V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

A LEVEL-SET BASED CUT-CELL METHOD FOR FLOWS WITH COMPLEX MOVING BOUNDARIES

Claudia Günther*, Daniel Hartmann, Matthias Meinke and Wolfgang Schröder

* Institute of Aerodynamics (AIA), RWTH Aachen University
Wüllnerstrae 5a, 52062 Aachen, Germany
e-mail: office@aia.rwth-aachen.de

Key words: Cartesian grid methods, cut-cell method, internal flow, complex geometry, moving boundaries

Abstract. *An extension of a strictly conservative Cartesian cut-cell method to meet the requirements for the simulation of internal flows in complex geometries including moving boundaries is presented. In the current method, the representation of boundaries which do not coincide with grid lines is done by assigning one ghost cell freely positioned in space to each cell cut by the geometry. A generalization of this approach providing multiple ghost cells for each cell is introduced such that the method is applicable to problems including technical objects with sharp concave features where cells multiply cut by the geometry can occur. The ability of this generalization to greatly simplify the application of robust boundary conditions to internal flows including cells with non-unique boundary conditions is demonstrated. An extension of this method to problems with moving boundaries represented by the zero-contour of an advected level-set function is outlined. Numerical results for the flow past a cube and in a pipe elbow show the correct modification of the flow field in the presence of sharp edges and the effective application to internal flows featuring interfaces with different boundary conditions.*

1 INTRODUCTION

To improve the performance of internal combustion engines, a detailed understanding of the in-cylinder flow field and mixing processes is crucial. Therefore, three-dimensional simulations of reciprocating engines are an important tool in engine development both in science and industry. This application puts certain special demands on a simulation method due to the complexity of the geometry and the presence of moving components such as piston and valves. A suitable method must allow simple and efficient mesh generation and update with respect to the moving parts of the boundary while providing a highly accurate representation of the geometrical details.

Conventional body-adapted grids usually require a considerable meshing effort and need to be updated after each time step in the presence of moving boundaries. In contrast, immersed boundary methods based on non-body-adapted grids allow automatic mesh generation and moving boundaries can be treated without remeshing. Recently, several methods using such non-boundary-conforming grids have been developed and reported in literature[1-4]. However, there exist only few examples for immersed boundary methods applied to compressible flow. A novel approach which allows the computation of three-dimensional viscous compressible flows in a strictly conservative way has been reported by Hartmann et al.[5-7]

Motivated by the application to the flow in an internal combustion engine, this work describes an extension of this method for flows bounded by highly complex geometries. Concave, especially acute-angled features of technical objects can lead to cells multiply cut by the geometry while internal flows exhibit interfaces with different adjacent physical boundary conditions. A method which is not able to resolve these specifics consistently is thus restricted to rather smooth geometries and will possess difficulties for cells with non-unique boundary conditions. The application of boundary conditions in the method of Hartmann et al. in the framework of Cartesian cut cells is done using ghost cells freely positioned in space. Exactly one ghost cell is assigned to each cell intersected by the geometry, representing one boundary surface per cut cell. The method is thus limited as stated above. To overcome these limitations multiple ghost cells for each cut cell, each accounting for a separate intersection surface and the respective boundary condition, are introduced. To enable the application to in-cylinder flow and other problems involving moving boundaries, the method has been coupled to a multi-purpose level-set solver. The moving surface is then represented by the zero-contour of a signed distance function. This approach, recently adopted by several authors[8-10], enables efficient parallelization and allows quick recomputation of the Cartesian cut cells necessary after each time step.

The outline of this paper is as follows: Section 2 starts with the governing equations while the developed numerical method is described in Section 3. Numerical Results are presented in Section 4 and final conclusions are drawn in Section 5.

2 GOVERNING EQUATIONS

The governing equations considered are the unsteady nondimensionalized Navier-Stokes equations for compressible fluids given by

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \overline{\mathbf{H}} = \mathbf{0}, \tag{1}$$

with $\mathbf{Q} = [\varrho, \varrho\mathbf{v}, \varrho\mathbf{E}]^T$ being the vector of the conservative variables and $\overline{\mathbf{H}} = [\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3]^T$ being the flux vector containing an inviscid part $\overline{\mathbf{H}}^i$ and a viscous part $\overline{\mathbf{H}}^v$

$$\overline{\mathbf{H}} = \overline{\mathbf{H}}^i + \overline{\mathbf{H}}^v = \begin{pmatrix} \varrho\mathbf{v} \\ \varrho\mathbf{v}\mathbf{v} \\ \varrho\mathbf{v}(\varrho E + p) \end{pmatrix} + \frac{1}{Re} \begin{pmatrix} 0 \\ \overline{\boldsymbol{\tau}} \\ \overline{\boldsymbol{\tau}}\mathbf{v} + \mathbf{q} \end{pmatrix}. \tag{2}$$

The Reynolds number based on the freestream velocity v_∞ and the characteristic length l reads $Re = \frac{\varrho_\infty v_\infty l}{\mu_\infty}$. Assuming a Newtonian fluid and zero bulk viscosity leads to the following formulation for the second-rank stress tensor $\overline{\tau}$

$$\overline{\boldsymbol{\tau}} = -2\mu\overline{\mathbf{S}} + \frac{2}{3}\mu(\nabla \cdot \mathbf{v})\bar{\mathbf{I}}. \tag{3}$$

Here, $\bar{\mathbf{I}}$ is the unit tensor and $\overline{\mathbf{S}}$ is the rate-of-strain tensor

$$\overline{\mathbf{S}} = \frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right). \tag{4}$$

Using Fourier's law and the Prandtl number $Pr = \frac{\mu_\infty c_p}{k_\infty}$ containing the specific heat at constant pressure c_p, the vector of heat conduction $\mathbf{q}$ can be written

$$\mathbf{q} = -\frac{k}{Pr(\gamma - 1)}\nabla T, \tag{5}$$

where γ is the ratio of specific heats. The thermal conductivity is evaluated from $k(T) = \mu(T)$, which holds for a constant Prandtl number.

Using the Gauss divergence theorem, the integral form of the Navier-Stokes equations is obtained as

$$\int_V \frac{\partial \mathbf{Q}}{\partial t} dV + \int_A \overline{\mathbf{H}}\,\mathbf{n}\, dA = \mathbf{0}, \tag{6}$$

where $\mathbf{n}$ is the outward unit normal vector on the surface dA.

3 NUMERICAL METHOD

3.1 Discretization

The time-integration of the Navier-Stokes equations (6) is done using a 5-stage second-order accurate Runge-Kutta scheme optimized for stability,

$$\begin{aligned} \mathbf{Q}^{(0)} &= \mathbf{Q}^w \\ \mathbf{Q}^{(k)} &= \mathbf{Q}^{(0)} - \alpha_k \Delta t L(\mathbf{Q}^{(k-1)}) \\ \mathbf{Q}^{w+1} &= \mathbf{Q}^{(5)}, \end{aligned} \tag{7}$$

with the coefficients $\alpha = \left(\frac{1}{4}, \frac{1}{6}, \frac{3}{8}, \frac{1}{2}, 1\right)$. The quantity $L(\mathbf{Q})$ denotes the numerical approximation of the term $\nabla\overline{\mathbf{H}}$ in equation (1). The superscript k indicates the Runge-Kutta stage, while the superscript w represents the time step Δt. With $a^n = \sqrt{\gamma \frac{p^n}{\varrho^n}}$ being the local speed of sound and C being the CFL number, the time step is computed globally by

$$\Delta t = \min_{n \in \mathcal{D}} \min_{j \in \{1,2,3\}} \left\{ \left(\frac{C h^n}{|v_j^n| + a^n} \right) \right\}. \tag{8}$$

For the evaluation of the inviscid flux $\overline{\mathbf{H}}^i(\mathbf{x}_s^m)$, the left and right interpolated states L and R in the surface centroid are computed at second-order accuracy using a monotone upstream centered schemes for conservation laws (MUSCL) approach for the primitive variables[11]

$$\mathbf{P}\left(\mathbf{x}_s^m\right)^{L/R} = \mathbf{P}\left(\mathbf{x}^{L/R}\right) + \nabla\mathbf{P}|^{L/R}\left(\mathbf{x}_s^m - \mathbf{x}^{L/R}\right). \tag{9}$$

The inviscid flux is computed using a modified version[12] of the advection upstream splitting method (AUSM)[13]. The cell center gradients $\nabla\mathbf{P}$ are computed using an unweighted linear least-squares reconstruction scheme as described in Hartmann et al.[7] These gradients are also used for the computation of the viscous fluxes $\overline{\mathbf{H}}^v$.

3.2 Embedded Boundary treatment

In Cartesian cut-cell methods, cells are usually classified in three categories. These are internal cells which are located completely inside the computational domain Ω, external or solid cells which are located completely outside the computational domain Ω, and boundary or cut cells which are cut by the boundary $\partial\Omega$. While external cells are discarded in the current approach, boundary cells need a special treatment. As proposed in[5,7], they are reshaped such that the entire control volume V^n of the reshaped cell lies inside the computational domain Ω. The reshaping process is done by first identifying the cut points of a boundary cell where the embedded geometry cuts through the edges of the respective cell. The faces of the reshaped cut cell are then obtained by linearly connecting the cut points on the cell edges of each face cut by the geometry while external faces, situated completely outside the computational domain Ω, are omitted as illustrated in Fig. 1.

The discretization of the integral form (6) of the Navier-Stokes equations in finite-volume methods is based on the evaluation of flux integrals over the cell faces. Therefore, besides the new surface area A^m the shifted centroid $\mathbf{x}_s^m$ of each cut face m has to be computed by

$$\mathbf{x}_s^m = \frac{1}{A^m} \int_{A^m} \mathbf{x}\, dA, \tag{10}$$

and the shifted volumetric cell center $\hat{\mathbf{x}}^n$ of each boundary cell n has to be computed by

$$\hat{\mathbf{x}}^n = \frac{1}{V^n} \int_{V^n} \mathbf{x}\, dV. \tag{11}$$

For a complete description of the boundary cells, the computation of the face area A^Γ, the normal vector $\mathbf{n}^\Gamma$, and the centroid $\mathbf{x}_s^\Gamma$ of the resulting boundary surface Γ which approximates the curved surface is necessary.

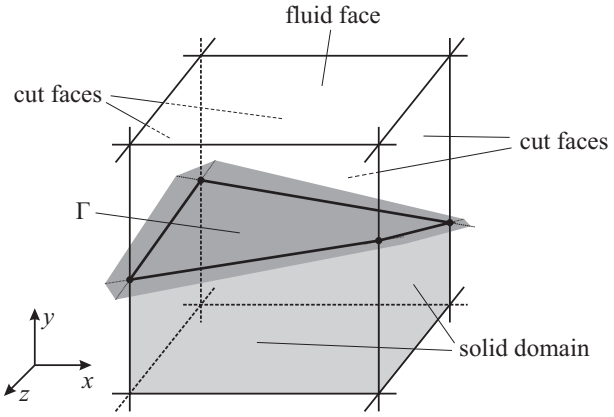


Figure 1: Reshaping of a Cartesian cut cell. The dark shaded surface Γ is the resulting boundary surface, the light shaded solid part of the cell is omitted.

Finally, following Hartmann et al.[6,7], the application of boundary conditions on the boundary surfaces Γ^n is done by prescribing the primitive variables on ghost cells one of which is created for each boundary cell. The ghost cells are connected to their corresponding boundary cell via the boundary surface Γ. The location of the ghost cells is computed on the straight line connecting the boundary cell center and the boundary surface centroid by

$$\tilde{\mathbf{x}^n} = \hat{\mathbf{x}}^n + 2\left(\mathbf{x}_s^{\Gamma^n} - \hat{\mathbf{x}}^n\right). \tag{12}$$

Thus, it is not restricted by the Cartesian grid lines. Dirichlet boundary conditions $\phi(\mathbf{x}, t) = \Phi(\mathbf{x}, t)$ for a generic quantity ϕ can be imposed by setting the ghost cell value $\tilde{\phi^n}$

$$\tilde{\phi^n} = 2\Phi^{\Gamma^n} - \phi^n. \tag{13}$$

This approach ensures that linear interpolation between the ghost cell and the boundary cell n recovers the prescribed value in the boundary surface centroid. A detailed discussion of the application of suitable Neumann boundary conditions and the computation of the wall shear stress on the embedded boundary can be found in Hartmann et al.[6,7] The ghost cells are used for the calculation of the surface flux through the boundary surfaces Γ as well as the gradient computation on the corresponding boundary cells using a least squares reconstruction.

3.2.1 Problem statement

Cells multiply cut by the geometry The treatment of the cut cells in the original method[6,7] as stated above requires that the geometric intersection of a boundary cell can

be described by exactly one surface. Therefore, only one ghost cell is assigned to each boundary cell. For relatively smooth geometries, this is sufficient and can still be ensured for rather coarse grid resolutions in the vicinity of the embedded boundary. However, if technical applications are considered, the embedded boundaries usually contain sharp edges. Their presence, especially in conjunction with convex features and acute angles, can lead to cells multiply cut by the geometry, where two or more cut surfaces are needed to describe the boundary cell resulting from the geometric intersection. Fig. 2 gives an example for such a cell occurring in the Cartesian grid generated for an internal combustion engine simulation.

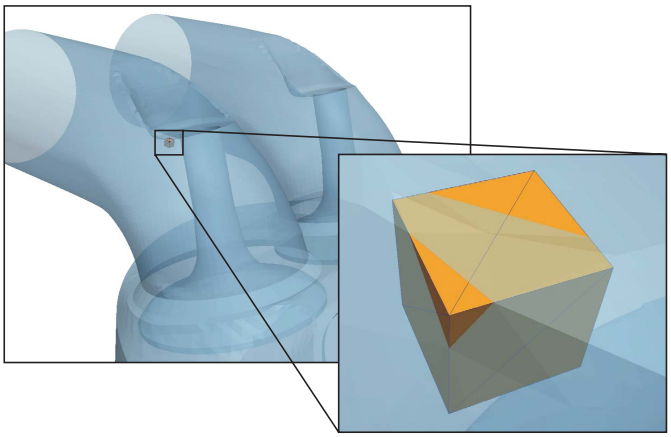


Figure 2: Cell multiply cut by the geometry due to a sharp edged indentation in the intake port of an internal combustion engine.

A possible remedy to this problem is the further refinement of problematic cells which can not be described by a single cut surface. However, there exist certain bizarre cases where further refinement does not eliminate the problematic cells or many refinement steps would be needed to resolve the problem as illustrated in Fig. 3. Thus, a general success of this approach can not be guaranteed. Furthermore, the extra refinement of cells which already have an adequate size from a physical point of view leads to an increased problem size and a reduced time step. Both effects increase computational costs without increasing the overall solution quality. As any user intervention should be avoided by all means to keep the benefits of automated mesh generation, especially regarding the treatment of moving boundaries, a more general representation of geometric intersections which allows to handle multiple boundary surfaces per boundary cell is required.

Internal flows When applied to internal flows, the method of Hartmann et al.[6,7] exhibits another difficulty which underlines the need for a more powerful boundary cell description. Internal flows usually contain interfaces with different adjacent physical boundary conditions where the wall of the bounding geometry and the inflow/outflow planes meet. Applying the original approach of Hartmann et al.[5,7] to a cell cut by such

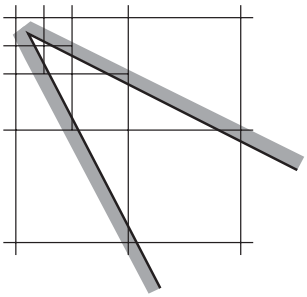


Figure 3: Configuration where many refinement steps are necessary to eliminate all cells multiply cut by the geometry. Each refinement step leads to a new problematic cell on the higher refinement level.

an interface, usually generates a single boundary surface which is not necessary planar, as it can be defined by four or more cut points (see Fig. 4). The boundary condition which should be applied to this surface is thus not well-defined and both possible choices lead to severe problems as will be discussed in the following.

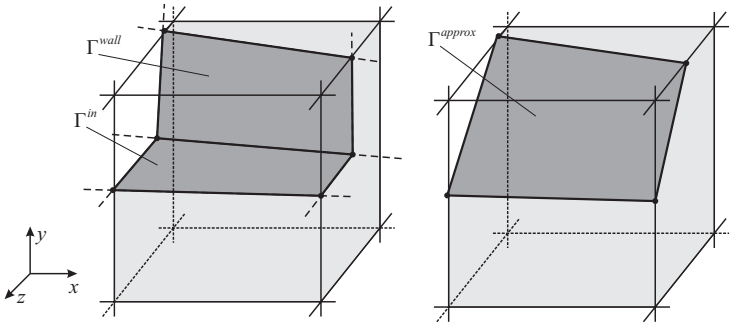


Figure 4: Cell with non-unique boundary conditions due to the approximate boundary surface generation. Left: original configuration; right: approximated boundary surface

If the resulting boundary surface is chosen to represent a wall, this conflicts with the usual assumption of a parallel inflow flow field. Moreover, depending on the geometry and the characteristics of the problem, the flow field can be disturbed significantly by this change of geometry. The assignment of an inflow/outflow boundary condition to the respective boundary surface can lead to even more severe numerical problems, which will be derived in the following.

Hartmann et al.[5,7] approximate the surface integral in Eq. (6) for a cell n by second-order accuracy to obtain

$$\left(\int_A \overline{\mathbf{H}}\mathbf{n}\, dA\right)^n = \sum_{m\in\sigma^n} \overline{\mathbf{H}}(\mathbf{x}_s^m)\mathbf{n}^m A^m + \mathcal{O}(h^n)^2. \tag{14}$$

Hence, the surface integral is replaced by the evaluation of the flux vector in the surface centroid $\overline{\mathbf{H}}(\mathbf{x}_s^m)$. To demonstrate that the surface integral is indeed of second-order accuracy for a cell n bounded by piecewise planar surfaces m let us consider a strictly linear

flux function $\mathbf{g}$ defined by

$$\mathbf{g}(\mathbf{x}) = \mathbf{k} + \overline{\mathbf{B}}\mathbf{x} = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix} + \begin{pmatrix} 0 & b_{12} & b_{13} \\ b_{21} & 0 & b_{23} \\ b_{31} & b_{32} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{15}$$

Furthermore, let us assume a time-independent problem such that using the flux function (15) Eq. (1) reduces to

$$\nabla \cdot \mathbf{g}(\mathbf{x}) = 0. \tag{16}$$

It can be easily verified that this equation holds for the flux defined by (15). Eq. (6) now reads

$$\begin{aligned} 0 &= \int_A \mathbf{g}(\mathbf{x}) \cdot \mathbf{n}\, dA = \sum_{m \in \sigma^n} \int_{\Gamma^m} \mathbf{g}(\mathbf{x}) \cdot \mathbf{n}\, dA \\ &= \sum_{m \in \sigma^n} \int_{\Gamma^m} k_1 n_x + k_2 n_y + k_3 n_z \\ &\quad + b_{12} y n_x + b_{13} z n_x + b_{21} x n_y + b_{23} z n_y + b_{31} x n_z + b_{32} y n_z \, dA. \end{aligned} \tag{17}$$

For piecewise planar surfaces this is equivalent to

$$\begin{aligned} 0 &= \sum_{m \in \sigma^n} \Big[(k_1 n_x^m + k_2 n_y^m + k_3 n_z^m) A^m \\ &\quad + b_{12} n_x^m \int_{\Gamma^m} y\, dA + b_{13} n_x^m \int_{\Gamma^m} z\, dA + \ldots + b_{32} n_z^m \int_{\Gamma^m} y\, dA \Big] \\ &= \sum_{m \in \sigma^n} \big[(k_1 n_x + k_2 n_y + k_3 n_z) A^m \\ &\quad + (b_{12} n_x^m y_s^m + b_{13} n_x^m z_s^m + b_{21} n_y^m x_s^m + b_{23} n_y^m z_s^m + b_{31} n_z^m x_s^m + b_{32} n_z^m y_s^m) A^m \big] \\ &= \sum_{m \in \sigma^n} \mathbf{g}(\mathbf{x}_\mathbf{s}^\mathbf{m}) \cdot \mathbf{n}^m A^m. \end{aligned} \tag{18}$$

This means that the approximation in (14) holds exactly for linear flux functions in this case. Let us now examine the situation displayed in Fig. 5 for a boundary cell. Here, the boundary surface Γ is not planar but can be decomposed into two piecewise planar surfaces Γ^1 and Γ^2. If we describe the cell using both piecewise planar surfaces in the description of the boundary cell n, $\{\Gamma^1, \Gamma^2\} \in \sigma^n$, Eq. (14) still holds exactly for the linear flux vector $\mathbf{g}$. We now try to find expressions for the area, the (substitute) normal vector and the surface centroid of a single surface Γ which can replace the piecewise planar surfaces Γ^1 and Γ^2 such that the surface flux is not changed. Thus, we require that

$$\int_\Gamma \mathbf{g}(\mathbf{x}) \cdot \mathbf{n}\, dA = \int_{\Gamma^1} \mathbf{g}(\mathbf{x}) \cdot \mathbf{n}\, dA + \int_{\Gamma^1} \mathbf{g}(\mathbf{x}) \cdot \mathbf{n}\, dA \tag{19}$$

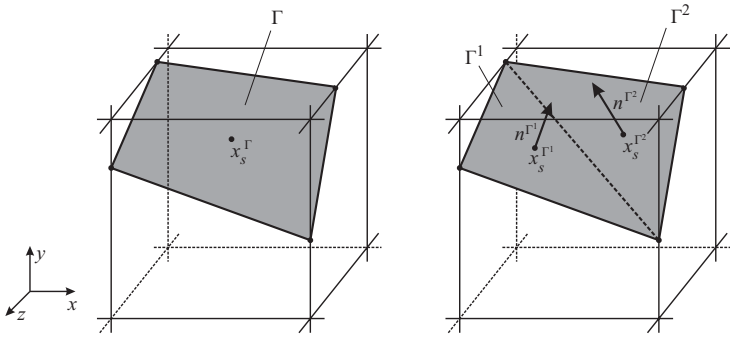


Figure 5: Boundary cell with nonplanar cut surface Γ (left) which can be decomposed into two piecewise planar surfaces Γ^1 and Γ^2 (right).

can be reformulated using approximation (14) such that

$$\mathbf{g}(\mathbf{x}_\mathbf{s}^\mathbf{\Gamma}) \cdot \mathbf{n}^\mathbf{\Gamma} \, A^\Gamma = \mathbf{g}(\mathbf{x}_s^{\Gamma^1}) \cdot \mathbf{n}^{\Gamma^1} \, A^{\Gamma^1} + \mathbf{g}(\mathbf{x}_s^{\Gamma^2}) \cdot \mathbf{n}^{\Gamma^2} \, A^{\Gamma^2}, \tag{20}$$

using suitable values for $\mathbf{x}_s^\Gamma$, $\mathbf{n}^\Gamma$ and A^Γ. Inserting the definition for $\mathbf{g}$ into this equation we get

$$\begin{gathered} k_1 n_x^\Gamma A^\Gamma + \ldots + k_3 n_z^\Gamma A^\Gamma + b_{12} y_s^\Gamma n_x^\Gamma A^\Gamma + \ldots + b_{32} y_s^\Gamma n_z^\Gamma A^\Gamma = \\ k_1 \left(n_x^{\Gamma^1} A^{\Gamma^1} + n_x^{\Gamma^2} A^{\Gamma^2} \right) + \ldots + k_3 \left(n_z^{\Gamma^1} A^{\Gamma^1} + n_z^{\Gamma^2} A^{\Gamma^2} \right) \\ + b_{12} \left(y_s^{\Gamma^1} n_x^{\Gamma^1} A^{\Gamma^1} + y_s^{\Gamma^2} n_x^{\Gamma^2} A^{\Gamma^2} \right) + \ldots + b_{32} \left(y_s^{\Gamma^1} n_z^{\Gamma^1} A^{\Gamma^1} + y_s^{\Gamma^2} n_z^{\Gamma^2} A^{\Gamma^2} \right). \end{gathered}$$

Since we require that this equation should hold for all choices of $k_i \in \mathbb{R}$ and $b_{ij} \in \mathbb{R}$, for $i \neq j$, we see that the resulting expression has to be fulfilled componentwise. First, we use the equations to be fulfilled for the components of the vector $\mathbf{k}$ to get expressions for $\mathbf{n}^\Gamma$ and A^Γ:

$$n_x^\Gamma A^\Gamma = n_x^{\Gamma^1} A^{\Gamma^1} + n_x^{\Gamma^2} A^{\Gamma^2} \tag{21}$$

$$n_y^\Gamma A^\Gamma = n_y^{\Gamma^1} A^{\Gamma^1} + n_y^{\Gamma^2} A^{\Gamma^2} \tag{22}$$

$$n_z^\Gamma A^\Gamma = n_z^{\Gamma^1} A^{\Gamma^1} + n_z^{\Gamma^2} A^{\Gamma^2}. \tag{23}$$

Together with the normalization requirement $\|\mathbf{n}\|_2 = 1$, these equations fully describe the normal vector and the area of the resulting non-planar surface Γ. Let us now turn to the equations for b_{21} and b_{31} as they both give conditional equations for x_s^Γ:

$$x_s^\Gamma n_y^\Gamma A^\Gamma = x_s^{\Gamma^1} n_y^{\Gamma^1} A^{\Gamma^1} + x_s^{\Gamma^2} n_y^{\Gamma^2} A^{\Gamma^2} \tag{24}$$

$$x_s^\Gamma n_z^\Gamma A^\Gamma = x_s^{\Gamma^1} n_z^{\Gamma^1} A^{\Gamma^1} + x_s^{\Gamma^2} n_z^{\Gamma^2} A^{\Gamma^2}. \tag{25}$$

Equations (24) and (25) have to be fulfilled at the same time to ensure that Eq. (20) holds, so the system of conditional equations is overdetermined. Solving both equations

for x_s^Γ and inserting Eq. (22) and (23)

$$\frac{x_s^{\Gamma^1} n_y^{\Gamma^1} A^{\Gamma^1} + x_s^{\Gamma^2} n_y^{\Gamma^2} A^{\Gamma^2}}{n_y^{\Gamma^1} A^{\Gamma^1} + n_y^{\Gamma^2} A^{\Gamma^2}} = \frac{x_s^{\Gamma^1} n_z^{\Gamma^1} A^{\Gamma^1} + x_s^{\Gamma^2} n_z^{\Gamma^2} A^{\Gamma^2}}{n_z^{\Gamma^1} A^{\Gamma^1} + n_z^{\Gamma^2} A^{\Gamma^2}} \tag{26}$$

has to be satisfied for the centroids, normal vectors, and areas of the two planar surfaces. By multiplying with the product of the denominators, Eq. (26) can be transformed to

$$x_s^{\Gamma^2} A^{\Gamma^1} A^{\Gamma^2} n_y^{\Gamma^2} n_z^{\Gamma^1} + x_s^{\Gamma^1} A^{\Gamma^1} A^{\Gamma^2} n_y^{\Gamma^1} n_z^{\Gamma^2} = x_s^{\Gamma^2} A^{\Gamma^1} A^{\Gamma^2} n_y^{\Gamma^1} n_z^{\Gamma^2} + x_s^{\Gamma^1} A^{\Gamma^1} A^{\Gamma^2} n_y^{\Gamma^2} n_z^{\Gamma^1} \tag{27}$$

which finally leads to

$$\left(x_s^{\Gamma^1} - x_s^{\Gamma^2}\right)\left(n_y^{\Gamma^1} n_z^{\Gamma^2} - n_y^{\Gamma^2} n_z^{\Gamma^1}\right) = 0. \tag{28}$$

When we repeat this process for the other two components of the surface centroid, y_s^Γ and z_s^Γ, we get the result that, apart from the meaningless solution $\mathbf{x}_s^{\Gamma^1} = \mathbf{x}_s^{\Gamma^2}$, the only possible constellation that allows to fulfill (20) requires

$$\mathbf{n}^{\Gamma^1} \times \mathbf{n}^{\Gamma^2} = \mathbf{0}. \tag{29}$$

This means, the normal vectors of the two planar surfaces are necessarily collinear which can be precised to identical. Consequently, for a nonplanar surface Γ which can be decomposed into two surfaces with non-identical normal vectors, Eq. (20) can not be fulfilled and thus approximation (14) is reduced to first-order accuracy.

This order reduction negatively influences the stability of the solution method. While it appears that the impact is quite small when such a nonplanar boundary surface represents a solid wall, the application of inflow or outflow boundary conditions to such a boundary surface leads to severe stability issues. In fact, for most configurations, divergence of the solution algorithm could be observed after only few iterations. This is due to the fact that a certain mass flux over an inflow/outflow boundary surface exists, while the mass flux is forced to zero in the case of a solid wall. A first-order approximation error in the first component of Eq. (6), the mass balance, has a much higher impact on solution stability than an error in the other components.

When a more general representation of the boundary cells is applied which allows to handle multiple boundary surfaces per boundary cell, the boundary surface on problematic cells as described above can be represented by piecewise planar surfaces. Each surface can represent the correct physical boundary condition of the respective part of the geometry and consequently, the difficulties explained in this section can be resolved.

3.2.2 Multiple ghost cell formulation

An extension of the method of Hartmann et al.[5-7] which accounts for the specifics of flow bounded by complex geometries as discussed in the previous section is straightforward. We omit the rule that each boundary cell contains exactly one boundary surface with one

ghost cell attached to it. Instead, multiple separate boundary surfaces, each equipped with a particular ghost cell, are allowed to be attached to a cell, see Fig. 6. For practical and memory reasons, the maximum number of separate surfaces allowed should be restricted. As the boundary conditions are implemented purely by computing appropriate values for the variable vector and gradients on the ghost cell related to a boundary surface, the general procedure to apply boundary conditions is not altered by this generalization. If different types of boundary conditions are assigned to two boundary surfaces belonging to one cell, the respective ghost cell values are calculated each by using the proper boundary condition.

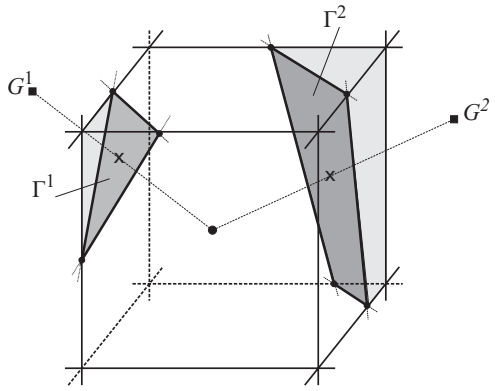


Figure 6: Representation of a boundary cell multiply cut by the geometry in the extended method. One ghost cell G^i is associated to each boundary surface Γ^i.

However, some care must be taken considering the implementation of Neumann boundary conditions. As explained in Hartmann et al.[6,7], Neumann boundary conditions on a cell n are applied using the gradient $\nabla\phi|^n$ in the cell center which is calculated with a least-squares reconstruction including a virtual ghost-point obtained by mirroring the boundary cell center at the boundary surface. The real ghost cell itself is not included in the reconstruction stencil. In the extended method, the reconstruction stencil and thus the reconstruction constants are not unique for one cell but differ depending on the boundary surface, respectively ghost cell, currently processed. For an improved efficiency, the reconstruction constants should therefore be calculated and stored dedicated to each ghost cell prior to the overall solution. Regarding the least-squares reconstruction of the cell-center gradients used for the evaluation of the surface fluxes, no further modifications are necessary apart from the inclusion of all ghost-cells of a boundary cell into its reconstruction stencil.

To avoid problems with numerical instabilities generated by cells of very small volume emerging at the embedded boundaries, Hartmann et al.[5,7] adopt a combined cell-merging/cell-linking approach. This approach merges the volume of a small cell with a neighboring cell of sufficient volume, called the master cell. The volumetric centers of the master and the small cell are thus shifted to the volumetric center of the merged

cell volume. As reported in [5], the surfaces between the original master and slave cells are discarded while the boundary surfaces are merged using the area-weighted average of the boundary surfaces originally created for master and slave cell. The ghost cell of the master cell is shifted according to the new position of the boundary surface centroid while the ghost cell of the small cell is discarded. This approach has to be modified to preserve the abilities of our extension also for boundary cells which get merged during this process. While the volume merging does not conflict with our modifications, the surfaces of both cells have to be retained together with the respective ghost cells. This is equivalent to transferring the boundary surfaces and ghost cells of the small cell to the resulting merged cell which now holds the combined set of boundary surfaces of both master and small cell as illustrated in Fig. 7.

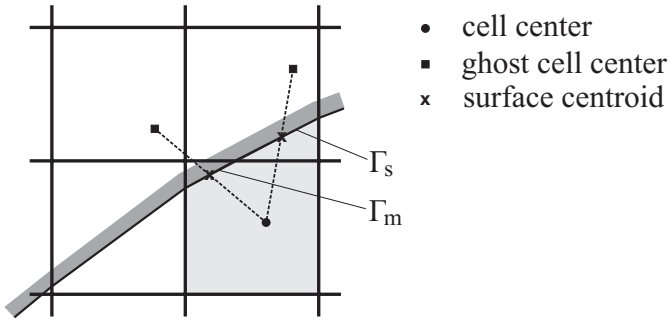


Figure 7: Treatment of small cells in the extended method. The merged cell holds the boundary surfaces of the master cell Γ^m and of the small cell Γ^s

3.2.3 Computation of the cut-cell information

The computation of the reshaped cells cut by the geometry is a nontrivial and relatively expensive task. Therefore, in problems which do not involve moving boundaries, it is usually implemented as a pre-processing step. The generalization of the boundary cell description adds additional complexity. To retain a high efficiency and enable parallelization, a suitable algorithm has to operate locally on a single cell and the result must be consistent for neighboring cells. Therefore, we have implemented a modification of the Marching Cubes algorithm which originates from the field of computer graphics. It is used for the computation a polygonal mesh of an isosurface from a three-dimensional scalar field. This algorithm has first been reported by Lorensen and Cline[14] and several improvements and extensions have been reported since[15-18].

The basic approach of the algorithm is simple. Based on information about the location (inside/outside the fluid region) of the corner vertices of a hexahedral cell, an index can be calculated which identifies the correct configuration for the examined cell out of a list of 256 possible polygon configurations within the cell and, using the cut points on the cell edges, all necessary cut cell information can be generated. The polygon configurations

can be grouped to the 15 base cases shown in Fig. 8(a). As discussed in [18], the set of 15 base cases for the different configurations presented in the original method has to be completed by several variations to guarantee that the resulting isosurface is watertight. Fig. 8(b) exemplifies this for case 6. The classification of a cell regarding a base case and its variations is ambiguous as the same inside/outside index is generated. However, the correct classification can be obtained using additional connectivity information from the description of the embedded geometry. Cases and variations resulting in grid configurations with degenerate least-squares stencils are omitted since for these cases mesh refinement is mandatory. The treatment of so called split cells which are generated if the geometry cuts the fluid volume of a cell into two independent pieces is not covered in the present approach and will thus not be discussed here.

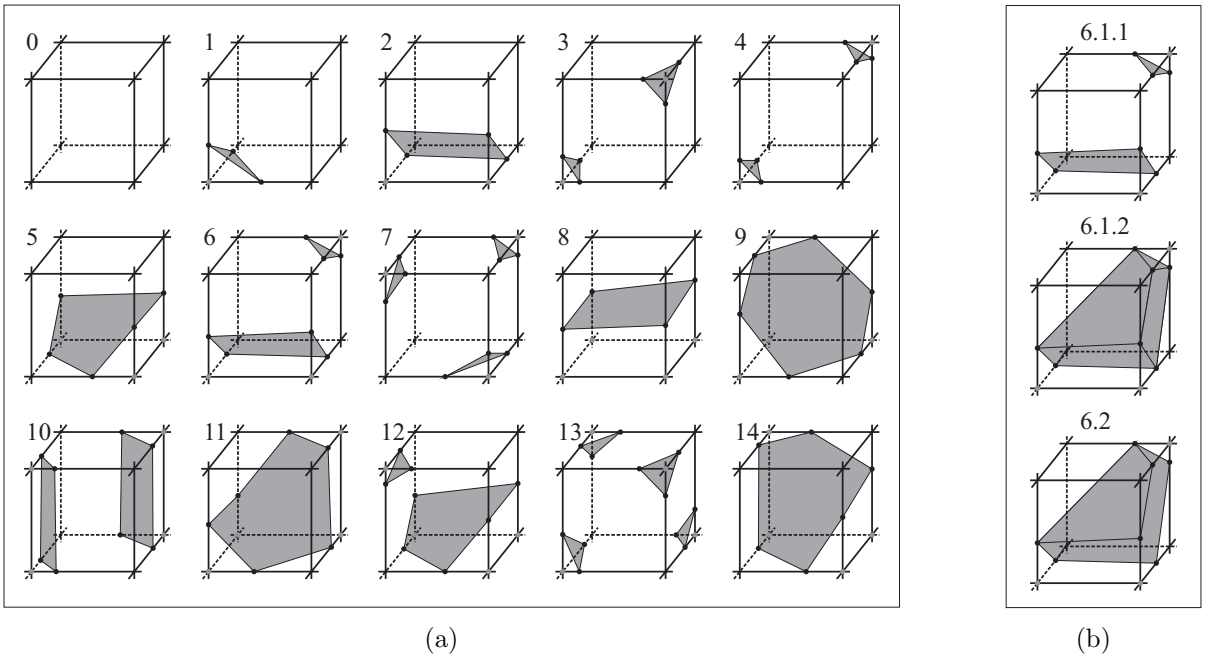


Figure 8: Polygon configurations in the Marching Cubes algorithm depending on the inside/outside states of the 8 corner vertices: (a) 15 base cases; (b) Case 6 and corresponding variations

As discussed in the previous section, cells which contain interfaces with different adjacent boundary conditions have to be treated more carefully as nonplanar boundary faces with inflow conditions have to be avoided. To achieve this goal, we process these cells separately to generate two different boundary surfaces for each of them. This allows to reproduce the original configuration in such cells as displayed in Fig. 4 in the left picture. However, a prerequisite for the success of this method is that the geometrical representation of the inflow boundary is completely planar. Based on the geometry given in our case as a surface triangulation, we can identify the interface between inflow/outflow surface and the solid wall, see Fig. 9. The cut points of this interface with the cell faces are used

to compute the relevant information of both boundary surfaces, the remaining surfaces of the boundary cell and its volume and volumetric center. Depending on the marching cubes case of the boundary cell, this procedure can be handled consistently and efficiently.

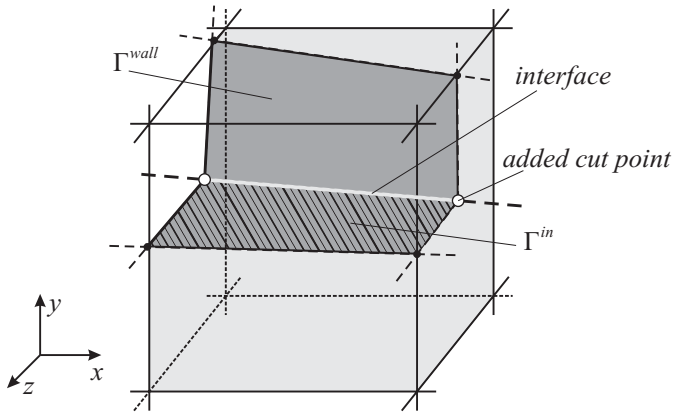


Figure 9: Reconstruction of the original geometrical configuration of boundary cells with non-unique boundary conditions. Cut point are added where the interface between the wall boundary surface Γ^{wall} and the inflow boundary surface Γ^{in} (thick light line) cuts through the cell faces.

3.3 Boundary motion

The present method is well suited for an extension to general three-dimensional moving boundary problems. According to the new boundary position after each time step, the cut-cell approach reduces the problem of adjusting the computational grid to the recomputation of the boundary cells cut by the moving parts of the geometry. A suitable approach for the description of the moving boundary is the representation by a scalar level set function $\phi(\mathbf{x}, t)$, which is specified as a signed distance function with respect to the moving boundary. This means the body outline is given by the zero contour of the level set function $\phi = 0$. Points inside the geometry are indicated by a negative value $\phi < 0$ and cells inside the fluid domain are indicated by a positive value $\phi > 0$. The evolution of ϕ in time using a given spatially constant extension velocity vector $\mathbf{f}(t)$ is then defined by

$$\frac{\partial \phi}{\partial t} + \mathbf{f} \cdot \nabla \phi = 0. \tag{30}$$

The computation of ϕ is carried out using a highly accurate level-set method developed by Hartmann et al.[19-21] coupled to the flow solver in a dual-mesh approach as reported in [22,23].

This methodology integrates well into the previously described modified marching cubes approach used for the generation of the cut-cell information. The cut points on the edges of a boundary cell can be obtained by interpolation of the scalar level set

function ϕ while its sign directly carries the information if a point is located inside or outside the fluid domain. Thus, the computation of the marching cubes index, which is quite expensive for a geometry given by a surface triangulation, is reduced to a simple sign evaluation. In addition, efficient parallelization is straightforward as non-localized operations such as the inside/outside state determination for a surface triangulation are completely avoided.

4 NUMERICAL RESULTS

4.1 Flow past a cube

To demonstrate the ability of the modified method to capture the flow field in the vicinity of sharp edges accurately, we consider the flow past a cube. The extended method preserves the sharp edges in their real position, while the original method bevels the edges of the cube, thus altering the flow field. Therefore, this test case is suitable to demonstrate the correct modification of the flow field when multiple ghost cells are attached to one cell. With L being the edge length of the cube the Reynolds number is defined $Re = \frac{\varrho_\infty v_\infty L}{\mu_\infty}$ based on the freestream values. A patchwise refined grid is used to discretize the computational domain $\Omega : [-5L, 10L] \times [-5L, 5L] \times [-5L, 5L]$, where the midpoint of the cube is located at $(x, y, z) = (0, 0, 0)$. The x-direction is the streamwise direction. A relatively coarse grid with a mesh width $h \approx 0.05L$ in the close vicinity of the cube has been used to increase the visible effect of the surface modification. We shall investigate the flow at Mach number $M = 0.1$ and a Reynolds number $Re = 100$, where the flow field is steady and symmetric. Flow separation at the leading edge occurs without any reattachments at the side walls[24,25].

Fig. 10 shows the velocity magnitude in the y-symmetry plane. The upper part of the picture shows the solution with the modified method, while the solution with the original method of Hartmann et al.[6,7] is depicted below the x-axis. The beveling of the leading

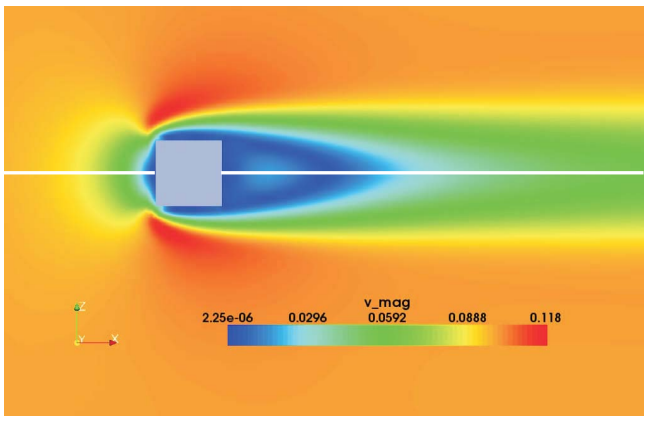


Figure 10: Velocity magnitude of the uniform flow past a cube (y-symmetry plane). Upper half: solution with the modified approach (sharp edge); lower half: solution with the original approach (beveled edge).

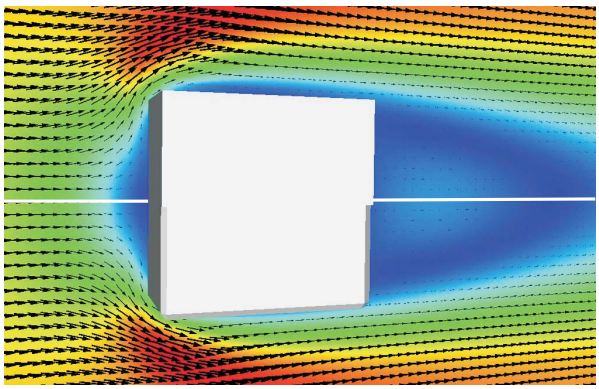


Figure 11: Arrow representation of the flow field (y-symmetry plane). Upper half: solution with the modified approach (sharp edge); lower half: solution with the original approach (beveled edge).

edge results in a smaller and narrower separation region while the modified method is able to capture the precise separation point and thus produces the correct bigger separation region. In Fig. 11, the same split view is applied for an arrow illustration of the flow direction. The upper part of the picture gives the solution for the sharp-edged cube with our generalized approach while the original solution together with the beveled cube is shown in the lower part. In both cases, the flow field follows the outline of the respective cube representation neatly. This exemplifies the ability of our method to modify the flow field appropriately in the presence of highly complex geometries.

4.2 Flow in a pipe elbow

The simulation of the flow in a pipe elbow shall be used to show the effect of our modification when applied to cells with non-unique boundary conditions occurring in internal flow simulations. A grid being refined near the wall to a minimum mesh width $h \approx 0.014D$ is used to discretize the computational domain. The pipe extends approximately $L = 4D$ in the y- and z-direction of the bend, where D is the pipe diameter. The Reynolds number used for the simulation is $Re = \frac{\varrho_\infty v_\infty D}{\mu_\infty} = 500$ and the Mach number is set to $M = 0.1$. The flow field is initialized with zero flow velocity. At time $t = 0$, a lower pressure is applied to the outflow boundary normal to the z-axis. Homogeneous Neumann boundary conditions are used for the flow velocities and the density on the outflow boundary. On the inflow boundary normal to the y-axis, the normal velocity is prescribed using a parabolic profile according to the instantaneous mass flux. Pressure and density are calculated to match the mass flux. Fig. 12 shows the situation after $t = 5.6\Delta t$ time units for the y-component of the velocity. Due to the representation of the geometry by piecewise planar surfaces for cells with non-unique boundary conditions in the inlet and outlet cross sections, no instabilities can be observed close to the respective cross sections. This is even more obvious when Fig. 13 is considered, where the normal components of velocity in the inlet and outlet cross sections are depicted.

Contrary, if the original method is applied nonplanar surfaces inlet/outlet boundary

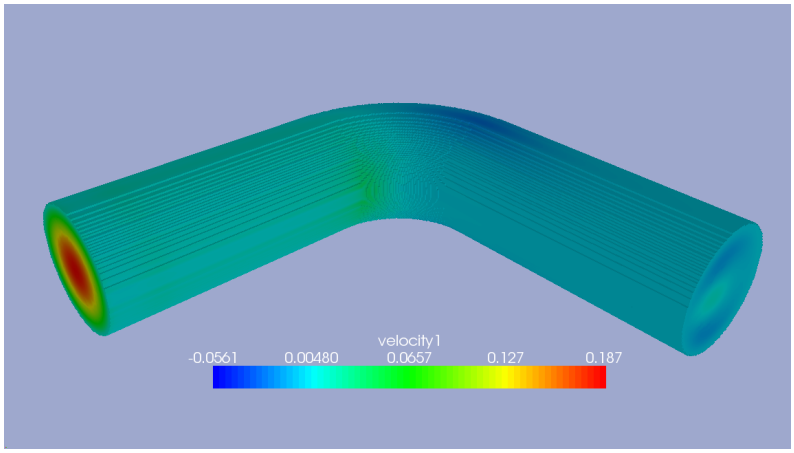
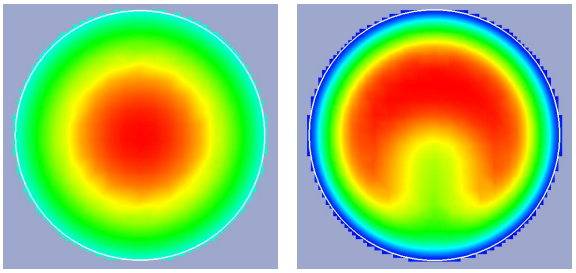


Figure 12: Laminar flow in a pipe elbow: velocity component normal to the inflow boundary.



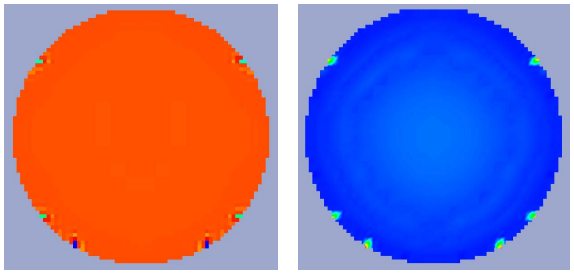
(a) (b)

Figure 13: Normal component of the velocity: (a) inlet cross section, (b) outlet cross section.

surfaces occur and the solution process diverges after few iterations. In Fig. 14, the pressure and the normal velocity in the outlet cross section after $t = 0.06\Delta t$ is shown. Nonphysical perturbations due to highly twisted boundary surfaces can be clearly detected which effect the instability of the solution method.

5 CONCLUSIONS

An extension to an existing strictly conservative Cartesian cut-cell method for general two- and three-dimensional problems of viscous flow was presented. In this method, embedded boundaries which do not coincide with the grid lines are represented by assigning multiple ghost cells freely positioned in space to each cut cell. This makes the extended method applicable to problems including highly complex geometries with sharp concave features. Furthermore, it was shown that the application of stable boundary conditions for internal flows including cells with non-unique boundary conditions is greatly simplified. A strategy for the application of this method to moving boundary problems using the level-set method has been outlined. The success of the extended method is shown by means of the simulation of the flow past a cube and in a pipe elbow.



(a) (b)

Figure 14: Nonphysical perturbations in the outlet cross section due to nonplanar cut surfaces: (a) pressure, (b) normal velocity.

ACKNOWLEDGMENTS

This research was performed as a part of the Cluster of Excellence "Taylor-Made Fuels from Biomass", which is funded by the German Excellence Initiative. The support is gratefully acknowledged.

REFERENCES

[1] D. Causon, D. Ingram, C. Mingham, G. Yang, R. Pearson, Calculations of shallow water flows using a Cartesian cut cell approach, *Adv. Water Res.*, **23**, 545 - 562 (2000)

[2] S. Cieslak, S. Ben Khelil, I. Choquet, A. Merlen, Cut cell strategy for 3-D blast waves numerical simulations, *Shock Waves*, **10**, 421-429 (2001)

[3] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, *J. Comput. Phys.*, **227**, 4825-4852 (2008)

[4] M. Barad, P. Colella, S. Schladow, An adaptive cut-cell method for environmental fluid mechanics, *Int. J. Numer. Meth. Fluids*, **60**, 473-514 (2009)

[5] D. Hartmann, M. Meinke and W. Schröder, An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods, *Comput. Fluids*, **37**, 1103-1125 (2008)

[6] D. Hartmann, M. Meinke and W. Schröder, A general formulation of boundary conditions on Cartesian cut cells for compressible viscous flows, *AIAA Paper* **2009-3878**, (2009)

[7] D. Hartmann, M. Meinke and W. Schröder, A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids, *Comput. Meth. Appl. Mech. Eng.*, **(submitted)**

[8] S. Marella, S. Krishnan, H. Liu, H. Udaykumar, Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations, *J. Comput. Phys.*, **210**, 1-31 (2005)

[9] R. Yang, F. Stern, Sharp interface immersed-boundary/level-set method for wave-body interactions, *J. Comput. Phys*, **228**, 6590-6616 (2009)

[10] Y. Cheny, O. Botella, The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties, *J. Comput. Phys.*, **229**, 1043-1076 (2010)

[11] B. van Leer, Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method, *J. Comput. Phys.*, **32**, 101-136 (1979)

[12] M. Meinke, W. Schröder, E. Krause, T. Rister, A comparison of second- and sixth-order methods for large-eddy simulation, *Comput. Fluids*, **31**, 695-718 (2002)

[13] M.-S. Liou, C. J. Steffen Jr., A new flux splitting scheme, *J. Comput. Phys.*, **107**, 23-39 (1993)

[14] W. Lorensen, H. Cline, Marching Cubes: A high resolution 3D surface construction algorithm, *Computer Graphics*, **21**, 163-169 (1987)

[15] G. Nielson, B. Hamann, The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes, *Proceedings of Visualization 91*, 29-38 (1991)

[16] C. Montani, R. Scateni, R. Scopigno, A modified lookup table for implicit disambiguation of marching cubes, *Visual Comput.*, **10**, 353-355 (1994)

[17] E. Chernyaev, Marching Cubes 33: Construction of Topologically Correct Isosurfaces, *Technical Report CERN*, **CN 95-17**,(1995)

[18] T. Lewiner, H. Lopes, A. Wilson Vieira, G. Tavares, Efficient implementation of Marching Cubes' cases with topological guarantees, *J. Graphics Tools*, **8**, 1-15 (2003)

[19] D. Hartmann, M. Meinke and W. Schröder, Differential equation based constrained reinitialization for level set methods, *J. Comput. Phys.*, **227**, 6821-6845 (2008)

[20] D. Hartmann, M. Meinke and W. Schröder, On accuracy and efficiency of constrained reinitialization, *Int. J. Numer. Meth. Fluids*, DOI:10.1002/fld.2135 (2009)

[21] D. Hartmann, M. Meinke and W. Schröder, The constrained reinitialization equation for level set methods, *J. Comput. Phys.*, **229** 1514-1535 (2010)

[22] D. Hartmann, M. Meinke and W. Schröder, An adaptive dual-mesh method for premixed combustion using the level set approach, AIAA Paper 2010-1068 (2010)

[23] D. Hartmann, M. Meinke and W. Schröder, A level-set based adaptive-grid method for premixed combustion, *Combust. Flame*, **(submitted)**

[24] R. Raul, P. Bernard, An application of the vorticity-vector potential method to laminar cube flow, *In. J. Numer. Meth. Fluids*, **10**, 875-888 (1990)

[25] A. Saha, Three-dimensional numerical simulations of the transition of flow past a cube, *Phys. Fluids*, **16**, 1630-1646 (2004)