DOI: https://doi.org/10.14311/TPFM.2023.015

ON REYNOLDS-AVERAGED TURBULENCE MODELING WITH IMMERSED BOUNDARY METHOD

L. Kubíčková [1, 2], M. Isoz [1, 2]

[1] Czech Academy of Sciences, Institute of Thermomechanics, Dolejškova 5, Prague 182 00, Czech Republic

[2] Department of Mathematics, Faculty of Chemical Engineering, University of Chemistry and Technology, Technická 5, Prague 166 28, Czech Republic

Abstract

The immersed boundary (IB) method is an approach in the computational fluid dynamics in which complex geometry conforming meshes are replaced by simple ones and the true simulated geometry is projected onto the simple mesh by a scalar field and adjustment of governing equations. Such an approach is particularly advantageous in topology optimizations (TO) where it allows for substantial speed-up since a single mesh can be used for all the tested topologies. In our previous work, we linked our custom IB variant, the hybrid fictitious domain-immersed boundary method (HFDIB), with a TO framework and successfully carried out an optimization under laminar flow conditions. However, to allow for optimizations of real-life components, the IB approach needs to be coupled with an affordable turbulence modeling. In this contribution, we focus on extending the HFDIB approach by the possibility to perform Reynolds-averaged simulations (RAS). In particular, we implemented the $k-\omega$ turbulence model and wall functions for closure variables and velocity.

Keywords: immersed boundary, RAS, wall functions, CFD, OpenFOAM

1 Introduction

With the growing available computational power, engineers more and more often utilize computational fluid dynamics (CFD) to design and test new components and devices. It is particularly advantageous to use CFD in topology optimizations (TO) where it can be easily connected with various optimization algorithms [1]. However, the speed, accuracy and stability of every CFD simulation relies on the quality of the computational mesh and mesh generation (meshing) is considered to be one of the biggest bottlenecks of CFD [2].

A way to circumvent the mesh-related difficulties is to use the immersed boundary (IB) method where the component geometry is not represented by the mesh conformation. Instead, a simple mesh is used and the geometry is projected onto it using a scalar field (λ) based on which the flow governing equations are adjusted, see, e.g., [3]. Utilization of a simple mesh significantly reduces the time spent on meshing and eliminates mesh quality related problems. Furthermore, usage of an IB method in TO allows for a substantial speed-up since the mesh can be generated only once and changes in topology are represented by changes in the λ field [1, 4].

Still, the vast majority of IB applications reported in the literature is performed in such flow regimes that the flow boundary layer is resolved [5]. Nevertheless, optimizations of components in real-life conditions requires the IB method to be compatible with affordable turbulence modeling approaches, e.g., Reynolds-averaged simulations (RAS), in which the boundary layer is commonly modeled via wall functions. So far, there were several attempts on coupling of an IB method with RAS [2, 5, 6, 7] showing acceptable accuracy yet not sufficient robustness for utilization in automated topology optimization.

In this work, we focus on development of a robust and general implementation of the RAS approach into our custom IB method variant, the hybrid fictitious immersed boundary method (HFDIB), see [3]. In Kubíčková and Isoz [8], we have already linked the HFDIB method to a TO framework to conduct a laminar TO. With the newly proposed HFDIB-RAS approach, we aim to perform TO under turbulent flow conditions. In the present state, the HFDIB-RAS approach is a combination of the HFDIB method, the k-ω turbulence model and wall functions for the closure variables (k and ω) and for velocity. The implementation is done in OpenFOAM [9] and the framework is general and allows for extension by any one or two equation RAS model. Finally, the method behavior is presented on several verification tests and the results show a qualitative agreement with standard geometry-conforming simulations.

2 Methods

Let $\Omega \subset \mathbb{R}^3$ be a finite volume discretization of an open, bounded, and connected computational domain, see Fig. 1-a). Based on the simulated component geometry, we can divide Ω into three subdomains

$$\Omega = \Omega_\mathrm{s} \cup \Omega_\mathrm{f} \cup \Omega_\mathrm{sf} \tag{1}$$

where Ω_s represents a part of the domain fully immersed in solid, Ω_f is a part of Ω immersed in fluid and Ω_sf comprises cells that are intersected by the fluid-solid interface, see Fig. 1-b). In the HFDIB-RAS approach, the division in (1) is represented by a scalar field λ, see Fig. 1-c). In each cell $\Omega_P \subset \Omega$, the λ field is defined as

$$\lambda = \begin{cases} 0 & \text{if } \Omega_P \subset \Omega_\mathrm{f} \\ 1 & \text{if } \Omega_P \subset \Omega_\mathrm{s} \\ \tilde{\lambda} \in (0,1) & \text{if } \Omega_P \subset \Omega_\mathrm{sf} \end{cases} , \quad \tilde{\lambda} = 0.5 \left[1 - \tanh \left(\frac{y_\perp}{\overline{V}^{\frac{1}{3}}} \right) \right] \tag{2}$$

where $\overline{V}$ is the average cell volume in Ω and $y_\perp$ is the signed perpendicular distance from P, the center of Ω_P, to the solid surface.

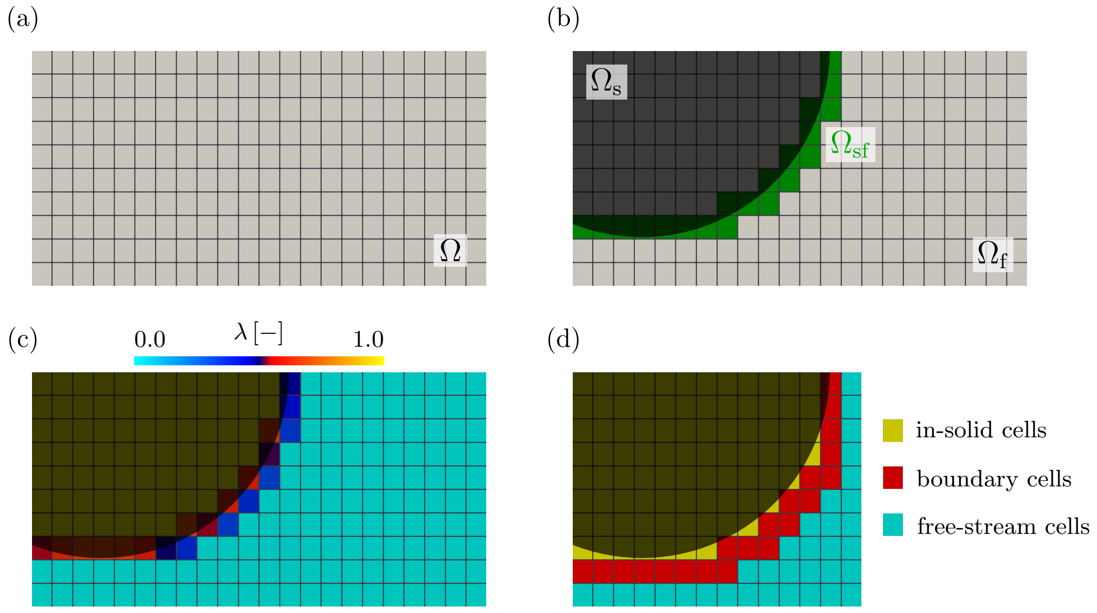

Figure 1: a) Uniform orthogonal mesh. b) Overlay of an immersed body (dark) and the mesh from a) with indicated division of Ω as is referred in (1). c) Representation of the immersed body geometry by a scalar field λ. d) Division of the mesh cells according to the influence of the solid and the fluid phase.

The λ field carries purely geometric information and for the purposes of RAS modeling and adjustment of governing equations, we further divide the cells in Ω into three disjoint sets based on the extent of influence of the two phases, see Fig. 1-d). The groups are (i) in-solid cells that have $\lambda \geq 1/2$ and are affected only by the solid phase, (ii) free-stream cells that are affected by the fluid flow only and are defined as cells that have $\lambda = 0$ and none of their vertex neighbors is an in-solid cell and (iii) boundary cells where a combined effect of the two phases is present. The boundary cells are either cells with $\lambda \in (0, 1/2)$ or cells with $\lambda = 0$ and an in-solid cell as a vertex neighbor.

The turbulent flow in Ω is described by Reynolds averaged Navier-Stokes equations that for an incompressible and isothermal flow of a Newtonian fluid have the form of

$$\begin{array}{c} \mathcal{M}(\boldsymbol{u}) = -\nabla \tilde{p} + \boldsymbol{f}_\mathrm{ib} \\ \nabla \cdot \boldsymbol{u} = 0 \end{array} , \quad \begin{array}{c} \mathcal{M}(\boldsymbol{u}) = \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \nabla \cdot \left[\nu_\mathrm{eff} \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\mathrm{T} \right) \right] \\ \boldsymbol{f}_\mathrm{ib} = \alpha(\lambda) \cdot \left(\mathcal{M}(\boldsymbol{u}_\mathrm{ib}) + \nabla \tilde{p} \right) \end{array} \tag{3}$$

where $\tilde{p}$ and $\boldsymbol{u}$ are averaged kinematic pressure and velocity, respectively. Note that, the momentum conservation equation is extended by an additional force term, $\boldsymbol{f}_{\text{ib}}$, which enforces the prescribed velocity boundary condition at the IB. The scope of effect of the $\boldsymbol{f}_{\text{ib}}$ is determined by a scalar field α where $\alpha = 1$ for in-solid and boundary cells and $\alpha = 0$ for free-stream cells. The values of $\boldsymbol{f}_{\text{ib}}$ are computed from immersed velocity values, $\boldsymbol{u}_{\text{ib}}$, which are enforced in the in-solid and boundary cells by an iterative solution process.

In RAS combined with the Boussinesq hypothesis, the effect of turbulence in (3) is accounted for by effective viscosity, ν_{eff}, which is computed as $\nu_{\text{eff}} = \nu + \nu_t$ where ν is the fluid viscosity and ν_t is turbulent viscosity. In general, the turbulent viscosity is computed via turbulence closure models. In the HFDIB-RAS solver, we chose to implement the $k - \omega$ turbulence model by Wilcox [10] in which $\nu_t = k/\omega$, where k is the turbulent kinetic energy and ω is the specific rate of dissipation of k. The behavior of the two closure variables, k and ω, is described by conservation equations

$$\begin{aligned} \mathcal{N}(k) = Q_{\text{ib}}, \quad \mathcal{N}(k) &= \nabla \cdot (\boldsymbol{u}k) - \nabla \cdot \left(\nu_{\text{eff,k}} \nabla k\right) - S^k, \quad Q_{\text{ib}} = \alpha(\lambda) \cdot \mathcal{N}(k_{\text{ib}}) \\ \mathcal{P}(\omega) = 0, \quad \mathcal{P}(\omega) &= \nabla \cdot (\boldsymbol{u}\omega) - \nabla \cdot \left(\nu_{\text{eff},\omega} \nabla \omega\right) - S^\omega \end{aligned} \tag{4}$$

where $\nu_{\text{eff,k}}$, S^k, $\nu_{\text{eff},\omega}$ and S^ω are computed according to Wilcox [10]. Similarly to the momentum conservation equation, the k conservation equation is extended by an additional source term, Q_{ib}. Its scope of effect is α-dependent and the same as for $\boldsymbol{f}_{\text{ib}}$. The values of Q_{ib} are computed from immersed values, k_{ib}, which are enforced in the in-solid and boundary cells. For the solution of the ω conservation equation, the immersed values, ω_{ib} are required as well. However, they are enforced by a direct modification of the system matrix for the purposes of simulation stability [10].

Thus, to enforce the prescribed boundary conditions at the IB, the immersed values are required to be set prior to the solution of equations (3) and (4). The immersed values in the in-solid cells are based on the continuum assumption that the $\boldsymbol{u}$ and k goes to zero and ω goes to infinity as we are getting closer to the wall. Hence, the values in the in-solid cells are set as

$$\boldsymbol{u}_{\text{ib}} = \boldsymbol{0}, \quad k_{\text{ib}} = 0, \quad \omega_{\text{ib}} = \max_{\Omega_{\text{sf}} \cup \Omega_{\text{wall}}} \left(\omega^{\text{old}}\right) \tag{5}$$

where Ω_{wall} comprises free-stream cells neighboring geometry-conforming walls and ω^{old} are values of ω from the previous iteration or the initial guess.

In the boundary cells, the situation is more complicated. In addition to the satisfaction of the boundary conditions, the boundary cells are responsible for simulation of the fluid boundary layer, which in more turbulent flows becomes very thin. In standard CFD codes, there are two approaches to the boundary layer simulation, (i) fully-resolved approach with local mesh refinement and (ii) modeled approach utilizing wall functions for $\boldsymbol{u}$, k and ω. In the HFDIB-RAS approach, we chose to focus on the latter, since we aim to use our solver mainly in optimizations and local mesh refinement makes the simulations more computationally expensive.

We leverage the standard assumptions utilized in boundary layer modeling. Namely, the description of the boundary layer via normalized variables [11] u^+, k and ω as functions of normalized perpendicular distance to the wall, y^+ [12], which is defined as

$$y^+ = \frac{y_\perp u_\tau}{\nu}, \quad u_\tau = \sqrt{\tau_w} \tag{6}$$

where u_τ and τ_w are the friction velocity and the wall shear stress for incompressible flows, respectively, and the normalized variables are defined as

$$u^+ = \frac{u_t}{u_\tau} \quad k^+ = \frac{k}{u_\tau^2}, \quad \omega^+ = \frac{\nu\omega}{u_\tau^2} \tag{7}$$

where u_t is the velocity component parallel to the wall.

The used wall functions implementation is chosen to be the piece-wise switch-based wall functions in a form similar to the one implemented in OpenFOAM [9]. In particular, the wall functions are defined separately for viscous sublayer and logarithmic region of the fluid boundary layer. Switching between the two definitions is based on a constant y^+_{lam}, which is defined as the value of y^+ when the two sub-functions intersect. The specific definitions are listed in Tab. 1.

variable	viscous sublayer ($y^+ < y^+_{\text{lam}}$)	logarithmic region ($y^+ \geq y^+_{\text{lam}}$)
u^+	y^+	$\frac{1}{\kappa}\ln(Ey^+)$
k^+	$\frac{2400}{C^2_{\varepsilon 2}}\left[\frac{1}{(y^++C)^4} + \frac{2y^+}{C^3} - \frac{1}{C^2}\right]$	$\frac{C_k}{\kappa}\ln(y^+) + B_k$
ω^+	$\frac{6}{\beta_1 (y^+)^2}$	$\frac{1}{\kappa\sqrt{C_\mu}y^+}$

Table 1: Wall functions for u^+, k^+ and ω^+ in the form they were implemented. The κ, E, $C_{\varepsilon 2}$, C, C_k, B_k, β_1 and C_μ are constant parameters of the $k-\omega$ turbulence model, for details see [10].

Commonly, the normalized variables given by the wall functions are used either to prescribe a Dirichlet boundary condition at the wall or to enforce the values directly in the wall-nearest cells. In OpenFOAM, the first approach is used for k and the second for ω [9]. Likewise, in HFDIB-RAS, we use the values of ω^+ for direct computation of the immersed values, ω_{ib}, via (7). On the other hand, the values of k^+ and u^+ are taken as prescribed at the immersed body surface and the immersed values in the boundary cells, $\boldsymbol{u}_{\text{ib}}$ and k_{ib}, are computed via interpolation between the value at the surface and values in the free stream.

To perform the interpolation, a local coordinate system for every boundary cell is created firs. In particular, we need to define normal and tangential directions with respect to the immersed body surface, see Fig. 2. The normal direction is used to identify interpolation points from the free stream. The tangential is required for velocity reconstruction since the wall functions can acquire only tangential velocity component, see (7). Let Ω_P be a boundary cell, then, the unit normal vector ($\boldsymbol{n}_P$) and the unit tangential vector ($\boldsymbol{t}_P$) are computed as

$$\boldsymbol{n}_P = -\frac{(\nabla\lambda)_P}{\|\nabla\lambda\|_P}, \quad \boldsymbol{t}_P = \frac{\boldsymbol{u}_{t,P_1}}{\|\boldsymbol{u}_{t,P_1}\|}, \quad \boldsymbol{u}_{t,P_1} = \boldsymbol{u}_{P_1} - (\boldsymbol{n}_P \cdot \boldsymbol{u}_{P_1})\boldsymbol{n}_P \tag{8}$$

where P is the center of Ω_P and P_1 is the first free-stream interpolation point.

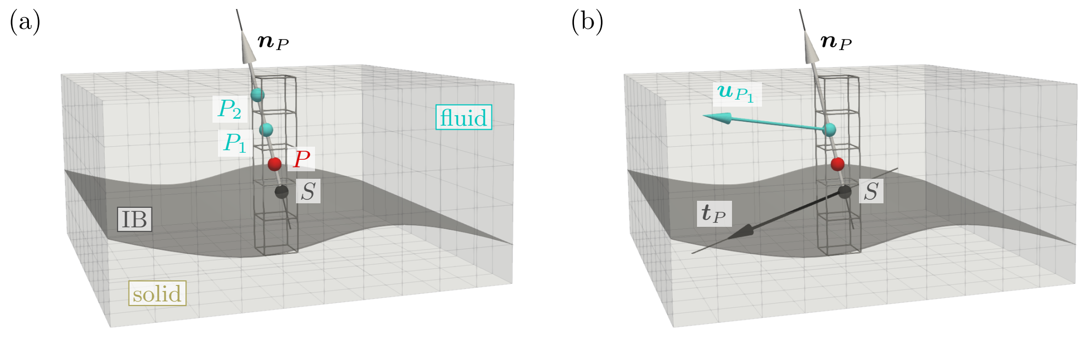

Figure 2: Local coordinate system for a boundary cell Ω_P. (a) Normal vector, $\boldsymbol{n}_P$, center point, P, surface point, S, and identified free-stream interpolation points. (b) Tangential vector, $\boldsymbol{t}_P$, and velocity in the first interpolation point, $\boldsymbol{u}_{P_1}$, used for its construction.

Utilizing the values from free-stream interpolation points and the surface value given by the wall functions, we compute the values in the boundary cells either via polynomial or logarithmic interpolation. The polynomial interpolation is valid if the boundary layer is fully resolved. As such, it is used for the tangential component of $\boldsymbol{u}_{\text{ib}}$ and for k when the center of the boundary cell lies in the viscous sublayer. Furthermore, it is always used to compute the normal component of $\boldsymbol{u}_{\text{ib}}$. For a scalar quantity φ, the polynomial reconstruction of φ_{ib} is defined as

$$\begin{gathered}\varphi_{\text{ib}} = Ay_\perp^2 + By_\perp + \varphi_S, \quad A = \frac{(\varphi_{P_2} - \varphi_{P_1})\,\delta_1 - (\varphi_{P_1} - \varphi_S)\,\delta_2}{\delta_1\delta_2\,(\delta_1 + \delta_2)} \\ B = \frac{(\varphi_{P_1} - \varphi_{P_2})\,\delta_1^2 + 2\,(\varphi_{P_1} - \varphi_S)\,\delta_1\delta_2 + (\varphi_{P_1} - \varphi_S)\,\delta_2^2}{\delta_1\delta_2(\delta_1 + \delta_2)}\end{gathered} \tag{9}$$

where δ_1 is the distance from the surface to the first interpolation point and δ_2 the distance between the first and the second interpolation point.

The logarithmic interpolation is used for the $\boldsymbol{u}_{\mathrm{ib}}$ tangential component and k when the center is located in the logarithmic region. It is defined as

$$\varphi_{\mathrm{ib}} = A \ln\left(B y_{\perp} + 1\right) + \varphi_S, \quad A = \frac{\varphi_{P_1} - \varphi_S}{B\delta_1 + 1}, \quad B = \begin{cases} \dfrac{E u_\tau}{\nu} & \text{if } \varphi = u_t \\ \dfrac{u_\tau}{\nu} & \text{if } \varphi = k \end{cases} \tag{10}$$

where the values for B were taken from the definitions of wall functions for the logarithmic region.

In order to enforce the immersed values in in-solid and boundary cells, HFDIB-RAS uses a modified solver iteration loop. The modifications on the SIMPLE solver connected with the $k-\omega$ turbulence model are depicted in Fig. 3. The iteration loop starts with computation of $\boldsymbol{u}_{\mathrm{ib}}$, using values from the previous iteration. Then, there is the momentum predictor step $(3)_1$ after which the algorithm enters a sub-loop over the pressure correction $(3)_2$, which is run until the difference between $\boldsymbol{u}$ and $\boldsymbol{u}_{\mathrm{ib}}$ is sufficiently small, i.e. smaller then an arbitrary small number $\Xi > 0$.

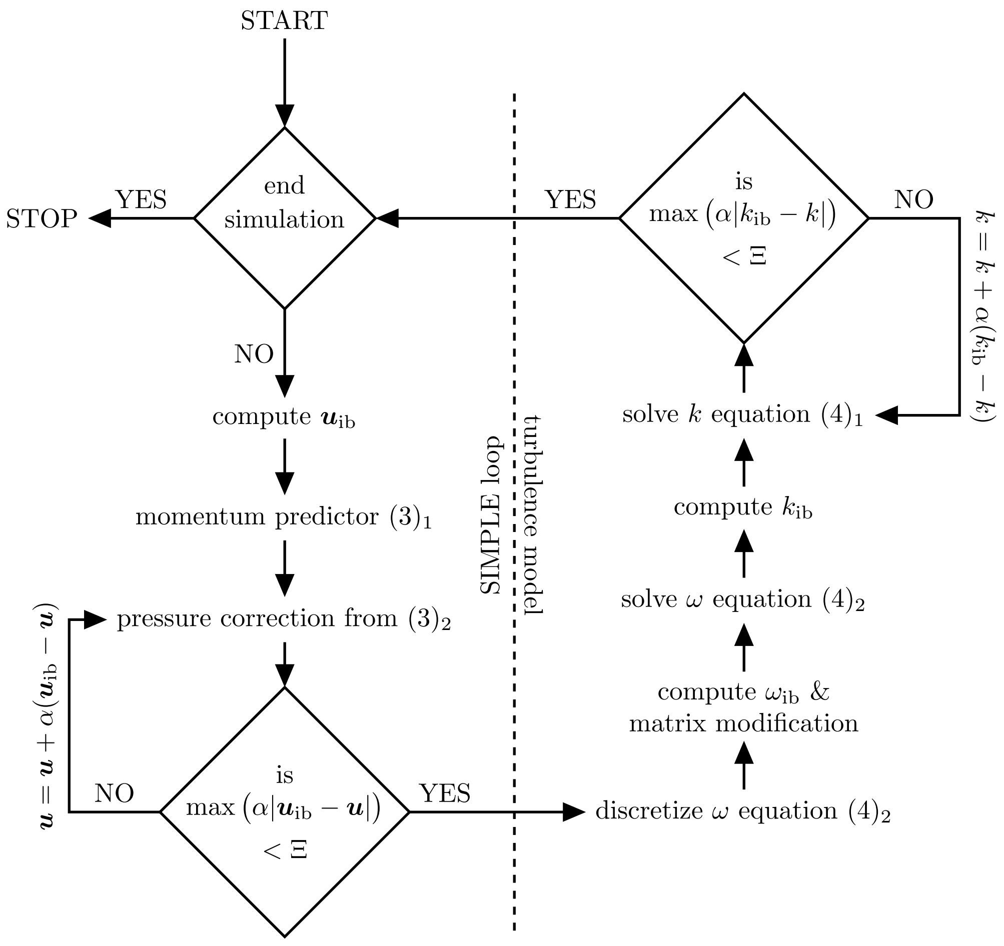
START
end simulation
YES
STOP
NO
compute $\boldsymbol{u}_{\mathrm{ib}}$
momentum predictor $(3)_1$
pressure correction from $(3)_2$
is $\max\left(\alpha|\boldsymbol{u}_{\mathrm{ib}} - \boldsymbol{u}|\right) < \Xi$
NO
$\boldsymbol{u} = \boldsymbol{u} + \alpha(\boldsymbol{u}_{\mathrm{ib}} - \boldsymbol{u})$
YES
SIMPLE loop
turbulence model
discretize ω equation $(4)_2$
compute ω_{ib} & matrix modification
solve ω equation $(4)_2$
compute k_{ib}
solve k equation $(4)_1$
is $\max\left(\alpha|k_{\mathrm{ib}} - k|\right) < \Xi$
YES
NO
$k = k + \alpha(k_{\mathrm{ib}} - k)$

Figure 3: SIMPLE loop connected to a $k-\omega$ turbulence model with modifications done for the HFDIB-RAS solver. The $\Xi > 0$ is an arbitrary small number.

Next, the algorithm enters the turbulence model loop, where the first step is the discretization of the ω equation $(4)_2$. Then the values of ω_{ib} are computed and the system matrix from the previous step is modified via tools available in OpenFOAM [9]. Subsequently, the algorithm solves for ω, computes the k_{ib} values and enters another sub-loop over the solution of the k equation $(4)_1$. Similarly to the pressure correction loop, the second sub-loop continues until the difference between k_{ib} and k is small enough. Finally, the algorithm checks the termination conditions and either ends the simulation or continues with the SIMPLE loop once again.

3 Results

Since the HFDIB-RAS development is an ongoing research, the capabilities of the HFDIB-RAS method are showed on several simple verification tests designed to show the research progress and not to represent a robust verification. In all the tests, we compared results from our HFDIB-RAS solver and from the standard OpenFOAM simpleFoam solver [9]. Every test was designed in such a way that it was possible to create two almost identical meshes, one geometry-conforming for simpleFoam and one for the HFDIB-RAS method where the geometry was represented by the scalar field λ. The meshes used in four of these tests are depicted in the Fig. 4.

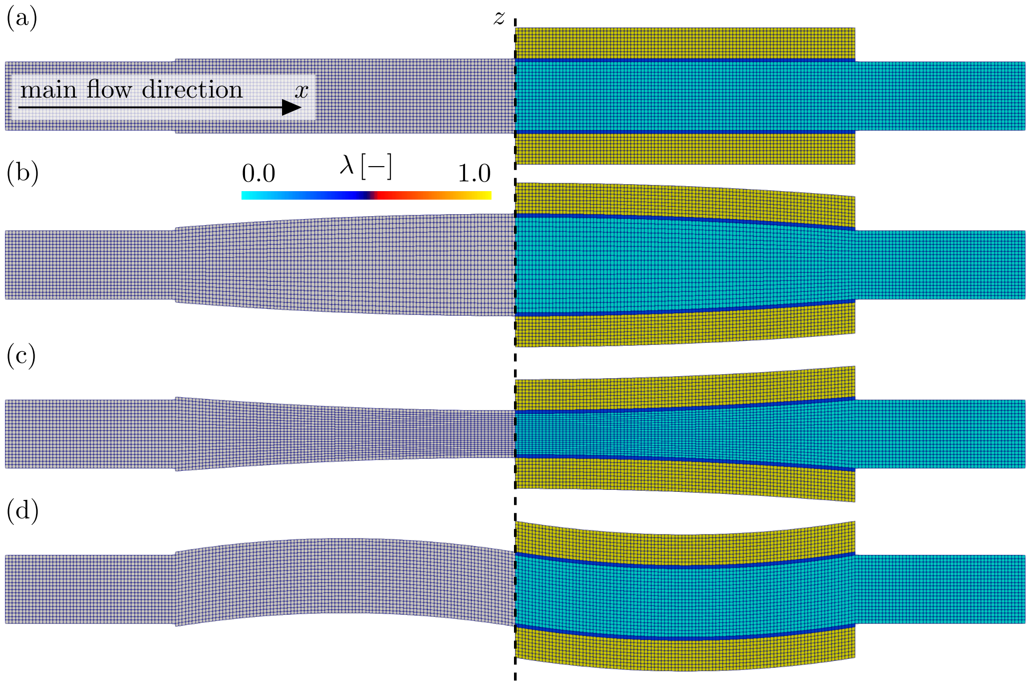

Figure 4: Computational meshes used in verification tests. On the left, there are geometry-conforming meshes and on the right, meshes used for HFDIB-RAS with the geometry indicated by the λ field.

With every mesh pair, we ran a number of simulations among which we changed the flow Reynolds number in a way that it varied from 10^1 to 10^6. First, we ran the verification tests with the meshes showed in Fig. 4-a) to see how the HFDIB-RAS solver behavior scales with the flow Reynolds number. The resulting $\boldsymbol{u}$ and ν_t profiles for the $\mathrm{Re} = \{10^2, 10^4, 10^6\}$ are compared in Fig. 5. Then, to test the generality of the local coordinate system creation, we ran simulations with the bent geometries, see Fig. 4-b)-c)-d). Comparison of $\boldsymbol{u}$ and ν_t profiles for $\mathrm{Re} = 10^6$ is depicted in Fig. 6.

From the results depicted in Fig. 5, it can be seen that qualitatively, the HFDIB-RAS solver scales well with growing Reynolds number. However, there is problem with the solution accuracy that is visible the most on the subplot with ν_t profiles for $\mathrm{Re} = 10^6$. The profiles from simulations with the bent geometries, depicted in Fig. 6, show a similar behavior to the profiles in Fig. 5. Hence, the creation of the local coordinate system is general enough, but the problems apparent from the results in Fig. 5 have non-negligible influence, i.e., the profiles are qualitatively accurate yet shifted in the absolute values. In the simulations with the non-symmetric mesh, see Fig. 4-d), the propagation of the absolute error even leads to a shift of the profiles along the domain itself.

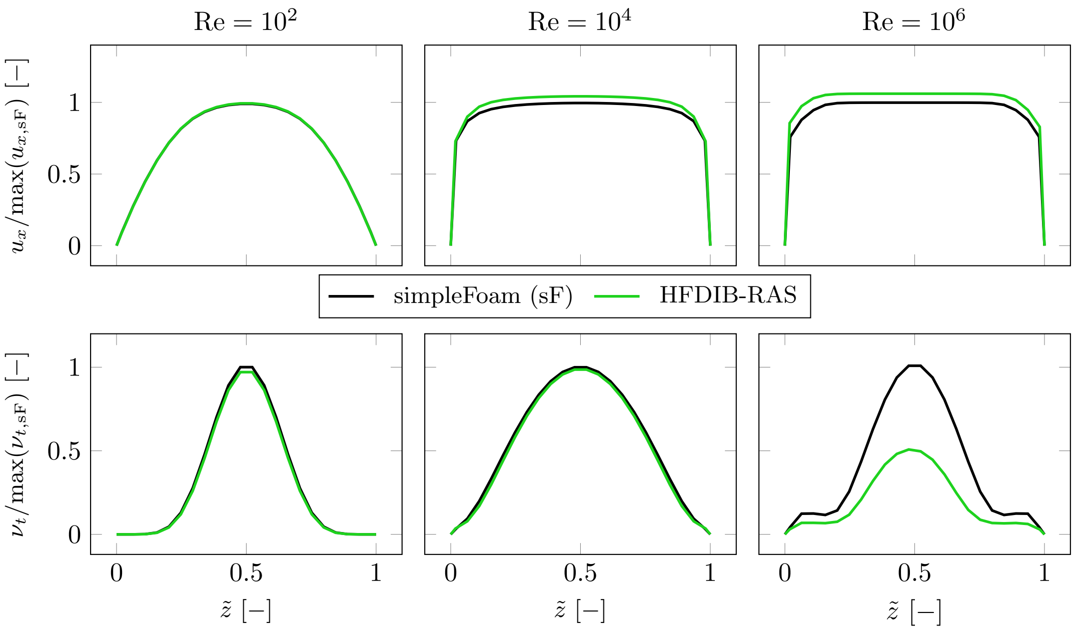

Figure 5: Comparison of $\boldsymbol{u}$ and ν_t profiles acquired from simulations for Re = $\{10^2, 10^4, 10^6\}$ and using the meshes depicted in Fig. 4-a). The $\tilde{z}$ is a normalized distance along the z-line depicted in Fig. 4 where $\tilde{z} = [z - \min(z_{\mathrm{sF}})]/[\max(z_{\mathrm{sF}}) - \min(z_{\mathrm{sF}})]$.

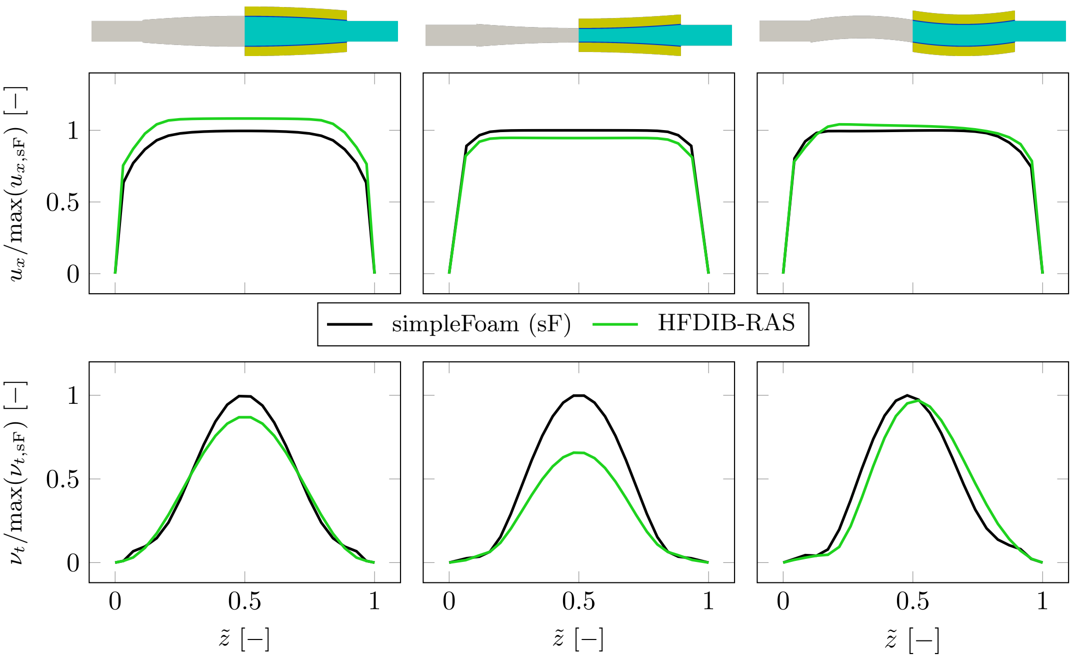

Figure 6: Comparison of $\boldsymbol{u}$ and ν_t profiles acquired from simulations for Re = 10^6 and using the meshes depicted in Fig. 4-b)-c)-d).

4 Conclusion

In this contribution, we presented a research progress in development of a HFDIB-RAS solver created as a connection of our custom HFDIB method and RAS turbulence modeling approach. In particular, we implemented a modified version of the Reynolds averaged Navier-Stokes equations, the $k - \omega$ turbulence closure model and switch-based wall functions. The HFDIB-RAS method behavior is presented on results from verification tests that showed a good qualitative agreement with a standard CFD solver. Nevertheless, there are non-negligible problems with the solution accuracy that shall be solved in future method development. Still, the shown method robustness with respect to bent geometries hints at its future applicability for topology optimizations of components under real-life conditions.

Acknowledgment

The work was financially supported by the institutional support RVO:61388998.

References

[1] Kubo, S., Koguchi, A., Yaji, K., Yamada, T., Izui, K. & Nishiwaki, S.: Level set-based topology optimization for two dimensional turbulent flow using an immersed boundary method. *Journal of Computational Physics.* vol. 446: (2021). page 110630.

[2] Capizzano, F.: A turbulent wall model for immersed boundary methods. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*: (01 2010).

[3] Isoz, M., Šourek, M. K., Studeník, O. & Kočí, P.: Hybrid fictitious domain-immersed boundary solver coupled with discrete element method for simulations of flows laden with arbitrarily-shaped particles. *Computers and Fluids.* vol. 244: (2022). page 105538.

[4] Kubíčková, L., Isoz, M. & Haidl, J.: Increasing ejector efficiency via diffuser shape optimization. In *Proceedings of Topical Problems of Fluid Mechanics 2021*: pp. 79–86. IT CAS: (02 2021).

[5] Troldborg, N., Sørensen, N. N. & Zahle, F.: Immersed boundary method for the incompressible Reynolds Averaged Navier–Stokes equations. *Computers and Fluids.* vol. 237: (2022). page 105340.

[6] Kalitzin, G. & Iaccarino, G.: Turbulence modeling in an immersed-boundary RANS method. *Center for Turbulence Research Annual Research Briefs.* (01 2002). pp. 415–426.

[7] Iaccarino, G.: Immersed boundary technique for turbulent flow simulations. *Applied Mechanics Reviews.* vol. 56: (05 2003). pp. 331–347.

[8] Kubíčková, L. & Isoz, M.: Hybrid fictitious domain-immersed boundary method in CFD-based topology optimization. In *Proceedings of Topical Problems of Fluid Mechanics 2022*: pp. 119–126. IT CAS: (02 2022).

[9] OpenCFD. *OpenFOAM: The Open Source CFD Toolbox, User Guide, OpenCFD Ltd.* Reading UK: (2016).

[10] Wilcox, D. *Turbulence modeling for CFD.* DCW Industries, USA: 3 edition: (2006).

[11] Spalding, D. B.: A single formula for the "law of the wall". *Journal of Applied Mechanics.* vol. 28: (1961). pp. 455–458.

[12] Moukalled, F., Darwish, M. & Mangani, L. *The finite volume method in computational fluid dynamics: an advanced introduction with OpenFOAM and Matlab.* Springer-Verlag: Berlin, Germany: 1 edition: (2016). ISBN 978-3-319-16874-6.