $\mu_{\text{lim}} = 0.87$; △: $\mu_{\text{lim}} = 0.96$; +: $\mu_{\text{lim}} = 1.00$; – –: slope for 8:1 coarsening ratio.

3.4. Choice of parameters

Apart from the parameter μ_{lim}, the number of smoothing sweeps (ν_1, ν_2) (before and after the coarse grid correction, respectively) and the over-relaxation factor ω need to be determined in order to fully specify the multigrid algorithm. For simplicity, only V-cycles have been considered in this work.

Eq. (1) is solved with the source term $q = 0$ and with an initial field that is randomized. The domain is a unit cube, i.e., $0 \leqslant x, y, z \leqslant 1$, and grids with varying degrees of stretching are used. The stretching is determined by

$$x_d^l = \frac{1}{2}\left[1 - \frac{\tanh(\gamma(1 - 2l/N_d))}{\tanh(\gamma)}\right], \qquad l = 0, \ldots, N_d, \quad d \in \{x, y, z\}, \tag{14}$$

where x_d^l is the d-coordinate of the lth grid line in that direction, N_d is the number of grid cells in that direction, and γ is a stretching parameter. All grids consist of 64^3 cells (i.e., $N_d = 64$), and the stretching of each grid is quantified by the maximum cell aspect ratio AR.

Initial tests show that the performance is relatively insensitive to the number of smoothing sweeps, but that $\nu \equiv \nu_1 + \nu_2 = 4$ or 5 give good results for the different grids. In the following, $(\nu_1, \nu_2) = (2, 3)$ is used exclusively.

An example of the convergence history is shown in Fig. 4, where $\|r\|$ is the L_2-norm of the residual. The residuals in the figures have been normalized by their initial values. The results from two different grids are shown, one uniform (AR = 1) and one moderately stretched (AR = 10). For the latter, three different values of the parameter μ_{lim} have been used.

In Fig. 4(a), the residual norm is plotted versus the number of multigrid cycles. First, it is seen that the convergence with $\mu_{\text{lim}} = 1$ (AR = 10) is very poor, even for this moderate stretching (note that the over-relaxation parameter ω had to be increased for this case to converge). For lower values of μ_{lim}, the rate of reduction of the residual norm is much improved.

In Fig. 4(b), the residual is plotted versus the number of work units (WU). One work unit is defined here as one smoothing sweep on the finest grid, neglecting the cost of residual evaluations and intergrid transfers. The fact that the uniform grid (AR = 1) requires the least amount of work is now clearly seen. This is to be expected, since it is a considerably less challenging problem to solve than the stretched grid case. For the stretched grid, the convergence when plotted versus work units is actually faster for the $\mu_{\text{lim}} = 0.87$ case

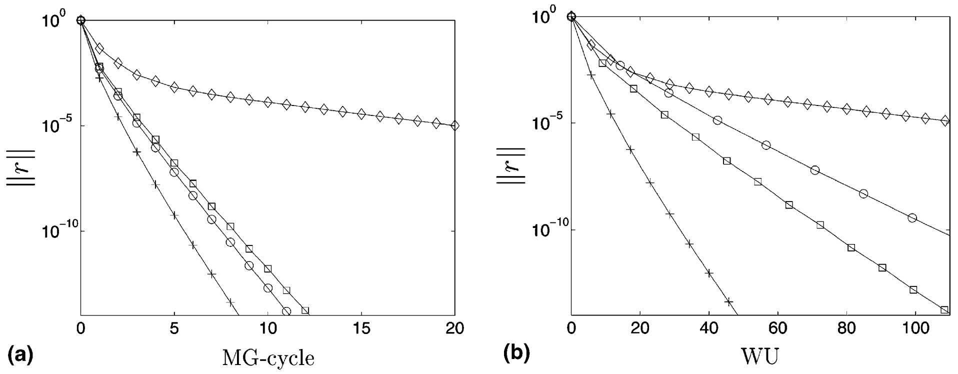

Fig. 4. Convergence history of residual norm $\|r\|$. +: AR = 1, $\omega = 1.15$; ○: AR = 10, $\omega = 1.15$, $\mu_{\text{lim}} = 0.71$; □: AR = 10, $\omega = 1.15$, $\mu_{\text{lim}} = 0.87$; ◇: AR = 10, $\omega = 1.5$, $\mu_{\text{lim}} = 1$. (a) Residual versus MG-cycle. (b) Residual versus work unit WU.

than the $\mu_{\text{lim}} = 0.71$ case, opposite from the case before. This is due to the fact that the lower value of μ_{lim} generates coarse grids with relatively more cells, and hence the cost of each smoothing sweep is higher.

To measure the performance of the algorithm, some definitions are needed. The convergence factor ρ is defined as [1]

$$\rho = \left(\frac{\|r^n\|}{\|r^k\|}\right)^{1/(n-k)}, \tag{15}$$

where $\|r^m\|$ is the residual norm after iteration m. From the definition, ρ is the average reduction in residual norm between iterations (MG-cycles) k and n. As seen in Fig. 4, the rate of convergence only approaches its asymptotic value after about four iterations, and hence $k = 4$ is used throughout this study, while n is the number of iterations to convergence (typically about 10).

In many studies of multigrid, the computational cost is measured in work units, as defined above. The cost of intergrid transfers (restriction and prolongation) is often neglected, as is the evaluation of residuals. For costly smoothers, such as block-implicit methods, this is probably a quite reasonable estimate of the cost, but for multigrid algorithms with cheap smoothers (such as RBPGS) this simplification is not justified at all. The cost of a residual evaluation is about one WU, and the restriction and prolongation operators are of similar cost. For these reasons, the concept of WU is not considered to be an accurate measure of the work for the present method, and hence the computational cost is measured by timing the code directly. This measure is of course machine dependent, and should be viewed in a relative sense.

The CPU time per iteration t is plotted as a function of μ_{lim} in Fig. 5(a) for a representative grid. The cost t decreases with increasing μ_{lim}, since higher values of μ_{lim} are less restrictive on the coarser grids. The algorithm then generates grids with fewer cells, which decreases the cost per iteration. The convergence factor ρ, which is plotted in Fig. 5(b), increases with μ_{lim} as expected. This is simply due to the fact that the coarser grids have been coarsened in directions where the smoother cannot adequately smooth the error. Note that as ω is increased, the convergence improves for high μ_{lim} but becomes worse for low μ_{lim}. The more aggressive coarsening with a higher μ_{lim} generates grids that are, on average, more suitable for a higher over-relaxation parameter as per Yavneh's [4] findings. Since the convergence factor is clearly a function of both μ_{lim} and ω, the optimal values of each parameter depends on the value of the other.

To get a good measure of algorithm performance, the CPU time per iteration is normalized as

$$\tilde{t} = \frac{t}{-\log_{10}\rho}, \tag{16}$$

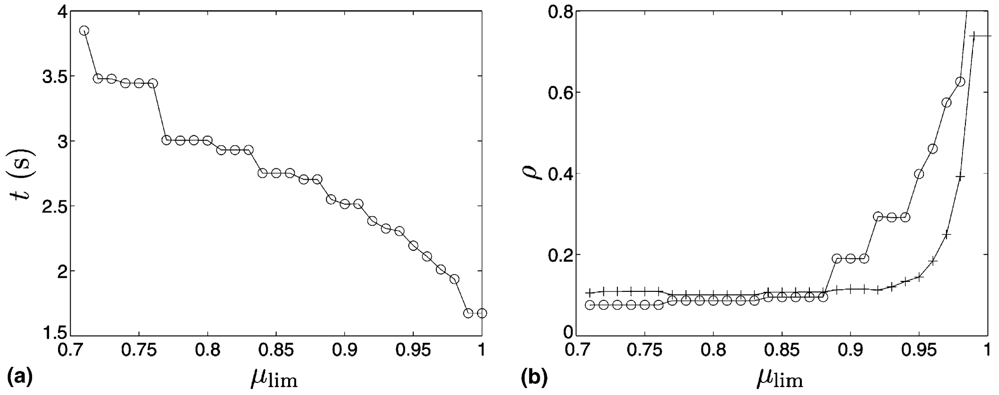

Fig. 5. Computational cost and convergence factor for grid with AR = 10. (a) CPU time per iteration. (b) Convergence factor ρ. ○: $\omega = 1.15$; +: $\omega = 1.50$.

which means that $\tilde{t}$ is the time it takes to reduce the residual by one order of magnitude. The normalized cost $\tilde{t}$ for three grids with different maximum aspect ratios is shown in Fig. 6(a)–(c). The optimum combination of $\mu_{\lim}$ and ω is different for the different grids, but some trends can be seen. First, higher values of

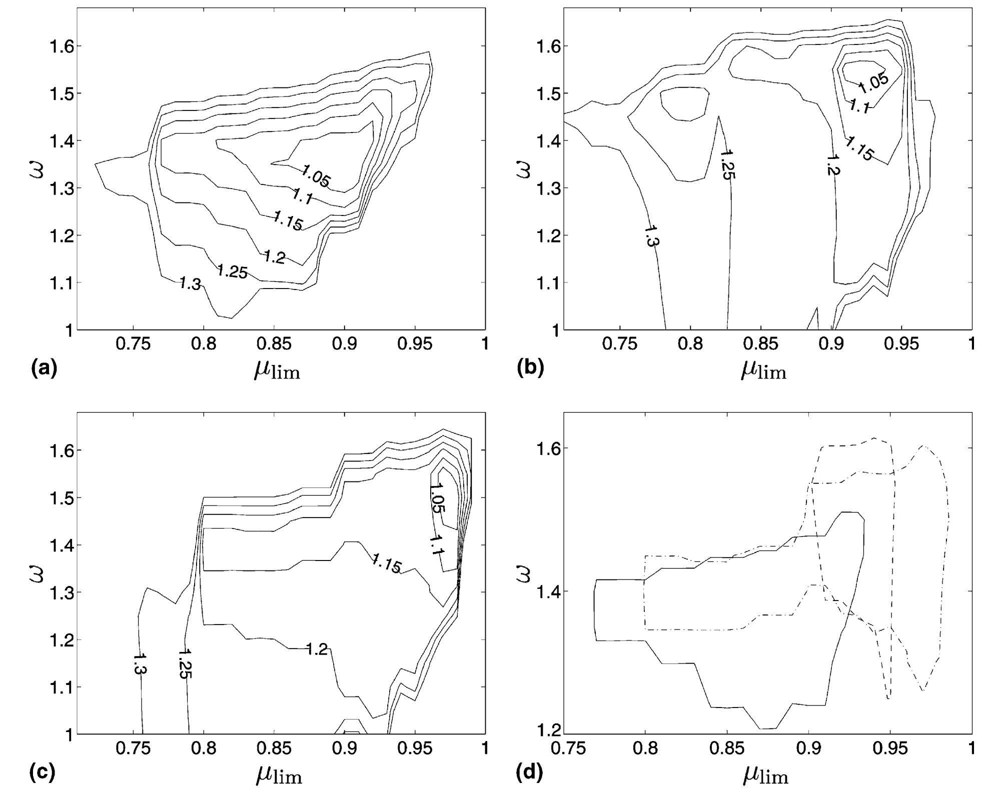

Fig. 6. Contours of $\tilde{t}$ scaled by the lowest value for each grid. (a) AR = 10. Six contours between 1.05 and 1.3. (b) AR = 100. Six contours between 1.05 and 1.3. (c) AR = 1000. Six contours between 1.05 and 1.3. (d) Contour at 1.15. —: AR = 10; – –: AR = 100; –·–: AR = 1000.

μ_{lim} require higher values of ω, which is an effect of the coarse grids being closer to those generated by standard, uniform, coarsening. For such grids, Yavneh [4] suggests over-relaxation parameters of 1.75 and above. Second, the optimum value of μ_{lim} increases with the maximum aspect ratio of the grid. This is due to the fact that the grid with higher stretching has a larger number of coarse grids, and hence a higher computational cost. This increased cost becomes more pronounced for lower values of μ_{lim}, which results in an optimum value that is higher.

A single contour of $\tilde{t}$ at a value 15% higher than the optimum is plotted for each grid in Fig. 6(d). From this, $\mu_{\text{lim}} = 0.92$ and $\omega = 1.45$ are taken as compromise values, yielding convergence within 15% of the optimum for each grid.

4. Results

To further test the multigrid algorithm, several test cases are considered. First, the three-dimensional unit cube from the previous section is used to examine the cost scaling and the parallel efficiency. Second, the LES grids in plane channel flow and the flow around a wall mounted cube are considered.

4.1. Three-dimensional unit cube

The Poisson equation is solved on domains and grids similar to those in Section 3.4. The number of cells in each direction is $N_d \in \{16, 32, 48, 64, 80, 96\}$, and two different degrees of stretching are used: AR = 1 (uniform grid) and AR = 100. The compromise values of $\mu_{\text{lim}} = 0.92$ and $\omega = 1.45$ are used for the stretched grids, while the uniform grids use the theoretically optimum value of $\omega = 1.15$.

The normalized computational cost is shown in Fig. 7, where $N = N_d^3$ is the total number of cells. The cost for the uniform grid scales linearly with N, which is characteristic of multigrid algorithms. The cost for the stretched grid does not scale linearly, but rather like $\tilde{t} \sim N^{1.18}$. Using curve fits, the normalized CPU time on a single CPU is

$$\tilde{t} \approx \begin{cases} 4.7 \times 10^{-6} N \text{ (s)}, & \text{AR} = 1, \\ 1.8 \times 10^{-6} N^{1.18} \text{ (s)}, & \text{AR} = 100. \end{cases} \tag{17}$$

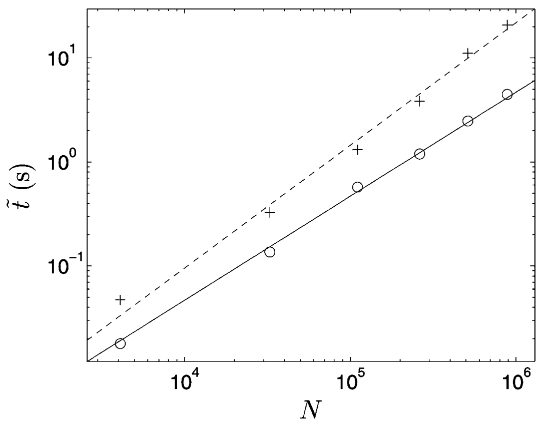

Fig. 7. Computational cost versus number of grid points. ○: AR = 1; +: AR = 100; —: Eq. (17) AR = 1; ––: Eq. (17) AR = 100.

Hence, the present semicoarsening algorithm does not show textbook multigrid efficiency. For the grid sizes considered here, the cost on the highly stretched grid is about 2–5 times that of the uniform grid. Comparing this to more complex multigrid solvers for anisotropic problems, such as ones with sweeping plane smoothers, the cost *scaling* is worse, but the cost *level* is probably comparable even for quite large grids. As will be seen later, the true benefit of the present semicoarsening algorithm is its high parallel efficiency, which results partly from the use of the RBPGS smoother.

The code is parallelized using MPI and run on up to 8 CPUs on a cluster of AMD Athlon XP machines. The communication is handled by an ethernet switch. The parallel efficiency is measured as

$$\eta(p) = \frac{\tilde{t}(1)}{p\tilde{t}(p)}, \tag{18}$$

where p is the number of processors. The results are plotted in Fig. 8. As expected, the efficiency improves with increasing grid size. The efficiency is slightly higher for the stretched grids, since there are more grid levels with large numbers of cells compared to the uniform grids. The efficiency is above 0.92 up to eight processors for grids with 32^3 cells or more, which corresponds to 4096 cells per processor. With 16^3 cells (512 cells per processor) the efficiency drops to about 0.73, a quite high value for such a coarse grid. The increases in parallel efficiency seen for some cases (most notably for the 32^3 stretched grid) are probably effects of cache performance. In this work, no cache optimization has been performed, and any such increases in performance are completely fortuitous.

To really test the parallel efficiency, computations on a larger number of CPUs would be necessary. This being said, the fact that the efficiency is high even on the coarse grids used here (~4000 cells per processor) suggests a high degree of parallelism.

4.2. Plane channel

The plane channel flow is a standard test case for large eddy simulation, and has been solved here using a fractional step algorithm similar to Zang et al. [6] for purposes of evaluating the multigrid algorithm for such types of grids. The size of the domain is $(2\pi, 2, \pi/2)$ for channel A, and $(2\pi, 2, 2\pi/3)$ for channels B and C, in the streamwise, wall normal, and spanwise directions, respectively. The number of cells ranges from 32 to 96 in the various directions. Three different types of grids are considered, with details given in Table 1. Channel A is at low Reynolds number and has a resolved inner region of the boundary layer. For this reason, the maximum cell aspect ratio is a relatively modest 32 close to the wall, due to the need to

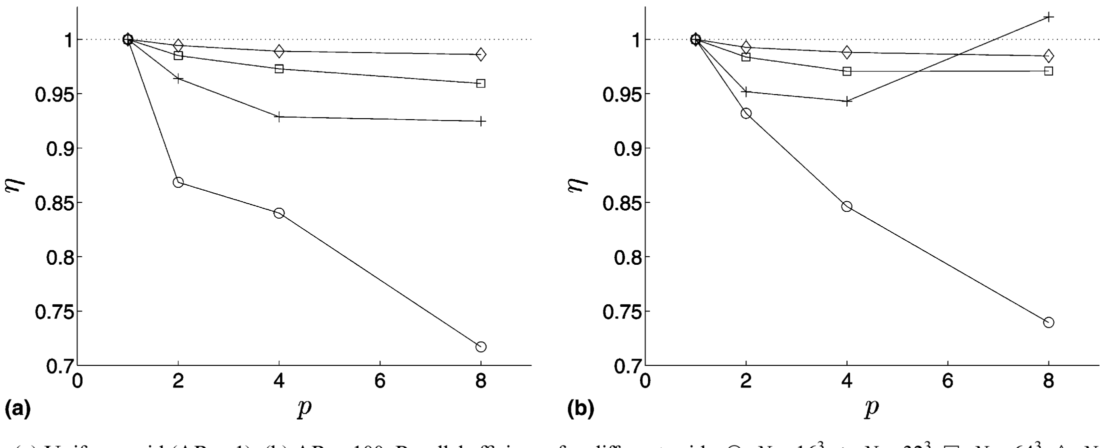

Fig. 8. (a) Uniform grid (AR = 1). (b) AR = 100. Parallel efficiency for different grids. ○: $N = 16^3$; +: $N = 32^3$; □: $N = 64^3$; ◇: $N = 96^3$.

Table 1
Channel flow details

Case	Re_τ	Wall BC	(N_x, N_y, N_z)	AR, wall	AR, core
Channel A	395	Resolved	(64, 64, 48)	32:1:11	3:2:1
Channel B	4000	Schumann	(32, 48, 32)	8:1:3	3:1:1
Channel C	2000	RANS	(32, 96, 32)	393:1:131	3:1:1

resolve structures in the spanwise and streamwise directions. Channels B and C are at high enough Reynolds numbers that the inner region of the boundary layer cannot be resolved, and hence it has to be modeled. The wall model in case B is the algebraic model by Schumann [7], for which the cell adjacent to the wall is very coarse in all directions. In channel C the wall layer is modeled by a Reynolds Averaged Navier–Stokes (RANS) model that only requires the wall normal direction to be well resolved, and hence this grid has a much higher AR.

The time-averaged streamwise velocity is shown in Fig. 9, and it is seen that the results are reasonable. The well resolved channel A is in close agreement with the analytical profiles, as is channel B with the Schumann [7] model. Channel C shows an interesting and well-known trend: the results are correct, except for a jump in the velocity immediately outside of the LES/RANS interface (located at $y^+ = 65$). This is an ongoing area of research and is related to the lack of turbulent structures in the LES region adjacent to the interface (see, e.g., Piomelli and Balaras [8]). The lack of turbulent structures forces the velocity gradient to increase in order to maintain balance of the stresses.

The convergence factor and the CPU time required to solve the Poisson equation are averaged over five timesteps, and the normalized CPU time $\tilde{t}$ is plotted in Fig. 10. Two results are plotted for each case: one using the compromise values of $\mu_{\text{lim}} = 0.92$ and $\omega = 1.45$, one using the combination found to give the best performance for each case. As can be seen, the compromise values yield convergence that is about 30% worse compared to the optimum for each case. On the one hand, this shows that some tuning of these parameters is necessary to obtain the fastest convergence possible. On the other hand, it also shows that the compromise values yield reasonable results even for widely different cases. This is taken as a sign that the algorithm is relatively, but not perfectly, robust as far as different grid anisotropies are concerned.

Also plotted in the figure are the curve fits to the results for the three-dimensional cube from Eq. (17). It is seen that the performance for the channel flows agrees quite well with the cube results. Channels A and B have maximum aspect ratios between 1 and 100 and are located between the two curve fits, while channel C, as expected due to its higher grid stretching, has a higher computational cost.

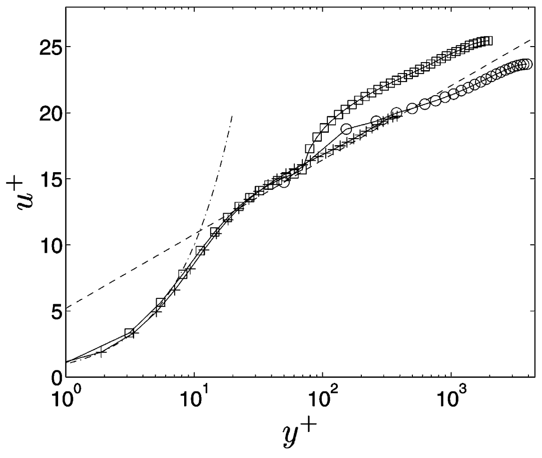

Fig. 9. Time-averaged streamwise velocity. +: channel A; ○: channel B; □: channel C; –·–: $u^+ = y^+$; ––: $u^+ = 2.44 \log y^+ + 5.2$.

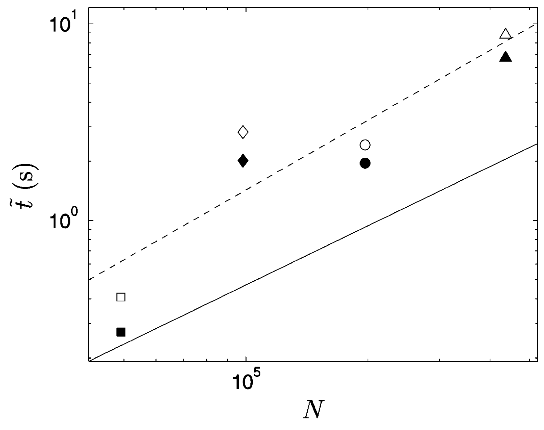

Fig. 10. Computational cost versus number of grid points. Open markers use $\mu_{\text{lim}} = 0.92$, $\omega = 1.45$. Filled markers use the best combination for each case. ○: channel A; □: channel B; ◇: channel C; △: wall mounted cube; —: Eq. (17) AR = 1; – –: Eq. (17) AR = 100.

4.3. Wall mounted cube

To test the algorithm on a complex geometry, the flow around a wall mounted cube is considered. A cube with dimension H is mounted on one wall of a plane channel, a flow that was studied experimentally by Martinuzzi and Tropea [9] and numerically by many authors (see, for example, Rodi et al. [10]). The

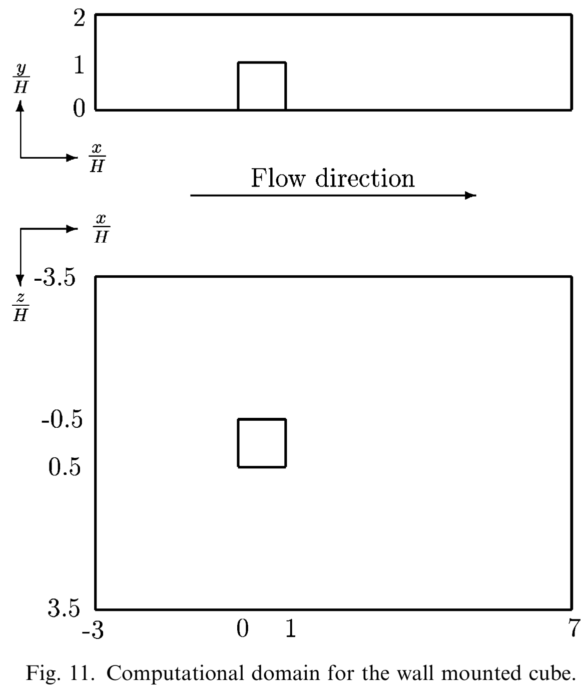

Fig. 11. Computational domain for the wall mounted cube.

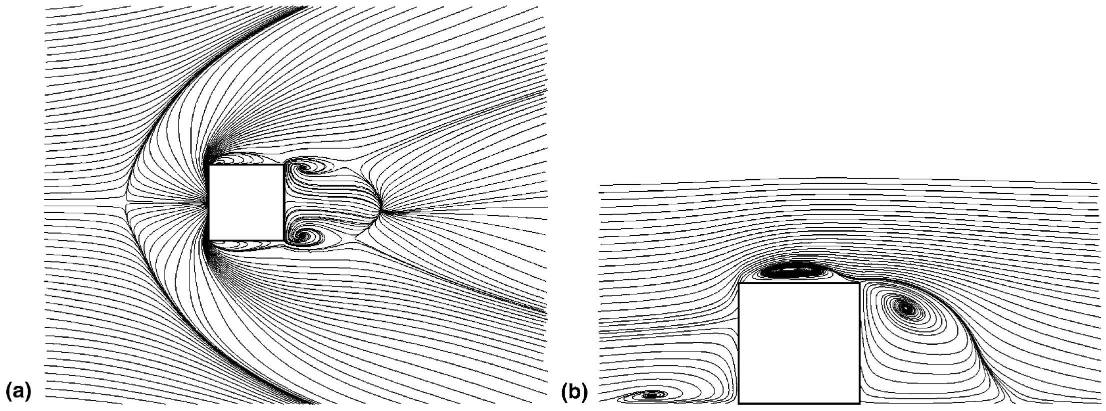

Fig. 12. Time-averaged streamlines around the wall mounted cube. (a) Close to the floor. (b) Slice at $z = 0$.

Reynolds number based on cube height and bulk velocity is 40,000, and the inlet velocity profile is taken from the experiments by Martinuzzi and Tropea [9].

The geometry is shown in Fig. 11. The grid used here is a multiblock grid with 17 blocks, with stretching performed in all directions. It is used with a simple algebraic wall function, and hence the maximum aspect ratio is a modest 35. The grid consists of 436,696 cells.

Fig. 12 shows time-averaged streamlines, and the complexity of the flow can be appreciated from the figures. A horseshoe vortex is formed upstream of the cube, and it curves around the sides of it. The flow is separated on the sides and on top of the cube, and a wake is formed downstream.

The normalized CPU time $\tilde{t}$ is plotted in Fig. 10, and as can be seen the computational cost agrees well with the results for the unit cube. This indicates that the semicoarsening algorithm is relatively robust for different geometries, since it yields predictable convergence performance for several different kinds of grids. As was the case for the channel flows, the compromise values of μ_{lim} and ω do not yield optimum performance, but rather about 30% worse than what can be achieved by case-dependent tuning.

5. Summary

A conditional semicoarsening multigrid algorithm that maintains a reasonably high convergence rate even on highly stretched grids has been investigated. The algorithm displays high parallel efficiency, mainly due to the inherent parallelism of the RBPGS smoother. The smoothing behaviour predicted by local Fourier analysis is used to determine how to coarsen the grid, and while the grids are not optimal in terms of how they reflect the local anisotropy, they do maintain a structured topology at all levels. Several numerical experiments were performed to determine the parameters involved. The combination $\mu_{\text{lim}} = 0.92$ and $\omega = 1.45$ was found to yield convergence within a computational cost of about 30% of the optimum for all cases studied. The optimum combination was found to be case dependent, and some tuning on a case-by-case basis is required to find this combination. This being said, the fact that performance within 30% of the optimum can be achieved by one combination for the many cases studied here suggests that the algorithm is relatively robust with respect to grid anisotropies.

The computational cost does not scale linearly with the number of grid cells, but rather like $\sim N^{1.18}$ for grids with a maximum aspect ratio of 100. While the scaling is not optimal, the actual cost level is competitive with other multigrid algorithms for anisotropic problems, especially in parallel situations. Parallel efficiencies of above 90% were found for grids with as few as 4096 cells per processor.

Acknowledgements

This work was supported by the National Sciences and Engineering Research Council of Canada (NSERC) and the Defence R&D Canada – Suffield. The authors are also grateful for the helpful comments by a reviewer, which considerably improved the manuscript.

References

[1] U. Trottenberg, C. Oosterlee, A. Schuller, Multigrid, Academic Press, New York, 2001.
[2] J.H. Ferziger, M. Peric, Computational Methods for Fluid Dynamics, Springer, Berlin, 1997.
[3] I. Yavneh, Multigrid smoothing factors for red-black Gauss–Seidel relaxation applied to a class of elliptic operators, SIAM J. Numer. Anal. 32 (1995) 1126–1138.
[4] I. Yavneh, On red-black SOR smoothing in multigrid, SIAM J. Sci. Comput. 17 (1996) 180–192.
[5] P.M. de Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, J. Comput. Appl. Math. 33 (1) (1990) 1–27.
[6] Y. Zang, R.L. Street, J.R. Koseff, A non-staggered grid, fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, J. Comput. Phys. 114 (1994) 18–33.
[7] U. Schumann, Subgrid scale model for finite difference simulations of turbulent flows in plane channels and annuli, J. Comput. Phys. 18 (1975) 376–404.
[8] U. Piomelli, E. Balaras, Wall-layer models for large-eddy simulations, Annu. Rev. Fluid Mech. 34 (2002) 349–374.
[9] R.J. Martinuzzi, C. Tropea, The flow around surface-mounted, prismatic obstacles placed in a fully developed channel flow, J. Fluid Eng. 115 (1993) 85–92.
[10] W. Rodi, J.H. Ferziger, M. Breuer, M. Pourquie, Status of large eddy simulation: results of a workshop, J. Fluid Eng. 119 (1997) 248–261.