RECENT IMPROVEMENTS IN CFD SOLVER FOR FULLY COUPLED PARTICLE-LADEN FLOWS

Martin Šourek 1, 2, Martin Isoz 2, 3

1 Department of Chemical Engineering, Faculty of Chemical Engineering, University of Chemistry and Technology, Technická 5, Prague 166 28, Czech Republic
2 Department of Mathematics, Faculty of Chemical Engineering, University of Chemistry and Technology, Technická 5, Prague 166 28, Czech Republic
3 Czech Academy of Sciences, Institute of Thermomechanics, Dolejškova 5, Prague 182 00, Czech Republic

Abstract

While new methods combining the computational fluid dynamics (CFD) and the discrete element method (DEM) have been developed to simulate freely moving solid particles, they tend to be focused on simulations with spherical particles only. Here, we present a strongly coupled CFD-DEM solver capable of simulating movement of arbitrarily-shaped particles dispersed in a fluid. The particles are assumed to be large enough to affect the fluid flow and distributed densely enough to come into contact with both the boundaries of the computational domain and with each other. In this paper, we will focus on the recent improvements of our solver; specifically, in the areas of (i) inclusion of solid bodies into the computational domain, (ii) general CFD-DEM coupling algorithm, and (iii) code parallelization and practical usability.

Keywords: CFD, DEM, HFDIB-DEM, OpenFOAM.

1 Introduction

Flows coupled with particle movement are often encountered both in nature and in industry. As a few examples, it is possible to mention a motion of sand including sand storms, sediment transport in river beds as well as water treatment or fluidized beds in chemistry. Due to the importance of processes involving coupled particle and fluid flow, there is a long lasting effort to complete available experimental data by detailed simulations of the fluid-particle interactions in such flows represented for example by the work of Hamzehei [4]. However, because these processes involve both fluid and particle motion, standard mesh-conforming methods of computational fluid dynamics (CFD) cannot be directly applied to simulate them. In particular, the potentially arbitrary solid motion inside the computational domain would require frequent and costly re-meshing.

Several approaches exist for simulations of particle-laden flows. Suppose that interactions between the solid and fluid phases need to be resolved, but the local behavior of individual particles can be neglected. In that case, it is possible to use a two-fluid model with the solid phase represented as a continuum [2]. On the other hand, if the particle-fluid coupling can be neglected, but individual particles’ behavior is essential, it is profitable to use a discrete approach such as the discrete element method (DEM) [6]. Finally, if both the particle-fluid coupling and local interactions between particles are important, it is necessary to couple the continuum (Eulerian, CFD) and discrete (Lagrangian, DEM) approaches. When talking about the Eulerian-Lagrangian coupling for solid particles that are highly irregular and/or span over several domain discretization elements, the main challenge lies in including the solid bodies into the Eulerian computational domain.

In the present paper, we build on our previous work [10], where we developed a custom CFD-DEM solver. We have utilized a combination of a fictitious domain (FD), and immersed boundary (IB) method [3, 12] for solid-phase inclusion in the computational domain and the discrete element method for movements of the bodies. In this contribution, we focus on recent improvements of the solver and discuss the effects of the made changes on the code behavior and usability. In particular, we present a change in the definition of the indicator function describing the position of solids that enables a more accurate reconstruction of the fluid-fictitious solid boundary, improvements to the fluid-solid interaction coupling leading to a better solver stability, and fundamental principles of the code parallelization, which facilitates using the solver for practical tasks. Also, for the reader convenience, the text is completed by a brief introduction to the DEM part of the code, including the contact treatment.
2 Computational methods

There are two main challenges connected to the implementation of a solver for four-way coupled particle-laden flows. The first challenge consists of implementing an efficient and accurate method suitable for an inclusion of solid bodies into the computational domain. The second problem lies in description of motion of the freely moving bodies inside the computational domain and in resolving potential body-body and body-wall contacts.

2.1 Flow governing equations and inclusion of solid bodies

In the solver, we take into account balance equations for momentum and mass in a finite open and simply connected domain $\Omega \subset \mathbb{R}^3$ with boundary $\Gamma = \partial \Omega$. Let us decompose $\Omega$ as $\Omega = \Omega_s \cup \Omega_f \cup \Gamma_{sf}$, where $\Omega_s$ represents the part of $\Omega$ occupied by a solid phase, $\Omega_f$ the part of $\Omega$ occupied by the fluid and $\Gamma_{sf} = \partial \Omega_s \cap \partial \Omega_f$ is the solid-fluid interface in $\Omega$. Note that in the case of moving solid bodies $\Omega_s$, $\Omega_f$ and $\Gamma_{sf}$ depend on time. Furthermore, for the case of a solid phase dispersed in $\Omega$, $\Omega_s$ and $\Gamma_{sf}$ may not be connected. The situation is illustrated in Fig. 1.

The considered flow governing equations correspond to the standard variant of laminar Navier-Stokes equations for an incompressible Newtonian fluid with added forcing term $s$,

$$
M(u) = -\nabla \tilde{p} + g + s \\
\nabla \cdot u = 0
$$

where $u$ is the fluid velocity, $\nu$ kinematic viscosity, $\tilde{p}$ kinematic pressure and $g$ the gravitational acceleration. The forcing term is constructed in a way that it generates a fictitious representation of $\Omega_s$ inside $\Omega$. This approach is known as hybrid fictitious domain-immersed boundary method and is based on the works of Patankar et al. [9], Blais et al. [1] and Municchi and Radl [7, 8].

In particular, the additional forcing term $s$ in (1) is defined as

$$
s = \text{ceil}(\lambda)\tilde{s}, \quad \tilde{s} = M(u_{ib}) + \nabla \tilde{p} - g, \quad \lambda = \begin{cases} 
0 & \text{in } \Omega_f \\
1 & \text{in } \Omega_s \\
\bar{\lambda} \in (0, 1) & \text{in } \Gamma_{sf}
\end{cases}
$$

where $\lambda$ is a color function defining the position of the solid bodies in the computational domain $\Omega$. Note that in the continuous setting $\Gamma_{sf}$ is infinitely thin. However, after the finite volume discretization of $\Omega$, the discrete variant of $\Gamma_{sf}$, $\Gamma_{sf}^h$, may span over several finite volume cells.

**Improved definition of the $\lambda$ field**
The principle of reconstruction of values of $u_{ib}$ (or any other intensive tensorial quantity $\varphi$) on $\Gamma_{sf}$ is depicted in Fig. 2-a). For details on the process the reader is referred to [10]. However, let us recall that the value of $\varphi$ at $C$ ($\varphi_C$) is obtained via polynomial extrapolation and based on values at $S$, $P1$ and $P2$. Successful computation of $\varphi_C$ depends on (i) estimating the position of $S$ and (ii) computing the direction vector of the line $r$, i.e. computing $n_{ib} := -\nabla \lambda/\|\nabla \lambda\|$.
Figure 2: (a) Principle of reconstruction of an arbitrary intensive tensorial quantity $\phi$ on the solid-fluid interface. $\Omega_s$ is in white, $\Omega_f$ in red and $\Gamma_{sf}$ in red and green. (b) Body surface reconstruction using $\lambda$ fields based on (3). (c) Same as (b) but based on (4).

Standard definition of the $\lambda$ field in the cell $\Omega^b_C$, $\lambda_C$, used for example in [1], is

$$
\lambda^b_C := \frac{|\{v_i\}_C| + |\{C_i\}_C| |\{v\}_C|}{2 |\{v\}_C|},
$$

where $|\{v\}_C|$ and $|\{v_i\}_C|$ are the number of all and immersed vertices of $\Omega^b_C$, respectively and $|\{C_i\}_C|$ is 1 if the cell centroid $C$ is immersed in the solid body and 0 otherwise.

The values of the $\lambda$ field based on (3) may be computed fairly efficiently. However, once $\lambda_C$ is computed, there is no relation between $\lambda_C$, $n_{ib,C}$ and the position of the point $S$. To keep a strong relation between these three quantities usable for computation of $\phi_C$, we proposed an alternative definition of the $\lambda$ field based on recomputing the values of $\lambda$ in the cells with $\lambda \in (0, 1)$, as obtained from (3), using signed distance ($\sigma_C$) between the cell centroid ($C$) and the point $S$,

$$
\lambda^b_C := \frac{1}{2} \left[ \pm \tanh \left( \frac{\delta_i \sigma_C}{\text{Volume}(\Omega^b_C)^{1/3}} \right) + 1 \right] \text{ if } \lambda_C \text{ from (3) } \in (0, 1),
$$

where the hyperbolic tangent is positive if $|\{C_i\}_C| = 1$ and negative otherwise and $\delta_i$ is the number of cells over which the final interface should span.

The most important property of introducing the relation (4) comes into play with the need to compute the point $S$ for $\varphi_C$ computation. Contrary to (3), the relation (4) provides clear relation between $\lambda_C$ and $S$,

$$
S \approx C + n_{ib,C} \sigma_C = C + n_{ib,C} \text{ atanh} \left( 2 \lambda_C - 1 \right) \frac{\text{Volume}(\Omega^b_C)^{1/3}}{\delta_i}.
$$

Note that the relation (5) is approximate only because of the numerical evaluation of $n_{ib,C}$. Comparison of positions of surface points computed based on (3) and (4) is depicted in Fig. 2-b) and c), respectively.

2.2 Bodies movement and contact resolution

The solids are included into $\Omega$ as described in the previous section. The movement of solids themselves, i.e. updating $\Omega_s$, is based on the discrete element method (DEM). DEM is a finite difference method for solving Newtons laws of motion for each included body individually. Splitting $\Omega_s$ into individual bodies as $\Omega_s = \bigcup_{i=1}^{\text{Bodies}} B_i$, the movement of the $i$-th body ($B_i$) is described by

$$
m_i \frac{d^2 \mathbf{x}_i}{dt^2} = f_i^g + f_i^d + f_i^v, \quad I_i \frac{d \mathbf{\omega}_i}{dt} = t_i^g + t_i^d + t_i^v,
$$

where $m_i$ stands for the mass of $B_i$ and $\mathbf{x}_i(t)$ for its centroid position at time $t$. Furthermore, $\mathbf{\omega}_i$ is the body angular velocity and $I_i$ is the matrix of inertial moments. The sums $\sum_j f_i^j$, $\sum_j t_i^j$, $j =$
\{g, d, c\} on the right hand sides of equations (6) represent all the forces and torques acting on the body \(i\), respectively. Furthermore, the superscripts \(g\), \(d\), and \(c\) mark gravity/buoyancy, drag, and contact, respectively.

The effects of gravity and buoyancy acting on \(B_i\) are computed as

\[
f^g_i = m_i g \left(1 - \frac{\rho_f}{\rho_s}\right), \quad t^g_i \equiv 0,
\]

with \(\rho_f\) and \(\rho_s\) being the fluid and solid density, respectively. The effects of flow on the movement of \(B_i\) are evaluated from \(s^h\), the finite volume discretization of \(s\), as

\[
f^d_i = \rho_f \sum_{\Omega^h_C \in \partial B_i} s^h_C \text{Volume}(\Omega^h_C), \quad t^d_i = \rho_f \sum_{\Omega^h_C \in \partial B_i} (C - x_i) \times s^h_C \text{Volume}(\Omega^h_C),
\]

where \(\partial B_i\) are the mesh cells appertaining to \(B_i\) with \(\lambda_C \in (0, 1)\), i.e. the surface cells of \(B_i\).

The contact forces and torques are significantly more complex than the remaining terms in (6). First, the particle-particle or particle-wall contact needs to be detected. Only afterwards, \(f^c_i\) and \(t^c_i\) can be evaluated to complete equations (6). A detailed description of contact detection and evaluation in the presented solver is beyond the scope of this contribution. However, below we try to at least outline the fundamentals of the implemented contact treatment.

![Figure 3: (a) \(B_i, B_j\), their surface cells and edge cells \(E_{ij}\). (b) Contact plane constructed based on \(E_{ij}\). (c) Overlap volume \(V_c\) between \(B_i\) and \(B_j\). (d) Vector \(r_i\) Computed based on \(x_i\) and the contact plane.](image)

In the present work, the contact is treated within soft-particle DEM framework, i.e. we allow for a small overlap between particles [6]. Contact between two bodies \(B_i\) and \(B_j\) is detected when they share at least a single cell, \(B_i \cap B_j \neq \emptyset\), see Fig. 3-a). We mark the shared surface cells of the two bodies as edge cells \(E_{ij} = \partial B_i \cap \partial B_j\) and use \(E_{ij}\) to compute the plane of contact between \(B_i\) and \(B_j\), see Fig. 3-b). The contact forces and torques acting on \(B_i\) are evaluated based on overlapping volume \(V_c\) between \(B_i\) and \(B_j\), Fig. 3-c) and vector \(r_i\) depicted in Fig. 3-d).

The particle-wall treatment is substantially less complicated because (i) cell in \(\mathcal{B}\) can be easily tested for adjacency to a wall-type boundary, and (ii) a normal to the wall boundary is readily present at each boundary face. Nevertheless, both particle-particle and particle-wall detection may be made more effective using concept of neighbor lists as presented by Verlet [11].

### 2.3 Improvement of CFD-DEM coupling

In the previous sections, we outlined fundamentals of computing the effects of solids on the flow via hybrid fictitious domain-immersed boundary method (HFDIB) and vice versa using DEM. In this section, we will focus on coupling HFDIB and DEM to obtain the final solver. First, note that the structure of equations (1) with the source term (2) permits the flow to be solved with standard semi-implicit PISO-like loop. Furthermore, both the solver parts are implemented in the OpenFOAM library.
In the original solver implementation [10], we followed the overall algorithm structure as presented by Blais et al. [1], which is summarized in Alg. 1. The flow and DEM parts of the code are segregated and coupled only weakly through updates of the forcing term $s$, see line 12 in Alg. 1. However, the update of $s$ does not affect the computation of the drag effects, which is directly based on $s$, see (8), computed on line 8 of Alg. 1.

Using only the weak coupling between HFDIB and DEM causes the converged $s$ term not to be the one affecting the solid bodies on the current time level. During the algorithm time-marching procedure, this fluid-solid uncoupling is self-amplifying and leads to nonphysical oscillations in the drag force during simulation. To overcome this issue, we leveraged the fact that both HFDIB and DEM are implemented in the same computational code (OpenFOAM). Consequently, all the HFDIB and DEM data are available in the computer memory in the same time and the computationally cheap drag evaluation may be done repeatedly inside the PISO loop, cf. Algs. 1 and 2.

Effects of the solution algorithm modification are illustrated in Fig. 4, where we depict comparison of drag forces computed using the old and the new solver variant. The test case corresponds to a sedimentation of a three-dimensional spherical particle inside a cylindrical container. Note that even in the modified solver, there are some oscillations in $f_d$ during the simulation. However, these oscillations are caused by numerical errors in other parts of the code, e.g. evaluation of $\nabla \lambda$ or computation of points $S$, $P_1$ and $P_2$ for each surface cell of the body, see Fig. 2.

### 2.4 Code parallelization

Another significant improvement in the presented code is its parallelization. Standard pre-built OpenFOAM applications scale well in parallelization even on thousands of cores given there is more than approximately 50,000 cells per one core. Furthermore, OpenFOAM provides a high-level language in which it is relatively easy for the user to make use of its built-in parallelization capabilities based on domain decomposition even when writing custom applications. Nevertheless, the presented implementation of HFDIB-DEM solver poses several difficulties that make its parallelization within the domain-decomposition framework a challenging topic.

The first challenge comes in play already during the $\lambda$ field reconstruction; in particular, during the computation of points $P_1$ and $P_2$ and during identification of the finite volume cells that contain

---

**Algorithm 1** Segregated coupling

**Require:**
- Tolerance $\varepsilon$
- Number of PISO corrections $N_{\text{corr}}^{\text{PISO}}$
- Final time $t_{\text{End}}$

1: while $t < t_{\text{End}}$ do
2: Evaluate current Courant number
3: Update time step
4: Update $\lambda$ field $\leftarrow$ see 2.1
5: Compute interpolation points
6: Compute drag from (8)
7: Move solid-phase
8: Update solid-phase movement
9: Detect potential contact $\leftarrow$ see 2.2
10: Resolve contacts
11: while $\varepsilon \leq$ Residuals do
12: Update $s$ from (2)
13: Optional momentum predictor
14: for $j = 1$ to $N_{\text{corr}}^{\text{PISO}}$ do
15: Solve the pressure equation
16: Update $u$ and $u_{ib}$
17: end for
18: end while
19: end while

**Algorithm 2** Strong coupling

**Require:**
- Tolerance $\varepsilon$
- Number of PISO corrections $N_{\text{corr}}^{\text{PISO}}$
- Final time $t_{\text{End}}$

1: while $t < t_{\text{End}}$ do
2: Evaluate current Courant number
3: Update time step
4: Update $\lambda$ field $\leftarrow$ see 2.1
5: Compute interpolation points
6: while $\varepsilon \leq$ Residuals do
7: Update $s$ from (2)
8: Optional momentum predictor
9: for $j = 1$ to $N_{\text{corr}}^{\text{PISO}}$ do
10: Solve the pressure equation
11: Update $u$ and $u_{ib}$
12: Compute drag from (8)
13: Update solid-phase movement
14: end for
15: end while
16: Move solid-phase
17: Detect potential contact $\leftarrow$ see 2.2
18: Resolve contacts
19: end while
them, see Fig. 2. During a parallel computation, the mesh and all the variable fields are decomposed into different subdomains, $\Omega^h = \bigcup_{i=1}^{n\text{Proc}} \Omega^h_i$, $\varphi^h = \bigcup_{i=1}^{n\text{Proc}} \varphi_i^h$, where $\Omega^h$ is a finite volume mesh and $\varphi^h$ represents a finite volume discretization of an arbitrary intensive tensorial quantity $\varphi$. Each processor handles its own part of the given field and the computation is made consistent by creating artificial processor-processor boundaries between individual parts of the decomposed fields.

Due to the domain decomposition principles, if $S \in \Omega^h_i$ and $P1 \in \Omega^h_j$, $i \neq j$, it is necessary to create a processor-processor communication channel outside of the standard OpenFOAM structure in order to locate the processor $j$ handling the cell $\Omega^h_{j,P1}$ containing the point $P1$ using a search originating in cell $\Omega^h_i$, $S$ on the processor $i$. Furthermore, this issue is encountered multiple times, (i) during computation of $P1$ and $P2$ as described above, (ii) during construction of the $\lambda$ field itself, which is based on OCTREE searches originating in a mesh cell previously belonging to a solid body, and (iii) during evaluation of $\varphi^{h,C}$ based on values of $\varphi^h$ in $S$, $P1$ and $P2$.

The second challenge regarding the code parallelization originates in the fact that the DEM part of the solver has to be parallelized in a manner completely independent of the standard OpenFOAM practices. Each solid body needs to have only a single center of mass, linear and angular velocities and other movement variables even if it is spanned over multiple subdomains. To prevent OpenFOAM from assigning different sets of movement variables to a single body, each body has its owner processor. The body owner processor is identified as the processor that contains the highest number of cells belonging to the body and it is the only processor that keeps the information on body movement variables.

To evaluate the scaling properties of the HFDIB-DEM solver, we set up a simple three-dimensional cavity test. The tests consists of a lid-driven flow in a closed cubical cavity with a spherical body placed in the center, see Fig. 5-b). The flow Reynolds number computed with respect to the size of the cavity is 10. The computational domain is discretized by 120 cells per cube side.

This case allows us to compare average times necessary to solve a single time level for a standard OpenFOAM CFD solver (pimpleFoam) and the newly developed HFDIB-DEM code. The results are depicted in Fig. 5-b). Note that the computational mesh used for HFDIB-DEM simulations contains a larger number of cells as the solid body is subtracted from the mesh used in pimpleFoam. Furthermore, even for a static body, the HFDIB-DEM solver performs additional solid-connected computations compared to the standard CFD solver. Consequently, the HFDIB-DEM solver is slightly slower per time level than pure pimpleFoam. On the other hand, the computational times are in the same order and scale similarly for both the solver.
Figure 5: (a) Comparison of time-step execution time for pimpleFoam and HFDIB-DEM solver for different number of used cores. (b) Visualization of the 3D cavity benchmark used for the test.

3 Example of solver usage

To show a possible usage of our solver we would like to study a flow of a water-based slurry through a porous medium. Such flows are present during filtration or in other engineering processes [5]. Particles comprising the slurry are depicted in Fig. 6-a). The porous medium defining the computational domain is shown in Fig. 6-b).

The problem length, velocity and viscosity scales are such that the flow is inside the laminar flow regime. The finite volume discretization of the computational domain consisted of approximately one million cells. At each time, roughly 100 individual particles were present inside the computational domain. Snapshot of the particle distribution and flow field during the simulation is given in Fig. 6-c) and d). Displayed simulation was computed within 46 hours using a parallel run on a standard eight core workstation.

4 Conclusion

Modeling of fully coupled particle-laden flows is still a challenging topic, especially if both the flow field at the particle-fluid interface and the particles movement are of interest. In this contribution,
we presented an extension to our previous work on development of a numerical solver for simulations of particle-laden flows. The presented solver modifications comprise a new definition of the \( \lambda \) field, refined fluid-solid interaction coupling and code parallelization, and increase the solver accuracy, stability and usability. In the future work, we plan to focus on further improving the code efficiency and on its applications on real-life engineering problems such as deposition of a catalytic material inside catalytic particulate filters in automotive exhaust gas aftertreatment systems.

**Acknowledgment**

The work was supported by the Grant project with No. GA19-22173S of the Czech Science Foundation, within institutional support RVO:61388998 and by the Centre of Excellence for Nonlinear Dynamic Behaviour of Advanced Materials in Engineering CZ.02.1.01/0.0/0.0/15_003/0000493 (Excellent Research Teams) in the framework of Operational Programme Research, Development and Education.

**References**


