Journal of Engineering Mathematics (2024) 146:5
https://doi.org/10.1007/s10665-024-10357-z

Implicit interpolation method for immersed boundary methods

Md. Sujaat Ali[1] · Renan de Holanda Sousa[1] · M. Ossman Awad[1] · Ricardo Camarero[1] · Jean-Yves Trépanier[1]

Received: 7 July 2023 / Accepted: 11 March 2024

Abstract
Immersed boundary (IB) methods have been successfully implemented for different applications. This paper focuses on the immersed boundary implementation for two different governing equations, namely the diffusion equation and Euler equations, using a bi-linear interpolation for the implementation of the boundary condition. The concept of implicit interpolation is introduced which eradicates the problems faced with the explicit interpolation in which it is required to move away from the boundary in the fluid domain in order to complete the interpolation stencil.

Keywords Bi-linear interpolation · Computational fluid dynamics · Immersed boundary method · Inviscid flow · Roe scheme

1 Introduction

Numerical simulations have been in use for the last few decades and have reached high attention for simulating the physical phenomena. A typical Computational Fluid Dynamics(CFD) problem includes geometry definition, mesh generation, numerical solver and post-processing of the simulation results. The most time consuming step is often the mesh generation, especially for complex geometries in traditional CFD. The mesh has to conform to the boundary to be able to depict the geometry precisely. Immersed boundary method is a recent approach to apply the boundary condition to a non-conforming mesh. It simplifies the mesh generation process [1], especially for complex geometries and for dynamic cases. It was proposed by Peskin [2] in 1972 to handle elastic boundaries for simulating blood flow in the heart [3]. As per Rajat Mittal et al. [1], the IB methods can be classified primarily in Continuous Forcing(CF)

✉ Md. Sujaat Ali
md-sujaat.ali@polymtl.ca

[1] Department of Mechanical Engineering, Polytechnique Montréal, Chem. de Polytechnique, Montréal, QC H3T 1J4, Canada

Published online: 17 April 2024

and Discrete Forcing(DF) approach. The CF approach is very useful for elastic bodies whereas DF approaches are useful for solid objects. As part of the DF approach, "sharp interface" immersed boundary approach is suitable for solid boundaries [1] and can be implemented in two different ways. The first one is the Indirect Boundary Condition(IBC) implementation and the second is the Direct Boundary Condition(DBC) implementation. The Indirect Boundary Condition implementation does not provide the required local accuracy. The DBC approach has proved to be the best candidate for simulating compressible flow cases as evident from the work presented in [4, 5] and [6]. Again, the sharp interface method can be implemented in multiple ways which includes ghost cell and cut-cell approaches. Ghost cells are the cells inside the solid region with at least one neighbour inside the fluid. The Cut-cell method uses a control volume which is entirely in the fluid domain but is an irregular polygon near the boundary [7]. Such polygons result from the modified cells containing the boundary.

The first step for IB implementation is the proper cell classification, called tagging. The cells in the solid domain, inside the geometry, are called solid cells. Fluid cells lie in the fluid domain with complete stencils and the cells in the fluid domain with at least one neighbouring cells in the solid domain are classified as intercepted cells. The intercepted cells need special treatment as these will be used for the boundary condition implementation using an interpolation function. The solid cells are not solved and for fluid cells typical Finite Volume(FV) schemes are used. The classification of solid and fluid cells can be done easily using any one of the methods including Ray tracing [8–11], Level set approach [12–14], point in a polygon method and explicit minimum distance approach [15].

There are numerous implementations of IBM for compressible and incompressible flows. The work by Kumar et al. [16] focused on the issues of mass conservation and pressure fluctuations. Picano et al. [17] implemented the pressure-driven turbulent flow in the presence of buoyant particles. Ji et al. [18] used an iterative IB method for solving viscous flows using finite volume. Roy and Acharya [19] coupled IBM with large-Eddy simulations for turbulent flows. Balaras [20] used the bi-linear interpolation for complex geometries and demonstrated flow over a cylinder as a benchmark case. Later Yang and Balaras [21] modified and improved the method for moving geometries as well. Udaykumar et al. [22] used a sharp interface method for moving geometries using cell modification, i.e. converting a cell to a trapezoid, more similar to a cut-cell method. Many of the schemes focused on the ghost cell approach which includes the work of Zhang et al. [23]. One major drawback of this method is that if the size of the geometry or a part of the geometry is relatively thin such that there is no ghost cell inside the geometry, it can not be captured by the numerical method. However, the one-sided interpolation technique discussed in Yang and Balaras et al. [21] removes these drawbacks.

Most of the work present in the literature are based on the explicit interpolation such as the one presented by Yang and Balaras [21]. In this method, for each intercepted cell, interpolation is done by completing a local interpolation stencil. There are different

interpolation schemes that can be applied for constructing the solution for the interface cells. The simplest of these interpolation methods for a 2-dimensional problem is the bi-linear interpolation as presented in Yang and Balaras [21]. Other notable work on linear interpolation includes the work by Mittal et al. [24] which used tri-linear interpolation for 3-D cases. Kim et al. [25] also used bi-linear interpolation but it reverts to a linear interpolation wherever the interpolation stencil is not complete [26]. Least square methods are also used by some authors as interpolation function [27–29]. Qu et al. [30] used the constrained moving least square method(CMLS) for the interpolation which eliminates the instabilities observed in moving least square interpolation. The moving least square enables the formulation with higher-order polynomials with flexible interpolation stencils. However, using high-order polynomials for interpolation could be a source of oscillatory behaviour in the solution [31]. Another popular interpolation scheme is radial basis function for which examples can be found in [32, 33] and [34]. The problem with explicit interpolation appears when the interpolation stencil is not complete and one or more cells constituting the interpolation stencil are part of the intercepted cells. In order to perform explicit interpolation in these cases, the interpolation stencil is moved further in the fluid domain. This is one of the problems reported in Yang Balaras et al. [20] and this has been explained in detail in [35].

In this paper, a one-sided bi-linear interpolation using an image point of the intercepted cell is presented for two different governing equations. At first, it is applied for the diffusion equation and then it is extended to the Euler equations. Moreover as stated previously, in some cases, an explicit interpolation can not be performed because of the incomplete interpolation stencils. For these cases, an implicit interpolation strategy is presented. This eliminates the necessity to identify these special cases where the interpolation cannot be performed and to apply a different interpolation strategy which includes moving away from the boundary and selecting the next nearest fluid cell available. A case with an inclined shock tube is presented which validates the tagging as well as the implicit interpolation strategy for compressible flow cases. It has to be noted here that, although the present implementation is for inviscid flows, the implicit interpolation strategy can be implemented in the same way for compressible and incompressible NS equations as well. Moreover, a new way of implementing slip boundary condition has been introduced.

This paper consists of five different sections. Section 1 introduces the Immersed Boundary approach and overall outlook of the paper. Section 2 focuses on the numerical modelling including the interpolation strategy, boundary conditions and the governing equations. Section 3 focused on the Results and discussions and Sect. 4 is the conclusion where the results obtained and drawbacks are discussed.

2 Numerical model

2.1 Governing equations

2.1.1 Governing equations for thermal diffusion

The heat diffusion equation with a constant thermal conductivity ($\kappa = 1$) and without any source term is given by

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

where T is the temperature.

2.1.2 Governing equations for inviscid compressible flows

The present study is based on the two-dimensional Euler equations. The Euler equations can be written as

$$\rho_t + (\rho u)_x + (\rho v)_y = 0, \tag{2}$$
$$(\rho u)_t + (\rho u^2 + p)_x + (\rho uv)_y = 0, \tag{3}$$
$$(\rho v)_t + (\rho v^2 + p)_y + (\rho uv)_x = 0, \tag{4}$$
$$E_t + [u(E + p)]_x + [v(E + p)]_y = 0, \tag{5}$$

where E is the total energy per unit volume:

$$E = \rho\left(\frac{1}{2}\mathbf{V}^2 + e\right) \tag{6}$$

Equations (2) to (5) can be expressed in a compact from, suitable for CFD applications:

$$U_t + F_x + G_y = 0,$$
$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \qquad F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E + p) \end{bmatrix}, \qquad G = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E + p) \end{bmatrix}. \tag{7}$$

2.2 Numerical method

2.2.1 Diffusion equation

The diffusion problem is solved using a typical Finite Volume Method as described in Versteeg and Malalasekera [36]. Once Eq. (1) is integrated over a control volume, it gives

$$\int_{\Delta V} \frac{\partial^2 \phi}{\partial x^2} \mathrm{d}x.\mathrm{d}y + \int_{\Delta V} \frac{\partial^2 \phi}{\partial y^2} \mathrm{d}x.\mathrm{d}y = 0. \tag{8}$$

Assuming the areas for east, west, north and south faces to be A_e, A_w, A_n and A_s, respectively, such that $A_e = A_w = \Delta y$ and $A_n = A_s = \Delta x$ and Δv as the control volume. Converting volume integral to boundary integral, Eq. 8 finally becomes

$$\left[A_e \left(\frac{\partial \phi}{\partial x} \right)_e - A_w \left(\frac{\partial \phi}{\partial x} \right)_w \right] + \left[A_n \left(\frac{\partial \phi}{\partial x} \right)_n - A_s \left(\frac{\partial \phi}{\partial x} \right)_s \right] = 0. \tag{9}$$

The numerically discretised fluxes through the faces for the control volume can be written as

$$\text{East Face flux} = A_e \frac{\phi_E - \phi_Q}{\delta_{QE}}, \tag{10a}$$

$$\text{West Face flux} = A_w \frac{\phi_Q - \phi_W}{\delta_{WQ}}, \tag{10b}$$

$$\text{South Face flux} = A_s \frac{\phi_Q - \phi_S}{\delta_{QS}}, \tag{10c}$$

$$\text{North Face flux} = A_n \frac{\phi_N - \phi_Q}{\delta_{NQ}}. \tag{10d}$$

Finally, the final fluxes can be discretised in the form

$$a_q \phi_Q = a_e \phi_E + a_w \phi_W + a_n \phi_N + a_s \phi_S, \tag{11}$$

where

$$\begin{aligned} &a_q = \left(\frac{A_e}{\delta_{QE}} + \frac{A_w}{\delta_{WQ}} + \frac{A_n}{\delta_{NQ}} + \frac{A_s}{\delta_{QS}} \right); \; a_e = \left(\frac{A_e}{\delta_{QE}} \right); \; a_w = \left(\frac{A_w}{\delta_{QW}} \right); \\ &a_n = \left(\frac{A_n}{\delta_{NQ}} \right); \text{ and } \; a_s = \left(\frac{A_s}{\delta_{QS}} \right), \end{aligned} \tag{12}$$

where δ_{QE} is the distance between points Q and E and all other distances are defined in the same way, as shown in Fig. 1.

Equation (11) is applied to all cells in the fluid region, except for the interface cells.

2.2.2 Euler equations

Roe scheme is essentially a flux difference splitting scheme. It was first presented by Roe [37] and since has been used and modified in a number of ways. For the present case, the original Roe scheme has been used as presented in [38].

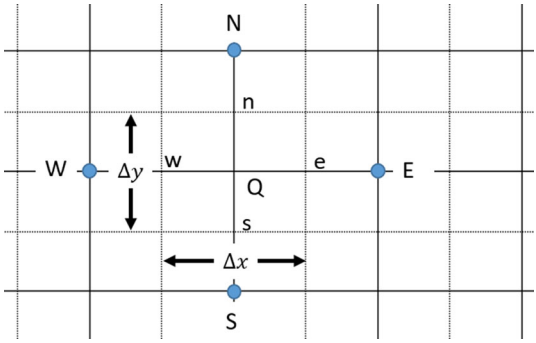


Fig. 1 2-Dimensional mesh representation for diffusion problem

The inter-cell flux, as per [38], is given as

$$F_{i+\frac{1}{2}} = \frac{1}{2}(F_R + F_L) - \frac{1}{2}\sum_{i=1}^{m} \tilde{\alpha_i} |\tilde{\lambda_i}| \tilde{K}^{(i)}, \tag{13}$$

where $m = 3$ for one-dimensional case and F_R and F_L are the flux values for the right and left face of the i^{th} cell calculated from the primitive variables from the previous time step. $\tilde{\alpha}$ is the wave strength and $\tilde{K}$ is the right eigenvector with averaged variables.

Dimensional splitting is used to get the 2-D Roe scheme from 1-D Roe scheme as defined in [38]. The sweeps are handled by a single subroutine. In the x-sweep, following equations are solved:

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}_x = 0.$$

In the y-sweep, following equations are solved:

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}_t + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}_y = 0.$$

The scheme is explicit and transient and for x-sweep, the explicit conservative form looks like

$$U_{i,j}^{n+\frac{1}{2}} = U_{i,j}^{n} + \frac{\Delta t}{\Delta x}\left[F_{i-\frac{1}{2},j}^{n} - F_{i+\frac{1}{2},j}^{n}\right]. \tag{14}$$

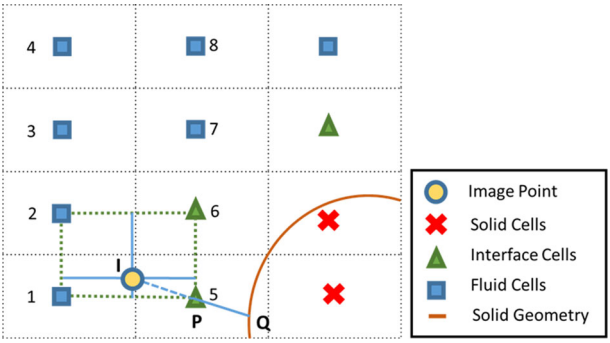


Fig. 2 Bi-linear interpolation with cell classification

$F^n_{i+\frac{1}{2},j}$ is the flux at the cell interface $x_{i+\frac{1}{2}}$ as described in Eq. (13). Similarly, for y-sweep

$$U^{n+1}_{i,j} = U^{n+\frac{1}{2}}_{i,j} + \frac{\Delta t}{\Delta x}\left[G^{n+\frac{1}{2}}_{i,j-\frac{1}{2}} - G^{n+\frac{1}{2}}_{i,j+\frac{1}{2}}\right]. \tag{15}$$

2.3 Tagging and immersed boundary implementation

For implementing immersed boundary, the tagging of the geometry with respect to the Cartesian mesh is required. In the present work, tagging has been done using point in a polygon approach. The tagging schematic is presented in Fig. 3. While moving in the clockwise direction along the geometric points, if, for a particular cell centre, $|\vec{A}X\vec{B}| < 0$ for all the points on the geometry, then the cell centre lies inside the geometry; otherwise, it lies outside. Once, inside and outside have been determined, one needs to find the intercepted cells. For that, one needs to look into the cells which lie inside the fluid domain and for which the stencils are not complete.

Once the tagging is done, the cells can be broadly classified into three different types namely intercepted/interface cells, solid cells and the fluid cells. After that, the numerical solver is applied for the fluid cells. The solid cells are not solved and for the interface cells, interpolation schemes have to be applied. Figure 2 shows the classification of the cell types and stencil for bi-linear interpolation. The beauty of this scheme is that it is possible to tag the geometry even if the width of the geometry is less than the mesh size ("thin geometries") and the interpolation function can be applied to it.

2.4 Bi-linear interpolation

2.4.1 Diffusion case

For implementing bi-linear interpolation with respect to immersed boundary, at first an image point "I" has to be found out which is $\overrightarrow{PQ} = \overrightarrow{IP}$ which can be interpreted

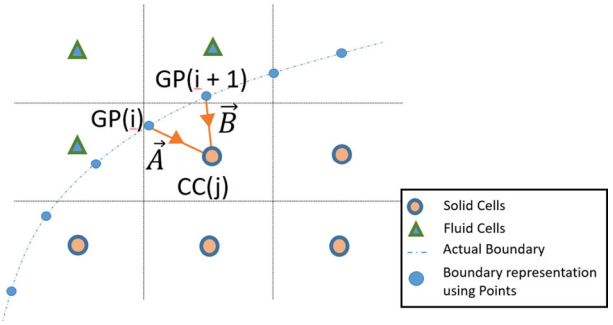


Fig. 3 Cross-product depiction

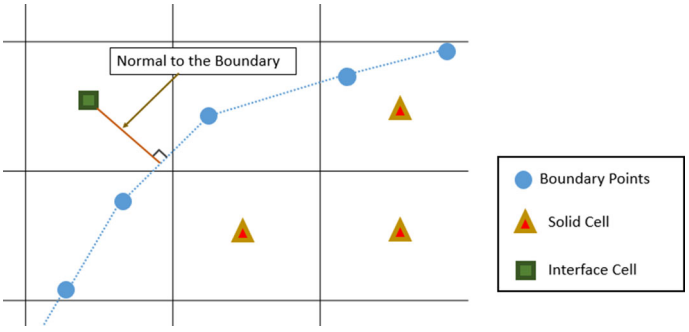


Fig. 4 Normal to the boundary

from Fig. 2. Also, it has to be noted that $\overrightarrow{PQ}$ is perpendicular to the boundary. The geometry consists of set of points and ultimately a set of line segments joining two consecutive points. $\overrightarrow{PQ}$ is perpendicular to the nearest line segments joining two consecutive points on the geometry. One example is shown in Fig. 4. After that, the image point is found out, upon this there are two different equations for a Dirichlet boundary condition:

$$\phi_P = (\phi_I + \phi_Q)/2. \tag{16}$$

From bi-linear interpolation, Fig. 2

$$\phi_I = a_1\phi_1 + a_2\phi_2 + a_6\phi_6 + a_5\phi_5; \qquad \text{where} \quad \phi_5 = \phi_p. \tag{17}$$

The coefficients depend on the geometric location of the cell centres. Thus, from Equations (16) and (17), ϕ_P can be found out. The image point, I as depicted in Fig. 2, lies along the normal to the geometry. There should be sufficient number of points to define the geometry, especially when the geometry consists of curves.

In Fig. 2, cell number 5 is an interface cell; therefore, the coefficients are to be found using the bi-linear interpolation described by Eq. (17). The ingenuity of the proposed method lies in the way interpolation is done. In some of the previous cited papers

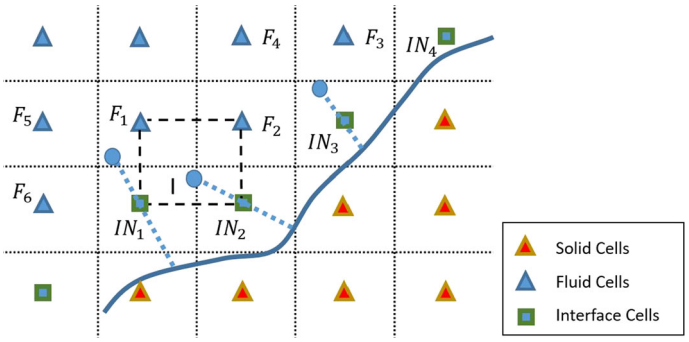


Fig. 5 Problem with explicit interpolation as presented in [20]

[20], the interpolation is done locally and if the interpolation stencil is not complete, it is required to move further away from the boundary in the fluid domain. However, this might induce error in the interpolation process. Moreover, finding whether the interpolation stencil is complete or not adds to the computational cost.

In this paper, we introduce the implicit interpolation strategy. This means that the interpolation is not done explicitly but through the construction of a system of equations which are solved at once. One such case is presented in Fig. 5 where the interpolation stencil is not complete. As per [20], it is not possible to do a regular bi-linear local or explicit interpolation for point IN_2 as the interpolation stencil includes another interface cell IN_1. This becomes a special case and in order to perform a bi-linear interpolation, the image point is moved further in the fluid domain. In this way, the stencil for bi-linear interpolation is complete.

With implicit interpolation, the problem described above is not encountered. The system of equations obtained for each of the interface cells are assembled in a matrix and solved. Considering IN_2 in Fig. 6, the equation for bi-linear interpolation for primitive variable represented as $f(x, y)$ can be written as

$$\begin{aligned} f(x,y) = {} & B_{11} f(x_{F_5}, y_{F_5}) + B_{21} f(x_{IN_2}, y_{IN_2}) + B_{12} f(x_{F_1}, y_{F_1}) \\ & + B_{22} f(x_{F_2}, y_{F_2}), \end{aligned} \tag{18}$$

where the coefficients B_{11}, B_{21}, B_{12} and B_{22} can be found by solving the following matrix system

$$\begin{bmatrix} B_{11} \\ B_{21} \\ B_{12} \\ B_{22} \end{bmatrix} = \left(\begin{bmatrix} 1 & x_{F_5} & y_{F_5} & x_{F_5} y_{F_5} \\ 1 & x_{IN_2} & y_{IN_2} & x_{IN_2} y_{IN_2} \\ 1 & x_{F_1} & y_{F_1} & x_{F_1} y_{F_1} \\ 1 & x_{F_2} & y_{F_2} & x_{F_2} x_{F_2} \end{bmatrix}^{-1} \right)^T \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}. \tag{19}$$

Considering a Dirichlet boundary condition and applying Eq. (16) and Eq. (17), Eq (18) will become

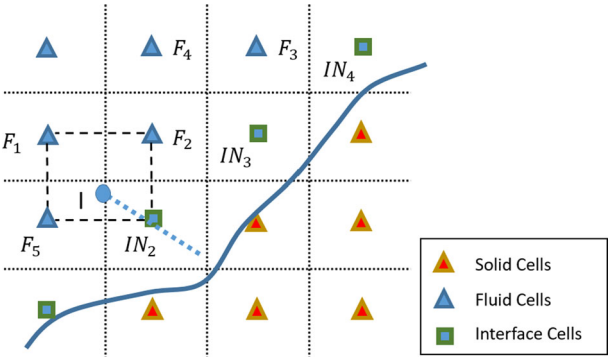


Fig. 6 Implicit interpolation; Only one interface cell in the interpolation stencil

$$f(x,y)(2-B_{21}) = B_{11}f(x_{F_5},y_{F_5}) + B_{12}f(x_{F_1},y_{F_1}) + B_{22}f(x_{F_2},y_{F_2}) + \phi_B, \tag{20}$$

where ϕ_B is the boundary value of the variable.

For a two-point system, the implicit interpolation scheme can be implemented easily. Considering interface cells IN_3 and IN_2 in Fig. 6 and assuming local coefficients for IN_3 as C_{11}, C_{21}, C_{12} and C_{22}, the matrix system to find the flow variables looks like

$$\begin{bmatrix} (2-B_{21}) & 0 \\ 0 & (2-C_{21}) \end{bmatrix} \begin{bmatrix} f(x_{IN_2},y_{IN_2}) \\ f(x_{IN_3},y_{IN_3}) \end{bmatrix} = \begin{bmatrix} B_{11}f(x_{F_5},y_{F_5}) + B_{12}f(x_{F_1},y_{F_1}) + B_{22}f(x_{F_2},y_{F_2}) + \phi_{IN_2} \\ C_{11}f(x_{F_2},y_{F_2}) + C_{12}f(x_{F_4},y_{F_4}) + C_{22}f(x_{F_3},y_{F_3}) + \phi_{IN_3} \end{bmatrix}. \tag{21}$$

This is a typical bi-linear interpolation that can be done in the explicit way as well. For the problem faced in Fig. 5, the implicit interpolation strategy has to be used to avoid the failure of explicit interpolation. Using Eqs. (16) and (17), the system of equations for two interface cells in a single bi-linear interpolation stencil can be written separately. For IN_1:

$$f(x_{IN_1},y_{IN_1})(2-B_{21}) = B_{11}f(x_{F_6},y_{F_6}) + B_{12}f(x_{F_5},y_{F_5}) + B_{22}f(x_{F_1},y_{F_1}). \tag{22}$$

Similarly, for IN_2

$$f(x_{IN_2},y_{IN_2})(2-D_{21}) - D_{11}f(x_{IN_1},y_{IN_1}) = D_{12}f(x_{F_1},y_{F_1}) + D_{22}f(x_{F_2},y_{F_2}). \tag{23}$$

Also for IN_3:

$$f(x_{IN_3},y_{IN_3})(2-C_{21}) = C_{11}f(x_{F_2},y_{F_2}) + C_{12}f(x_{F_4},y_{F_4})$$

$$+C_{22} f(x_{F_3}, y_{F_3}), \tag{24}$$

where B_{ij}, C_{ij} and D_{ij} are the coefficients obtained using Eq. (19).

Therefore, the final matrix will look like

$$\begin{bmatrix} (2-B_{21}) & 0 & 0 \\ -D_{11} & (2-D_{21}) & 0 \\ 0 & 0 & (2-C_{21}) \end{bmatrix} \begin{bmatrix} f(x_{IN_1}, y_{IN_1}) \\ f(x_{IN_2}, y_{IN_2}) \\ f(x_{IN_3}, y_{IN_3}) \end{bmatrix} = \begin{bmatrix} B_{11} f(x_{F_6}, y_{F_6}) + B_{12} f(x_{F_5}, y_{F_5}) + B_{22} f(x_{F_1}, y_{F_1}) \\ D_{12} f(x_{F_1}, y_{F_1}) + D_{22} f(x_{F_2}, y_{F_2}) \\ C_{11} f(x_{F_2}, y_{F_2}) + C_{12} f(x_{F_4}, y_{F_4}) + C_{22} f(x_{F_3}, y_{F_3}) \end{bmatrix}. \tag{25}$$

Therefore, for a problem with N interface cells, the left side matrix will be $[N * N]$.

2.4.2 Euler equations

The sequence for tagging and the implicit interpolation is same as that presented in the previous section for diffusion. For Euler equations, a Neumann boundary condition is required for pressure and density. Therefore, the value of the primitive variable at I is considered equal to its value at IN_2, in Fig. 6, thus Eq (18) becomes

$$f(x, y)(1 - B_{21}) = B_{11} f(x_{F_5}, y_{F_5}) + B_{12} f(x_{F_1}, y_{F_1}) + B_{22} f(x_{F_2}, y_{F_2}). \tag{26}$$

For a two-point system, considering interface cells IN_3 and IN_2 in Fig. 6 and assuming local coefficients for IN_3 as C_{11}, C_{21}, C_{12} and C_{22}, the matrix system to find the flow variables looks like,

$$\begin{bmatrix} (1-B_{21}) & 0 \\ 0 & (1-C_{21}) \end{bmatrix} \begin{bmatrix} f(x_{IN_2}, y_{IN_2}) \\ f(x_{IN_3}, y_{IN_3}) \end{bmatrix} = \begin{bmatrix} B_{11} f(x_{F_5}, y_{F_5}) + B_{12} f(x_{F_1}, y_{F_1}) + B_{22} f(x_{F_2}, y_{F_2}) \\ C_{11} f(x_{F_2}, y_{F_2}) + C_{12} f(x_{F_4}, y_{F_4}) + C_{22} f(x_{F_3}, y_{F_3}) \end{bmatrix}. \tag{27}$$

Similarly, for a three-point system, as in Fig. 5, for IN_1 and IN_3, the implementation of interpolation is very straightforward. Assuming the local coefficients for IN1 as D11, D21, D12 and D22. The interpolation equation for IN_2 can be written as

$$f(x, y)(1 - B_{21}) - B_{11} f(x_{IN_1}, y_{IN_1}) = B_{12} f(x_{F_1}, y_{F_1}) + B_{22} f(x_{F_2}, y_{F_2}). \tag{28}$$

The matrix system to find the flow variables looks like

$$\begin{bmatrix} (1-D_{21}) & 0 & 0 \\ -B_{11} & (1-B_{21}) & 0 \\ 0 & 0 & (1-C_{21}) \end{bmatrix} \begin{bmatrix} f(x_{IN_1}, y_{IN_1}) \\ f(x_{IN_2}, y_{IN_2}) \\ f(x_{IN_3}, y_{IN_3}) \end{bmatrix} = \begin{bmatrix} D_{11} f(x_{F_6}, y_{F_6}) + D_{12} f(x_{F_5}, y_{F_5}) + D_{22} f(x_{F_5}, y_{F_5}) \\ B_{12} f(x_{F_1}, y_{F_1}) + B_{22} f(x_{F_2}, y_{F_2}) \\ C_{11} f(x_{F_2}, y_{F_2}) + C_{12} f(x_{F_4}, y_{F_4}) + C_{22} f(x_{F_3}, y_{F_3}) \end{bmatrix}. \tag{29}$$

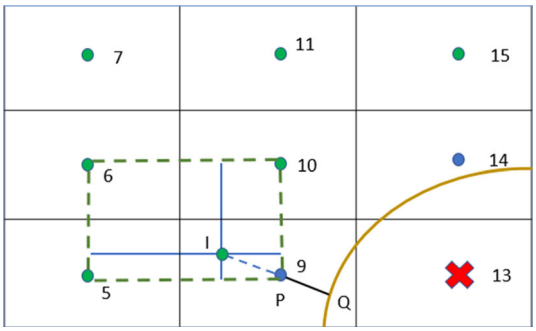


Fig. 7 Bi-linear interpolation

Again, for a problem with N interface cells, the left side matrix will be $[N * N]$. One thing to be noted here is that after x-sweep, Eq. (14), the implicit bi-linear interpolation is done for the interface cells such that the primitive variables are present for calculating the fluxes for the y-sweep. Therefore, the interpolation is done twice for a single time step: for x-seep and then for y-sweep. If dimensional splitting is not used, the interpolation is done once for one time step.

2.5 Boundary conditions

The boundary can be either Dirichlet or Neumann. Both can be represented easily with the interpolation scheme. For the Euler equations case, Neumann boundary condition has to be applied on the wall for pressure and density and for velocity, a slip boundary condition has to be implemented.

The Neumann boundary condition for pressure and density is very straight forward. As shown in Fig. 7, the values are interpolated at point I, assuming that the primitive variable value does not change in the normal direction such that $pressure$ or $density$ at point I is same as that of P and Q. For velocity, in order to implement slip boundary condition, a two-step procedure is implemented:

1. The image point I, shown in Fig. 7, is not considered and the velocity is extrapolated at point P using the points 5, 6 and 10.

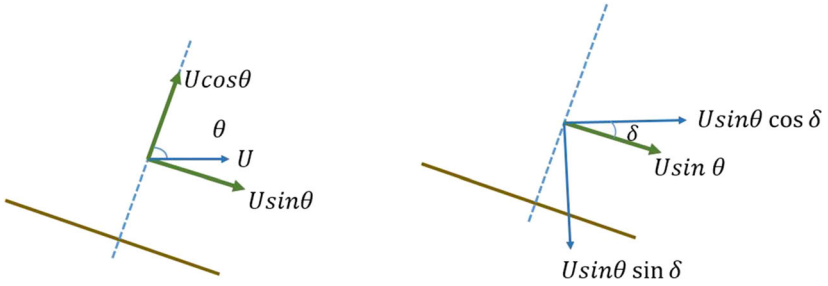
2. The velocity U is divided into tangential and normal components to the solid surface. The normal component is neglected as it will be zero for a slip boundary condition. The tangential component is broken into X and Y-directions contributing to the velocity in those directions accordingly, as presented in Fig. 8. This is repeated for V.

From Fig. 8 and 9, final velocity obtained in X-direction is $Usin\theta cos\delta - Vsin\theta sin\delta$ and in y-direction is $Vsin\delta cos\theta - Usin\theta sin\delta$.

The implementation of the Neumann boundary condition is also unique and gives a more geometric sense to the boundary condition implementation.

The sequence of steps for IB implementation for Euler equation includes:

1. Defining the geometry discretely using a set of points. Define the number of elements in X and Y-directions for the mesh.



(a) Normal and tangential components of U (b) Final U velocity components

Fig. 8 U velocity components

(a) Normal and tangential velocity components of V (b) Final V velocity components

Fig. 9 V velocity components

2. Tagging of the geometry with respect to the background mesh. Identify the solid, fluid and interface cells.
3. Initialise the fluid domain
4. Use Roe scheme for X-sweep
5. Use implicit bi-linear interpolation as described in Sect. 2.4.1 for pressure and density. For velocity, use the slip boundary condition presented in Sect. 2.5
6. Use Roe scheme for Y-sweep
7. Use implicit bi-linear interpolation as described in Sect. 2.4.1 for pressure and density. For velocity, use the slip boundary condition presented in Sect. 2.5.

3 Results and discussions

3.1 Diffusion problem

The schematic of the problem is described in Fig. 10, consisting of a domain bounded by two concentric circles with Dirichlet boundary condition. The inner circle has a temperature T_1 and the outer circle has temperature T_2. The heat conduction for this problem is governed by Eq. 1, which is the heat diffusion equation with a constant thermal conductivity, ($\kappa = 1$), and without any source term.

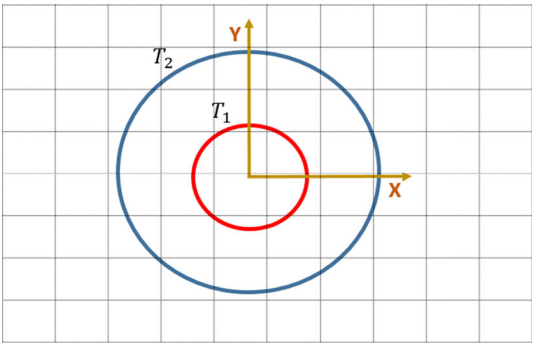


Fig. 10 Geometry Definition

The analytical solution, as presented in [39], for this case, is given by Eqs. (30) and (31):

$$\text{Heat Transfer[watts]} = q_r = 2\pi k(T_1 - T_2)/ln(r_2/r_1), \tag{30}$$

$$\text{Temperature[K]} = T(r) = \frac{(T_1 - T_2)}{ln(r_2/r_1)} ln\left(\frac{r}{r_2}\right) + T_2, \tag{31}$$

where q_r is the heat transfer rate, and $T(r)$ is the temperature at radius r.

The numerical solution is shown in Fig. 11 for a 200 x 200 cells in the mesh:

The variation of temperature in the radial direction is compared with the analytical solution in Fig. 12. The results match with the analytical solution and are accurate.

A convergence analysis is performed to verify the order of convergence that is obtained by the present numerical algorithm. This consists of finding the error(norm) (L2 or L1), against the number of points. In present case, convergence analysis has been done using the L2 norm and is given by Eq. (32):

$$Error = L2 = \sum_{i=1}^{M} \sqrt{(T_{i,analytical} - T_{i,numerical})^2}/Nx, \tag{32}$$

where Nx is the total number points considered for the numerical calculation.

The convergence analysis for the numerical solution using L2 norm is shown in Fig. 13. The order of convergence obtained is of second order, which is expected.

The error analysis for the change in the temperature in the normal direction is presented in Fig. 15, which is basically the error calculation for the flux. Calculation of the analytical fluxes has been modified such that the numerical and analytical fluxes are compared on the same basis. For calculating the analytical fluxes, Eq. (33) is used.

In Fig. 14,

$$\left(\frac{dT}{dn}\right)_{\text{Analytical}} = \frac{T_{\text{Boundary}} - T_{\text{Analytical}}}{\Delta n}. \tag{33}$$

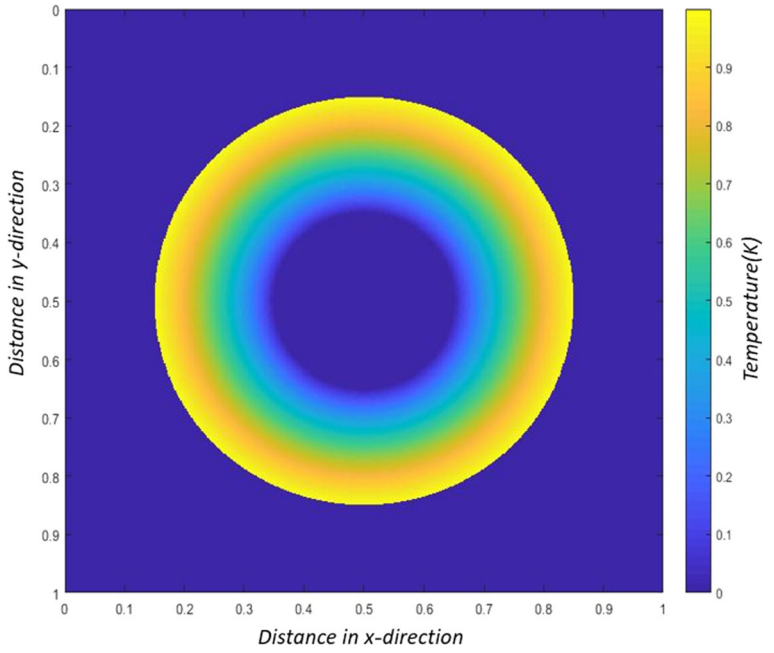


Fig. 11 Numerical temperature distribution in x and y-directions : Concentric cylinders at a temperature differential

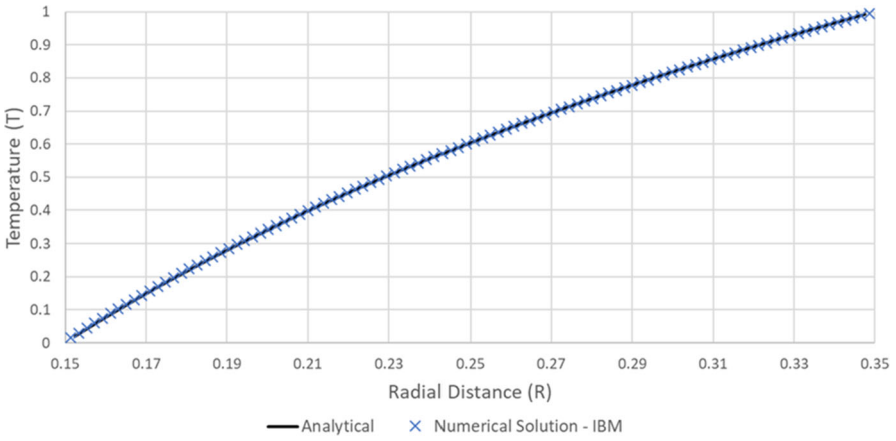


Fig. 12 Comparison of Temperature[T] for numerical and analytical solutions for concentric cylinders in radial direction[R]

Similarly, for the numerical solution,

$$\left(\frac{dT}{dn}\right)_{\text{Numerical}} = \frac{T_{\text{Boundary}} - T_{\text{Numerical}}}{\Delta n}, \tag{34}$$

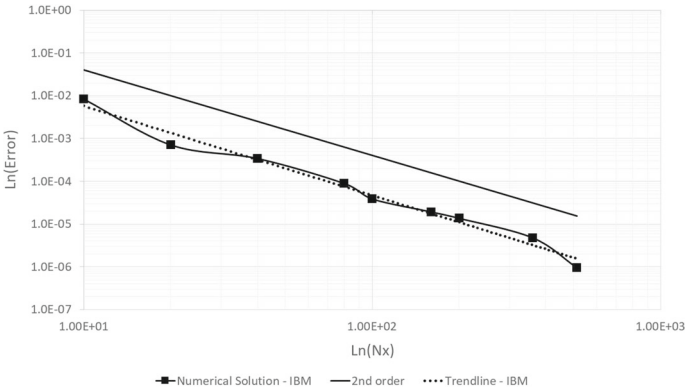


Fig. 13 Convergence Analysis: Temperature

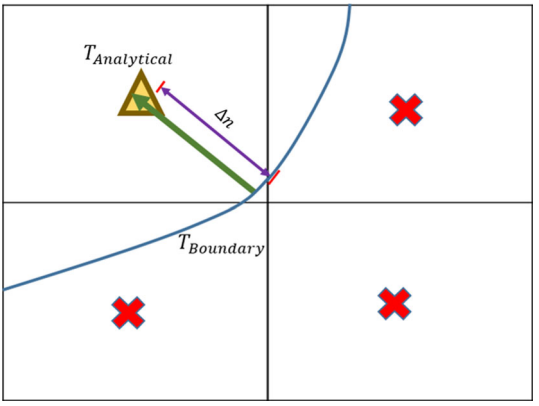


Fig. 14 Calculating analytical temperature for concentric cylinder case. Here, $T_{Analytical}$ is the analytical temperature calculated at the cell centre and $T_{Boundary}$ is the temperature on the boundary

where $T_{\text{Numerical}}$ is the interface cell centre where solution is obtained using the interpolation function.

3.2 Euler equations

The shock tube problem schematic is shown in Fig. 16. The high pressure $P_l = 10$, high density $\rho_l = 11.6$, $u_l = 0$ and $V_l = 0$ is set on the left. The low pressure $P_r = 1.0$, low density $\rho_r = 1.16$, $u_r = 0$ and $V_r = 0$ is set on the right. When the diaphragm ruptures, the shock wave moves to the right and the rarefaction wave moves to the left. This problem has an exact solution which is explained in Toro [38], and therefore, this problem is a good benchmark test for the numerical method.

The results obtained for this problem are presented in Fig. 17c. The convergence for the inclined shock tube is represented in Fig. 18. It has to be noted that the order of convergence for the shock tube case is found to be around 0.5 which is less than

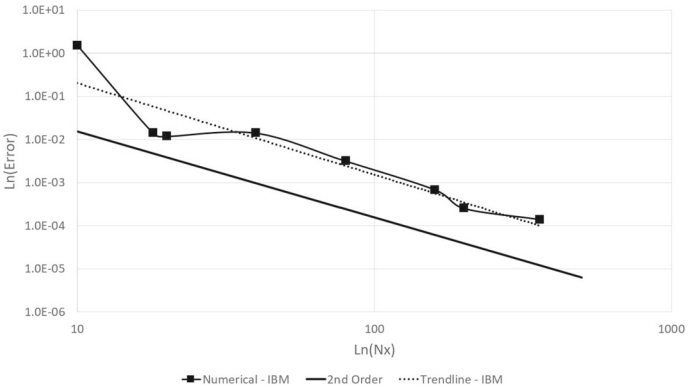


Fig. 15 Convergence for dT/dn with 1000 points on the geometry and assuming that the geometry is represented by straight line in between two points. The error has been reported with respect to L2 norm

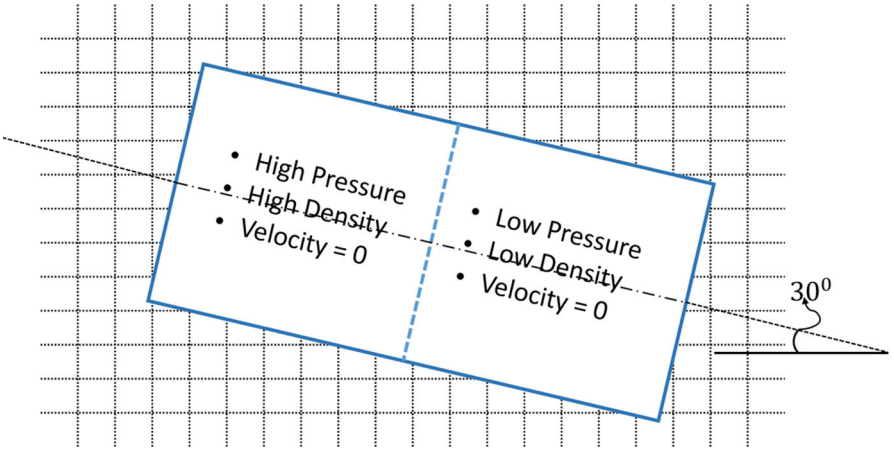
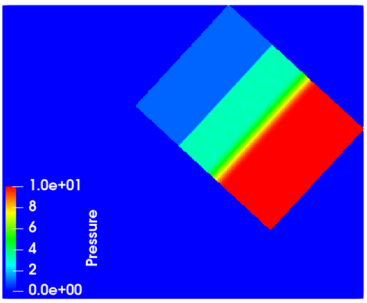


Fig. 16 Inclined Shock Tube schematic with an inclination of 30^0

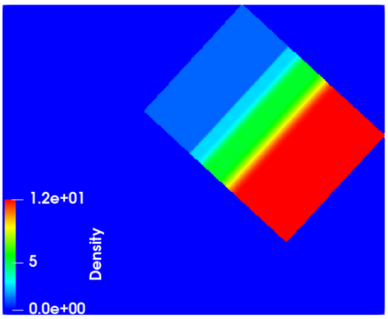
the expected value of 1 and this is in agreement with the similar test cases of flow discontinues and shock waves for a first-order flow solver [40].

3.3 Computational efficiency

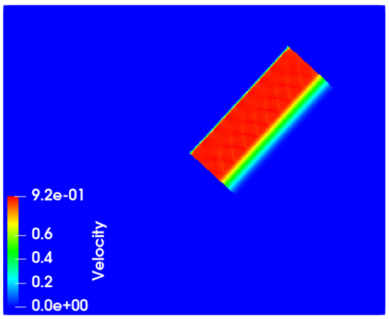
The explicit and implicit interpolation methods are compared based on the CPU time that is required for each type of interpolation. For this, a case of straight immersed shock tube is taken. The test case of inclined shock tube cannot be used for the comparison as the explicit interpolation will fail in that case. The comparison is presented in Table 1. The time for implicit interpolation is normalised by the CPU time of the explicit interpolation. As expected, the implicit interpolation is more computationally expensive than the explicit interpolation, but the increase is reasonable considering the improvement in the overall robustness of the approach.



(a) Pressure variation inside the shock tube



(b) Density variation inside the shock tube



(c) Velocity variation inside the shock tube

Fig. 17 Pressure, Velocity and Density contours for inclined shock tube for 500 X 500 background mesh with the angle of inclination of 140^0

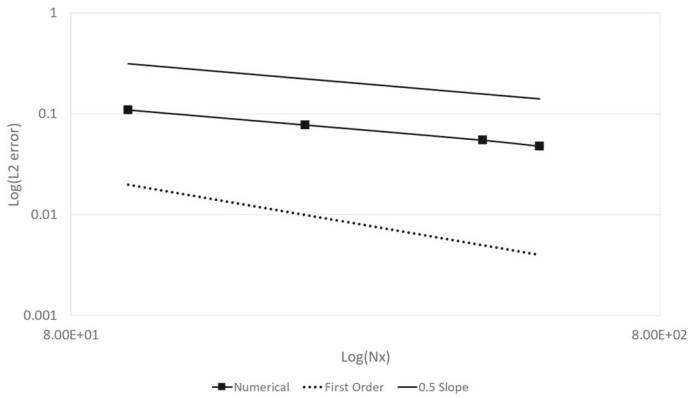


Fig. 18 Pressure convergence for inclined shock tube with implicit bi-linear interpolation for an angle of rotation of 140^0

Table 1 Comparison for computational time of explicit and implicit interpolation

S. No	Mesh size	Explicit time (ET)	Implicit time (IT)
1	300*300	1	1.151

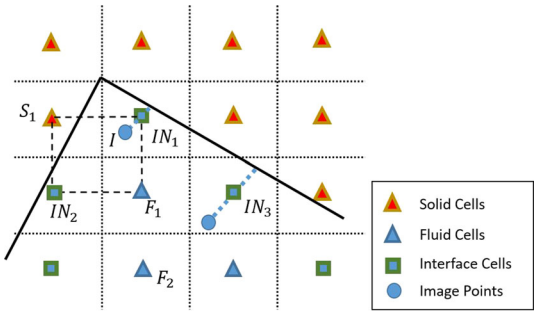


Fig. 19 Demonstration of corner cell problem for shock tube

3.4 Problems with shock tube for corner cells

For the two-dimensional cases, some problems can still be present with the sharp corner cells, as found at the four corners of the shock tube problem presented earlier. The non-availability of the fluid cells to fill the coefficients in the implicit bi-linear equation matrix leads to the failure of even the implicit approach. As presented in Fig. 19, it is clear that the bi-linear interpolation cannot be implemented at the corner cells as one of the cells that constitute the interpolation stencil is a solid cell. For these cases, other interpolation strategies like least square or radial basis functions are recommended.

In order to get rid of this problem, one suggestion is to use a linear interpolation with three points. Even at the corners, it is ensured that three nearest points are available from the fluid and interface regions and the interpolation has to be done implicitly. Mathematically, it is similar to the bi-linear interpolation expressed in Eq. (19) except that the fourth point has to be dropped.

4 Conclusion

The idea of implicit interpolation gets rid of the problems faced with explicit interpolation especially when the interpolation stencil includes another interface cells. The use of the image point is helpful for implementing the Neumann boundary condition for pressure and density. However, as with most of the IB methods drawback, the problem of the corner cells for the shock tube case remains and can be solved even using the three-point linear implicit interpolation. Although it is implemented for inviscid compressible flows(Euler equations), same can be implemented easily for NS solvers. Also, the implicit interpolation can be implemented for higher order interpolation methods as well in a similar way. However, the computational cost is expected to increase, as is the case for bi-linear implicit interpolation.

Acknowledgements Authors would like to acknowledge GE Grid Solutions, Arc Technology Research Centre, France, for their financial and technical support.

Author Contributions All authors reviewed the manuscript. Md Sujaat Ali wrote the manuscript and coded all the stuff. Ricardo, Renan and Moustafa provided the tagging methodology. Jean-Yves helped in the implicit interpolation strategy and the slip boundary condition.

Declarations

Conflict of interest The authors declared that they have no conflict of interest.

References

1. Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37(1):239–261
2. Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):250–271
3. Wang X, Deiterding R, Liang J, Cai X, Zhao W (2022) A second-order-accurate immersed boundary ghost-cell method with hybrid reconstruction for compressible flow simulations. Comput Fluids 237:105314
4. Boukharfane R, Eugenio Ribeiro FH, Bouali Z, Mura A (2018) A combined ghost-point-forcing/direct-forcing immersed boundary method (ibm) for compressible flow simulations. Comput Fluids 162:91–112
5. Liu J, Zhao N, Hu O, Goman M, Li XK (2013) A new immersed boundary method for compressible Navier-Stokes equations. Int J Comput Fluid Dyn 27(3):151–163
6. Zhang Y, Zhou CH (2014) An immersed boundary method for simulation of inviscid compressible flows. Int J Numer Methods Fluids 74(11):775–793
7. Nikfarjam F, Cheny Y, Botella O (2018) The ls-stag immersed boundary/cut-cell method for non-Newtonian flows in 3d extruded geometries. Comput Phys Commun 226:67–80
8. de Tullio MD, De Palma P, Iaccarino G, Pascazio G, Napolitano M (2007) An immersed boundary method for compressible flows using local grid refinement. J Comput Phys 225(2):2098–2117
9. De Palma P, de Tullio MD, Pascazio G, Napolitano M (2006) An immersed-boundary method for compressible viscous flows. Comput Fluids 35(7):693–702. Special Issue Dedicated to Professor Stanley G. Rubin on the Occasion of his 65th Birthday
10. Khalili ME, Larsson M, Müller B (2019) High-order ghost-point immersed boundary method for viscous compressible flows based on summation-by-parts operators. Int J Numer Methods Fluids 89(7):256–282
11. Al-Marouf M, Samtaney R (2017) A versatile embedded boundary adaptive mesh method for compressible flow in complex geometry. J Comput Phys 337:339–378
12. Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J Comput Phys 79(1):12–49
13. Edwards JR, Choi J-I, Ghosh S, Gieseking DA, Eischen JD (2010) An immersed boundary method for general flow applications. In: Fluids engineering division summer meeting, vol. ASME 2010 3rd joint US-European fluids engineering summer meeting: volume 1, symposia—parts A, B, and C, pp. 2461–2469
14. Kapahi A, Mousel J, Sambasivan S, Udaykumar HS (2013) Parallel, sharp interface eulerian approach to high-speed multi-material flows. Comput Fluids 83:144–156. Numerical methods for highly compressible multi-material flow problems
15. Luo K, Mao C, Zhuang Z, Fan J, Haugen NEL (2017) A ghost-cell immersed boundary method for the simulations of heat transfer in compressible flows under different boundary conditions part-ii: complex geometries. Int J Heat Mass Transf 104:98–111
16. Kumar M, Roy S, Ali MS (2016) An efficient immersed boundary algorithm for simulation of flows in curved and moving geometries. Comput Fluids 129:159–178

17. Picano F, Breugem W-P, Brandt L (2015) Turbulent channel flow of dense suspensions of neutrally buoyant spheres. J Fluid Mech 764:463–487
18. Ji C, Munjiza A, Williams JJR (2012) A novel iterative direct-forcing immersed boundary method and its finite volume applications. J Comput Phys 231(4):1797–1821
19. Roy S, Acharya S (2012) Scalar mixing in a turbulent stirred tank with pitched blade turbine: role of impeller speed perturbation. Chem Eng Res Des 90(7):884–898
20. Balaras E (2004) Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. Comput Fluids 33(3):375–404
21. Jianming Yang EB (2005) An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 215:12–40
22. Udaykumar HS, Mittal PRKR (2001) A sharp interface cartesian grid method for simulating flows with complex moving boundaries. J Comput Phys 174:345–380
23. Zhang X, Gu X, Ma N (2021) A ghost-cell immersed boundary method on preventing spurious oscillations for incompressible flows with a momentum interpolation method. Comput Fluids 220:104871
24. Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852
25. Kim J, Kim D, Choi H (2001) An immersed-boundary finite-volume method for simulations of flow in complex geometries. J Comput Phys 171(1):132–150
26. Gilmanov A, Sotiropoulos F (2005) A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies. J Comput Phys 207(2):457–492
27. Bale R, Bhalla APS, Griffith BE, Tsubokura M (2021) A one-sided direct forcing immersed boundary method using moving least squares. J Comput Phys 440:110359
28. Spandan V, Lohse D, de Tullio MD, Verzicco R (2018) A fast moving least squares approximation with adaptive Lagrangian mesh refinement for large scale immersed boundary simulations. J Comput Phys 375:228–239
29. Haji Mohammadi M, Sotiropoulos F, Brinkerhoff J (2019) Moving least squares reconstruction for sharp interface immersed boundary methods. Int J Numer Methods Fluids 90(2):57–80
30. Qu Y, Shi R, Batra RC (2018) An immersed boundary formulation for simulating high-speed compressible viscous flows with moving solids. J Comput Phys 354:672–691
31. Al-Marouf M, Samtaney R (2017) A versatile embedded boundary adaptive mesh method for compressible flow in complex geometry. J Comput Phys 337:339–378
32. Xin J, Chen Z, Shi F, Shi F, Jin Q (2021) A radial basis function-based ghost cell method for complex rigid or flexible moving boundary flows. Int J Comput Methods 18(01):2050025
33. Toja-Silva F, Favier J, Pinelli A (2014) Radial basis function (rbf)-based interpolation and spreading for the immersed boundary method. Comput Fluids 105:66–75
34. Liu J, Zhao N, Hu O, Goman M, Li XK (2013) A new immersed boundary method for compressible Navier-Stokes equations. Int J Comput Fluid Dyn 27(3):151–163
35. Ali MS (2020) Two dimensional compressible flow solver for moving geometries using immersed boundary method. Master's thesis, Polytechnique Montréal
36. Versteeg HK, Malalasekera W (1995) An introduction to computational fluid dynamics—the finite volume method. Addison-Wesley-Longman, p 1257
37. Roe PL (1997) Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 135(2):250–250
38. Toro EF (2009) Riemann solvers and numerical methods for fluid dynamics, chap. 11, 16, 3rd edn. Springer, Berlin, pp 345–578
39. Bergman TL, Lavine AS, Incropera FP, DeWitt DP (2017) Fundamentals of heat and mass transfer. Wiley, New York
40. Oberkampf WL, Roy CJ (2010) Verification and validation in scientific computing. Cambridge University Press, Cambridge