RANS solvers with adaptive structured boundary non-conforming grids

By Sekhar Majumdar, Gianluca Iaccarino, AND Paul Durbin

1. Motivation and objectives

The objective of the present work is to develop robust RANS solvers for accurate computation of flow and heat transfer in complex geometries, such as arise in engineering design. In CFD analysis, the most time-consuming process is often the generation of an acceptable grid — whether it is a boundary-conforming, curvilinear mesh, or even a completely unstructured mesh. The present work focuses mainly on RANS solution of viscous and turbulent flows, where the relevant governing equations always impose strict requirements on the grid and on the solution algorithm. The special problems that arise in RANS solvers, coupled to turbulence models, have not been addressed in the literature on locally refined, Cartesian meshes, or in papers on the use of boundary non-conforming grids.

The ideal would be to develop a RANS code, with advanced turbulence models, that can perform flow and heat transfer analysis directly on the CAD representation of a surface, avoiding the cumbersome process of surface meshing. Such a development would have the potential for an enormous impact on applied computational analysis. The present research aims to exploring promising avenues in this direction.

1.1. *Issues with Cartesian meshes*

Flow solvers using Cartesian meshes have two major issues: (1) how to define a complex shaped surface accurately and (2) how to satisfy the relevant boundary conditions for the governing equations of pressure, velocity and turbulence scalars at any point lying on a curved boundary surface. These issues are addressed in available literature under two different approaches. An approach using forcing functions was originally proposed by Peskin (1972) and pursued further by Mohd–Yusof (1997), Verzicco *et al* (2000) and Fadlun *et al* (2000). The forcing technique is most natural when applied to the Navier-Stokes equations for momentum; but formulation of forcing terms for turbulence model equations could be very cumbersome. The basic idea is that a forcing term is added to the equations in cells intersected by a boundary; the magnitude and direction of the force is adjusted so that the boundary conditions are satisfied.

The other approach is designated as cell cutting: here, Cartesian faces of cells that are intersected by the curved surface are replaced by the surface face. Such splitting generates non-rectangular cells, which may be handled efficiently only by an unstructured flow solver.

In the present work a new approach, designated as an extended Ghost Cell Method, is being explored for RANS applications. It promises to provide the flexibility needed for imposing the various boundary conditions of interest in turbulence modeling. Indeed, the approach is simply to impose the specified boundary condition by interpolation onto a surface that does not coincide with a computational mesh surface – or, more correctly,

to use the condition on that non-computational surface to extrapolate the variable value to a ghost node, inside the body.

1.2. *Special issues in RANS computation*

The RANS equations for incompressible flow are

$$\begin{aligned} D_t U &= \frac{1}{\rho}\nabla P + \nabla \cdot \left[(\nu + \nu_T)(\nabla U + \nabla^t U)\right] \\ \nabla \cdot U &= 0 \end{aligned} \tag{1.1}$$

The point to note here is that ν_T is an eddy viscosity which is determined by further transport equations. A variety of models are discussed in Durbin & Pettersson Reif (2000). Generic issues in solving turbulence models can be illustrated by reference to the widely used $k - \varepsilon$ model.

The transport equations for k and ε are

$$\begin{aligned} D_t k &= 2\nu_T |S|^2 - \varepsilon + \nabla \cdot [(\nu + \nu_T)k] \\ D_t \varepsilon &= \frac{C_{\epsilon 1} 2\nu_T |S|^2 - C_{\epsilon 2}\varepsilon}{T} + \nabla \cdot [(\nu + \nu_T/\sigma_\epsilon)\nabla \varepsilon] \end{aligned} \tag{1.2}$$

A distinction from conservation equations is obvious: the right-hand sides contain source and sink terms. In the major part of a turbulent shear layer there is a rough balance between sinks and sources. The source and sink terms strongly couple the k equation to the ε equation and this argues for a coupled solution. (This is believed to be one reason why explicit solution algorithms are not effective on turbulence models. The usual rationale is that explicit methods are not efficient for bringing the source terms into balance. Implicit schemes are better and are the basis of most robust routines for solving turbulence models.)

Another source of stiffness occurs near the boundaries. The no-slip conditions on Eq. (1.2) are

$$k(0) = 0, \qquad \varepsilon(0) = \lim_{y \to 0} 2\nu k/y^2.$$

This is imposed numerically by evaluating the right-hand side at the first computational point above the surface. The factor of $1/y^2$ can cause numerical stiffness if the k and ε equations are not solved as a coupled, implicit system.

Adequate near-wall resolution is required to properly compute heat transfer, skin friction and flow separation. In general the grid should be strongly anisotropic, with small spacing in the wall-normal direction. This grid requirement is tied to the geometry, and mandated by the boundary layers. Separated shear layers can exist within the flow; they also demand local grid refinement, although now tied to the flow field, not the geometry. To capture such features the anisotropic refinement needs to be adapted to the solution.

The established approach to handle the above-mentioned issues is to construct body fitted meshes – structured or unstructured – with high resolution next to solid surfaces. Sometimes the mesh is adapted in the course of the solution – usually on unstructured grids. The present approach differs in both respects: the near-wall refinement is intended to be done with a Cartesian grid, without body fitting. Boundary conditions are imposed by an interpolation method designated as a generalized 'Ghost Cell' approach. Similarly, flow as well as the boundary adaptation are carried out by locally refining a quasi-

structured grid inside the flow and the refinement is non-isotropic which can be easily achieved in a Cartesian grid environment.*

During the last six months, attempts have been made to develop the two different aspects of this research project in parallel. The first part concentrates on the method of local refinement in structured Cartesian grids: the second part explores the appropriate interpolation or reconstruction schemes to handle the boundary conditions, while solving the RANS equations using a finite-difference solver and a Cartesian grid not conforming to the body surfaces. Accomplishments so far are described below in two different subsections.

2. Accomplishments

2.1. *Local refinement using Cartesian grids*

2.1.1. *Methodology*

Most research on solution-adaptive gridding has been reported for unstructured solution methods (Mavripilis, 1995). Local adaptation involves inserting nodes within a pre-existing mesh. Local mesh refinement is considered to be suited only to unstructured solution algorithms because the element connectivity is explicit and local. However, in the present work we propose a somewhat analogous method that can be developed for structured grid algorithms.

The locally adaptive method proposed herein is motivated by the idea of *iblanking* (Benek *et al*, 1985); the terminology, *iblank*, comes from a variable name used in computer codes. Originally *iblanking* was a device to insert geometry into a structured grid by extending the grid inside the body, then blanking out the interior portion. The region inside the body is decoupled from the fluid via boundary conditions. In the original approach the interior region is solved, although the equations there are arbitrary It is straightforward to revise such algorithms so that the blanked region is skipped, requiring neither storage of variables nor solution of equations. For present purposes, the notionally blanked portion of the grid could be larger than the active portion. Our solution algorithm skips the blanked portion.

Conceptually, our method consists of adding grid lines where needed, and blanking out all but the portion of those lines that lies in the area where higher resolution is required. Since the blanked portion is skipped, the actual method adds line segments (refer to Fig. 4). The line segments lie on an underlying, notional, structured grid. The notional grid would be constructed from active and blanked nodes, as illustrated by Fig. 1. The black circles denote an initial, coarse grid. The gray circles indicate points added through local refinement. They amount to inserting an extra grid line in the k-direction, a portion of which is active, the rest blanked — as indicated by the open circles. A discrete solution scheme must be adapted for this class of grids.

The node labeled H (usually called a 'hanging node') in Fig. 1a is connected to three active nodes. To complete the finite difference stencil a solution value is interpolated to the point labeled I. Note that I is *only* used to complete the stencil at H; otherwise it is treated as blanked. The interpolation stencil determines the effective finite difference scheme, and its local accuracy.

For instance, if the value at I is linearly interpolated between its vertical neighbors, then the 6-point stencil in Fig. 1b is implied by the 5-point stencil in figure 1a. A centrally-differenced j derivative is second-order accurate for a symmetric stencil:

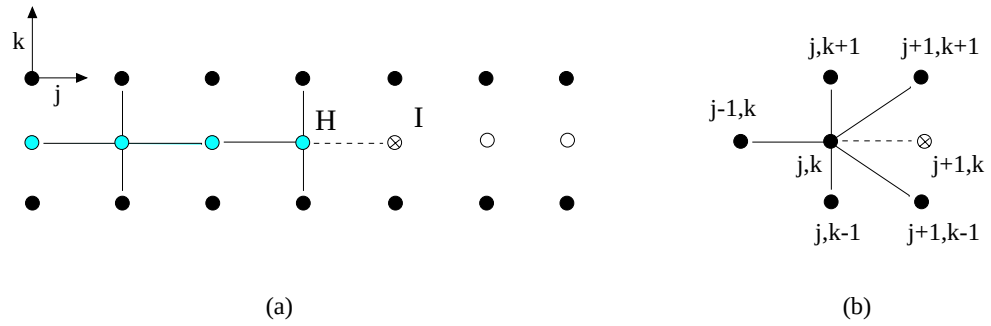

FIGURE 1. Finite difference stencils for hanging nodes. (a) Active (●, ●) and blanked (○) grid points. Points indicated by ● show the initial grid; those indicated by ● are added through refinement; ⊗ is an interpolated point (H Hanging Node; I Interpolation Node). (b) Effective stencil at an interpolation node.

$\delta_j u = \frac{1}{2}(u_{j+1} - u_{j-1})$. However, with an interpolated value as in figure 1b, the accuracy becomes first order because of the asymmetry:

$$\delta_j u_{j,k} = \frac{1}{2}\left(\frac{1}{2}(u_{j+1,k+1} + u_{j+1,k-1}) - u_{j-1,k}\right) \tag{2.1}$$

irrespective of the fact that the value $u_{j+1,k} = \frac{1}{2}(u_{j+1,k+1} + u_{j+1,k-1})$ is a second-order interpolation in the k-direction.

In a 2-D grid of N^2 nodes, the number of hanging nodes will be $O(N)$. Hence the global error of a second-order method on complete stencils would not be reduced by first-order accuracy adjacent to interpolation points. However, the local dissipative error due to first-order convection might be undesirable. Local accuracy can be increased by modifying the interpolation stencil. The formula

$$u_{j+1,k} = \frac{1}{2}(u_{j+1,k+1} + u_{j+1,k-1}) - \frac{1}{2}(u_{j,k+1} + u_{j,k-1} - 2u_{j,k}) \tag{2.2}$$

provides a second-order difference.

2.1.2. *Application examples*

Fig. 2 shows the block refinement of the grid for flow over a backstep. The coarse grid is refined uniformly in two blocks. The upper right corners of each block contain unacceptable interpolation points; in order to apply the interpolation formula Eq. (2.2) no other inactive points should appear in the stencil. The encircled regions contain locations where deletions are needed to form acceptable interpolation stencils.

A flow computation has been performed on the grid of Fig. 2. The full domain extends over $-4 < x < 35$, $0 < y < 6$ with a symmetry condition at $y = 6$. Streamlines, velocity vectors and convergence history are displayed in Fig. 3. This particular grid is simply block-refined; it is not adapted locally to the solution. It serves primarily to illustrate the effectiveness of the solution algorithm and to show that the solution continues smoothly across the interpolation points. Also, no numerical instabilities are encountered.

Second order formulae for the interpolation Eq. (2.2) are used in this computation. The convergence history of Fig. 3 is for a pseudo a time-step of $\Delta t = 2.5$ based on free-stream speed and step height of unity; this corresponds to a fine grid CFL number of 6.5. With

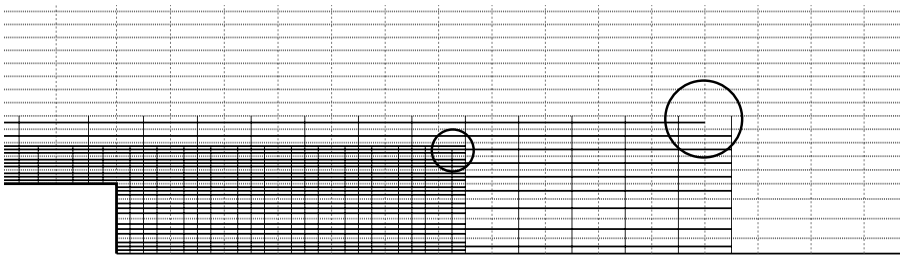

FIGURE 2. Block refinement for flow over backstep. Two levels of refinement are shown. The y-axis has been magnified by a factor of 2. Faint lines are the coarse grid. Circles indicate where the interpolation points were revised.

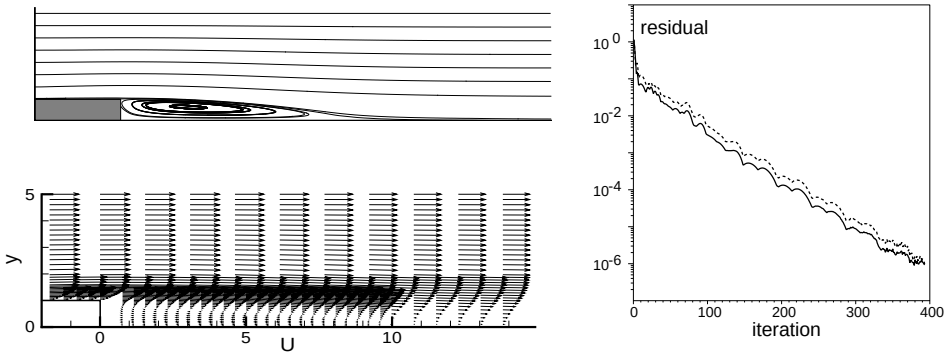

FIGURE 3. Streamlines and velocity vectors for backstep at step height Reynolds number=200 on grid of figure 2. The full domain is $-4 < x < 35$, $0 < y < 5$. Residual plot shows the maximum absolute residual (solid) and maximum divergence (dashed).

$\Delta t = 10.5$ the same level of convergence (*i.e.*, to single precision) was obtained in only 300 iterations.

As a further example of application of the present technique, an adaptive grid refinement for the driven cavity flow proposed by Ghia *et al* (1985) is shown in Fig. 4. The adaptation function is based on a linear combination of velocity and pressure gradients. The initial (uniform) grid consists of 20×20 cells; three successive refinement steps are employed. The final mesh is shown at the left of Fig. 4. Active nodes are clustered in the eddies and near to the walls. The refinement is distinctly non-isotropic.

The streamlines at the right of Fig. 4 show the presence of secondary recirculation regions at the lower corners, in good qualitative agreement with benchmark results of Ghia *et al* (1985). A more quantitative comparison is reported in Fig. 5: velocity profiles on vertical and horizontal centerlines are reported there for the adapted grid, and are compared to the solution on a uniform 100×100 grid. The agreement between these two solutions is quite good. Note that the locally-refined solution cuts through the highly-irregular grid at the center of the cavity. The grid in Fig. 4 contains 4,683 active points; the finest level corresponds to a 80×80 full grid.

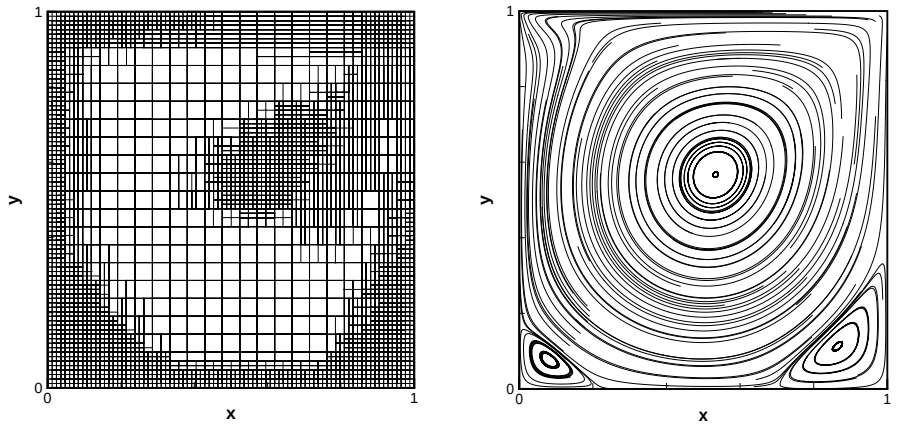

FIGURE 4. Computational grid and streamlines for the flow in a square cavity at Re = 1,000 using adaptive locally refined grids. Hanging nodes are interpolated using Eq. (2.2).

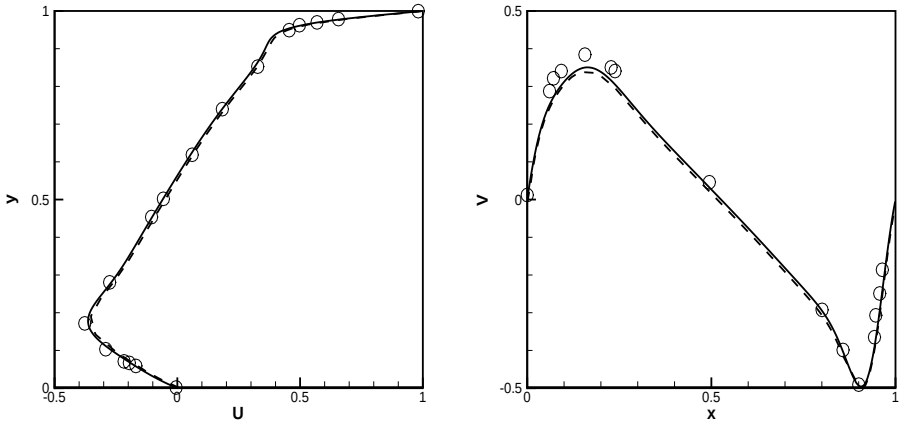

FIGURE 5. Velocity components in vertical and horizontal centerline; flow in a square cavity at Re = 1,000. ——— Uniform mesh; - - - - Locally refined grid, ○ : Ghia *et al.*, (1982)

2.2. *Ghost-cell method for RANS solvers*

2.2.1. *Basic concept*

The first step in this method is to tag all grid nodes according to whether they are inside, outside or bound the body. Points that bound the body are inside, but have at least one neighbor outside as shown in Fig. 6a. The body around which the flow is to be computed is usually represented as a closed polygon. The tagging can be done using an efficient search method. The computational node in question is connected by a linear segment to a point chosen far outside the body polygon to be scanned. One then determines whether this line segment is intersected by the body polygon. If the number

of intersections is even or zero, the node is outside the body: if the number of intersection is odd the node is inside.

Once the inside nodes are identified, the boundary nodes lying inside the body and connected to at least one computation node in the flow domain are marked as the Ghost nodes or Ghost points in the flow computation. The flow solver senses the presence of the boundary through the values of flow variables at the ghost nodes, which are computed using a local Reconstruction Scheme involving the ghost node and some of its neighboring flow nodes. The interpolation function and the relevant boundary and domain where the interpolation function can be evaluated matters of choice. Polynomial functions are usually efficient since these are compatible with the finite-difference stencils which link the ghost node to the flow computation nodes. The solution accuracy depends to a great extent on the degree of the interpolating polynomials used. But polynomials of higher degree, although expected to be more accurate, may often lead to serious boundedness problems and hence to numerical instability. Three different interpolation procedures have been attempted so far for a few validation test cases. The variable values at the ghost cell can be updated either in an implicit or an explicit manner, depending on the coupled flow solver. Implicit updating is in general observed to have enhanced numerical stability whereas explicit updating of the ghost node values often requires very low underrelaxation factors for numerical convergence.

2.2.2. *Interpolation procedures*

1. Linear interpolation in triangular domain

The simplest approach to linking the ghost node to the surrounding fluid nodes is to construct a triangle where the ghost node forms one of the vertices and the other two nodes lie in the flow domain. These triangles can be chosen by the user depending on the configuration: see Fig. 6a, where G is the ghost node, $F1$ and $F2$ are the fluid nodes and B, the midpoint of the intercept, is assumed to be the wall node where the boundary conditions are satisfied through the interpolation procedure.

A simple linear interpolation formula in a 2D space is taken as

$$\phi = a_1 x + a_2 y + a_3 \tag{2.3}$$

The flow solver usually needs the variable values at the ghost node as weighted combination of the values at the neighboring nodes, in the following form

$$\phi_G = w_1 \phi_1 + w_2 \phi_2 + w_B \phi_B \tag{2.4}$$

It is therefore most convenient to evaluate the relevant weighting coefficients and the neighboring node indices in a pre-processor and use them later in the flow solver. The weighting coefficients can always be expressed in terms of the interpolating polynomial in the following form:

$$[w_1, w_2, w_3]^t = T^{-1}[x_G, y_G, 1]^t \tag{2.5}$$

where in the case of a linear interpolation T is a 3×3 matrix, whose elements can be computed from the coordinates of the three boundary points of the interpolation space. Either the values or the normal derivatives are specified on that boundary. x_G, y_G are the coordinates of the ghost point.

In the preliminary calculations this simple linear relationship is used to extrapolate the ghost-point value at G — which obviously lies outside the interpolation space. The major drawback experienced with such extrapolation is that large, negative weighting

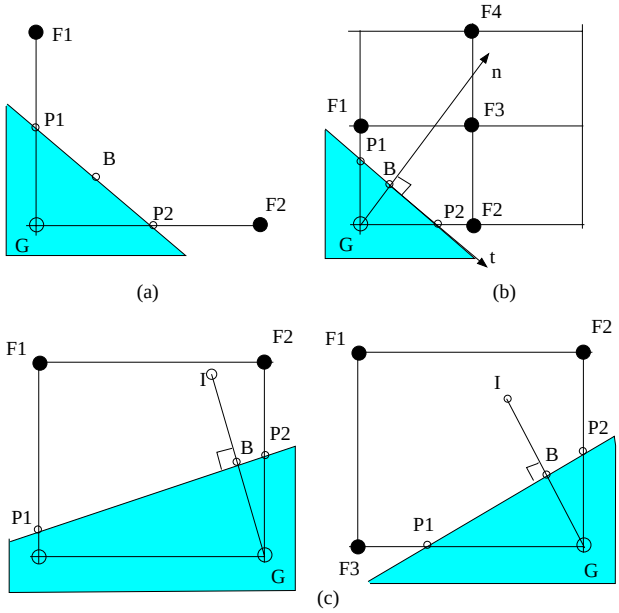

FIGURE 6. Schematic representation of the interpolation procedure; (a) Linear interpolation, (b) Quadratic interpolation, (c) Bilinear interpolation (G Ghost point, I Image point)

coefficients are often encountered. Although these are algebraically correct, they can lead to to severe numerical instability of the coupled flow-solution algorithm.

Using the concept of an image point inside the flow domain for each ghost node is a better procedure to ensure suitable weighting coefficients of the neighboring nodes. In this approach, an image point I is identified in the fluid, along the normal to the boundary through the ghost node G. It lies at a distance equal to that of the ghost node from the same boundary (Fig. 6c). The value of the flow variable is first evaluated at this image point using the chosen interpolation scheme. The principle of maxima can be used to show that if a linear (or bilinear) interpolation scheme is employed, the weighting coefficients are guaranteed to be positive and less than unity. However this argument is not valid when normal derivative conditions on a solution variable are applied at any boundary point of the interpolation space. The value at the ghost node can be expressed, assuming a linear variation along the line connecting G and I, as

$$\phi_G = 2\phi_B - \phi_I. \tag{2.6}$$

The wall boundary conditions are satisfied at B, the foot of the perpendicular from the ghost point G on the wall segment. For velocities and some of the turbulence scalars, the value of ϕ may be set to zero at B on a non-moving wall;. but for pressure or turbulence dissipation (ϵ), the wall value also needs to be evaluated from the neighboring node information in order to obtain the ghost-point value.

2. Bilinear interpolation in rectangular domain

If the interpolating polynomial is assumed to be bilinear in a 2D space, it involves four constants and hence needs four conditions to be specified on the boundary of the interpolation space. For a given body polygon, it is quite straightforward to compute the intersection-point coordinates on the different cell boundaries of the grid network.Thus any ghost node forming one corner of a grid rectangle may have two different types of

intersection as shown in Fig. 6c. In both cases, however, the two fluid nodes and the two wall-intersection nodes may be used for evaluation of the relevant weighting coefficients for the image point. The polynomial in this case may be written as

$$\phi = a_1 x + a_2 y + a_3 xy + a_4. \tag{2.7}$$

The wall intersection points $P1$ and $P2$ on the computational grid lines are used in evaluation of the interpolating polynomial, and the wall boundary conditions are satisfied at the point B. The required weighting coefficients are computed using the same principle described for linear interpolation in triangular space – only the matrix T connecting the interpolation point coordinates will now be different.

3. Linear-quadratic interpolation along wall-normal direction

Most second-order-accurate, finite-difference flow solvers use quadratic variation of flow variables in the direction normal to the wall. The higher order interpolation during reconstruction is therefore expected to retain the formal second order accuracy of the scheme. If the flow variables are assumed to vary in a quadratic manner along the wall normal direction and linearly along the wall, the interpolating polynomial is:

$$\phi = a_1 n^2 + a_2 n + a_3 t + a_4 nt + a_5 \tag{2.8}$$

where the wall coordinates n and t as shown in Fig. 6b. The normal to the wall intersects the adjacent grid lines at two points and the variable values at these two points in turn depend on the neighboring flow node values. Hence the five constants of the assumed polynomial are evaluated from the four neighboring flow nodes, marked with filled-in circles, and the wall point (Fig. 6b). The ghost-node value is either extrapolated or evaluated using the concept of an image point. The use of the image point in case of a quadratic interpolation may produce better weighting coefficients, but their values are no longer guaranteed to be positive and less than unity as they are in the linear interpolation scheme.

2.2.3. *Test cases*

The numerical method of the RANS flow solver to which the present ghost cell approach is coupled is based on a Cartesian, two-dimensional Navier Stokes code. The discretization technique is cell-vertex, finite differences over structured meshes. The convection term is third-order upwind biased; diffusion terms are second-order centered. The pseudo-time integration is implicit and the equations are solved in a coupled manner. The implicit matrices are inverted by Gauss-Seidel line relaxation. The ghost-cell procedure has so far been tested for two laminar flows, using linear or bilinear interpolation, and for a simple turbulent flow using quadratic interpolation.

1. Laminar flow around a semi-circular cylinder at Reynolds number of 150

A semi-circular shape was chosen to avoid flow unsteadiness and vortex shedding observed for circular cylinders at this Reynolds number. This calculation uses the linear interpolation scheme in triangular domains around the ghost points. Three grids of different grid fineness are used, covering the cylinder zone by 20×20, 40×40 and 80×80 uniformly-spaced grids. These provide 43, 87 and 177 ghost points respectively to represent the curved boundary. The finest Cartesian grid used, superimposed on a boundary-fitted (radial polar) grid of approximately the same grid spacing, is shown in Fig. 7. Results obtained from the boundary-fitted grid calculation are compared in Fig. 8 to those of the present ghost-cell procedure.

The present results with the finest grid are observed to be the closest to the solution

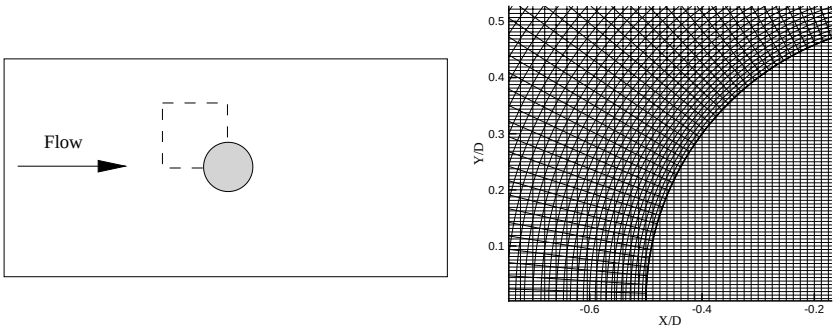

FIGURE 7. Test problem for semi-circular cylinder and close view of the Cartesian grid overlapped with the boundary-conforming radial polar grid

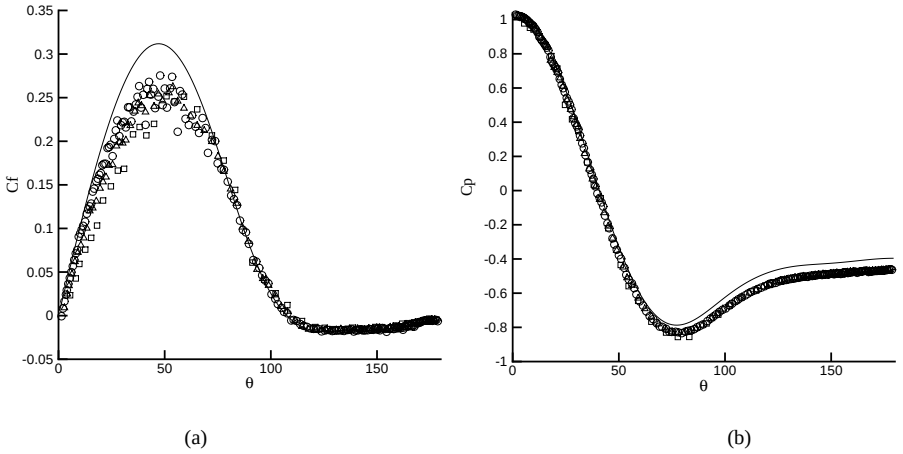

FIGURE 8. Circumferential variation of (a) skin-friction and (b) pressure coefficient for flow around a cylinder ($Re = 150$). ——— : Radial polar grid, (○ , △ , □) Cartesian grid with ○ 177 ghost points , △ 87 ghost points and □ 43 ghost points.

obtained from the boundary-fitted grid. The discrepancies observed in the maximum value of the skin friction and for the surface pressure in the post-separation region may be attributed to the inaccuracy introduced by the linear interpolation procedure used for local reconstruction. The mild kinks or wiggles are always observed in Cartesian grid calculations with ghost cells. This may perhaps be attributed to the circumferential variation of the distance between the wall point where the boundary conditions are satisfied, and the corresponding ghost point. In a boundary-fitted grid, the normal distance of the near-wall point can be systematically controlled and maintained to vary continuously along the body surface. On the other hand in the boundary non-conforming situation, the distance between the wall and the ghost point, primarily decided by the geometry of the body and the Cartesian grid intersection, may often have an erratic variation along the body surface.

2. Laminar flow past a two-dimensional bump

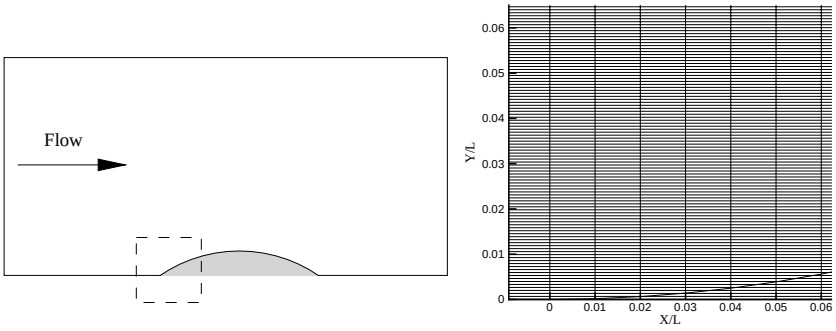

FIGURE 9. Test problem for flow past a two-dimensional bump and close view of the Cartesian grid.

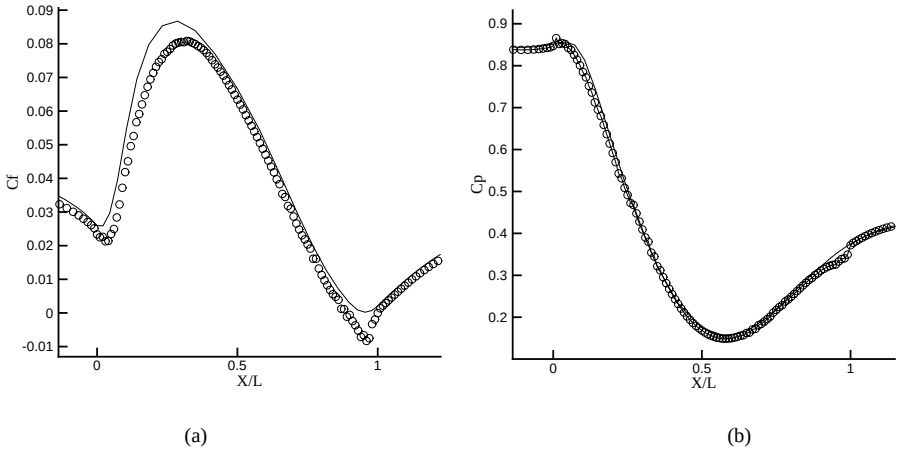

FIGURE 10. Longitudinal variation of (a) skin-friction and (b) pressure coefficient for flow past a bump ($Re = 1000$). ——— : boundary-conforming grid, ∘ non-conforming grid.

This test case has been chosen to validate the present ghost-cell procedure for two-dimensional turbulent flow in a geometry for which detailed measurement data of Webster *et al* (1996) are available. Because handling the boundary conditions for turbulence scalars is not so straightforward, attempts have first been made to analyse laminar flow over the same geometry and to compare the results with other calculations using boundary-conforming grids. The surface bump on the lower wall of a channel is defined by three tangential circular arcs and is shown schematically in Fig. 9. The boundary layer experiences the effects of significant surface curvature and streamwise pressure gradient. The present calculation uses uniform plug flow at the channel inlet at a flow Reynolds number of 1000, based on the bump chord length.

A close-up view of the rectangular grid used near the concave root of the bump is also shown in Fig. 9 . As stated earlier, the difficulty in using the boundary-non-conforming grid is the lack of control on the resolution near the boundary. The only way to obtain

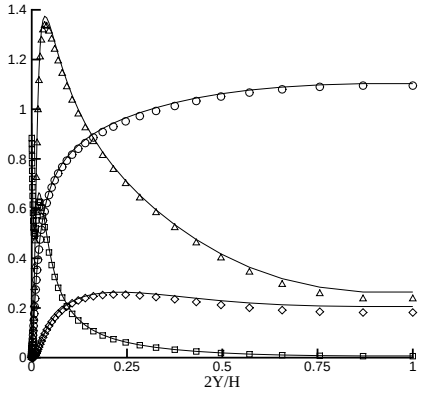

FIGURE 11. Transverse variation of mean velocity and turbulence quantities for fully developed channel flow (Re based on half channel height = 10,000). ——— boundary conforming grid, ○ △ □ ◇ non-conforming grid (○ U, △ 100k, □ 50 ε and ◇ 100 v^2).

adequate resolution is to use a very fine uniformly-spaced grid in the region covered by the intersecting curved boundary. The present calculation uses a 201×201 Cartesian grid, out of which 101×101 points cover the bump region. Since the bump height is only about 6.5% of the chord length, the grid aspect ratio is always of the order of 15 near the bump surface region. This computation uses bilinear interpolation in rectangular interpolation space coupled to the image-point concept discussed in the previous section.

Fig. 10 shows the longitudinal variation of the surface pressure and the skin-friction coefficient for the flow, computed in the present procedure directly from the spatial interpolation scheme used in the local reconstruction. Comparison to results obtained from boundary-conforming grid for the same configuration at the same Reynolds number shows excellent agreement for the surface pressure distribution; but the skin friction is observed to be under-predicted by about 5%, especially near the top of the bump. Usually skin friction is very sensitive to the resolution of the boundary layer. In a boundary-non-conforming grid environment, this resolution would always vary along the body surface and inadequate resolution at some of the longitudinal sections may lead to inaccurate near-wall velocities and hence a wrong value of C_f.

Some mild kinks are observed in the results based on the ghost-cell procedure, especially near the beginning and end of the bump, where the weighting coefficients of the interpolation procedure suddenly change from the uniform value over the flat part of the channel to new values as the bump starts or ends. It is further observed that as the flow Reynolds number is increased, the amplitude of these wiggles grows, and may eventually lead to numerical instabilities.

3. Fully developed turbulent flow in a plane 2D channel

This simple example is chosen to test the capability of the present ghost-cell procedure when coupled to a RANS solver with a standard turbulence model. The boundary non-conformity is maintained by placing the channel wall between two horizontal grid lines. Quadratic interpolation along the boundary-normal direction is used in this case.

The $v^2 - f$ turbulence model (Durbin, 1995) is used for this RANS computation of channel flow. In the reconstruction procedure, the boundary conditions satisfied at the wall point are $U = V = k = v^2 = f = \partial p/\partial y = 0$. Evaluation of the non-zero value of the

turbulence dissipation ϵ at the wall point is however not so obvious. The value of ϵ at wall points is set to $2\nu k/y^2$ where k is the turbulence kinetic energy at the near-wall node at a normal distance y from the wall surface. As mentioned in Section 1, such a boundary condition can cause undesirable numerical stiffness and was therefore implemented in a coupled. implicit manner.

Fig. 11 shows the transverse profiles of the mean longitudinal velocity and different turbulence quantities computed using the conventional body-conforming as well as the present body-non-conforming grids. Reasonable agreement between the two results demonstrates the accuracy of the ghost cell procedure with quadratic interpolation, at least when the body boundaries are parallel to either of the Cartesian grid directions.

3. Conclusions and future tasks

The ghost-cell procedure to handle complex-shaped curved boundaries using a Cartesian grid has been validated for a few two-dimensional flow situations. Work is in progress to study the numerical stability of the procedure — especially how to ensure numerically amenable values of the interpolation weights, and to eliminate the large-amplitude oscillations of the solution which are often observed at high flow Reynolds number.

The validation examples have shown that adequate near-wall resolution can be obtained in the present method only by using a very fine grid. This usually implies many wasteful, inactive nodes in a fixed Cartesian grid. Therefore the local refinement procedure, already developed for structured Cartesian grids, now needs to be coupled to the ghost-cell procedure to develop an efficient and accurate RANS solver. We then propose to extend the methodology to three-dimensional flow and validated against measurement and/or other computation data for some turbulent flow examples of industrial interest.

4. Acknowledgements

The first author gratefully acknowledges the financial support provided by the Center for Turbulence Research and by the National Aerospace Laboratories, Bangalore, India.

REFERENCES

Benek, J. A., Buning, P. G., Steger, J. L., 1985 A 3-D Chimera grid embedding technique, *AIAA* Paper 85-1523.

Durbin, P. A. & Pettersson Reif, B. A. 2000 *Statistical Theory and Modeling for Turbulent Flow*. Wiley, New York.

Durbin, P. A. 1995 Separated flow computations with the $k-\epsilon-v^2$ model. *AIAA J.* **33**, 659–664.

Fadlun, E.A., Verzicco, R., Orlandi, P. & Mohd–Yusof, J. 2000 Combined immersed–boundary/finite–difference methods for three–dimensional complex flow simulations. *J. Comp. Phys.* **161**, 35.

Ghia, U., Ghia, K. N., Shin, C. T. 1985 High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comp. Phys.* **48**, 387–411.

Mavripilis, D. J. 1995 Unstructured mesh generation and adaptivity. *Von Karman Inst. for Fluid Dynamics*, Lecture Series 1995-02.

Mohd–Yusof, J. 1997 Combined immersed–boundary/B–spline methods for simulations of flows in complex geometries, *A*nnual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ. 317–327.

Peskin, C. S 1972 Flow patterns around heart valves: a numerical method. *J. Comp. Phys.*, **10**, 252–271.

Verzicco, R., Mohd–Yusof, J., Orlandi, P. and Haworth, D. 2000 Large–eddy–simulation in complex geometric onfigurations using boundary body forces. *AIAA J.* **38**, 427.

Webster, D. R., DeGraaff, D. B. and Eaton, J. K. 1996 Turbulence characteristics of a boundary layer overa two-dimensional bump. *J. Fluid Mech.* **320**, 53-69.