INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **38**:21–41 (DOI: 10.1002/fld.202)

Local block refinement with a multigrid flow solver

C. F. Lange[1], M. Schäfer[2,*,†] and F. Durst[3]

[1]*Department of Mechanical Engineering, University of Alberta, Edmonton, Alberta, Canada, T6G 2G8*
[2]*Department of Numerical Methods in Mechanical Engineering, Technische Univ. Darmstadt, Petersenstr. 30, D-64287 Darmstadt, Germany*
[3]*Institute of Fluid Mechanics, University of Erlangen-Nürnberg, Cauerstr. 4, D-91058 Erlangen, Germany*

SUMMARY

A local block refinement procedure for the efficient computation of transient incompressible flows with heat transfer is presented. The procedure uses patched structured grids for the blockwise refinement and a parallel multigrid finite volume method with colocated primitive variables to solve the Navier–Stokes equations. No restriction is imposed on the value of the refinement rate and non-integer rates may also be used. The procedure is analysed with respect to its sensitivity to the refinement rate and to the corresponding accuracy. Several applications exemplify the advantages of the method in comparison with a common block structured grid approach. The results show that it is possible to achieve an improvement in accuracy with simultaneous significant savings in computing time and memory requirements. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: grid refinement; block-structured grids; multigrid; finite volume method

1. INTRODUCTION

A significant challenge in computational fluid dynamics is the simulation of flows in domains with realistically complex geometries and steep gradients in the solution, which often occur in engineering practice. Besides the general difficulty of generating an adequate grid for geometrically complicated configurations, the mesh has to be very fine in regions with steep gradients in the solution, in order to achieve the required numerical accuracy. An overall refinement of the mesh, however, may also lead to a high resolution in regions where it is not required. This implies an unnecessary waste of memory and CPU-time, often limiting the quality of the simulation. This problem can be circumvented by a procedure that enables a local refinement of the mesh, limited to regions where it is actually needed.

*Correspondence to: Prof. M. Schäfer, Dept. of Numerical Methods in Mechanical Engineering, Technische Univ. Darmstadt, Petersenstr. 30, D-64287, Germany.
†E-mail: schaefer@fnb.tu-darmstadt.de

Received 8 August 2000
Revised 21 December 2000

Several alternatives have been proposed in the literature for the implementation of local grid refinement. Existing methods can be generally grouped into three categories:

- p-method: the approximation order of the discrete operator is increased in the refinement region;
- h-method: new points are added to the grid in the refinement region;
- r-method: the grid points are redistributed and clustered in the refinement region.

The p-method is sometimes implemented in combination with finite element methods, but the h- and r-refinements are by far more frequently used. Finite element codes very often work with unstructured grids, which have the inherent ability to locally restrict the refinement, either by the h-, or by the r-method, but they also have more complex data management and difficulty in the development of efficient solution algorithms. A separate data structure for the finite element connectivity is always needed. Some examples of this methods are found in References [1–3].

Structured grids are very common in conjunction with finite volume or finite difference discretisation methods. Structured grids have a very simple data management, because connectivity of the control volumes is known *a priori*. The grid structure enables a very efficient implementation of the code. Unfortunately, local refinement of structured grids is not a straightforward task. The simple relocation of grid points following the characteristic lines of the problem allows for an improvement in the solution without any change in the original data structure and has been extensively investigated [4–10]. This r-refinement method, however, may result in very skewed and distorted grids at high refinement rates, especially in the case of convection dominated flows [11], if no constraints are imposed [6, 10]. Moreover, the mesh refinement in a region of the domain is always achieved at the expense of a mesh coarsening in the remaining regions.

One approach to the grid refinement of structured grids by the addition of new computational points (h-method) is the simple subdivision of the control volumes into smaller ones in the region, where refinement is required. In some cases, the refined regions are treated as different grid levels and the results on a coarse level serve as boundary condition and initial guess to the computation of a finer level. By this way the grid structure is maintained at each grid level. A fairly successful example of this strategy is the AMR method of Quirk [12–14], following the idea of Berger and Collela [15] and designed initially for use with Cartesian grids. Several authors adopted the idea and extended the method, or applied it to different physical problems [16–21]. This method seems to be best suited for single processor computations, but complex data management is required and a complete parallelisation is not straightforward. In order to preserve the efficiency of the multigrid method in the parallel implementation of a similar multi-level approach [22], each processor had to solve a compatible grid that covered the full domain with refinement confined to the processor's partition of the total grid.

Other authors couple refined and non-refined regions at the same level, computing the whole grid simultaneously [23–28]. Special discretisation schemes are required for the coarse–fine grid transition. Properly speaking, this type of grids must be considered unstructured, although mainly composed of quadrilaterals (2D) or hexahedrals (3D). Hence, this kind of h-refinement of an initially structured grid loses some of the structured grid advantages, like simple data management, while imposing limitations to the employment of multigrid solvers [29]. On the other hand, this class of methods is able to restrict the refinement more exactly to the required region.

A convenient way to approach the grid refinement problem with structured grids is to divide the domain in several blocks. The blocks in the refinement region are then discretised with finer meshes. This also simplifies the grid generation process, because each block can match a part of a complex boundary geometry without becoming very skewed, or loosing its simple structure, which is important for the achievement of high numerical efficiency. Moreover, such a block structure is advantageous with respect to parallel computing, because it can serve as a base for the parallelisation of the method by grid partitioning. This kind of refinement corresponds to the *h*-method and was the approach implemented in the present study.

Using a block-structuring technique of local grid refinement, the treatment of the block interfaces needs special attention in order to ensure a proper coupling of the subdomains. There are several ways by which the blocks can be interconnected. The highest degree of simplicity in the grid generation process is achieved when the blocks connect to each other by arbitrary overlapping regions (so-called *Chimera grids*, e.g. References [30, 31]). However, it is difficult to ensure conservation and to treat regions with strong gradients in this case [32]. Also, the interpolation between the overlapping regions may have to be constructed differently for different configurations, restricting the generality of the code.

Another way to handle the connection between the blocks is to patch their internal boundaries together, making them share a common interface line, but allowing a different point distribution for each block (non-intersecting grid lines). This strategy results in a highly efficient implementation, since redundant regions are avoided. These *patched grids* are also called *zonal grids* [33], *block adaptive grids* [34] or, in a more general classification, *partially discontinuous structured composite grids* [35].

Considering the attributes of the code that served as basis to the present grid refinement method, i.e. very efficient implementation of a block-structured parallel multigrid solver, the best compromise of an *h*-method that preserves these attributes was a blockwise refinement using patched grids. Patched grids have been used mainly in aerodynamics to solve the Euler equations, e.g. see Reference [36]. Rai [33, 37, 38] studied the adaptation of several solution techniques to the application of patched grids. He also adapted a combined Euler/Thin-Layer-Navier–Stokes solver to the simulation of rotor/stator interaction, using these kind of grids to handle the relative movement of different parts of the flow domain [39]. Other publications on the subject were concerned with applications or extensions of this technique (e.g. References [40–42]). In a different approach, Seidl *et al.* [34] employed patched grids to the solution of the Navier–Stokes equations and showed the suitability of the blockwise refinement to parallel computation.

The local refinement method described in this paper, called local block refinement (LBR), is aimed at increasing accuracy and efficiency in the computation of flow problems. Special care was taken to allow for the parallel computation of the locally refined blocks and to preserve the good convergence behaviour of the multigrid method. The general discretisation procedure is first outlined in the next section, followed by a detailed description of the local refinement method in Section 3. In Section 4, the LBR method is then validated and analysed by means of simple and complex flow problems, regarding convergence behaviour and gains in accuracy and computational costs.

2. DISCRETISATION PROCEDURE

Considering the flow of an incompressible Newtonian fluid in a two-dimensional geometry, the relevant governing equations are the conservation of mass, momentum and energy. In the case of turbulent flows, the governing equations are considered to be in the Reynolds averaged form and equations for the conservation of the turbulent kinetic energy and its dissipation rate (k-ε turbulence model) are additionally included. The governing equations are solved via a block-structured finite volume method, whereby the solution domain is divided into blocks and each block is discretised in a structured mesh of quadrilateral (in general non-orthogonal) control volumes (CVs).

To calculate the balance of the conserved quantities, the governing equations are integrated over each CV. Representing by ϕ the transported quantity in each equation, the integral form of all governing equations can be expressed in a general form

$$\zeta\left[\int_\Omega \frac{\partial(\rho\phi)}{\partial t}\,\mathrm{d}\Omega + \int_\Omega \frac{\partial(\rho U_i\phi)}{\partial x_i}\,\mathrm{d}\Omega\right] = \int_\Omega \frac{\partial}{\partial x_i}\left(\Gamma_\phi \frac{\partial\phi}{\partial x_i}\right)\mathrm{d}\Omega + \int_\Omega Q_\phi\,\mathrm{d}\Omega \tag{1}$$

where Ω is the volume of the CV, which in the 2D case corresponds to its area, U_i and x_i $(i=1,2)$ are the Cartesian velocity components and coordinate directions respectively, t is the time, ρ is the density, Γ_ϕ is a diffusion coefficient and Q_ϕ is a distributed source. The values of these variables in each balance equation are shown in Table I, where μ is the dynamic viscosity, P is the pressure, c_p is the specific heat, k is the thermal conductivity and Φ is the viscous dissipation function. For the simulation of turbulent flows, the k-ε-model of Launder and Spalding [43] is employed, whereby k_t is the turbulent kinetic energy, μ_t is the eddy viscosity, P_k is the rate of production and ε is the dissipation rate. The four remaining variables (σ_k, σ_ε, c_{ε_1} and c_{ε_2}) are adjustable parameters of the model.

The solution of the coupled set of non-linear equations is based on the well-known *SIMPLE* algorithm [44]. The discretised momentum equations are linearised using values for the pressure and mass fluxes from the previous iteration. With the resulting velocity field a pressure correction equation is assembled and solved. The mass fluxes, velocities, and pressure are then corrected and other possible scalar equations are solved. This procedure is repeated until the maximum sum of the absolute residuals in all equations is reduced to a prescribed value. Details about the discretisation process and the pressure–velocity coupling can be found in References [45] and [46].

Table I. Values of variables in the general balance (Equation (1)).

Equation	ζ	ϕ	Γ_ϕ	Q_ϕ
Continuity	1	1	0	0
Momentum	1	U_j	μ	$-\frac{\partial P}{\partial x_j} + \frac{\partial}{\partial x_i}\left(\mu\frac{\partial U_i}{\partial x_j}\right)$
Energy	c_p	T	k	Φ
Turb. kin. energy	1	k_t	$\frac{\mu_t}{\sigma_k}$	$P_k - \rho\varepsilon$
Dissipation	1	ε	$\frac{\mu_t}{\sigma_\varepsilon}$	$\frac{\varepsilon}{k}(c_{\varepsilon_1}P_k - c_{\varepsilon_2}\rho\varepsilon)$

An iterative ILU decomposition method is used for the solution of the sparse linear systems of the resultant equations [47] and a non-linear multigrid scheme is employed for convergence acceleration [48]. For the parallel computation, a message passing method and a block structured grid partitioning with non-overlapping subdomains and auxiliary CVs along the block interfaces are employed [49].

3. LOCAL BLOCK REFINEMENT

The present refinement strategy is based on the block structure of the grid. The fully structured case, when grid lines at internal block boundaries must match on both sides of the interface, is a particular case of a generic block structure. The generalisation consists in relaxing the condition of zero order continuity of grid lines across block interfaces. The point distribution of the grid can now be arbitrarily defined at both sides of the internal boundary. This generic block structure is called *patched grid* and Figure 1 shows an example of it.

In the flow simulation program, the coupling of the generic block structure demands a special treatment of the block interfaces. If we consider, for example, that two blocks are located on different processors, all that each processor needs to know about the neighbour is the information stored in the layer of auxiliary CVs at the interface. This information is used as a boundary condition for the block and actualised from sweep to sweep. In other words, for each CV in the vicinity of an internal boundary, there must be an auxiliary CV on the other side of the interface, which contains the data of an equivalent CV in the neighbour block. When the point distribution along the interface coincides, a simple exchange of CV data at the interface suffices. Patched grids, however, require an additional data structure and an interpolation procedure, since the number of CVs on each side of the interface is no longer the same.

To supply the information to the neighbour, an array of virtual CVs fitting the mesh distribution of the neighbour is computed at each interface segment. This provides a data structure identical to that expected by the neighbour block.

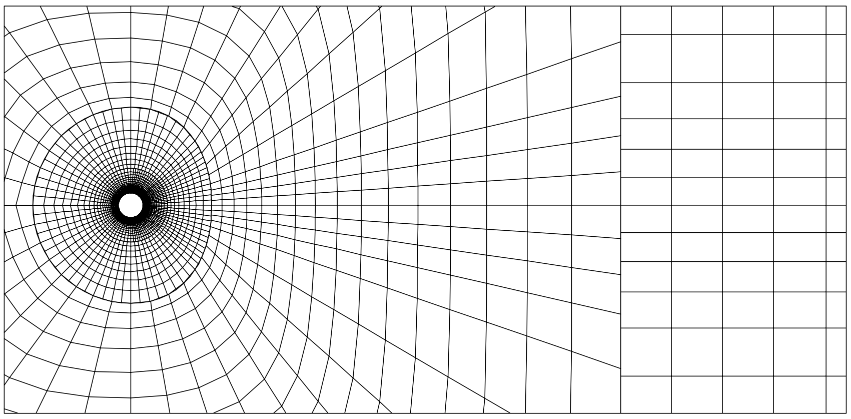


Figure 1. Example of patched grid around a cylinder.

During the computation, the data needed by the block at the interface are of two types:

- *geometrical data*: data of this kind need to be exchanged only once, since they do not change during the calculation;
- *dependent variables and matrix coefficients*: this kind of data has to be interpolated and exchanged at each inner (sweep) and outer iteration.

To follow the goals of having, at the same time, the lowest storage overhead and the highest efficiency, each kind of data is treated in a different manner.

Interpolation of geometrical data

The geometrical quantities of the virtual CVs are interpolated at the beginning of the computation in a side-by-side procedure. Immediately after the interpolation, they are exchanged and stored in the neighbour block in the usual arrays for geometrical data. This way, the supplementary arrays needed for the interpolation are kept small and local to the subroutine. This causes the compiler to reserve only the memory amount needed for one side, because this space is deallocated at the end of the subroutine and can be reused for the next side.

In the flow simulation program, all virtual quantities that need to be exchanged repeatedly lie at the centre of the CVs. Therefore, all can be interpolated by the same procedure. Because these exchanges have to be performed many times during the computation, the efficiency of this procedure is more important than memory saving. Hence, two additional global arrays for each block side were introduced, providing for a very efficient transfer of virtual data during the computation.

It is important to emphasise the difference between auxiliary CVs and the recently introduced virtual CVs. Auxiliary CVs are a regular part of the grid structure in a block, containing information received from the neighbour block and matching the grid point distribution of a block at the interface. In contrast, virtual CVs do not fit, in general, in the grid structure of a block. They are generated in order to match the grid point distribution of the neighbour at the interface and serve to the transmission of data to auxiliary CVs at the neighbour block.

Since it is assumed that the grid does not move, values of the length of the CV face and interpolation factor for the face centre of virtual CVs need to be computed only once. After receiving from the neighbour block (processor) information about the number of CVs and the position of its grid points at the interface segment, a processor is able to compute the geometrical properties of virtual CVs, which would be exactly the CVs of the first row at the interface, if there was no local block refinement (see Figure 2). The virtual CVs have similar inclinations as the actual CVs, causing all quantities to be interpolated, never extrapolated, which is clearly recognised in Figure 3. The main advantage of this strategy is the guarantee that all virtual CVs lie within the boundaries of the block, which computes them.

First of all, the *virtual boundary points* received from the neighbour are ordered between two real boundary points and this order is stored in a database. A linear interpolation factor is then computed based on the relative position of each virtual boundary point. With this factor, *virtual internal points* are computed for each virtual boundary point, following the same inclination of the local grid lines and forming the frame of the virtual CV layer. This ensures that the virtual layer of CVs will never be defined outside the boundaries of the current block. Finally, the virtual CV centres and interpolation factors from the virtual centres to the middle of the faces are calculated.

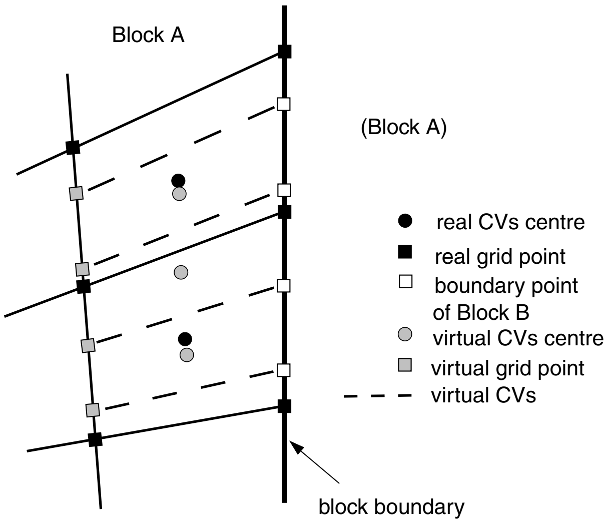


Figure 2. Virtual CVs generated in Block A. The boundary points of Block B (□) are the only information transmitted to Block A.

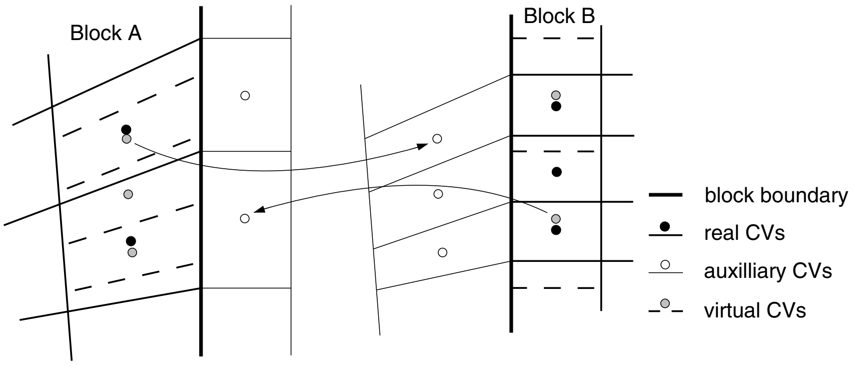


Figure 3. Blocks with auxiliary CVs after interpolation and exchange of geometrical data. Also shown is the data transfer from virtual to auxiliary CVs.

All virtual geometrical values are then sent to the neighbour block, where they are stored in the usual geometrical arrays (see Figure 3). After being sent to the neighbours, the virtual geometrical data are overwritten, since they are no longer needed in the current block. The allocated memory is therefore freed to be used by other local arrays, ensuring the smallest requirement of additional memory.

Interpolation of coefficients and dependent variables

All coefficients and dependent variables are needed at the centre of the virtual CVs and, thus, their interpolations and exchanges can all follow the same procedure. In order to prepare

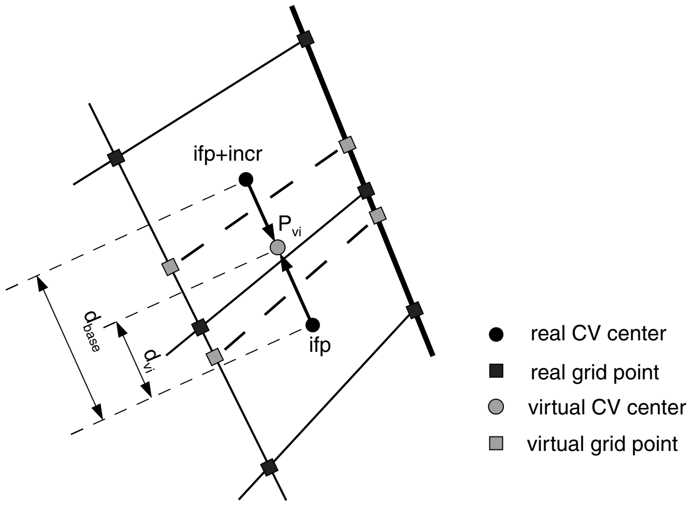


Figure 4. Interpolation of virtual centre values.

for the interpolation, the two real centre points closest to a virtual centre $\mathsf{P}_{\mathsf{V}_i}$ are identified, the first real centre being called the *base point for the interpolation* (see Figure 4). Note that these real centre points are not necessarily related to the real boundary points used to determine the virtual boundary points. The linear interpolation factor *fp* based on the distance to the real centres is calculated by

$$fp_{\mathsf{V}_i} = \frac{\mathsf{d}_{\mathsf{V}_i}}{\mathsf{d}_{\mathsf{base}}} \tag{2}$$

where $\mathsf{d}_{\mathsf{V}_i}$ and $\mathsf{d}_{\mathsf{base}}$ are defined in Figure 4. This interpolation factor is stored in a global array, together with the index of the base point for the interpolation ifp. This means that just one integer and one real (double-precision) array per block side are stored permanently, representing an increase of only few per cent caused by the grid refinement procedure in the memory requirement. The calculation of *fp* has to be done during the initialisation, immediately after the calculation of the geometrical data, while these data are still available. During the computation, no additional information is required.

Every time an exchange of variables or coefficients has to be performed, the corresponding variable is linearly interpolated at the virtual CV's centre by

$$\phi_{\mathsf{V}_i} = \phi_{\mathsf{ifp}} + fp_{\mathsf{V}_i}(\phi_{\mathsf{ifp+incr}} - \phi_{\mathsf{ifp}}) \tag{3}$$

and then sent to the corresponding auxiliary CV at the neighbour (see Figure 3).

So, the interpolation of dependent variables and coefficients during the computation becomes simple and very efficient, causing the lowest possible increase in effort for the exchange of interface data, compared to the implementation without local block refinement.

An important property of the virtual CV definition and interpolation procedure described above is the exact resembling of the original data transfer method in case of no local block refinement. If regular fully structured grids are used, the virtual CVs coincide with actual CVs and no interpolation is performed.

Special attention had to be devoted to the definition of virtual CVs and the exchange of data at block corners in junctions of three or four blocks. In the case of non-orthogonal grids, cross-derivatives of the velocity field along the CV faces may have non-zero values. These cross-derivative terms are part of the molecular (diffusive) transport of momentum. The discretisation of the cross-derivative terms introduces an influence of the 'diagonal' neighbours (NW, NE, SW and SE) on each CV. This contribution increases with larger deviation from orthogonality and with an increased diffusion dominance in the flow. In addition to this, the consistent definition of block corner CVs plays a key role in the convergence of the multigrid method. Because of these considerations a special procedure was developed for the treatment of block corners (see Reference [50], for details).

Mass flux calculation at internal boundaries

Fuchs [51] performed one of the rare investigations on patched grids in connection with the multigrid method. He advised that, besides the consistent information exchange between the blocks, some conservation laws have to be satisfied very accurately (to the level of round-off errors) in order to obtain convergence. Depending on the system of partial differential equations and on the type of data exchange between the blocks, other conservation laws may be satisfied approximately, i.e. up to the level of truncation errors. In the present flow simulation program, the mass conservation law is of the former type and requires a specific data exchange procedure. The remaining quantities (momentum, thermal energy, etc.) admit the satisfaction of conservation up to the overall truncation error of the scheme, which is ensured by the exchange procedure described in the previous section.

Because the SIMPLE method uses the continuity equation for the coupling of velocities and pressure, it is essential that the mass fluxes through the block interfaces agree in both blocks precisely. The present interpolation method, however, does not satisfy intrinsically this requirement, as shown in the following.

The exact mass flux through the interface can be expressed by

$$\dot{m} = \int \rho U_i n_i \,\mathrm{d}S \tag{4}$$

where U_i are the components of the velocity vector, ρ is the fluid density, S is the common surface between the blocks (in the 2D case it is represented by a line) and n_i are the components of the outward directed normal vector to this surface. In the discretisation, this integral is approximated for the east side of a block A by the sum

$$\dot{m} \approx \dot{\mathrm{m}}|_{\mathrm{A}} = \sum_{N_{\mathrm{A}}} \dot{\mathrm{m}}_{\mathrm{e}} = \sum_{N_{\mathrm{A}}} \rho U_{i_{\mathrm{e}}} n_i \delta s \tag{5}$$

where $U_{i_{\mathrm{e}}}$ are the components of the velocity vector interpolated in the centre of the east face of a CV adjacent to the interface, δs is the area of the same face with its outward normal components n_i. N_{A} is the number of control volumes located in block A along the interface to block B. At the neighbour block B, a similar summation takes place, producing $\dot{\mathrm{m}}|_{\mathrm{B}}$.

Note that using patched grids, in general, we have $N_{\mathrm{A}} \neq N_{\mathrm{B}}$ and, therefore, $\dot{\mathrm{m}}|_{\mathrm{A}} \neq \dot{\mathrm{m}}|_{\mathrm{B}}$ even if the two velocity profiles are the same. The approximated fluxes, $\dot{\mathrm{m}}|_{\mathrm{A}}$ and $\dot{\mathrm{m}}|_{\mathrm{B}}$, will only coincide in the case of an equal distribution of CVs along the interface or of a constant gradient of the momentum $\rho\mathbf{u}$. In both cases, the summations agree up to round-off errors irrespective of the number of CVs. However, if the distribution of the momentum $\rho\mathbf{u}$ over the interface is not linear and the CV distribution does not coincide, the approximations will differ, since the approximation in the side with more CVs will be more accurate than the other.

To force the satisfaction of the mass conservativeness condition, the present method has to execute a correction every time the mass flow rate is calculated or exchanged. Two kinds of corrections were tested. The first kind, called correction by product, used a factor to correct the mass fluxes on each CV along the interface of one of the neighbouring blocks, forcing the summation to equal the total mass flux of the neighbour block. This correction method was found to be unstable, amplifying oscillations, when there was recirculation or the mass flux through the interface was close to zero.

The second kind of mass flux correction, called correction by sum, was preferred. Instead of using a factor, one of the blocks, for instance block B, distributes the difference between the two total flow rates, $\Delta\dot{m}$, among its CVs, adding a correction to the mass flux of each CV at the interface side. To perform a distribution that accounts for an eventual recirculation through the interface, the amount of correction corresponding to an individual CV is based on the sum of the absolute mass fluxes and is given by

$$\Delta\dot{m}_{\mathrm{w}_l} = \left[\frac{|\dot{m}_{\mathrm{w}_l}|}{\sum_{j=1}^{N_\mathrm{B}} |\dot{m}_{\mathrm{w}_j}|}\right]\Delta\dot{m} \quad l = 1, N_\mathrm{B} \tag{6}$$

The resulting corrected mass fluxes

$$\dot{m}^*_{\mathrm{w}_l} = \dot{m}_{\mathrm{w}_l} + \Delta\dot{m}_{\mathrm{w}_l} \quad l = 1, N_\mathrm{B} \tag{7}$$

satisfy the equality

$$\dot{m}|_\mathrm{A} = \dot{m}^*|_\mathrm{B} \tag{8}$$

Finally, after the correction of the mass flux, the velocity vector of the auxiliary CV at the boundary, used originally to compute the flux, is correspondingly corrected in order to preserve the consistency of the method.

In general, the mass flux correction has to be seen as a link between the two regions with different mesh spacing. Thereafter, as the solution converges, the correction $\Delta\dot{m}$ may converge to a value slightly different from zero, i.e. the correction continues to be needed, even when the exact numerical solution is achieved. The magnitude of this value tends to increase as the refinement rate $N_\mathrm{B}/N_\mathrm{A}$ departs from the unity.

4. NUMERICAL RESULTS

The present LBR procedure was validated and evaluated by means of three test cases. The plane Poiseuille flow in a channel served to the study of convergence behaviour and robustness of the procedure. Two practical, more complex applications, namely the laminar, confined flow around a cylinder and the turbulent flow around a car allowed the estimation of practical

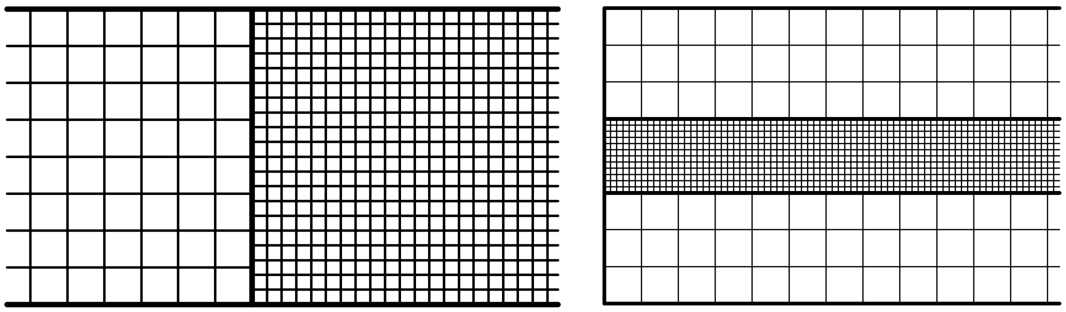


Figure 5. Detail of grids for channel flow.

performance enhancements and accuracy gains through LBR, specially in conjunction with the multigrid method.

A very important characteristic of the new local block refinement scheme is its consistency with the original flow simulation solver. In the case of a conventional grid, with no local refinement, both versions showed exactly the same convergence history and numerical behaviour and, therefore, they converged to the same solution. The increase in the computation time was always smaller than 5 per cent, compared to the original version of the code. This was verified for all test problems in serial as well as in parallel computations. This consistency with the original version of the code allows the assumption of a parallel performance similar to that version (see e.g. Reference [49]).

First test case: channel flow

The plane Poiseuille flow was used to test the proposed local block refinement scheme with respect to its sensitivity to the refinement rate *RR*, defined as the ratio between the number of CVs on both sides of the block interface. Two cases were considered: in one the interface between the refined and non-refined blocks was perpendicular to the flow, and in the other two interfaces, parallel to the flow, separated the blocks. The first case tested the procedure regarding the convective terms, while the second tested mainly the diffusion terms, because these were the dominant fluxes through the interfaces. An example of each case is shown in Figure 5.

The number of outer iterations required for convergence after a sudden start from rest to a Reynolds number of 100 was used as the comparison parameter. The results can be seen in Table II. At first, we note that the remarkable advantages of the multigrid acceleration method in the non-refined case ($RR = 1$) are preserved and even extended when LBR is applied. Table II also shows that the number of outer iterations with the multigrid method remains relatively stable up to $RR = 4$. For larger refinement rates, the number of outer iterations increased, in the case of the vertical interface, but not as fast as in the single grid computations. Because of memory restrictions and to preserve the same computational conditions, the computations were limited to a refinement rate of eight, but tests starting with a coarser grid showed a limited increase in computational effort even at higher values of *RR*. The number of iterations at a refinement rate of 16 increased only by a factor smaller than three compared with the case with $RR = 8$. With the horizontal interfaces, the method became unstable

Table II. Convergence for channel flow.

Refinement rate	Number of iterations			
	Vertical interface		Horizontal interface	
	Multigrid	Single grid	Multigrid	Single grid
1.0	22	312	31	424
1.5	25	359	49	380
2.0	25	416	53	352
2.5	25	489	45	386
3.0	29	576	41	484
4.0	41	831	37	760
5.0	61	—	45*	1108*
6.0	85	—	57*	—
8.0	161	—	85*	—

*Initial velocity $= 1.0\,\mathrm{m\,s^{-1}}$.

at $RR > 4$ and diverged. Divergence occurred because of velocity oscillations that appeared along the interfaces when the fluid started from rest. At large values of RR, the oscillations could no longer be damped by the mass flux correction procedure described in the previous section. The problem was circumvented by prescribing an initial velocity to the fluid, which prevented the occurrence of oscillations.

The effective advantage of employing the multigrid method cannot be extracted from Table II, because of the additional effort needed by V-cycles compared to single grid iterations. Therefore, the only suitable criteria is the computing time. To allow the comparison of cases with different numbers of CVs and to analyse just the influence of the numerical scheme, the computing time needed for the finest grid was divided by the total number of CVs on this grid level. The resulting time per CV was then divided by the equivalent time of the reference case, i.e. the case with multigrid and no local refinement. This normalisation resulted in a relative time that can be expressed as

$$\mathrm{RelativeTime}_{RR_i} = \frac{\left(\dfrac{\text{comp. time}}{\text{No. of CVs}}\right)_{RR_i}}{\left(\dfrac{\text{comp. time}}{\text{No. of CVs}}\right)_{RR_1-\mathrm{MG}}} \tag{9}$$

where RR_i stands for a refinement rate of i, computed either with the single grid or with the multigrid method. The relative computing time obtained, permitted the comparison of test runs on different machines, which would be otherwise impossible. It is also a measure of the numerical efficiency of the method.

Figure 6 shows the evolution of the relative computing time with the refinement rate for both cases and the clear advantage and stabilizing effect of the multigrid solver. This demonstrates, at the same time, the compatibility of the developed LBR scheme with the implemented multigrid solver. The graphics also contain the relative time needed by the code without LBR (black points at $RR = 1$), which was with multigrid only 4 per cent and without 2 per cent faster.

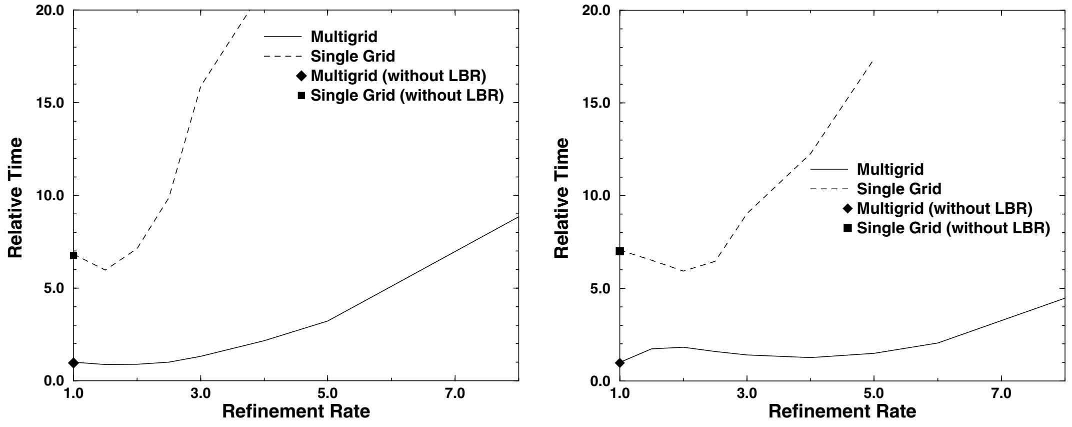


Figure 6. Comparison of the relative computing time for the channels with vertical interface (left) and with horizontal interfaces (right).

Based on the foregoing analysis, the developed method can be regarded as robust until a refinement rate of four, if diffusive transport through the interface dominates, and up to a much higher value, if the convection through the interface plays the main role. Numerical efficiency, represented by the number of outer iterations required for convergence was also significant up to a refinement rate of four, leading to $RR \leqslant 4$ as a recommended limit for practical applications. Higher refinement rates should be split into two or more steps, using transition blocks.

Second test case: confined flow around a cylinder

A more complex test case was employed to investigate the order of the local refinement scheme and its numerical behaviour in the case of a recirculating flow. To this end, the flow around a circular cylinder confined in a channel was used. Since there is no exact solution available for the considered problem, geometry and boundary conditions were chosen to meet a standard, well-defined test case, published by Schäfer and Turek [52]. This standard problem was utilised for the comparison of a series of flow simulation codes, based on a variety of numerical schemes. Schäfer and Turek [52] presented and analysed the benchmark results of about 15 different codes. Although the results showed a relatively large scatter due to large differences in spatial and time resolution, it was possible to establish lower and upper bounds for the correct solution based on the more accurate computations. These lower and upper bounds differed at some quantities, e.g. drag coefficient, by only 0.3 per cent, demonstrating the reliability of the obtained results.

The geometry of the standard problem is shown in Figure 7, where all length scales are given relative to the cylinder diameter $D = 0.1$ m. An incompressible Newtonian fluid with a kinematic viscosity of $\nu = 10^{-3}\ \mathrm{m^2\,s^{-1}}$ and a density of $\rho = 1.0\ \mathrm{kg\,m^{-3}}$ was prescribed. The flow case computed was a steady flow with $Re = 20$, corresponding to case 2D-1 in the mentioned benchmark.

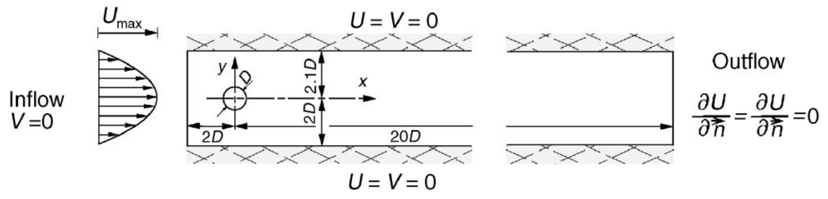

U_{max}
$U = V = 0$
Inflow
$V=0$
y
D
$2.1D$
$2D$
x
$20D$
Outflow
$\frac{\partial U}{\partial \vec{n}} = \frac{\partial U}{\partial \vec{n}} = 0$
$U = V = 0$

Figure 7. Geometry and boundary conditions for the confined flow around a cylinder.

Table III. Results for steady test case 2D-1 in Reference [52].

Code	Unknowns	C_D	C_L	L_a	ΔP	Mem.	CPU time
FASTEST-LBR	150 528	5.5843	0.0109	0.0827	0.1176	39 MB	102 s
	37 632	5.5807	0.0111	0.0814	0.1176	11 MB	27 s
	9408	5.5632	0.0115	0.0775	0.1175	3 MB	6 s
FASTEST	294 912	5.5846	0.0106	0.0846	0.1176	75 MB	192 s
	73 728	5.5852	0.0105	0.0845	0.1176	19 MB	47 s
	18 432	5.5755	0.0102	0.0842	0.1175	5 MB	13 s
‘Correct solution’	lower bound	5.5700	0.0104	0.0842	0.1172		
	upper bound	5.5900	0.0110	0.0852	0.1176		

The benchmark in Reference [52] revealed the previous (non-refined) version of the code as one of the most efficient and accurate of all codes compared. This renders more significance to any enhancement in the performance of the cited code. The grid chosen to run the problem with LBR was a local refined version of the original grid used in the benchmark. This allows a more realistic comparison of the best performance of both versions.

The physical quantities proposed for comparison are the drag coefficient C_D, the lift coefficient C_L and the pressure difference ΔP between the foremost and the rearmost points of the cylinder, as well as the length of the recirculation region L_a. The results for the proposed quantities and the performance data obtained are summarised in Table III, where the present version of the code is called FASTEST-LBR. Following the rules prescribed for the benchmark, results for the last three grid levels (of a total of five multigrid levels) are presented. The results obtained with the previous version of the code (FASTEST) and published by Schäfer and Turek [52] (method 7a) are also presented in Table III for a better contrast. Computations were carried out on the same machine, a workstation HP735 with a Linpack1000 performance rate of 13 Mflops, allowing a direct comparison of the computing time. The memory requirements are given in Mbytes and the number of unknowns correspond to the sum of the points for calculation of U, V and P, i.e. three times the number of CVs.

Table III allows also an analysis concerning the accuracy of the solution, based on the proposed physical quantities. Since no exact solution for this problem is available, it is impossible to make a quantitative statement about the relative error of these results. It can be said, however, that the results converge to similar values, well inside the bounds given by Schäfer and Turek [52] for the ‘correct solution’ in almost all cases. The recirculation region L_a showed some lack of accuracy because the local refinement emphasised the cylinder vicinity, leaving part of the recirculation region out. This result stresses the importance of a careful choice of the locally refined region, depending on the objective of the computation.

The computation of the confined flow around a cylinder served also to estimate the order of accuracy of the spatial discretisation. With the results of three successive grid levels for a general quantity the order of accuracy of the numerical scheme can be estimated (see Reference [29]). The results of FASTEST-LBR, listed in Table III, indicate that the order of the scheme is approximately two, confirming that the order of the underlying discretisation scheme (second order) was preserved by the local refinement procedure.

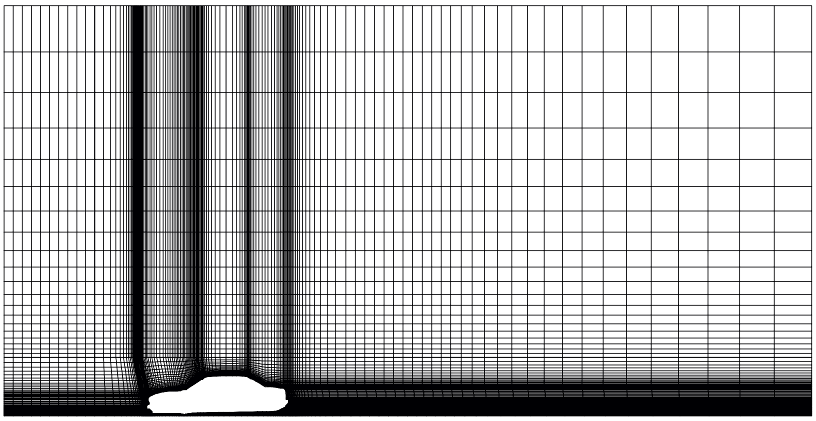
The results of Table III demonstrate that both memory requirement and computing time of the already highly efficient original code could successfully be halved by the application of local block refinement. These enhancements were obtained without loss of accuracy in quantities, such as C_D, C_L and ΔP, which clearly demonstrate the potentiality of the LBR method.

Third test case: turbulent flow around a 2D car model

As a final test case, the turbulent flow around a 2D car model was chosen to demonstrate the robustness and general applicability of the developed local block refinement method. This flow problem was investigated thoroughly by Angelis in his thesis [53]. Angelis' work dealt with the complementary numerical and experimental investigation of the flow around a 2D car model and studied, among others, the effect of the gap between the model and the ground. This 2D model consisted of a beam with a car profile that stretched over the entire cross-section of the wind tunnel. Precisely measured inflow boundary conditions and access to the original experimental results allowed a highly detailed comparison of results.

The geometry and boundary conditions adopted for this computation corresponded exactly to one of the cases investigated by Angelis [53], namely the case of a 3-mm gap between the model and the ground and inflow velocity of $15\,\mathrm{m\,s^{-1}}$ at the middle of the wind tunnel. The Reynolds number based on the model length was 3.38×10^5. Likewise in the previous test case, a conventional grid computed with the former version of the code was compared with a locally refined grid calculated with the present code version. The conventional grid chosen was the original grid used by Angelis [53], the first level of which can be seen at the top of Figure 8. The need for a very fine grid resolution in the vicinity of the model and the requirement to continue the grid lines up to the boundary led to the acceptance of extremely large grid aspect ratios in the bottom of the car and in the outer flow regions. As a compromise to avoid a fine grid resolution over the whole domain, the large aspect ratios have a negative effect on the convergence process and reinforce the damping effect of the employed upwind discretisation of the convection terms.

The locally refined grid had approximately the same resolution in the vicinity of the model and in the near wake region as the conventional grid (see bottom of Figure 8). Some grid points were added in the streamwise direction at the gap region between the model and the ground to reduce the large aspect ratios of the original grid. In the rest of the domain the new grid was heavily coarsened. This represented an economy of 34 per cent in the total number of CVs. In addition, the better aspect ratios and the reduction of the number of blocks from 19 to 12, resulting from the grid coarsening, had a very positive effect on the convergence behaviour, as it is shown below.



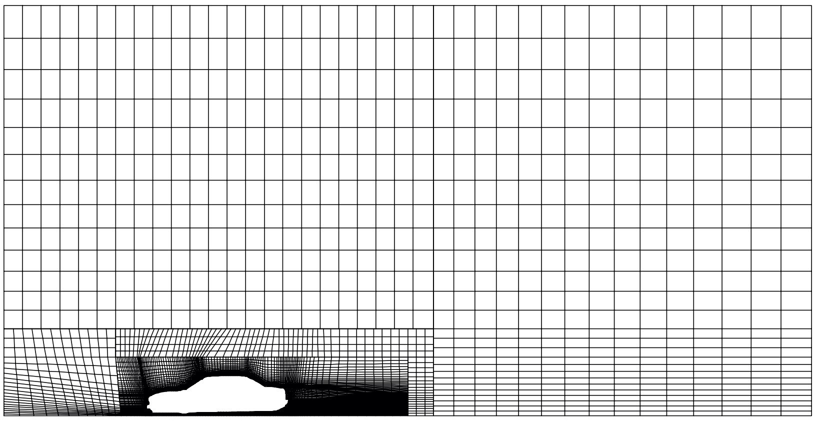


Figure 8. Conventional and locally refined grids for the computation of the third test case (first grid level).

While Angelis [53] calculated this problem using only two grid levels, a further fine grid level was employed for the present comparison. The convergence history for the computation of each grid level, the first with single grid and the others with multigrid method, is shown in Figure 9. The convergence improvement obtained with the locally refined grid is evident. While the first conventional grid level needed twice as many and the second level three times the number of iterations of the locally refined grid to converge, the third grid level failed completely to achieve the prescribed convergence limit (10^{-4}). The poor quality of the grid caused the residual norm of the multigrid solution to oscillate around a value of 5×10^{-4}.

The effect of the improved convergence behaviour of the locally refined grid together with the smaller computational effort needed per iteration (due to the smaller number of CVs) resulted in a total computing time for the first level of a third of the time needed for the conventional grid. The second grid level was computed more than five times faster by the locally refined grid, not to mention the third grid, for which no relation could be determined.

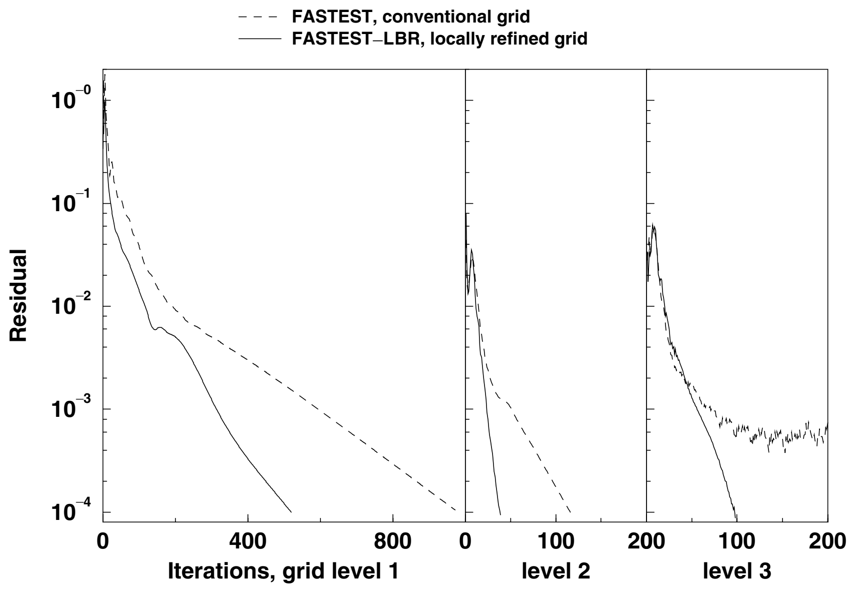
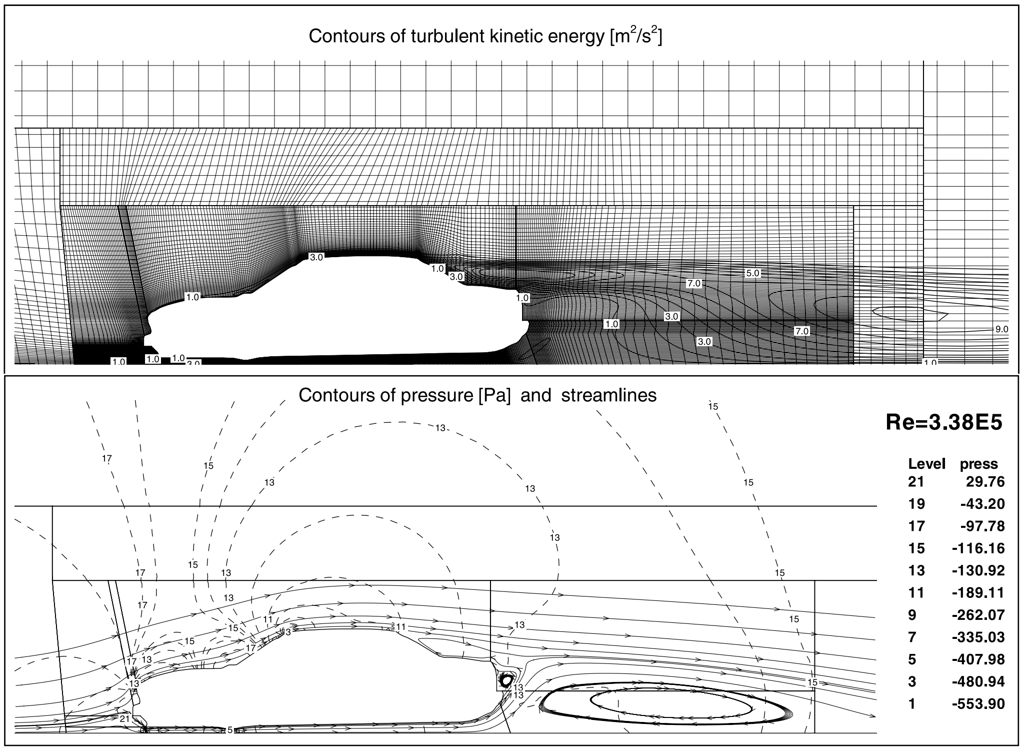


Figure 9. Comparison of the convergence history for the computation of the turbulent flow around a 2D car model.

The solution corresponding to the second grid level can be seen in Figure 10. At the top of the figure, a detail of the grid around the model is shown in the background of isolines of the turbulent kinetic energy k_T. The bottom part of Figure 10 shows streamlines and pressure contours for the present test case. It is worth noting the smooth transition of all isolines and streamlines from locally refined blocks around the model to coarser blocks in the outer flow region. Moreover, the form of the streamlines is very similar to the streamlines based on the experimental data reported by Angelis [53].

The quality of the results can be verified by a comparison with experimental values of the velocity components. In Figure 11 experimental results are plotted together with the numerical profiles of the third grid level for the V-velocity. The figure shows the good agreement between experimental and numerical profiles at the frontal part of the model. A small improvement in the solution compared with the original results was verified mainly for the V velocity component in the frontal region (data not shown).

The computation of this third test case demonstrated successfully the general applicability of the developed local block refinement scheme. The test case also revealed a very favourable effect of the locally refined grid on the convergence behaviour. This was a consequence of a significant improvement in grid quality, made possible by the flexibility introduced through the local refinement method in the grid generation process. Moreover, the improved grid allowed the computation of an additional fine grid level, which resulted in improvements in the accuracy. This more complex test case shows that the profit of the developed local block refinement method can be twofold: a significant gain in performance combined with a simultaneous increase in accuracy, if the grid is carefully generated.



Level	press
21	29.76
19	-43.20
17	-97.78
15	-116.16
13	-130.92
11	-189.11
9	-262.07
7	-335.03
5	-407.98
3	-480.94
1	-553.90

Figure 10. Results for turbulent flow around a 2D car model: contour lines of turbulent kinetic energy with grid (left) and contour lines of pressure and streamlines (right).

The LBR method has also been successfully applied to practical computations, such as the investigation of hot-wires in the vicinity of walls [54], where the local block refinement allowed for an unprecedented level of accuracy in the hot-wire simulations.

5. CONCLUSION

A new developed local grid refinement procedure, called local block refinement, was implemented, validated and investigated by means of several test cases. The method proved to preserve the order of accuracy of the underlying numerical scheme and showed excellent compatibility with the multigrid solver, fully exploiting its advantages of convergence acceleration and stability. Moreover, the local block refinement was found to be robust and provided substantial gains in performance and reduction of memory requirements to the employed numerical code. In the test cases considered, computing time gains up to 80 per cent and reduction of memory requirements up to 50 per cent could be verified.

Although the implementation and results presented correspond to the two-dimensional version of the code, the extension of the procedure to three dimensions is, in principle, straightforward, but the special treatment of block corners will likely be very complex. However,

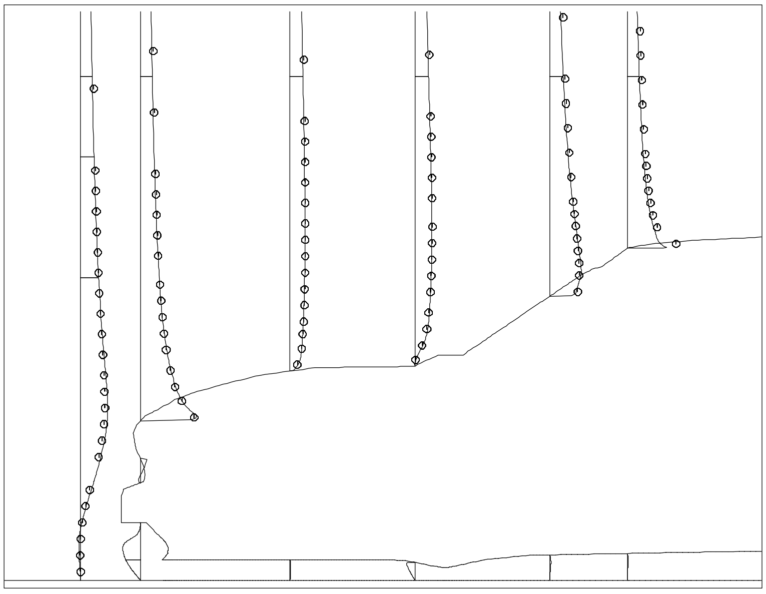


Figure 11. Comparison of numerical (lines) and experimental (circles) profiles of velocity component V.

it is expected that the local block refinement procedure will provide even larger gains in performance and memory requirements on three dimensions.

ACKNOWLEDGEMENTS

The authors thank W. Angelis for the help in the comparison with the computational and experimental results from his thesis. This work has been supported by a fellowship from the German Academical Exchange Program (DAAD) and by the Bavarian Consortium for High Performance Scientific Computing (FORTWIHR), which are gratefully acknowledged.

REFERENCES

1. Karamyshev V, Kovenya V, Sleptsov A. Adaptive methods for Navier–Stokes equations. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, Paris, Désidéri JA, Hirsch C, Le Tallec P, Pandolfi M, Périaux J (eds). Wiley: Chichester, 1996; 301–307.
2. Ofengeim D, Timofeev E, Voinovich P, Galyukov A, Drikakis D, Satofuka N. A locally adaptive structured/unstructured 2-D/3-D Navier–Stokes finite-volume solvers for steady and unsteady compressible flows. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, Paris, Désidéri JA, Hirsch C, Le Tallec P, Pandolfi M, Périaux J (eds). Wiley: Chichester, 1996; 187–192.
3. Zhmakin AI. A memory-efficient unstructured grid refinement algorithm for computation of 3D steady viscous flows. *Comunications in Numerical Methods in Engineering* 1997; **13**:219–228.
4. Carcaillet R, Dulikravich GS, Kennon SR. Generation of solution-adaptive computational grids using optimization. *Computational Methods in Applied Mechanical Engineering* 1986; **57**:279–295.
5. Nakahashi K, Deiwert GS. Three-dimensional adaptive grid method. *AIAA Journal* 1986; **24**:948–954.
6. Jacquotte OP. Generalization, optimization and adaptation of multiblock grids around complex configurations in computational fluid dynamics. *International Journal for Numerical Methods in Engineering* 1992; **34**: 443–454.
7. Eiseman PR. Control point grid generation. *Computational Mathematical Applications* 1992; **24**:57–67.

8. Lin SY, Wu TM. An adaptive multigrid finite-volume scheme for incompressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids* 1993; **17**:687–710.
9. Yang JC, Soni BK. Structured adaptive grid generation. *Applied in Mathematical Computation* 1994; **65**: 265–278.
10. Le Tallec P, Martin C. A non-linear elasticity model for structured mesh adaptation. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, Paris, Désidéri JA, Hirsch C, Le Tallec P, Pandolfi M, Périaux J (eds). Wiley: Chichester, 1996; 275–281.
11. Zegeling PA. *r*-refinement for evolutionary PDEs with finite elements or finite differences. *Applied Numerical Mathematics* 1998; **26**:97–104.
12. Quirk JJ. An adaptive mesh refinement algorithm for computational shock hydrodynamics. PhD thesis, Cranfield Institute of Technology, Cranfield, U.K., 1991.
13. Quirk JJ. A cartesian grid approach with hierarchical refinement for compressible flows. In *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Wagner S, Périaux J, Hirschel EH (eds). Wiley: Chichester, 1994; 200–209.
14. Quirk JJ. On the dynamics of a shock-bubble interaction. Report TR-94-75, ICASE, Hampton, VA, 1994.
15. Berger MJ, Collela P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**:64–84.
16. Fischer J. Selbstadaptive, lokale Netzverfeinerung für die numerische Simulation kompressibler, reibungsbehafteter Strömungen. PhD thesis, Universität Stuttgart, Stuttgart, 1993.
17. Hentschel R, Hirschel EH. Self adaptive flow computations on structured grids. In *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Wagner S, Périaux J, Hirschel EH (eds). Wiley: Chichester, 1994; 242–249.
18. Hentschel R, Hirschel EH. AMRFLEX3D—flow simulation using a three-dimensional self-adaptive, structured multi-block grid system. In *Flow Simulation with High-Performance Computers II*, Notes on Numerical Fluid Mechanics, Hirschel EH (ed.). Vieweg: Braunschweig, 1996; 400–415.
19. Jouhaud JC, Borrel M. A hierarchical adaptive mesh refinement method: Application to 2D flows. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, Paris, Désidéri JA, Hirsch C, Le Tallec P, Pandolfi M, Périaux J (eds). Wiley: Chichester, 1996; 268–274.
20. Uphoff U, Thill CH, Hänel D. Structured mesh-refinement techniques for reactive and multi-phase flow. In *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, Paris, Désidéri JA, Hirsch C, Le Tallec P, Pandolfi M, Périaux J (eds). Wiley: Chichester, 1996; 287–293.
21. Friedel H, Grauer R, Marliani C. Adaptive mesh refinement for singular current sheets in incompressible magnetohydrodynamic flows. *Journal of Computational Physics* 1997; **134**:190–198.
22. Mitchell WF. The full domain partition approach to distributing adaptive grids. *Applied Numerical Mathematics* 1998; **26**:265–275.
23. Altas I, Stephenson JW. A two-dimensional adaptive mesh generation method. *Journal of Computational Physics* 1991; **94**:201–224.
24. Coelho P, Pereira JCF, Carvalho MG. Calculation of laminar recirculating flows using a local non-staggered grid refinement system. *International Journal for Numerical Methods in Fluids* 1991; **12**:535–557.
25. Coelho PJ, Pereira JCF. Calculation of confined axisymmetric laminar diffusion flame using a local grid refinement technique. *Combustion Science and Technology* 1993; **92**:243–264.
26. Lazarov RD, Mishev ID, Vassilevski PS. Finite volume methods with local refinement for convection–diffusion problems. CAM Report 93-20, *Computational and Applied Mathematics*, UCLA, 1993.
27. Davis RL, Dannehofer, III JF. Decomposition and parallelization strategies for adaptive grid-embeding techniques. *Computational Fluid Dynamics* 1993; 79–93.
28. Blom JG, Verwer JG. VLUGR3: A vectorizable adaptive grid solver for PDEs in 3D. I. Algorithmic aspects and applications. Report NM-R9404, Centrum voor Wiskunde en Informatica: Amsterdam, 1994.
29. Ferziger JH, Perić M. *Computational Methods for Fluid Dynamics*. Springer: Berlin, 1996.
30. Nirschl H, Dwyer HA, Denk V. Three-dimensional calculations of the simple shear flow around a single particle between two moving walls. *Journal of Fluid Mechanics* 1995; **283**:273–285.
31. Wehr D, Stangl R, Wagner S. Interpolation schemes for intergrid boundary value transfer applied to unsteady transonic flow computations on overlaid embedded grids. In *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Wagner S, Périaux J, Hirschel EH (eds). Wiley: Chichester, 1994; 383–389.
32. Thompson JF. Grid generation techniques in computational fluid dynamics. *AIAA Journal* 1984; **22**:1505–1523.
33. Rai MM. A conservative treatment of zonal boundaries for Euler equation calculations. *Journal of Computational Physics* 1986; **62**:472–503.
34. Seidl V, Perić M, Schmidt S. Space- and time parallel Navier–Stokes solver for 3D block-adaptive cartesian grids. In *Parallel Computational Fluid Dynamics—Implementation and Results Using Parallel Computers*, Ecer A, Periaux J, Satofuka N, Taylor S (eds). Elsevier: Amsterdam, 1996.
35. Shih TIP, Bailey RT, Nguyen HL, Roelke RJ. Algebraic grid generation for complex geometries. *International Journal for Numerical Methods in Fluids* 1991; **13**:1–31.

36. Børresen Jr. B. A multi-block methodology for solving flow problems in complex geometries. In *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Wagner S, Périaux J, Hirschel EH (eds). Wiley: Chichester, 1994; 377–382.
37. Rai MM. An implicit, conservative, zonal-boundary scheme for Euler equation calculations. *Computational Fluids* 1986; **14**:295–319.
38. Rai MM. A relaxation approach to patched-grid calculations with the Euler equations. *Journal of Computational Physics* 1986; **66**:99–131.
39. Rai MM. Navier–Stokes simulations of rotor/stator interaction using patched and overlaid grids. *Journal of Propulsion* 1987; **3**:387–396.
40. Kassies A, Tognaccini R. Boundary conditions for Euler equations at internal block faces of multi-block domains using local grid refinement. Paper 90-1590, AIAA, 1990. *AIAA 21st Fluid Dynamics*, *Plasma Dynamics and Lasers Conf*, Seattle, WA, 1990.
41. Rangwalla AA, Madavan NK. Application of an unsteady Navier–Stokes solver to transonic turbine design. *Journal of Propulsion Power* 1992; **8**:1079–1086.
42. Amato M, Paparone L. On the application of the multizone modelling and the local grid refinement technique for high-lift airfoil analysis. In *Proceedings of the Second European Computational Fluid Dynamics Conference*, Stuttgart, Wagner S, Périaux J, Hirschel EH (eds). Wiley: Chichester, 1994; 370–376.
43. Launder BE, Spalding DB. The numerical computation of turbulent flow. *Computational Methods in Applied Mechanical Engineering* 1974; **3**:269–289.
44. Patankar SV, Spalding DB. A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows. *International Journal of Heat and Mass Transfer* 1972; **15**:1787–1806.
45. Perić M, Kessler R, Scheuerer G. Comparison of finite-volume numerical methods with staggered and colocated grids. *Computational Fluids* 1988; **16**:389–403.
46. Demirdžić I, Perić M. Finite volume method for prediction of fluid flow in arbitrary shaped domains with moving boundaries. *International Journal for Numerical Methods in Fluids* 1990; **10**:771–790.
47. Stone HL. Iterative solution of implicit approximations of multi-dimensional partial differential equations. *SIAM Journal of Numerical Analysis* 1968; **5**:530–558.
48. Hortmann M, Perić M, Scheuerer G. Finite volume multigrid prediction of laminar natural convection: Benchmark solutions. *International Journal for Numerical Methods in Fluids* 1990; **11**:189–207.
49. Durst F, Schäfer M. A parallel blockstructured multigrid method for the prediction of incompressible flows. *International Journal for Numerical Methods in Fluids* 1996; **22**:549–565.
50. Lange CF. Numerical predictions of heat and momentum transfer from a cylinder in crossflow with implications to hot-wire anemometry. PhD thesis, Lehrstuhl für Strömungsmechanik, Friedrich–Alexander-Universität Erlangen-Nürnberg, Erlangen, 1997.
51. Fuchs L. Numerical flow simulation using zonal-grids. Paper 85-1518, AIAA, 1985.
52. Schäfer M, Turek S. Benchmark computations of laminar flow around a cylinder. In *Flow Simulation with High-Performance Computers II*, Notes on Numerical Fluid Mechanics, Hirschel EH (ed.). Vieweg: Braunschweig, 1996; 547–566.
53. Angelis W. Komplementäre Methodik in der Aerodynamik bodennaher Fahrzeuge. PhD thesis, University of Erlangen-Nürnberg, 1996.
54. Lange CF, Durst F, Breuer M. Wall effects on heat losses from hot-wires. *International Journal of Heat Fluid Flow* 1999; **20**:34–47.