INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **31**: 1157–1172 (1999)

A LOCAL GRID REFINEMENT METHOD FOR THREE-DIMENSIONAL TURBULENT RECIRCULATING FLOWS

G. PAPADAKIS* AND G. BERGELES[1]

Department of Mechanical Engineering, Laboratory of Aerodynamics, National Technical University of Athens, PO Box 64070, 15710 Zografou, Athens, Greece

SUMMARY

A local grid refinement method is presented and applied to a three-dimensional turbulent recirculating flow. It is based on the staggered grid arrangement. The computational domain is covered by block-structured subgrids of different refinement levels. The exchange of information between the subgrids is fully conservative and all grids are treated implicitly. This allows for a simultaneous solution of one variable in all grids. All variables are stored in one-dimensional arrays. The solver selected for the solution of the discretised finite difference equations is the preconditioned bi-conjugate gradient (Bi-CG) method. For the case examined (turbulent flow around a surface-mounted cube), it was found that the latter method converges faster than the line solver. The locally refined mesh improved the accuracy of the pressure distribution on cube faces compared with a coarse mesh and yielded the same results as a fine single mesh, with a 62% gain in computer time. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: local refinement; staggered mesh; finite volumes; conjugate gradient method; incompressible recirculating flows

1. INTRODUCTION

In the local grid refinement method, the computational domain is first covered by a relatively coarse numerical mesh. In the most critical regions of the domain, the grid is locally refined, i.e. the cells are subdivided in one or more directions, so as to capture important flow characteristics without the necessity of extending the grid lines away from that region. The clustering of grid lines away from the critical regions results in cells with high aspect ratios, which affects numerical stability. The selection of these regions is made *a priori* and not during the solution process as in adaptive grids. It is possible to perform successive refinements in an already refined region; in this way different refinement levels are formed.

Coelho *et al.* [1] presented a local refinement method which they applied in two-dimensional laminar flow cases. They used the collocated grid arrangement. The embedded cells are generated by halving the mesh spacing on a cell-by-cell basis in both the x- and y-directions. Each flow variable is solved simultaneously for the whole solution domain, thus providing a

* Correspondence to: Department of Mechanical Engineering, King's College London, Strand, London WC2R 2LS, UK. E-mail: george.papadakis@kcl.ac.uk
[1] E-mail: bergeles@fluid.mech.ntua.gr

CCC 0271–2091/99/231157–16$17.50
Copyright © 1999 John Wiley & Sons, Ltd.

Received July 1998
Revised September 1998

strong coupling between grids of different refinement level. The method was applied to the solution of two scalar transport equations, to cavity flows driven by body and shear forces and to sudden plane contraction flow. Good agreement was found between the predictions and the analytical solutions or the experimental measurements. The results show that neither the convergence rate nor the stability of the method is affected by the embedded grids.

Lockwood *et al.* [2] proposed a local refinement method using a refinement ratio of 1:3. Again, the discretised equations were solved simultaneously for the whole solution domain inclusive of the regions where local refinement is applied. The method was applied to three-dimensional isothermal turbulent flows, where indeed it captured flow characteristics that single grids failed to do. Details about the savings in computer time and memory were not given.

Schneider *et al.* [3] applied a grid refinement method where subgrids were overlapped with one grid line. Contrary to the two previous works, the grids were considered independent of each other and in the boundaries Dirichlet conditions were applied. Thus, at each iteration, data were transferred from the coarse grid to the refined grid and *vice versa* and were used as boundary conditions until convergence. The method was applied in isothermal and reacting flow cases inside a coal fired boiler. The results showed that the stability of the code was hardly affected by the local refinement method. The savings in computer time per iteration between the coherent fine mesh and the multidomain version (where the near burner region was discretised with a separate grid using the same grid spacing as the fine grid) was 25.8%.

Smith and Gosman [4] combined the local grid refinement method with the multigrid technique. The use of a collocated variable arrangement together with a common cell face between the coarse and fine grid structure allowed the simple implementation of flux conservation and led to a particularly simple prolongation operator. The solution procedure was assessed through application to a plane laminar jet at 45° to the grid and to the well-known driven cavity at Reynolds numbers of 100 and 1000. It was demonstrated that the local refinement can, very nearly, reproduce the accuracy of full fine grid solutions and, in accordance with the multigrid theory, solution times varying linearly with the grid size were achieved in most cases. Recently, Emvin and Davidson [5] implemented the local refinement strategy in a multigrid environment and applied it successfully in two test cases (backward-facing step and a three-dimensional ventilated enclosure).

The above-mentioned works have emphasised the advantages of local refinement using an arbitrary number of structured blocks: complex geometries with a large variation in length scale can be handled easily, parallelisation is straightforward while the simple local indexing system eases vectorisation. On the other hand, adaptation is difficult, while the generation of structured grids may not be easy for very complex geometries.

Fully unstructured meshes can cope with very complex geometries, the aspect ratio is easily controlled while the grid may be easily locally refined. These advantages are, however, offset by the irregularity of the data structure, the higher memory requirements (since neighbouring connections need to be specified explicitly), the matrix of the algebraic equation system no longer has a regular diagonal structure, while the bandwidth needs to be reduced by reordering of the points (see Ferziger and Peric [6]). Demirdzic and Muzaferija [7] consider the local refinement as a special case of unstructured mesh with cells of arbitrary topology. In their methodology, the control volume can be of an arbitrary polyhedral shape, i.e. it can have an arbitrary number of cell faces. Their method shares all the advantages and disadvantages of unstructured meshes. The present method is simpler as it is based on the block-structured approach, albeit less general.

Ferziger and Peric [6] and Peric [8] discuss methods that are more close to the one presented in this paper, i.e. block-structured. These authors stress that conjugate gradient-type solvers are a good choice since they are capable of handling the irregular matrix structure that the local refinement incurs. In the present work such a solver is used, while its effect on the improvement of the convergence behaviour (as compared with a line solver) is quantified.

In conclusion, both block-structured meshes and unstructured meshes have their strengths and weaknesses. In view of ease of programming and code transparency, it was decided to further pursue the block-structured local refinement methodology.

In this paper, a local refinement method is presented for staggered grids. The characteristics of the method that will be analysed in the following sections are

(a) The computational domain in covered with structured subgrids. The subgrids are in contact in a common interface and do not overlap. The transfer of information is fully conservative, i.e. account is taken to ensure reciprocity of the fluxes on either side of an interface. In this way, conservativeness is easily checked and applied.
(b) A flow variable is solved in all grids simultaneously and not in each grid separately using Dirichlet boundary conditions. For the solution of the finite difference algebraic system, the preconditioned bi-conjugate gradient (Bi-CG) method is used.
(c) The method can be applied with any refinement ratio. In the next sections the method is described for ratio of 1:2 in all directions.
(d) All variables are stored in one-dimensional arrays. The subgrids are stored consecutively, i.e. one after the other. This method of storage is preferred to the alternative way of storing the elements in a four-dimensional array, i.e. $\Phi(NI, NJ, NK, n_{\text{grids}})$. Since each subgrid is structured, there is no need to use unidimensional arrays for the storage of the neighbours of each grid node. Only the neighbours of nodes that are adjacent to grid interfaces need to be stored.

In what follows, the transfer of information across the subgrid interfaces will be examined in detail. The method is then applied for the calculation of the turbulent flow around a cube mounted on the surface of a wind tunnel. The conjugate gradient (CG) method and the line-by-line solver are compared and the effect of the local refinement on the accuracy of the results and the execution times is discussed.

2. METHOD FORMULATION

2.1. Transfer of information from the coarse to the fine grid

Figures 1(b)–4(b) show the nodes of the coarse grid that are involved in the transfer of information from the west side to node P of the fine grid for the scalar variables U, V and W velocity respectively. Due to halving the distance in all directions, there are nodes of the finer grid close to the interface that do not have actual upstream nodes on the coarser grid. Thus, it is necessary to introduce fictitious nodes; their location is denoted by $\times$ in Figures 1(b)–4(b). The discretised transport equation of the general variable Φ is

$$A_P\Phi_P = \sum_{i=E,N,S,D,U} A_i\Phi_i + A_W\Phi_W + S_\Phi,$$

where the node W is fictitious. The value of each variable stored in the fictitious node is calculated from the linear interpolation between the neighbouring nodes

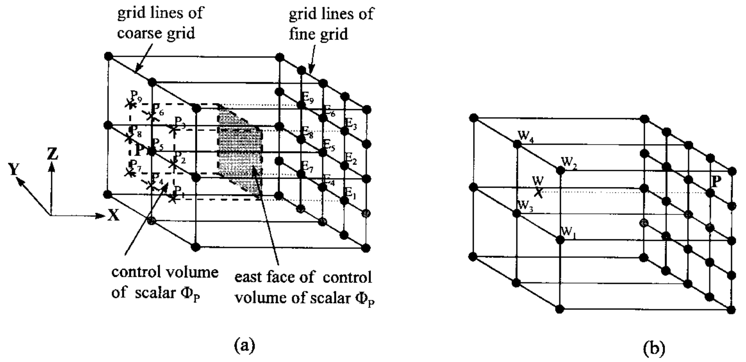

grid lines of coarse grid
grid lines of fine grid
control volume of scalar Φ_P
east face of control volume of scalar Φ_P
(a)
(b)

Figure 1. (a) East neighbours of scalar Φ_P of the coarse grid, (b) west neighbour of scalar Φ_P of the fine grid. Full circles indicate interacting nodes.

$$\Phi_W = \sum_{i=1}^{4} \mathrm{wf}_i \Phi_{W_i},$$

where wf_i are the weighting factors for the linear interpolation and W_i are existing nodes of the coarse mesh; these are shown in Figures 1(b)–4(b) for all variables. Thus, the discretised equation becomes

$$A_P\Phi_P = \sum_{i=E,N,S,D,U} A_i\Phi_i + A_W \sum_{i=1}^{4} \mathrm{wf}_i \Phi_{W_i} + S_\Phi.$$

Thus, the treatment of the common interface is reduced to the treatment of the second term of the right-hand-side of the above equation. The incorporation of this term with the source term will significantly reduce the coupling between the grids (explicit coupling). In this case, a reduction of the convergence rate is expected. Thus, the values Φ_{W_i} are not incorporated into the source term and are considered instead as unknown in every iteration. In this way, the fully implicit treatment between grids of different refinement levels is retained. The interfaces that are located on any other side of node P are treated in a similar manner.

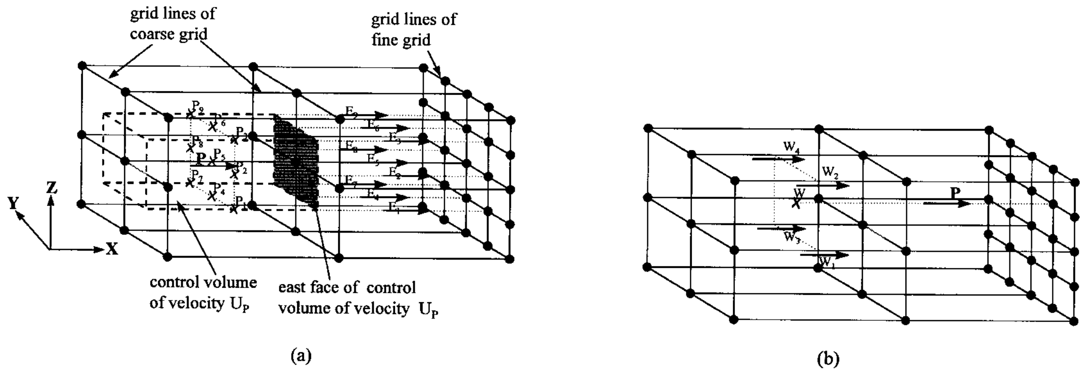

grid lines of coarse grid
grid lines of fine grid
control volume of velocity U_P
east face of control volume of velocity U_P
(a)
(b)

Figure 2. (a) East neighbours of velocity U_P of the coarse grid, (b) west neighbour of velocity U_P of the fine grid.

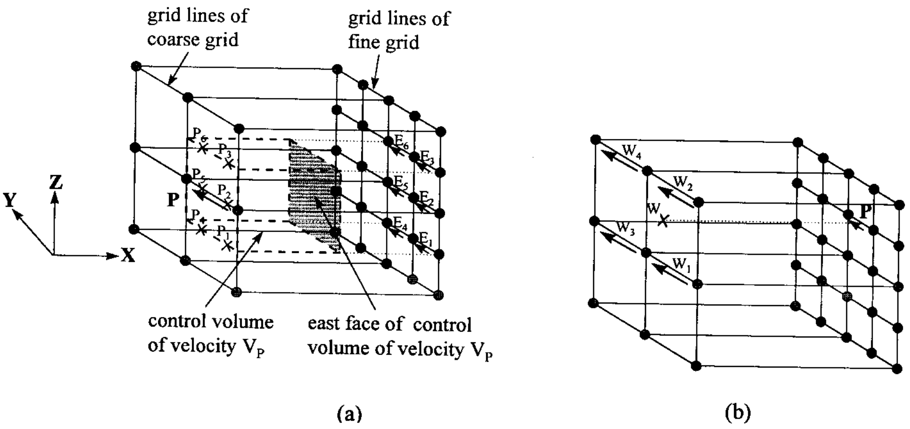

grid lines of coarse grid
grid lines of fine grid
control volume of velocity V_P
east face of control volume of velocity V_P
P
P_1
P_2
P_3
P_4
P_5
P_6
E_1
E_2
E_3
E_4
E_5
E_6
W
W_1
W_2
W_3
W_4
X
Y
Z
(a)
(b)

Figure 3. (a) East neighbours of velocity V_P of the coarse grid, (b) west neighbour of velocity V_P of the fine grid.

2.2. Transfer of information from the fine to the coarse grid

The integrated transport equation is written as

$$J_e - J_w + J_n - J_s + J_d - J_u = S_\Phi,$$

where, for example,

$$J_e = \left[\rho u \Phi - \Gamma_\Phi \frac{\partial \Phi}{\partial x}\right]_e A_e,$$

where A_e is the east area of the control volume.

The variable J_i ($i =$ e, w, n, s, d, u) is the flux of Φ with convection and diffusion through the interface i. For the case when the i face of a control volume contacts more than one face of adjacent volumes, the total flux is calculated from the sum of the individual fluxes. Figures 1(a)–4(a) show the east neighbours (denoted as E_i) for node P of the coarse mesh for the scalar variables and the velocities. The number of neighbours is either nine (for scalars and velocity U) or six (for velocities V and W). For the hybrid differencing scheme used here, the

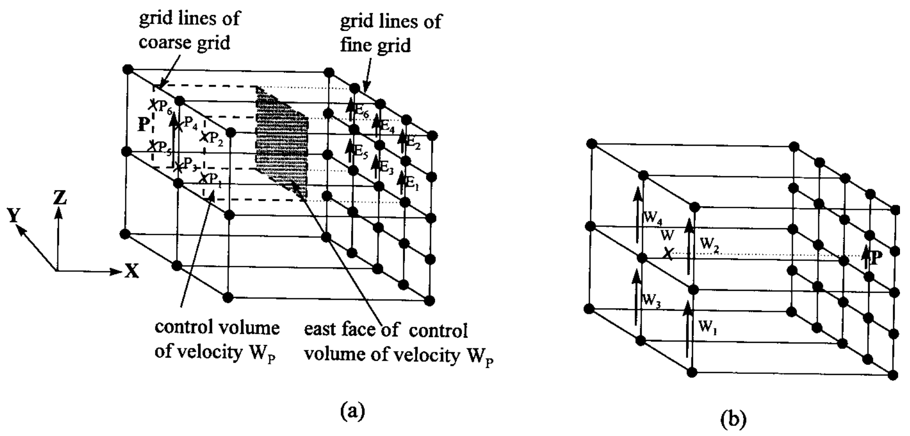

grid lines of coarse grid
grid lines of fine grid
control volume of velocity W_P
east face of control volume of velocity W_P
P
P_1
P_2
P_3
P_4
P_5
P_6
E_1
E_2
E_3
E_4
E_5
E_6
W
W_1
W_2
W_3
W_4
X
Y
Z
(a)
(b)

Figure 4. (a) East neighbours of velocity W_P of coarse grid, (b) west neighbour of velocity W_P of the fine grid.

total flux from the east side of the control volume around node P is calculated as (see Patankar [9] for the case when there is only one neighbour)

$$J_e = \sum_{i=1}^{9} [A_{E_i}(\Phi_{P_i} - \Phi_{E_i}) + C_{e_i}\Phi_{P_i}].$$

In the above relation, C_{e_i} is the contribution of the node E_i ($i = 1$, 9 or 6) to the convective flux through the east face, i.e. $C_{e_i} = [\rho u A]_{w_i}$, where A_{w_i} is the area of the volume surrounding E_i that is in contact with the east face of the volume surrounding P. The nodes P_i are also shown in Figures 1(a)–4(a) and are indeed the fictitious nodes discussed earlier. Relations that involve only one neighbour can be written for the fluxes through the other faces. Substituting all the relations into the integrated transport equation and rearranging the terms we get

$$A_P\Phi_P = \sum_{i=W,N,S,U,D} A_i\Phi_i - \sum_{i=1}^{9} [A_{E_i}(\Phi_{P_i} - \Phi_{E_i}) + C_{e_i}(\Phi_{P_i} - \Phi_P)] + S_\Phi,$$

where

$$A_P = \sum_{i=W,N,S,U,D} A_i,$$

and the east nodes have been excluded from the summation.

This equation governs the transfer of information from the fine mesh to the coarse mesh. Again, the values Φ_{E_j} are treated implicitly, while Φ_{P_i} are calculated through linear interpolations from surrounding nodes.

The pressure correction equation is handled in exactly the same way.

3. SOLUTION OF THE LINEAR SYSTEM

Since many terms are not incorporated in the source term, the linear system is no longer tridiagonal in each direction for those cells that are in contact with the interface between subgrids of different refinement levels. Therefore, the well-known TDMA method is replaced by the preconditioned Bi-CG. This method is very flexible and capable of handling matrices with arbitrary sparsity patterns. In order to accelerate the convergence rate, the incomplete lower/upper (ILU) decomposition with no fill-in was used for preconditioning. This solver is used for all variables, including the pressure correction equation. In the latter case, the fact that the matrix is symmetric is taken into account and thus the number of operations is significantly reduced (compared with unsymmetric matrices). Details for the use of the method are given in the next section.

4. APPLICATION OF THE METHOD. RESULTS AND DISCUSSION

The method was applied for the flow around a cube of 200 mm height mounted on the surface of a wind tunnel. Experimental measurements for velocities and pressure distribution on the cube faces were made by Castro and Robins [10]. The velocity of the incoming flow has a boundary layer type profile. The infinite velocity is 0.5 m s^{-1} and the Reynolds number (based on the infinite velocity and the cube height) is 4000. The boundary layer height is 10 cube heights.

Table I. Comparison of the Bi-CG and the line solver

Grid	$29 \times 25 \times 20$		$39 \times 35 \times 28$	
	Iterations	CPU time (s)	Iterations	CPU time (s)
Line solver	686	5278	1145	25 535
CG	106	1933	231	12 911
Change (%)	−84.5	−63.3	−79.8	−49.4

The local refinement method was applied in three successive steps: in the first step, a coarse mesh $29 \times 25 \times 20$ was constructed. The second step involved the local refinement of this mesh in a region close to the cube. The two meshes were the $29 \times 25 \times 20$ (coarse mesh) and $23 \times 23 \times 19$ (local refinement). The third step involved the construction of a single fine mesh, which offers the same grid line distribution close to the cube as the locally refined mesh. This mesh is constructed by extending the grid lines of the local refinement (i.e. grid $23 \times 23 \times 19$) until they meet the boundaries of the computational domain. The resulting grid is $39 \times 35 \times 28$. These meshes in the x–z-plane are shown in Figure 5.

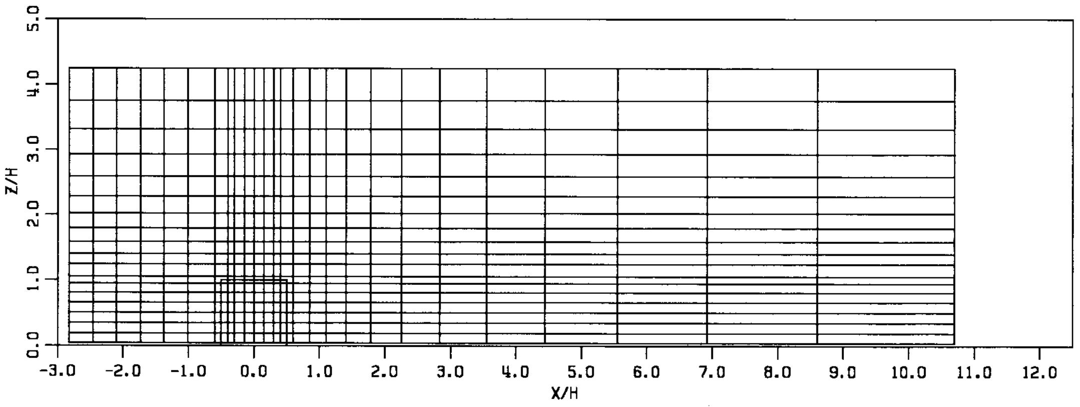
In order to demonstrate the superiority of the method, it must be proved that the locally refined mesh is capable of providing more accurate results than the coarse mesh in less computer time than the time needed for the single fine mesh. Also, the results of the locally refined mesh and the single fine mesh should be identical (or at least very close to each other).

4.1. Comparison of the CG method with the line-by-line solver

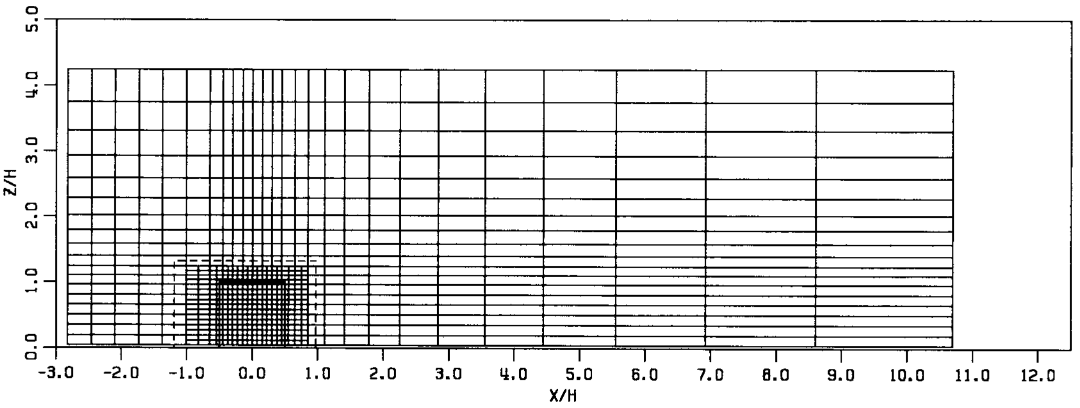
In this section, the selection of the CG method as a system solver will be examined in the single meshes ($29 \times 25 \times 20$ and $39 \times 35 \times 28$). A comparison will be made with the line-by-line solver (tridiagonal matrix algorithm).

The inner iterations of the latter solver is five for the pressure correction equation and three for the rest of the variables. For the transport equations, the convergence of the CG method for each outer iteration is monotonic and very fast: typically two or three iterations are needed in order to reduce the residual by three orders of magnitude. However, many more iterations are needed for the pressure correction equation; typically 20–70 inner iterations are needed in order to reduce the residual by one order of magnitude. The reason for this dramatic increase has to do with the type of the equation: this is a diffusion-type equation, which implies that all six (or more in case of local refinement) neighbours affect the value of the each node. On the other hand, for the transport equations, due to upwind differencing, only upwind nodes affect its value.

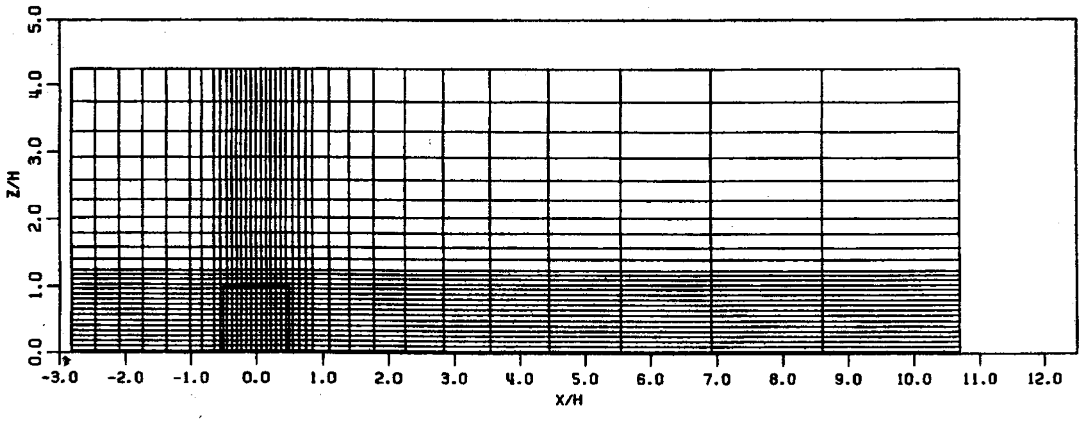
Table I shows the results of the computer time and the number of iterations for convergence for both solvers for both grids. The solution is assumed fully converged when all normalised residuals are below 10^{-4}. All runs were made in a HP 710 Apollo computer with full optimisation during the code compilation. The underrelaxation factor for all variables was 0.5. No attempt was made to optimise these factors. The reduction in computer time using the CG method is obvious (1.9 and 2.7 times). Even more dramatic is the decrease in the number of iterations (5 and 6.4 times). On the other hand, an increase in computer time per iteration is observed (2.3 and 2.5 times). These values justify the use of the CG method not only for the single meshes but for the locally refined meshes as well, with the additional advantage of allowing fully coupled treatment between the subgrids. It is important to note that the figures quoted above are case-dependent and cannot be considered as general.



(a)

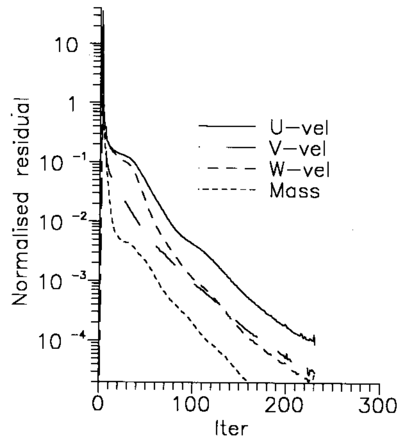


(b)

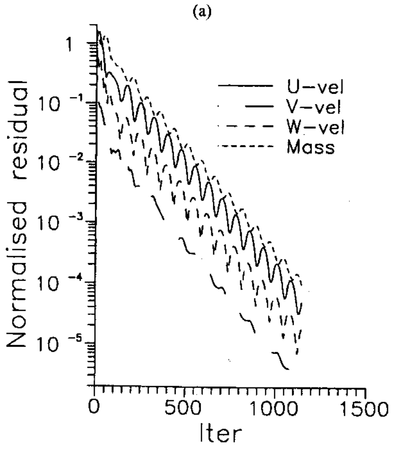


(c)

Figure 5. Grid lines in the x–z-plane (a) grid $29 \times 25 \times 20$, (b) grids $29 \times 25 \times 20$ and $23 \times 23 \times 19$, (c) grid $39 \times 35 \times 28$. Dashed line shows the interface between grids of different refinement level.



(a)



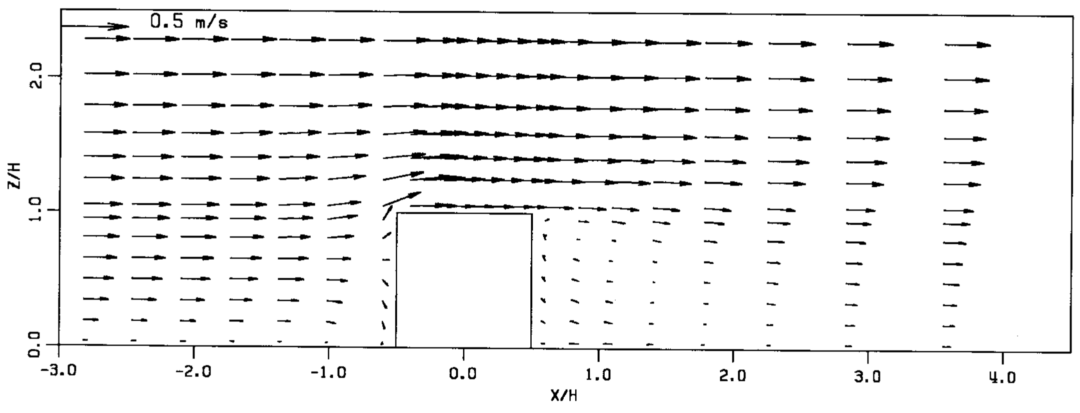
(b)

Figure 6. Convergence history for the single fine mesh: (a) with the CG method, (b) with the line-by-line solver.

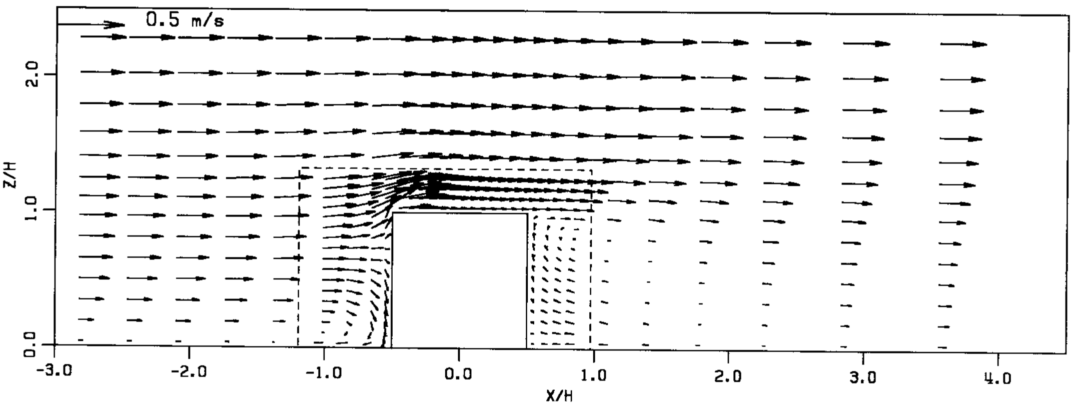
Figure 6 shows the convergence history using the two solvers for the fine single mesh ($39 \times 35 \times 28$). The CG method leads to monotone convergence while the line solver to oscillatory behaviour. This is the reason for the dramatic decrease in the number of iterations and the total execution time.

Table II. Number of iterations and execution times for the three cases (Bi-CG solver)

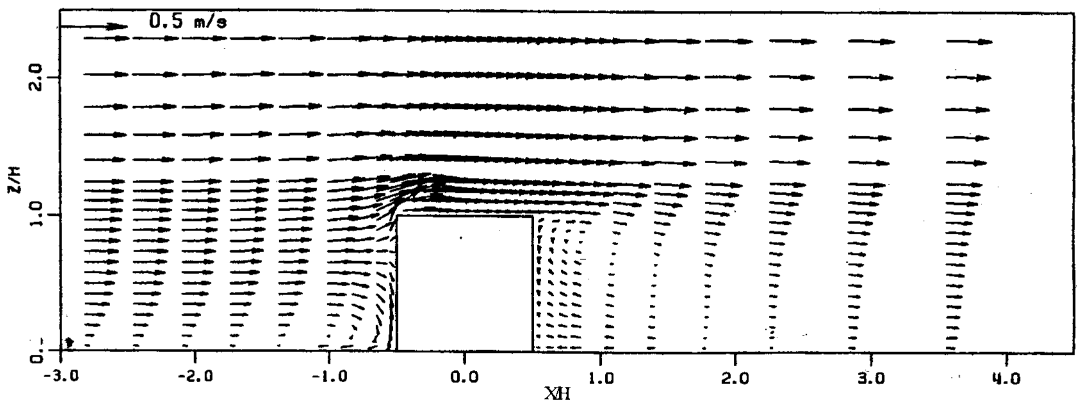
	Iterations	CPU time (s)
Grid $29 \times 25 \times 20$	106	1933
Grids $29 \times 25 \times 20$ and $23 \times 23 \times 19$	149	4864
Grid $39 \times 35 \times 28$	231	12 911
Savings between local refinement and single fine mesh (%)	−35	−62



(a)



(b)



(c)

Figure 7. Velocity vectors in the x–z-plane in the cube symmetry plane (detail); (a) grid $29 \times 25 \times 20$, (b) grids $29 \times 25 \times 20$ and $23 \times 23 \times 19$, (c) grid $39 \times 35 \times 28$.

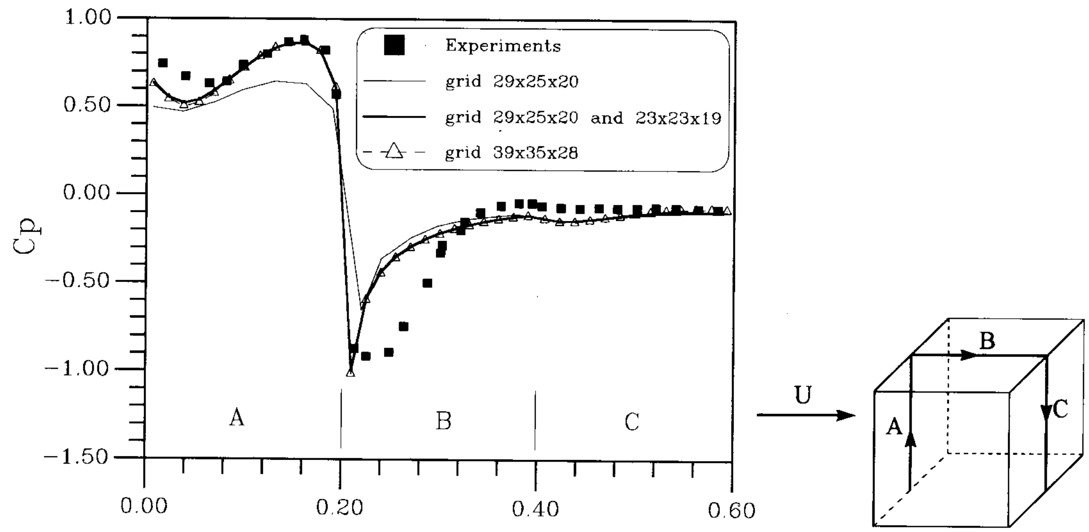


Figure 8. C_p distribution on the cube surface along the line ABC (x–z-plane of symmetry).

4.2. *Execution time required by the locally refined meshes*

Table II shows the number of iterations and the total execution time for the three meshes examined (with the CG solver). Also given is the percentage gain in the case of the locally refined mesh and the single fine mesh. It is to be noted that we have a decrease of 35% in the number of iterations and of 62% in the total execution time. Also, the memory needed is reduced due to the smaller number of grid points. Hence, there is a significant gain when a locally refined mesh is used. However, it is important to prove that the results obtained have increased accuracy compared with the coarse mesh (or have the same accuracy as the results of the single fine mesh) in order to prove the superiority of the method.

4.3. *The effect of the local refinement on the accuracy of the results*

The accuracy of the locally refined grid will be assessed against the available experimental data and the results that the single fine mesh gives. The velocity field in the middle of the x–z-plane for the three grids is shown in Figure 7. It is obvious that with the coarse mesh it is not possible to clearly pick up the recirculation region in front of the cube. This zone is captured only when a locally refined mesh or a single fine mesh are used. Also in the case of local refinement, flow enters and exits from the grid interface without any effect on the stability of the method. Although a small recirculation region in the leading-edge of cube appears in the experiments, it can not be captured by the present simulations. This phenomenon has also been reported by other investigators and it is attributed to deficiencies of the k–ε model. The problem is traced to the calculation of the generation of the turbulent kinetic energy and is dealt in detail by Murakami [11] and Murakami *et al.* [12]. The same behaviour is also noted in the vertical leading-edges of the cube.

Figure 8 shows the distribution of surface pressure coefficient C_p in the symmetry x–z-plane as computed with the three meshes. The calculation of C_p is based on the boundary layer velocity on the cube height. Obviously, the accuracy is greatly enhanced in the front face of the cube and the minimum value of C_p is correctly calculated in the leading-edge when the locally refined mesh is used. The same results are also computed with the single fine mesh. However, there are two regions where differences with the experiments are observed: the first region is in the front face close to the cube basis and the second is in the first half of the top face. The first region is associated with the recirculation region that is created in front of the cube. Detailed measurements of mean and root-mean-square (rms) velocities are not available in this region so as to trace with certainty the cause of the discrepancies. However, the same behaviour is also reported by Baetke *et al.* [13]. It is noted that Song and He [14] predict, with very good accuracy, this region using a large eddy simulation (LES) method for turbulence simulation. The discrepancy in the second region (very rapid pressure recovery immediately after the leading-edge) is related to the inability of the k–ε model to capture the small recirculation zone in that area. Other models, like the algebraic stress model or the LES model, are capable of capturing this region (Murakami [11]) and thus delaying the pressure recovery. In the downstream face, the C_p distribution is well predicted.

Figure 9 shows the C_p distribution in the x–y-plane at a half cube's height. Again the locally refined mesh leads to increased accuracy in the front and close to the lateral faces. Discrepancies with respect to the experimental measurements are observed close to the lateral leading-edges for the same reasons that have already been discussed. In the upstream and downstream faces, the C_p distribution is well predicted.

Figures 10 and 11 show the C_p distribution in the lateral faces along the vertical direction and the top face along the transverse direction. In both cases, the C_p distributions are satisfactorily predicted while there is slight improvement when the locally refined grid is used.

Figure 12 presents the comparison between measurements and predictions for the vertical distribution of the main velocity in three locations (on the cube and downstream). The non-dimensionalisation has been done with the reference velocity U_r, which is equal to the infinite velocity (0.5 m s^{-1}). The predictions are generally satisfactory, even inside the wake region behind the cube. The effect of grid distribution is generally small.

Finally, Figure 13 shows transverse distributions of the main velocity in two stations downstream the cube. There is some overestimation of the velocities (especially in the

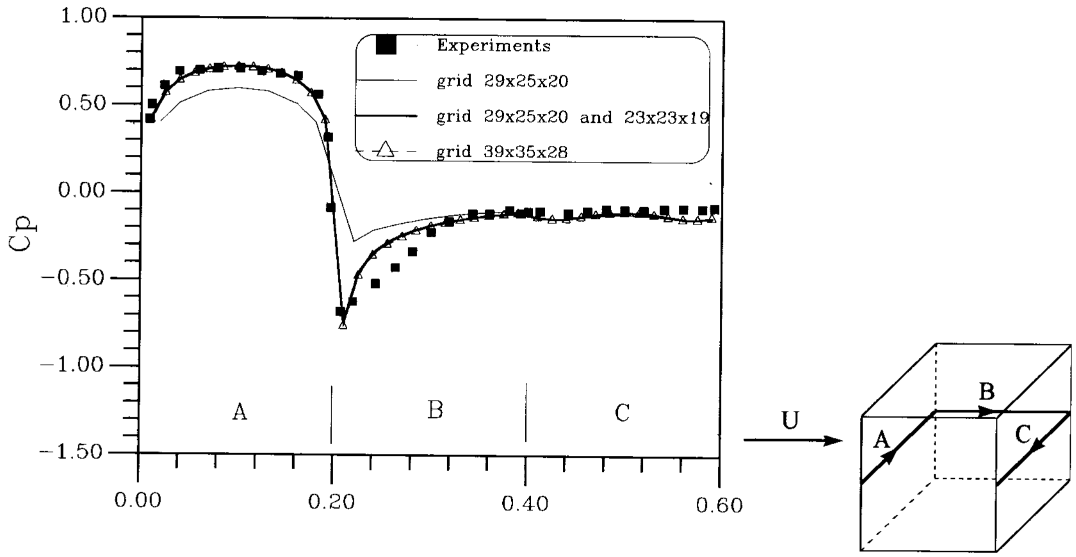


Figure 9. C_p distribution on the cube surface along the line ABC (x–y-plane at half a cube height).

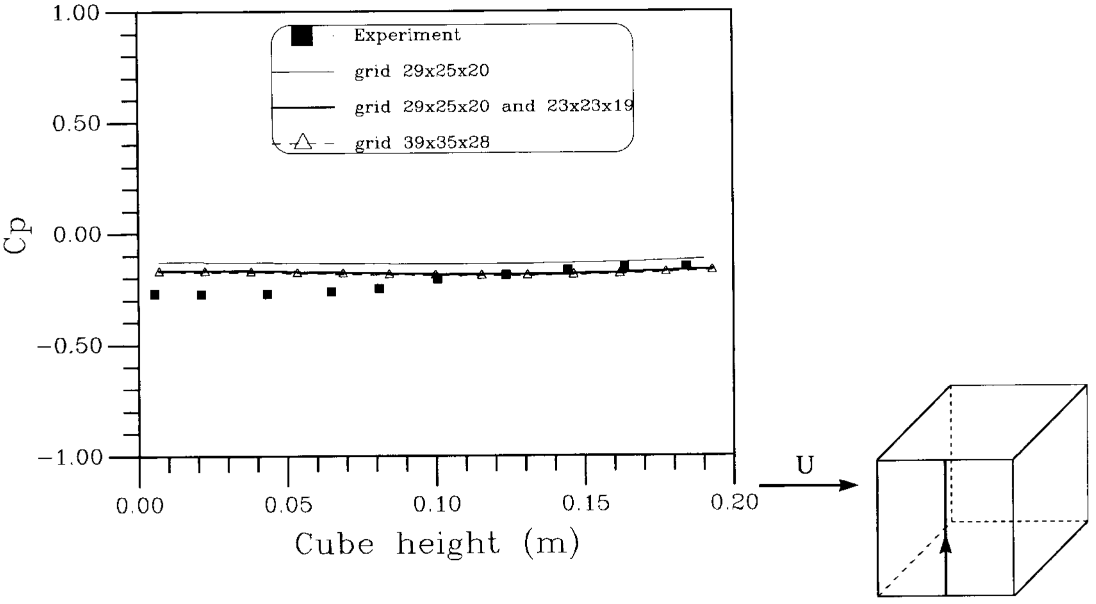


Figure 10. C_p distribution on a lateral cube surface (along the line shown).

second profile) which should be attributed to the isotropical character of the $k-\varepsilon$ model and its inability to predict the flow characteristics after the recirculation zones. Also, it may be attributed to the unsteady character of the wake in this region.

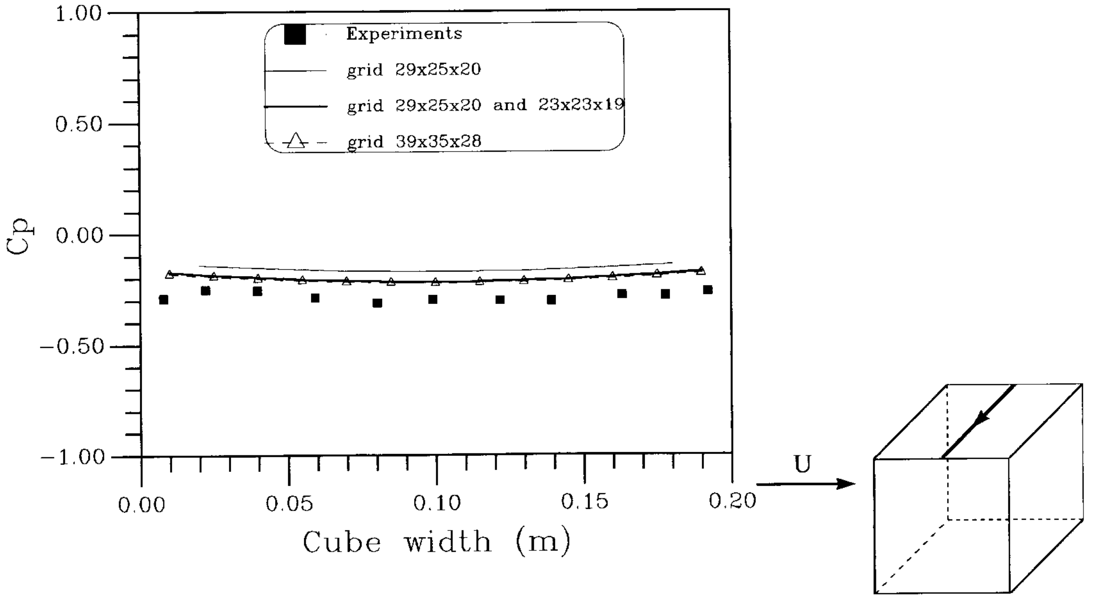


Figure 11. C_p distribution on the top cube surface (along the line shown).

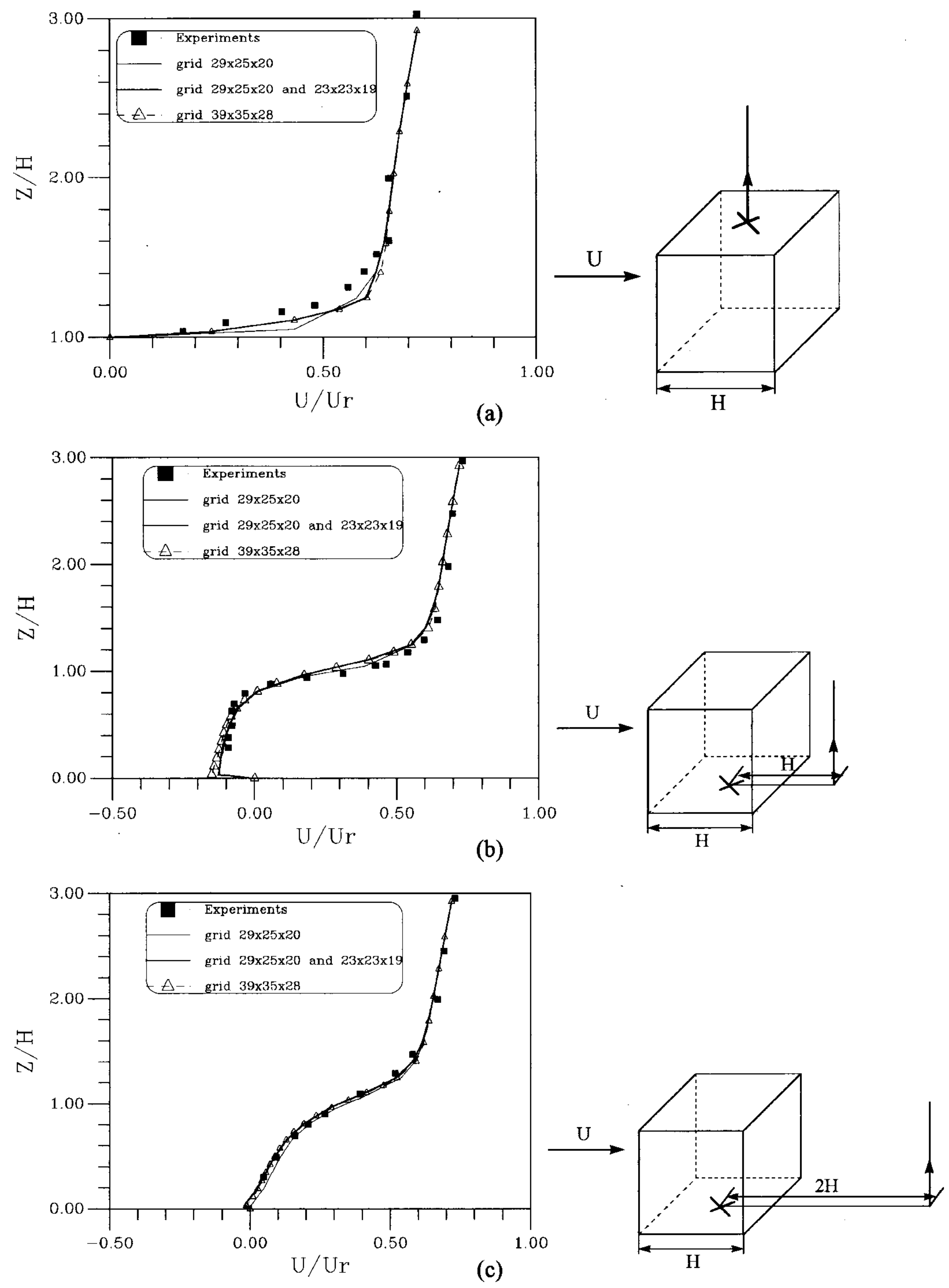
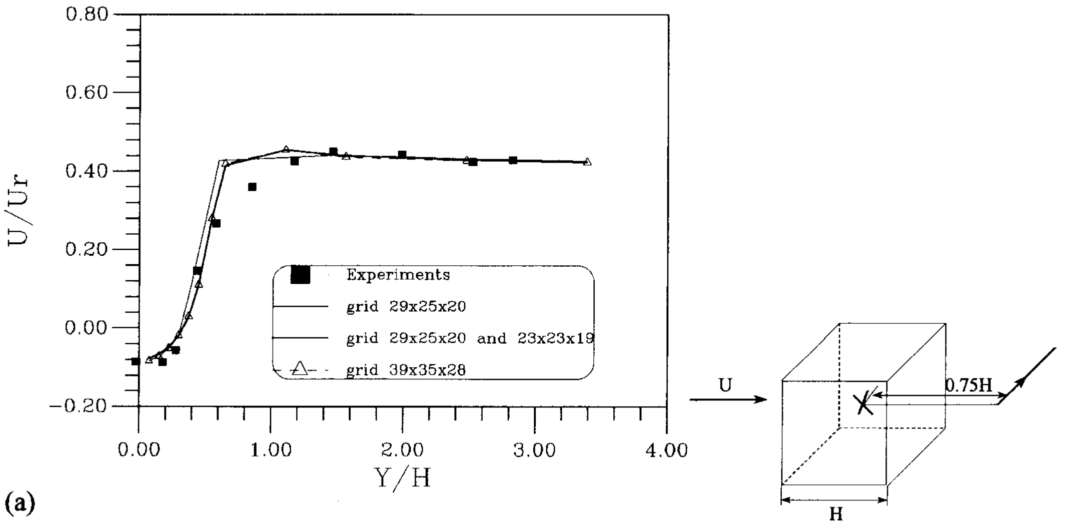


Figure 12. Main stream velocity distribution along the z-axis at (a) $x = 0H$, $y = 0H$, (b) $x = 1H$, $y = 0H$, (c) $x = 2H$, $y = 0H$ (H is cube height).



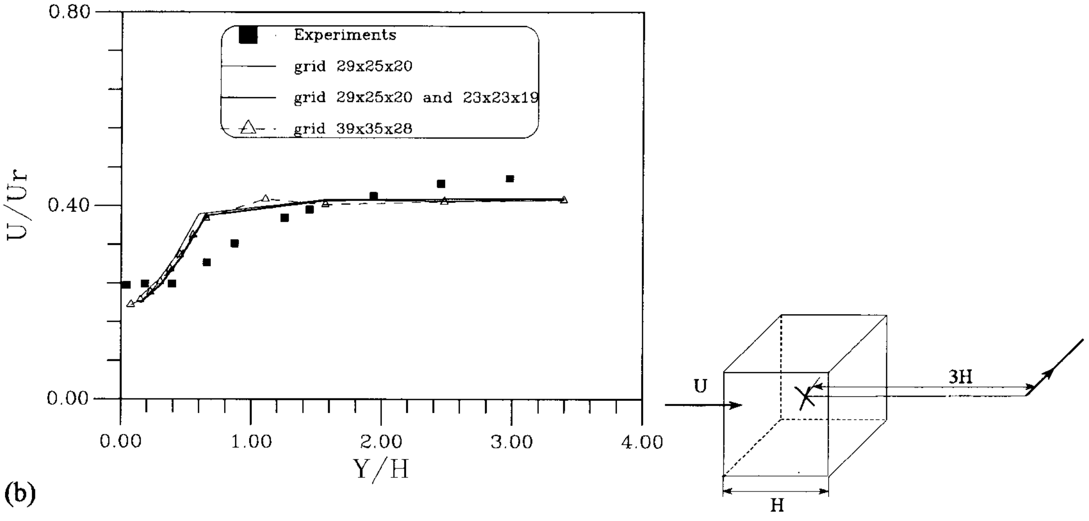


Figure 13. Main stream velocity distribution along the y-axis at (a) $x = 0.75H$, $z = 0.5H$, (b) $x = 3H$, $z = 0.5H$ (H is cube height).

5. CONCLUSIONS

It was found that for the case of the turbulent flow around a surface-mounted cube, the preconditioned Bi-CG method leads to shorter execution times than the line-by-line solver. The number of iterations for full convergence is significantly reduced, although the time per iteration is increased. The predictions with the locally refined mesh and the fine single grid are almost indistinguishable. The use of the locally refined grid in the cube vicinity leads to increased accuracy in the C_p distribution with a 62% gain in computer time. Velocity distributions are slightly affected.

REFERENCES

1. P. Coelho, J.C.F. Pereira and M.G. Carvalho, 'Calculation of laminar recirculating flows using a local non-staggered grid refinement system', *Int. J. Numer. Methods Fluids*, **12**, 535–557 (1991).
2. F.C. Lockwood, C. Pikoulas and N.G. Shah, 'A new multilevel local grid refinement technique for the solution of the 3D Navier–Stokes equations', *British Flame Days Meeting*, British Flame Research Committee, 1992.
3. R. Schneider, B. Risio, U. Schnell and K.R.G. Hein, 'Numerical simulation of reacting flows in coal-fired utility boilers using a domain decomposition method', *Proc. 2nd Europ. Computational Fluid Dynamics Conference*, Stuttgart, 1994.
4. D.M. Smith and A.D. Gosman, 'An application of multigrid with local grid refinement to fluid flow calculations', *Proc. 5th Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, 1991.
5. P. Emvin and L.A. Davidson, 'Local mesh refinement algorithm applied to turbulent flow', *Int. J. Numer. Methods Fluids*, **24**, 519–530 (1997).
6. J.H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*, Springer, Berlin, 1996.
7. I. Demirdzic and S. Muzaferija, 'Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology', *Comput. Methods Appl. Mech. Eng.*, **125**, 235–255 (1995).
8. M. Peric, 'Numerical methods for computing turbulent flows', *Lecture Series 03*, von Karman Institute for Fluid Dynamics, 1997.
9. S.V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere–McGraw-Hill, New York, 1980.
10. I.P. Castro and A.G. Robins, 'The flow around a surface-mounted cube in uniform and turbulent streams', *J. Fluid Mech.*, **79**, 307–335 (1977).
11. S. Murakami, 'Comparison of various turbulence models applied to a bluff body', *J. Wind Eng. Ind. Aerodyn.*, **46/47**, 21–36 (1993).
12. S. Murakami, A. Mochida and Y. Hayashi, 'Examining $k-\epsilon$ model by means of a wind tunnel test and large eddy simulation of the turbulence structure around a cube', *J. Wind Eng. Ind. Aerodyn.*, **35**, 87–100 (1990).
13. F. Baetke, H. Werner and H. Wengle, 'Numerical simulation of turbulent flow over surface-mounted obstacles with sharp edges and corners', *J. Wind Eng. Ind. Aerodyn.*, **35**, 129–147 (1990).
14. C.C.S. Song and J. He, 'Computation of wind flow around a tall building and the large-scale vortex structure', *J. Wind Eng. Ind. Aerodyn.*, **46/47**, 219–228 (1993).