Toward immersed boundary simulation of high Reynolds number flows

By Georgi Kalitzin and Gianluca Iaccarino

1. Motivation and Background

In the immersed boundary (IB) method, the surface of an object is reconstructed with forcing terms in the underlying flow field equations. The surface may split a computational cell removing the constraint of the near wall gridlines to be aligned with the surface. This feature greatly simplifies the grid generation process which is cumbersome and expensive in particular for structured grids and complex geometries.

The IB method is ideally suited for Cartesian flow solvers. The flow equations written in Cartesian coordinates appear in a very simple form and several numerical algorithms can be used for an efficient solution of the equations. In addition, the accuracy of numerical algorithms is dependent on the underlying grid and it usually deteriorates when the grid deviates from a Cartesian mesh.

The challenge for the IB method lies in the representation of the wall boundaries and in providing an adequate near wall flow field resolution. The issue of enforcing no-slip boundary conditions at the immersed surface has been addressed by several authors by imposing a local reconstruction of the solution. Initial work by Verzicco *et al.* (2000) was based on a simple linear, one-dimensional operator and this approach proved to be accurate for boundaries largely aligned with the grid lines. Majumdar *et al.* (2001) used various multidimensional and high order polynomial interpolations schemes. These high order schemes, however, are keen to introduce wiggles and spurious extrema. Iaccarino & Verzicco (2003) and Kalitzin & Iaccarino (2002) proposed a tri-linear reconstruction for the velocity components and the turbulent scalars. A modified implementation that has proven to be more robust is reported in this paper.

The issue of adequate near wall resolution in a Cartesian framework can initially be addressed by using a non-uniform mesh which is stretched near the surface. In this paper, we investigate an unstructured approach for local grid refinement that utilizes Cartesian mesh features.

The computation of high Reynolds number wall bounded flows is particularly challenging as it requires the consideration of thin turbulent boundary layers, i.e. near wall regions with large gradients of the flow field variables. For such flows, the representation of the wall boundary has a large impact on the accuracy of the computation. It is also critical for the robustness and convergence of the flow solver.

2. Numerical Technique

The computational code IBRANS is based on the steady-state incompressible RANS equations closed either with an one-equation turbulence model (Spalart & Allmaras 1994) or a two-equation model (Kalitzin & Iaccarino 2002). A second-order, implicit, cell centered Cartesian discretization is used within a SIMPLE pressure-velocity coupling algorithm with a segregated solution of the field equations (Ferziger & Peric 2002).

The equations are discretized in three-dimensions with a typical 7-point stencil. In this paper, however, we will refer for simplicity to the corresponding two-dimensional equations. For each cell (i, j), the discretized field equation for momentum, turbulent scalar and pressure correction can be written as:

$$a_W^n \phi_W^{n+1} + a_E^n \phi_E^{n+1} + a_S^n \phi_S^{n+1} + a_N^n \phi_N^{n+1} + a_{(i,j)}^n \phi_{(i,j)}^{n+1} = s_{(i,j)}^n \tag{2.1}$$

where the indices E, W, S, N denominate the neighboring cells that have a common face with cell (i, j), e.g. E for cell $(i+1, j)$ etc. The superscript n indicates the iteration. The equation is under-relaxed implicitly to increase the diagonal dominance of the implicit matrix. The Strong Implicit Procedure (SIP) by Stone (1968) is used for the solution of (2.1).

The flow solver is set up as a virtual wind tunnel. The x-direction is the main flow direction with inflow and outflow, the other directions have either the usual wall boundary, symmetry or periodicity conditions. The domain is partitioned in x-direction and the equations are solved in parallel using MPI. An object is placed into the flow domain and the immersed boundaries, described in the next section, are applied to the velocity components and the turbulence variables. There is no special treatment of the pressure at the immersed boundary. The immersed boundaries affect the Poisson equation for the pressure correction only through the source term which represents the divergence of the velocity field. Thus, the Poisson equation is solved in the entire computational domain. A parallel geometric multigrid is being used to accelerate the convergence of the Poisson equation.

The non-uniform computational grid and the IB interpolation stencil is generated automatically and the pre-processing time to start the computation is negligible. The procedure is based on the ray tracing technique that allows one to identify the cells cut by the immersed boundary. An initial coarse (and typically) uniform mesh is specified by the user; a three-steps iterative procedure is employed: (a) tag the grid identifying the cells cut by the immersed boundary, (b) split these cells in the direction that moves the cell center closer to the immersed boundary, (c) regenerate a structured grid by propagating the cell split to the domain boundaries, and restart from (a). This procedure is repeated until a prescribed distance from the wall is achieved for all the interface cells. The geometry definition is based on the StereoLitography format and therefore the CAD model can be used directly.

3. Immersed Boundary Treatment

After the pre-processor detects the computational cells that are cut by the body, the cells are divided in those that are inside and outside of the body as shown in Fig. 1. The cut cells are separated in two type of cells corresponding to the location of their cell center. The cells that have their cell center outside or on the surface are labeled as interface cells. The other cut cells are treated like inside cells.

The flow variable ϕ, which represents a velocity component or a turbulence variable, is set to zero in the inside cells. Note that the turbulence models considered have zero wall boundary conditions for all turbulence variables. At the interface cells, the nearby wall is modeled with an off wall boundary condition which in general consists of an implicit linear interpolation stencil with an explicit (non-linear) correction.

A sketch of the linear interpolation algorithm is shown in Fig. 1a. Like the plotted tangential velocity component, the scalar variable ϕ in the interface cell (i, j) is interpolated

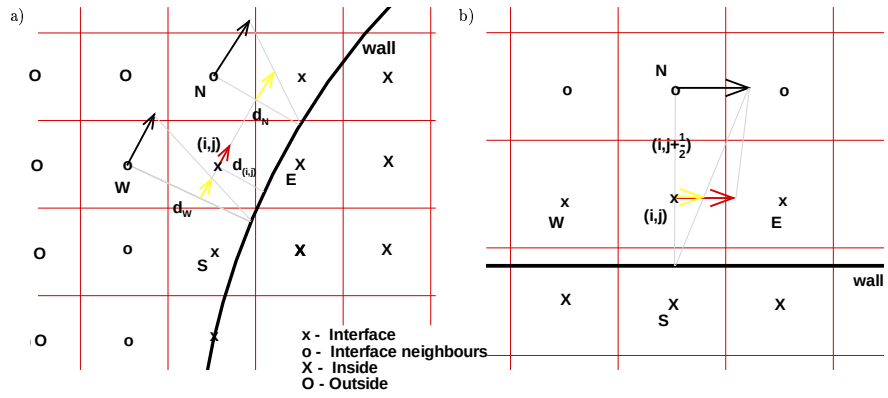

FIGURE 1. Schematic representation of interpolation algorithm for the tangential velocity. General case (left) and wall function approach for a grid aligned case (right). Tangential velocity: ⟶ computed, ⟶ interpolated linearly on intermediate location, ⟶ imposed in interface cell.

using only the neighboring cells that are further away from the body. The interpolation is carried out in two steps. First, for each neighboring cell an interpolation is carried out normal to the wall to an intermediate location that has the same distance to the wall as the interface cell center (i, j). This interpolation might be linear, quadratical or other, depending on the near wall behavior of the considered variable. The second interpolation is on a surface that is parallel to the wall. Here we use the inverse-distance weighted method proposed by Franke (1982) that has the property of preserving local maxima and producing smooth reconstructions. The interpolation of $\phi_{(i,j)}$ in the interface cell is:

$$\phi_{(i,j)} = \sum_m w_m \tilde{\phi}_m / q \quad \text{with } w_m = \left(\frac{H - h_m}{H h_m} \right)^p, \quad q = \sum_m w_m \quad \text{and} \quad p = 2 \tag{3.1}$$

where $\tilde{\phi}_m$ represents the previously interpolated values, h_m is the distance between the location of $\phi_{(i,j)}$ and the location of $\tilde{\phi}_m$ and H represents the maximum of the h_m's. The sum is over all neighboring cells with $m = W, E, S, N$. The weights w_m are zero for all non-qualifying neighbors. For each interface cell the described linear interpolation results in a set of weights β_m which correspond to the effect of the neighboring cells on the interface cell:

$$\phi_{(i,j)} = \beta_W \phi_W + \beta_E \phi_E + \beta_S \phi_S + \beta_N \phi_N \tag{3.2}$$

By considering only the neighboring cells (W, E, S, N) that have a common face with the interface cell (i, j) the linear interpolation (3.2) can easily be treated implicitly; the implicit matrix of the discretized equation (2.1) can be modified with the weights β_m without introducing new diagonals with non-zero elements. The inclusion of other neighbors such as corner cells in the interpolation stencil (3.2) can be done explicitly. For a non-linear near wall behavior of ϕ an explicit correction $\Delta\phi^{corr}_{(i,j)}$ is added to the right hand side of equation (3.2). This correction is computed as the difference of the non-linear value of $\phi_{(i,j)}$ and the linear interpolated value, both computed with values from the current iteration.

The described interpolation can be applied to the velocity vector $\vec{v}$ without coupling

implicitly the momentum equations for the Cartesian velocity components. For this, the velocity components are transformed into a wall aligned 2-D coordinate system with the axis pointed in the tangential $\vec{t}$ and normal $\vec{n}$ velocity direction: $\vec{v}_n = (\vec{v} \cdot \vec{n})\vec{n}$ and $\vec{v}_t = \vec{v} - \vec{v}_n$, respectively. The normal unit vector is obtained from the wall distance d as: $\vec{n} = \nabla d/|\nabla d|$. Both components v_t and v_n are interpolated according to their physical behavior normal to the wall and using the interpolation (3.1) along the surface. The tangential and normal velocity components in the interface cell are then rotated back into the base Cartesian system. Because of linearity the weights β remain the same for the Cartesian components. As before, the non-linearity is treated as an explicit correction.

For low Reynolds numbers, the near wall behavior of the tangential velocity is linear. As plotted in Fig. 1a, the tangential component on the intermediate location for cell N is: $(\tilde{v}_t)_N = (v_t)_N d_{(i,j)}/d_N$. The corresponding quadratically interpolated normal component is: $(\tilde{v}_n)_N = (v_n)_N d^2_{(i,j)}/d^2_N$. For high Reynolds numbers, the near wall behavior of the tangential velocity is in general non-linear and the value at the intermediate location is determined with a non-linear procedure that utilizes adaptive wall functions.

In Kalitzin *et al.* (2003), an adaptive wall function concept has been developed for body fitted grids and zero pressure gradient flow. These wall functions are independent of the location of the first grid point above the wall. A look-up table provides an explicit dependency of the friction velocity on the local Reynolds number. This look-up table consists of cubic splines that approximate piecewise a solution obtained on a very fine grid for a zero pressure gradient boundary layer flow. This universal law is written in terms of the local Reynolds number Re_y rather than in terms of y^+. The local Reynolds number is defined as $Re_y = U^+ y^+ = Uy/\nu$. The look-up table is turbulence model specific as the universal law varies slightly depending on the turbulence model. Similar look-up tables provide the explicit dependency of the eddy-viscosity and turbulence variables on the local Reynolds number or on the wall distance y^+. The tables also include the derivative of the velocity and turbulence variables.

In this paper, we only consider a high Reynolds number case in which the plate is aligned but not coincident with the grid lines as shown in Fig. 1b. For zero pressure gradient flows, the Navier-Stokes equations simplify near the wall to:

$$(\nu + \nu_t)\frac{dU}{dy} = u_\tau^2 \tag{3.3}$$

In the current approach, the velocity in cell (i, j) is chosen such that the flux at the face $(i, j + \frac{1}{2})$ fulfills equation (3.3). The tangential velocity and the wall distance in cell N define the local Reynolds number Re_y which is related to the friction velocity u_τ via the adaptive wall function table. The friction velocity u_τ and the distance between the face $(i, j + \frac{1}{2})$ and the wall allow to introduce a wall distance y^+ which determines an eddy-viscosity value $(\nu_t)_{(i,j+\frac{1}{2})}$. The discretized equation (3.3) defines the tangential velocity in (i, j):

$$u_{(i,j)} = u_N - \frac{u_\tau^2}{\nu + (\nu_t)_{(i,j+\frac{1}{2})}}(d_N - d_{(i,j)}) \tag{3.4}$$

This approach can not simply be generalized for a situation shown in Fig. 1a as the tangential velocity in (i, j) is not uniquely determined by one single interface flux. For the general non-aligned case we employ therefore a crude approach by simply extrapolating of the tangential velocity component and eddy-viscosity using wall function information about the derivative in cell N.

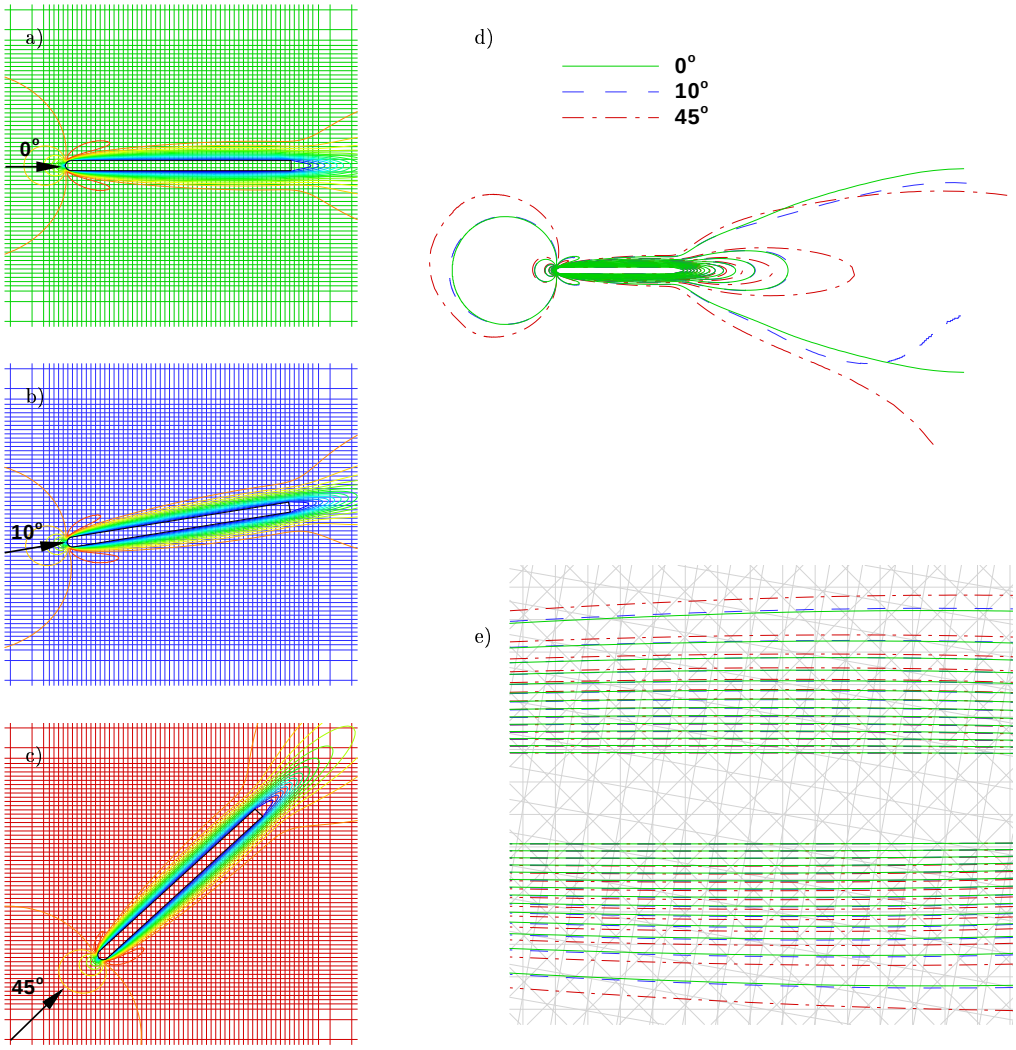

FIGURE 2. Velocity magnitude for immersed boundary layers simulations on non-aligned grids, $Re = 1000$, mesh with every $8th$ line plotted.

Immersed boundary results for a flat plate boundary layer at $Re = 1,000$ are shown in Fig. 2. The three simulations are carried out on uniform, non-aligned grids with a different angle between the plate and the gridlines. The flow is laminar and y^+ of the first cell center is about 0.1. The case of 10^o is the most challenging as the wall distance of the first grid point above the wall fluctuates by moving in the streamwise direction along the plate. There are about 30 cells in the boundary layer and 20 cells across the plate resolving the cylindrical leading edge. The solution is nearly independent of the angle between plate and gridlines as shown with the velocity magnitude contours in Fig. 2d

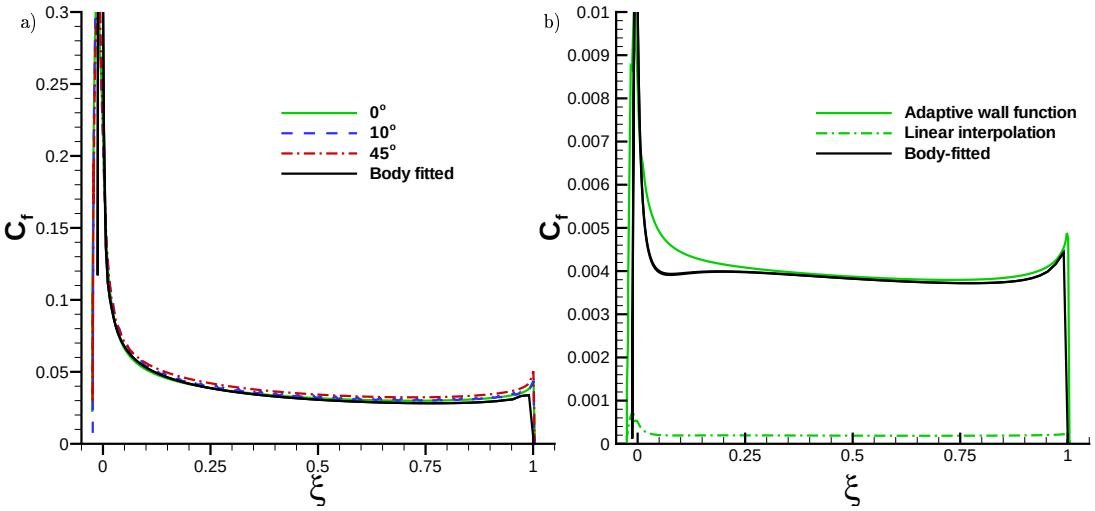

FIGURE 3. Skin friction distribution for immersed boundary layer simulations over a flat plate for various grid alignments for $Re = 1000$ (left) and for grid aligned case for $Re = 10^6$ (right)

and Fig. 2e. The contour lines for larger velocity values deviate slightly for the 45^o what is likely to be a numerical dissipation effect. However, the skin friction along the plate matches remarkably well the body fitted solution shown in Fig. 3a.

In Fig. 3b, skin friction is reported for an immersed boundary simulation of a flat plate boundary layer at $Re = 10^6$. The plate is aligned with the grid and the turbulent flow is computed with the Spalart-Allmaras turbulence model. The wall distance y^+ of the first cell center above the wall is about 25. The results obtained using wall functions are compared to the ones obtained using the simple linear interpolation and to a body fitted solution. The skin friction levels for the linear interpolation are incorrect whereas the wall functions results are in good agreement with the body fitted solution downstream of $\xi = 0.25$. Upstream of this location, the discrepancy between the wall function and body fitted result is related to the grid resolution and wall function implementation. The resolution of the cylindrical leading edge is very coarse for this Reynolds number. The grid is the same as the one used for the simulations in Fig. 2.

4. Local grid refinement

A new version of IBRANS with a local grid refinement capability is in development for an efficient clustering of cells in the boundary layers. The present implementation is an extension of the "classical" adaptive mesh refinement (AMR) technique to allow non-isotropic refinement. It can also be interpreted as a generalization of the procedure used for building coarse grids for geometric multigrid on structured meshes.

The basic idea was introduced in Durbin & Iaccarino (2002) for a finite difference discretization. The AMR grid is considered as a *coarsened* version of an underlying, structured grid; on this underlying grid the cells are defined (in two-dimensions) by a couple of vertices with indices (i, j) and $(i + 1, j + 1)$. On the AMR grid each element

is bounded by the gridlines passing through the vertices (i, j) and $(i + \Delta_i, j + \Delta_j)$. The effective element size in the AMR grid is not constant ($\Delta_i \neq \Delta_j$ and both indices depend on (i, j)). Therefore, the cells are not organized in a structured way with one-to-one neighbors in each Cartesian direction. This requires a modification of the algorithm to deal with *hanging nodes*. In Durbin & Iaccarino (2002) this was simply based on a second-order interpolation. In the current finite volume implementation, flux conservation is enforced and the algorithm resembles the unstructured face-based algorithm described in Ferziger & Peric (2002) and, more closely, the AMR discretization used in Ham *et al.* (2002). Each face-flux is computed using the two adjacent cells and for each cell the fluxes (in general more than four) are collected to build the corresponding diffusive and convective operators. In two dimensions, the implicit discretization yields a sparse matrix with elements not organized in five diagonals as for its structured counterpart. This complexity in the matrix structure prevents the use of the SIP procedure and a Krilov-type algorithm with a simple Jacobi pre-conditioner was implemented. Standard conjugate gradient is used for the pressure equation and the BiCGStab (Van den Vorst 1992) for the momentum and turbulent scalars.

The major advantage of the present approach with respect to classical OCTREE-based (Berger & Aftosmis 1998) and fully-unstructured (Ham *et al.* 2002) schemes lies in the economy and flexibility of storing and retrieving connectivity information due to the underlying grid. In particular, only N cells are effectively defined on a $N_i \times N_j$ underlying grid and they are defined by the two couples (i, j) and $(i + \Delta_i, j + \Delta_j)$. The total storage cost is $4N$ integers. In addition, an array of integers, $ID_{(i,j)}$, is defined on the fine grid to store the correspondence between the underlying cell and the actual AMR element. In other words, all the underlying cells included in the range $(i, i+\Delta_i-1)$ and $(j, j+\Delta_j-1)$ are tagged using the AMR cell number. The total storage required is, therefore, $N_i \times N_j$. The connectivity information for each cell are retrieved consistently to a structured framework by indirectly querying the array $ID_{(i,j)}$. The neighbors of a AMR cell are $ID_{(i-1,k)}$ and $ID_{(i+\Delta_i+1,k)}$ for k ranging between j and $j+\Delta_j-1$, in the positive and negative i-direction respectively. It is evident that the approach handles multiple hanging nodes for each cell and, eventually, allows to reconstruct additional connectivity information without any increase in storage; for example it is straightforward to identify all the vertex-based neighbors.

The generation of AMR grids is carried out by creating the underlying (fine) grid as discussed before and then coarsening it in the region away from the immersed boundary. The advantage of this approach is that all the cell tagging (ray tracing) can be performed on a structured grid taking full advantage of the alignment of the cell centers and the grid nodes. The coarsening and the generation of the connectivity information is the last step of the grid generation process.

An example of the application of this procedure is shown in Fig. 4 where both the underlying and the AMR grids are reported; note that the AMR grid contains only 9% of the underlying cells. A sample calculation has been performed on both grids and it is reported in Fig. 5 in terms of turbulent kinetic energy. The solution on the AMR grid is remarkably smooth and consistent with the one obtained on the structured mesh. It converges significantly faster as reported in Fig. 6. This is mainly due to a reduced aspect ratio of the cells away from the immersed boundary. In terms of computational cost the savings is about 70%.

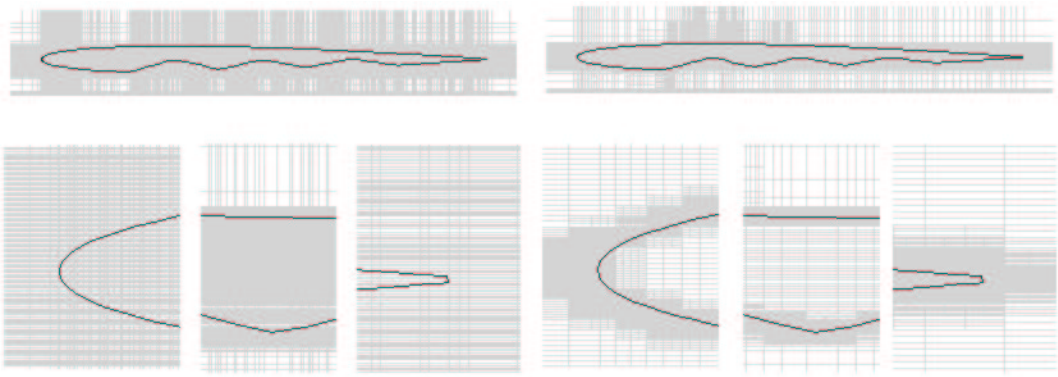

FIGURE 4. Computational grid for the flow around an airfoil. Structured, underlying grid on the left with every second grid line plotted and locally refined grid on the right.

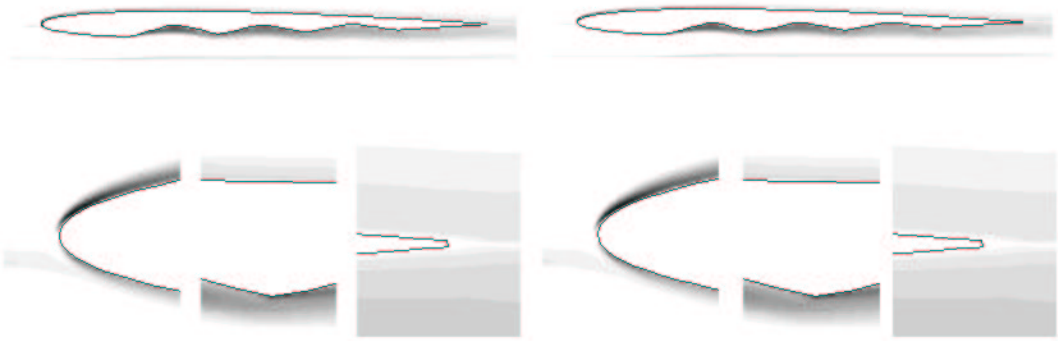

FIGURE 5. Turbulent kinetic energy for structured grid (left) and locally refined grid (right).

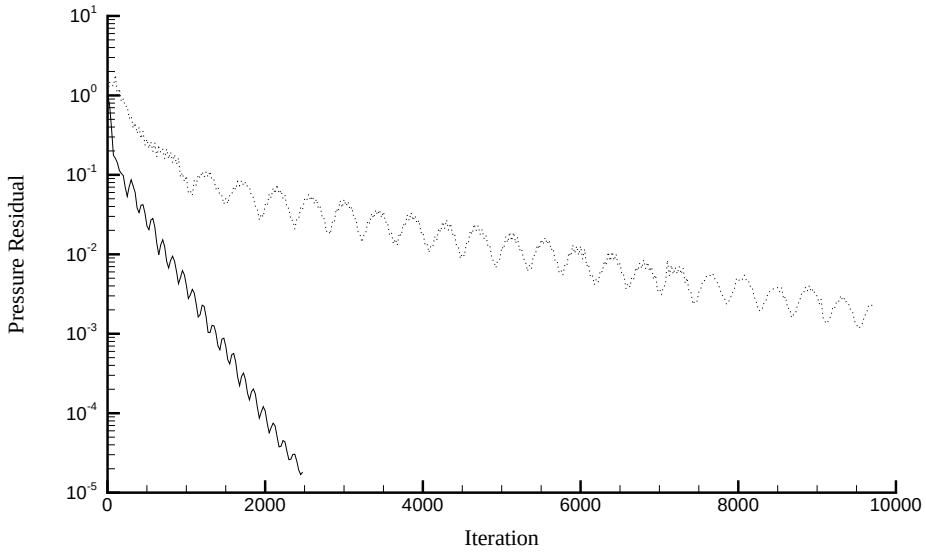

FIGURE 6. Convergence history for structured grid (grey) and a locally refined grid (black).

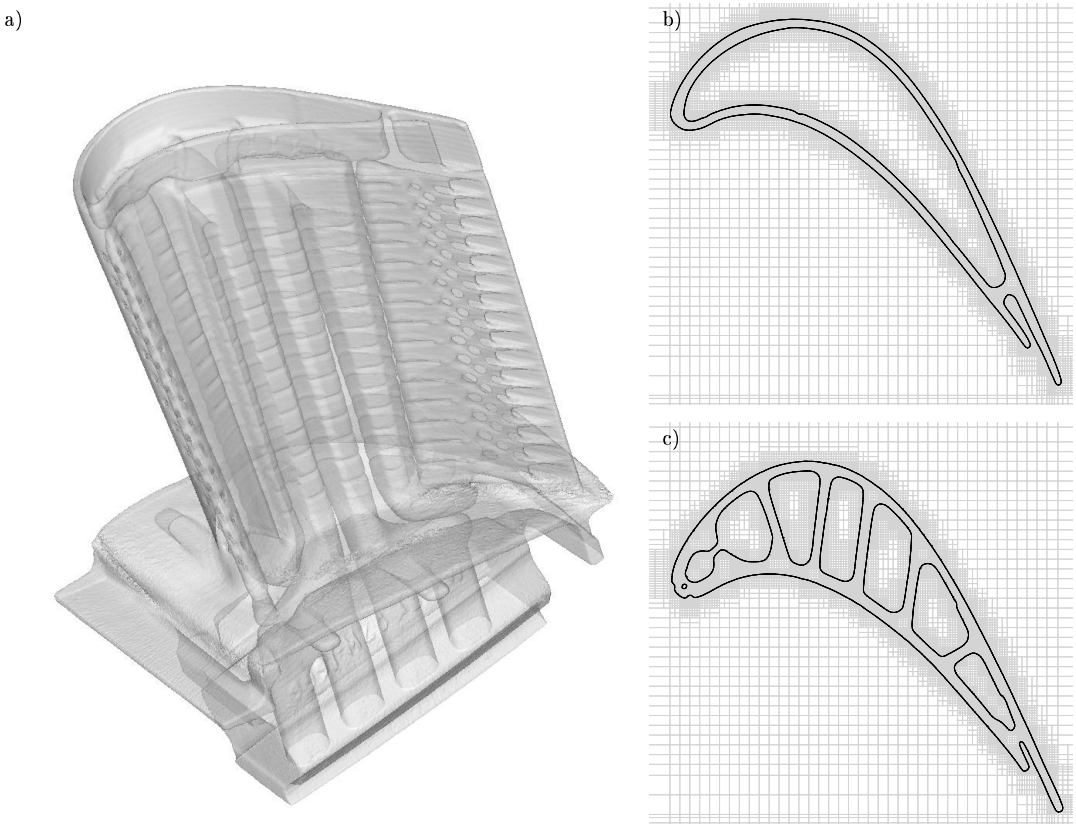

FIGURE 7. Turbine blade with internal cooling passages (a) and planar cuts through locally refined grid: (b) cut near the tip, (c) cut near mid-span

The local grid refinement algorithm has been implemented in three-dimensions and it works well even with very complex geometries as the real turbine blade shown in Fig. 7. The blade is twisted and the internal cooling system of the blade consist of extremely intricate passages with multiple serpentines, ribs, tip ejection holes, film cooling holes, leading edge impingement, etc. Planar cuts through the locally refined grid are shown in the same figure. A comparison between both cuts reveal a remarkable consistent refinement in all internal flow regions. The local grid refinement also follows the twisted outer surface of the blade.

5. Conclusions and future work

The paper presents details of a Cartesian immersed boundary method for RANS flow simulations. It focuses on the IB implementation for mean flow and turbulence variables. The immersed boundary is represented with an implicit, linear, off wall interpolation with an explicit, non-linear correction. An extension of the method with adaptive wall functions is presented for grid aligned cases. The paper also addresses the issue of flow resolution at the immersed boundary and discusses a local grid refinement algorithm.

Skin friction distributions are presented for a boundary layer over a flat plate that is non-aligned with the gridlines at $Re = 1,000$. The results are nearly independent of

the angle between the plate and the gridlines. For the grid aligned case, skin friction is presented for a flat plate boundary layer at $Re = 10^6$.

Future work consists in continuing the development of IBRANS and combining the described technologies into one mature computational code. The code is in the process of being extended for conjugate heat transfer computations. The energy transport equation has been implemented and it needs the development of appropriate heat transfer immersed boundary conditions. The adaptive wall functions for immersed boundaries need to be extended for general grids and heat transfer problems. It is planned to parallelize and adapt the multigrid procedure for the flow solver with local grid refinement.

6. Acknowledgment

The authors would like to acknowledge Gorazd Medic (Stanford University) for his contribution to the adaptive wall function formulation for boundary fitted grids.

REFERENCES

Berger M. & Aftosmis M., 1998 Aspects (and aspect ratios) of Cartesian mesh methods. 16*th Int. Conf. on Numerical Methods in Fluid Dynamics*.

Durbin, P. A. & Iaccarino, G., 2002 An Approach to Local Refinement of Structured Grids. *J. of Comput. Phy*, **181**, 639–653.

Ferziger, J. H. & Peric, M., 2002 Computational Methods for Fluid Dynamics. *Springer-Verlag*, third-edition.

Franke, R., 1982 Scattered data interpolation: Tests of some methods. *Math. Comput.*, **38**, 181–200.

Iaccarino, G. & Verzicco, R., 2003 Immersed Boundary Technique for Turbulent Flow Simulations. *Appl. Mech. Rev.*, **56**, 331–347.

Ham, F. E., Lien, F. S., & Strong, A. B., 2002 A Cartesian grid method with transient anisotropic adaptation. *J. Comput. Phys.* **179**, 469-494.

Kalitzin, G., Medic, G., & Iaccarino, G., 2003 On wall functions for RANS. In preparation.

Kalitzin, G. & Iaccarino, G., 2002 Turbulence Modeling in an Immersed Boundary RANS Method. *Annual Research Briefs*, Center for Turbulence Research, 415–426.

Majumdar, S., Iaccarino, G. & Durbin, P. A., 2001 RANS solver with adaptive structured boundary non-conforming grids. *Annual Research Briefs*, Center for Turbulence Research, 353–366.

Spalart, P. R. & Allmaras, S. R., 1994 A One-Equation Turbulence Model for Aerodynamic Flows. *La Recherche Aerospatiale*, **1**, 1–23.

Stone, H. L., 1968 Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM J. Numer. Anal.*, **5**, 530–558.

Van den Vorst, H. A., 1992 BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **13**, 631–644.

Verzicco, R., Mohd-Yusof, J., Orlandi, P. & Haworth D., 2000 LES in complex geometries using boundary body forces. *AIAA J.*, **38**, 427–433.