LA-UR-15-21544
Approved for public release; distribution is unlimited.

Title: Three-Dimensional Local ALE-FEM Method for Fluid Flow in Domains Containing Moving Boundaries/Objects Interfaces

Author(s): Carrington, David Bradley

Intended for: International Journal for Numerical Methods in Fluids

Issued: 2015-03-05 (rev.1)

Three-Dimensional Local ALE-FEM Method for Fluid Flow in Domains Containing Moving Boundaries/Objects Interfaces

David B. Carrington
Los Alamos National Laboratory
Theoretical Division T3
Los Alamos, NM 87545

A. K. M. Monayem H. Mazumder and Juan C. Heinrich
University of New Mexico
Department of Mechanical Engineering
Albuquerque, NM 87131

Abstract

A three-dimensional finite element method for the numerical simulations of fluid flow in domains containing moving rigid objects or boundaries is developed. The method falls into the general category of Arbitrary Lagrangian Eulerian methods; it is based on a fixed mesh that is locally adapted in the immediate vicinity of the moving interfaces and reverts to its original shape once the moving interfaces go past the elements. The moving interfaces are defined by separate sets of marker points so that the global mesh is independent of interface movement and the possibility of mesh entanglement is eliminated. The results is a fully robust formulation capable of calculating on domains of complex geometry with moving boundaries or devises that can also have a complex geometry without danger of the mesh becoming unsuitable due to its continuous deformation thus eliminating the need for repeated re-meshing and interpolation. Moreover, the boundary conditions on the interfaces are imposed exactly. This work is intended to support the internal combustion engines simulator KIVA developed at Los Alamos National Laboratories. The model's capabilities are illustrated through application to incompressible flows in different geometrical settings that show the robustness and flexibility of the technique to perform simulations involving moving boundaries in a three-dimensional domain.

1. Introduction

The accurate numerical simulation of fluid flow in the presence of moving structures that modify the fluid domain is of great practical importance in many areas of engineering and science. These include the automotive, medical, and aeronautics industries. In the automotive industry design refinements through modeling of the combustion inside the engines has become economically important and, in general, internal combustion engines are an important part of our everyday life, being used in a myriad of tools as well as in trains, ships, aircraft and automobiles. An example of modeling software as a tool for the design of internal combustion engines is the Los Alamos National Laboratory engine simulation code "KIVA" [1]. Designs made with these types of software benefit from increased ability to estimate efficiency, power output, and other metrics of a prototype before manufacture [2-6].

Fluid flow finite element simulators with moving boundaries capabilities are also used in the medical field [7-9]. The methods used for medical studies fall under the category of immersed boundary techniques [10, 11], these have also been used to investigate the mechanism of insect flight, as small aircraft are pushed to their operational limits at very low Reynolds numbers [12, 13]. Finite element numerical models for problems with moving boundaries for the most part are based on Arbitrary Lagrangian Eulerian (ALE) methods. The earliest ALE applications involved fluid structure interactions, which has been a crucial driver of these efforts especially in aeronautics, also of great interest is a large class of free surface flows [14-21].

The ALE process begins at each time step by displacing the boundaries in the Lagrangian (moving) framework. The second step is to solve the equations in the fluid domain in the Eulerian (stationary) framework. The classical ALE approach is to use continuously deforming meshes [22-27]. In these schemes the mesh is attached to the moving boundary, and continuously deformed or re-generated to adapt to the changing domain geometry during the simulation. Because of the mesh deformation, it is often the case that the mesh has been degraded past the point where it can be used in the calculations. At these times the program operator must stop and re-mesh, or have a routine handy that automates this process. This and other practical difficulties make moving mesh schemes undesirable for a variety of practical applications. In this study, the newly developed ALE method eliminates these handicaps by calculating on a locally changing mesh adapted only in the immediate vicinity of the moving interface; leaving the global mesh outside of this area un-deformed.

There are a variety of other techniques such as embedded mesh, fictitious domain, level set, phase field, etc. that have also been used for many years. However these do not enjoy the same degree of popularity as ALE methods and will not be addressed here; a general discussion on numerical techniques for evolving spatial domains is given in [10, 11, 28, 29], where some of the drawbacks of these methods are also discussed.

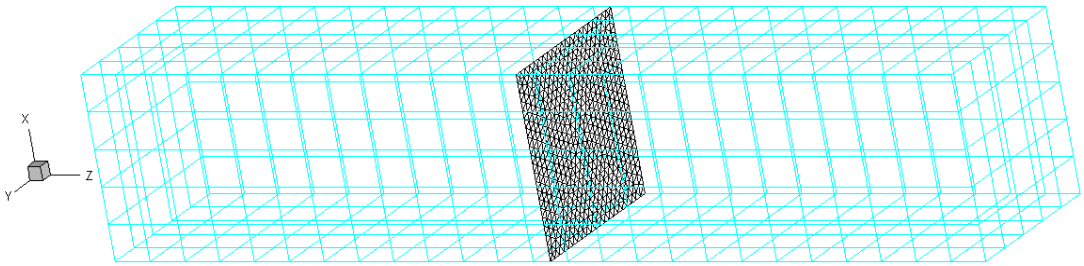
Because in numerical simulations of fluid flow in domains with moving solid boundaries based on the finite element method when a moving mesh is used that mesh must be constantly deformed and periodically regenerated during the simulation. The motion and continuous deformation of the mesh often leads to the mesh becoming inadmissible, referred to as *mesh entanglement* that requires the mesh to be regenerated, sometimes very often. Additionally interpolation of the variables between meshes is required, that can lead to instabilities and loss of accuracy [30, 31], and can make the computations expensive.

The objective of this work is to develop a method to solve moving boundary problems without the need of re-meshing. This is accomplished using a fixed mesh numerical scheme that is constantly adapted locally in space and time to the moving interfaces. A second objective is to validate this local ALE method to ensure that it has the correct accuracy, flexibility and robustness. For this purpose the scheme is applied to realistic geometries and a semi-formal local error analysis has been performed, the details of which are reported in [32], the main results are also mentioned here.

2. Domain and interfaces discretization

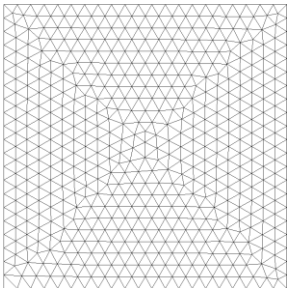
A two-dimensional implementation of the ideas presented here has already been published [33]; the present work concentrates on the three-dimensional methodology and considers only the case when Dirichlet boundary conditions are imposed on the moving interfaces.

Denote by Ω_0 the complete domain occupied by fluid at any time during the complete simulation, this domain can also be used as the reference domain in the ALE formulation [31, 34] and is referred to as the *base* or *reference* domain. Let the domain Ω_0 be discretized using a finite element mesh that is deemed as appropriate for all stages of the simulation (this does not preclude the use of adaptive refinement or higher order elements during the simulation, but these extensions of the method will not be discussed here). In this work, the mesh is made out of hexahedral tri-linear isoparametric elements; it can also be composed of linear pyramidal elements, which makes the implementation much simpler. The moving interfaces are defined independently using grids of linear triangular two-dimensional elements that describe the boundaries of the interfaces in three-dimensional space. The triangular elements that make up the interfaces are called *marker triangles* and their nodes *marker nodes* or *marker points*. The interfaces so defined can move (slide) through the base domain according to their velocity, that can be prescribed by a given function or data set, or can be calculated as part of the solution. In this work, it is assumed that the interfaces are rigid and the interfaces velocity is prescribed by a given function. Figure 1 illustrates the ideas in a simple hexahedral domain intersected by a plane interface. The interface separates the domain into two parts, one containing the fluid, referred to as the *fluid* portion and the rest of the domain denoted as the *inactive* portion.



Fluid side Inactive side

(a)



(b)

Figure 1: (a) Square cylinder subdivided by a uniform mesh of 5 by 5 by 25 tri-linear elements and intersected by a plane interface perpendicular to the z-axis slightly before the midpoint in the z-direction. (b) Interface intersecting the domain in figure 1(a); defined by marker triangles with vertices that are marker nodes.

At each new time step in the calculation an interface intersects the mesh elements at a new position, the intersections of the marker triangles with the element edges are determined and the intersected elements are adapted to fit only that portion of the element that lies in the fluid assuming that within an element the intersecting surface is either planar or bilinear. The adapted mesh is used to carry out the flow calculation and once the velocity and pressure are known the adaptation is discarded, the interface position is advanced to the next time step and a new adaptation performed. Figure 2 shows the resulting adapted mesh assuming that the left hand side of the domain contains the fluid. Once the moving interface goes past an element, the element regains its original form; the mesh adaptation is performed only in those elements intersected by an interface and is local both in space and in time. If Dirichlet conditions are imposed at the interface, the calculation of velocity involves only mesh nodes contained on the fluid side and interpolation is never required; the pressure needs to be calculated at the interface also. Note that the mesh in the inactive part does not enter the calculation and therefore it does not matter how deformed it becomes. This is important in cases involving complex geometries where the mesh in the inactive side may become inadequate for calculation.

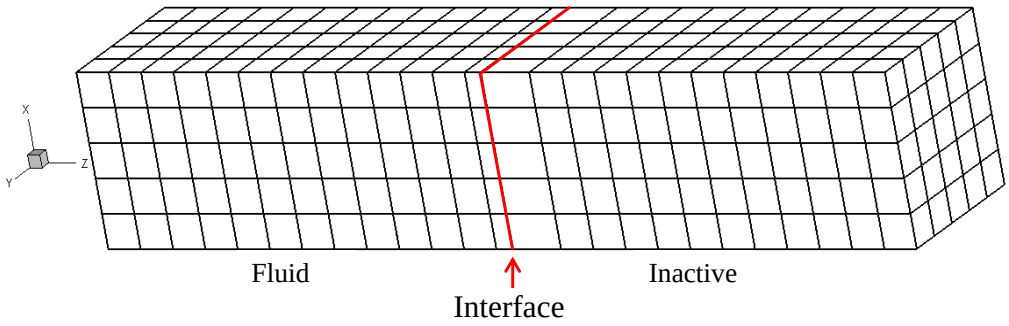


Figure 2: Schematic of a mesh adapted to the intersection by an interface.

In three-dimensions there are eight possible different situations that can arise when a hexahedral element is intersected by a plane. These situations are characterized by the number of nodes that are contained in the fluid part as listed below:

Case 1: Pyramidal intersection with 1 node in the fluid side.
Case 2: Prismatic intersection with 2 nodes in the fluid side.
Case 3: Irregular hexahedral intersection with 3 nodes in the fluid side, the intersecting plane has five corners and two sides in the fluid part are triangular.
Case 4: Regular hexahedral intersection with 4 nodes in the fluid side.
Case 5: Irregular heptahedral intersection with 4 nodes in the fluid side, the intersecting plane has six corners, three of the sides on the fluid side have five corners and the other three sides are triangular.
Case 6: Same as case 3 with the fluid and inactive sides interchanged and 5 nodes in the fluid side.

Case 7: Same as case 2 with the fluid and inactive sides interchanged, 6 nodes in the fluid side.
Case 8: Same as case 1 with the fluid and inactive sides interchanged, 7 nodes in the fluid side.

Figure 3 illustrates the intersections leading to types 1 through 6. If the finite element mesh is based on tetrahedral pyramids only three different cases arise and the geometric setting is considerably simpler.

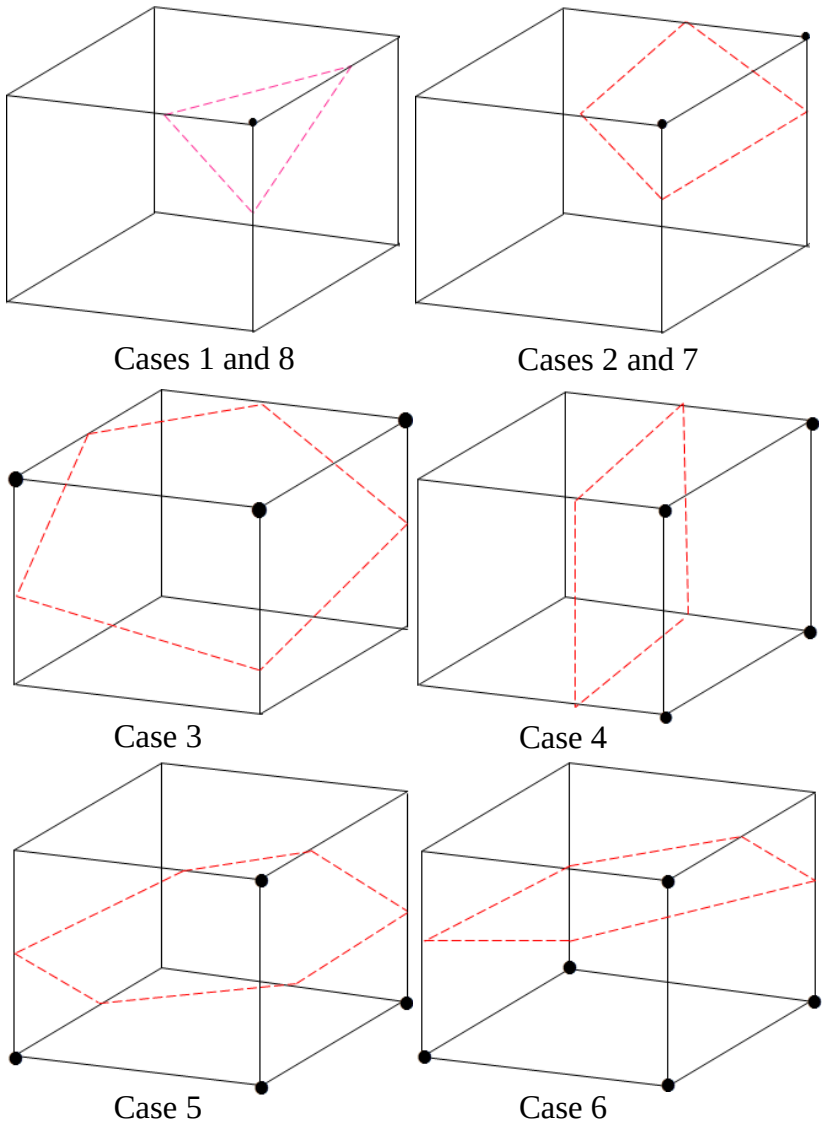


Figure 3: The eight possible ways a plane may intersect a hexahedron dividing it into a fluid part where the nodes are shown, and an inactive part. The intersections in Case 3 and Case 6 are of the same type, but the fluid and inactive parts are switched around. In the same way Case 8 is obtained from Case 1 and Case 7 from Case 2.

Most important is the fact that these are the only eight situations that may arise, therefore, once they have been programmed there is never a new different case to be

accounted for. The above figures show that the interface may intersect as many as six element edges (case 5), creating fluid parts in the elements with shapes that are not hexahedral. To address this difficulty two additional elements are allowed at the interface, the linear tetrahedral pyramids in case 1 and the prismatic pentahedron in case 2. For cases 3 and 5 through 8, nodes in the inactive part of the elements are moved to create hexahedral element that are a close approximation to the fluid part geometry. The way this is done is shown in section 4.

The problem of maintaining the mesh quality is thus eliminated and the result is a robust formulation on arbitrary geometrical configurations. However, elements that are relatively very small or with very large aspects ratios may be generated; to avoid these extreme situations the present implementation neglects fluid element intersections with a volume less than 0.1% of the original volume of the element.

3. Governing equations and finite element approximation

The incompressible Navier-Stokes equations, assuming zero body forces are written in non-dimensional form as

$$\frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U}\cdot\nabla)\mathbf{U} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{U} \tag{1}$$

$$\nabla\cdot\mathbf{U} = 0 \tag{2}$$

Where $\mathbf{U} = (u\,\mathbf{i} + v\,\mathbf{j} + w\mathbf{k})$ is the velocity; $\nabla = (\frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k})$ is the gradient operator, t is time, p is the pressure and $\mathrm{Re} = \frac{UL}{\nu}$ is the Reynolds number. U is a characteristic velocity, L is a characteristic length and ν is the kinematic viscosity of the fluid. Equations (1) and (2) are defined over the domain $[0,T]\times\Omega(t)$ where T is a real number, and $\Omega(t)$ is a connected time dependent domain in $\mathbb{R}^3$ with a sufficiently smooth boundary $\Gamma(t)$.

At time $t_0 = 0$, $\Omega(t_0) = \Omega_0$ and the initial condition is $\mathbf{U}(\mathbf{x},0) = \mathbf{U}_0$.

The boundary conditions for each velocity component may be of the Dirichlet or Neumann type over different portions of the boundary, those portions of the boundary where Dirichlet boundary conditions are imposed are denoted by Γ_D and those with Neumann boundary conditions by Γ_N. Note that in any portion of the boundary a Dirichlet condition may be imposed on one of the velocity components and a Neumann boundary condition on another, so this notation needs to be interpreted accordingly for each velocity component. Denote $\Omega(t) \equiv \Omega_t$ at each time t ; define the space $L^2(\Omega_t)$ as the space of functions defined in Ω_t that are square integrable in Ω_t, and the space $H^1(\Omega_t)$ as the space of functions defined in Ω_t such that the function and its first partial derivatives are square integrable in Ω_t. Finally, let $S_k(t) = \left\{\mathbf{x}_i, i = 1, n_k \,/\, \mathbf{x}_i \in \bar{\Omega}(t)\right\}$ be

finite sets points that define k interfaces/boundaries contained in the reference domain that move within the domain with prescribed velocity $\mathbf{v}_k$. Theoretical considerations related to this algorithm have already been analyzed in detail in [31, 34-36] and are not repeated here. Notice that in this formulation there is no *mesh velocity* , only the interfaces move modifying the mesh in their immediate vicinity.

The ALE formulation combined with the projection method is:

1. Lagrangian step: Update the position of the interfaces $S_k(t)$ from time $t = t_n$ to time $t = t_{n+1} = t_n + \Delta t$ according to the prescribed velocity $\mathbf{v}_k(t)$ of each interface.
2. Eulerian step: Solve the Navier-Stokes equations to find $\mathbf{U}(\mathbf{x}, t_{n+1})$ and $p(\mathbf{x}, t_{n+1})$. This is done using a first order in time projection method [37], described below.

Let $\mathbf{U}^n(\mathbf{x}) \equiv \mathbf{U}(\mathbf{x}, t_n)$ be known. At time $t = t_{n+1}$ decompose the velocity as $\mathbf{U}^{n+1} = \mathbf{U}^* + \mathbf{U}'$ where $\mathbf{U}^*$ is an intermediate or viscous velocity that does not satisfy continuity and $\mathbf{U}'$ is a correction or inviscid velocity that enforces the mass conservation.
To simplify the explanation, the fractional step formulation is given using only the x-component of velocity u, the equations for the other two components is similar. The time derivative is discretized using a first order backward Euler difference, the intermediate velocity component u^* is obtained solving

$$\int_{\Omega_{t_{n+1}}} \left\{ \frac{1}{\Delta t} w_u u^* + \frac{1}{\mathrm{Re}} \nabla w_u \cdot \nabla u^* \right\} d\Omega = \int_{\Omega_{t_n}} \left\{ -w_u (\mathbf{U}^n \cdot \nabla) u^n + \frac{1}{\Delta t} w_u u^n \right\} d\Omega \quad (3)$$

where w_u denotes weighting functions in $H^1(\Omega_t)$ that satisfy homogeneous Dirichlet boundary conditions in Γ_D.

Because the convective term is kept explicit, the algorithm is subject to the Courant-Friedrich-Levy (CFL) stability condition $c \leq 1$, where c is the local Courant number. Let the subscript e denote an element, $d\Omega_e$ denote the element differential of volume and N_i , $i = 1,8$ denote the trilinear shape functions of the element. The full Galerkin finite element discretization of Eq. (3) using eight-node isoparametric elements results in the element equations

$$\left(\frac{1}{\Delta t} \mathbf{M}^e + \mathbf{K}^e \right) \left(\mathbf{u}^* \right)^e = \mathbf{F}_u^e \quad (4)$$

where

$$\mathbf{M}^e = \left[m_{ij}^e \right] = \iiint_e N_i N_j d\Omega_e \tag{5}$$

is the element mass matrix

$$\mathbf{K}^e = \left[k_{ij}^e \right] = \frac{1}{\mathrm{Re}} \iiint_e \left[\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] d\Omega_e \tag{6}$$

is the element stiffness matrix, $\left(\mathbf{u}^*\right)^e = \left(u_1^*, u_2^*, ..., u_8^*\right)^T$ are the u^* component degrees of freedom contained in the element and

$$\mathbf{F}_u^e = \left[\left(f_u^e \right)_i \right] = \iiint_e N_i \left[-\left(\sum_{k=1}^{8} N_k u_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j^n \right) - \left(\sum_{k=1}^{8} N_k v_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j^n \right) - \left(\sum_{k=1}^{8} N_k w_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial z} u_j^n \right) + \frac{1}{\Delta t}\left(\sum_{j=1}^{8} N_j u_j^n \right) \right] d\Omega_e \tag{7}$$

Two more sets of element equations, $\left(\frac{1}{\Delta t}\mathbf{M}^e + \mathbf{K}^e \right)\left(\mathbf{v}^*\right)^e = \mathbf{F}_v^e$ and $\left(\frac{1}{\Delta t}\mathbf{M}^e + \mathbf{K}^e \right)\left(\mathbf{w}^*\right)^e = \mathbf{F}_w^e$ for the v and w components of velocity in the *y*- and *z*-direction are obtained that differ only by the right hand sides $\mathbf{F}_v^e$ and $\mathbf{F}_w^e$ which are given by

$$\mathbf{F}_v^e = \left[\left(f_v^e \right)_i \right] = \iiint_e N_i \left[-\left(\sum_{k=1}^{8} N_k u_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial x} v_j^n \right) - \left(\sum_{k=1}^{8} N_k v_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j^n \right) - \left(\sum_{k=1}^{8} N_k w_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial z} v_j^n \right) + \frac{1}{\Delta t}\left(\sum_{j=1}^{8} N_j v_j^n \right) \right] d\Omega_e \tag{8}$$

and

$$\mathbf{F}_w^e = \left[\left(f_w^e \right)_i \right] = \iiint_e N_i \left[-\left(\sum_{k=1}^{8} N_k u_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial x} w_j^n \right) - \left(\sum_{k=1}^{8} N_k v_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial y} w_j^n \right) - \left(\sum_{k=1}^{8} N_k w_k^n \right)\left(\sum_{j=1}^{8} \frac{\partial N_j}{\partial z} w_j^n \right) + \frac{1}{\Delta t}\left(\sum_{j=1}^{8} N_j w_j^n \right) \right] d\Omega_e \tag{9}$$

respectively.

After the three assembled systems of equations have been solved for the intermediate velocity components, the pressure is obtained from the solution to the Pressure Poisson equation (PPE)

$$\int_{\Omega_{t_{n+1}}} \nabla w_p \cdot \nabla p^{n+1} \, d\Omega = -\frac{1}{\Delta t} \int_{\Omega_{t_{n+1}}} w_p \, \nabla \cdot \mathbf{U}^* d\Omega \tag{10}$$

with the weighting function w_p in $\mathrm{H}^1\left(\Omega_t\right)$. The pressure is interpolated to the same order as the velocity and the Galerkin discretization yields the element equations

$$\mathbf{B}^e \mathbf{p}^e = \mathbf{F}_p^e \tag{11}$$

$$\mathbf{B}^e = \left[b_{ij}^e \right] = \iiint_e \left[\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right] d\Omega_e \tag{12}$$

$\mathbf{p}^e = \left(p_1^{n+1}, p_2^{n+1}, ..., p_8^{n+1} \right)^T$ are the pressure degrees of freedom in the element and

$$\mathbf{F}_p^e = \left[f_i^e \right] = -\frac{1}{\Delta t} \iiint_e N_i \left[\sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j^* + \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j^* + \sum_{j=1}^{8} \frac{\partial N_j}{\partial z} w_j^* \right] d\Omega_e \tag{13}$$

After assembling and solving for the pressure at $t = t_{n+1}$ the velocity is obtained from

$$\mathbf{U}^{n+1} = \mathbf{U}^* - \left(\Delta t \right) \nabla p^{n+1} \tag{14}$$

Discretized, for each velocity component at the element level Eq.(14) takes the form

$\mathbf{M}^L \mathbf{u}^{n+1} = \mathbf{f}_u$, $\mathbf{M}^L \mathbf{v}^{n+1} = \mathbf{f}_v$ $\mathbf{M}^L \mathbf{w}^{n+1} = \mathbf{f}_w$, where $\mathbf{M}^L$ denotes the *lumped mass matrix* [38], used to avoid the solution of the extra systems of equations that would result without this modification. The above matrices are

$$\mathbf{M}^L = \left[m_i^L \right] = \left[\sum_{j=1}^{8} \iint_e N_i N_j d\Omega_e \right] \tag{15}$$

which is the same for the three components of velocity; $\mathbf{u}^{n+1}$, $\mathbf{v}^{n+1}$, and $\mathbf{w}^{n+1}$ contain the eight degrees of freedom of each component contained in the element and the right hand sides are

$$\mathbf{f}_u = \left[\left(\mathrm{f}_u \right)_i \right] = \left[\sum_{j=1}^{8} \iiint_e N_i N_j u_j^n d\Omega_e \right] - \Delta t \left[\iiint_e N_i \sum_{j=1}^{3} \frac{\partial p_j^{n+1}}{\partial x} \right]$$

$$\mathbf{f}_v = \left[\left(\mathrm{f}_v \right)_i \right] = \left[\sum_{j=1}^{8} \iiint_e N_i N_j v_j^n d\Omega_e \right] - \Delta t \left[\iiint_e N_i \sum_{j=1}^{3} \frac{\partial p_j^{n+1}}{\partial y} \right] \qquad (16)$$

$$\mathbf{f}_w = \left[\left(\mathrm{f}_w \right)_i \right] = \left[\sum_{j=1}^{8} \iiint_e N_i N_j w_j^n d\Omega_e \right] - \Delta t \left[\iiint_e N_i \sum_{j=1}^{3} \frac{\partial p_j^{n+1}}{\partial z} \right]$$

The systems of linear equations resulting from the assembly of Eqs. (4) and (11) can be solved by any appropriate method, in this work the total number of degrees of freedom involved in the simulations is rather modest (maximum 12,000 nodes) and a direct skyline method [39] has been used. Obtaining the corrected velocity at time step t_{n+1} from Eq. (14) involves an uncoupled linear system with a diagonal coefficients matrix that is readily solved.

4. Mesh intersections and adaptation

The reference domain Ω_0 is assumed to be discretized using a mesh of isoparametric tri-linear hexahedra as shown in Figure 1(a). The domain is intersected by one or more interfaces discretized using linear two-dimensional marker triangles as shown in Figure 1(b). A new adapted mesh is needed to advance from time step t_n, when the velocity and pressure are known, to time step t_{n+1}. The procedure to find the new adapted mesh goes along the following steps:

1. Find all the intersections between the marker triangles and the edges of the FEM mesh elements. The first time step this involves a search over all elements in the mesh and all marker triangles in the interface. After the first time step the search is narrowed to only the elements intersected at time t_n and their immediate neighbors.
2. In all intersected elements find the nodes that are in the fluid part. To do this the vector normal to the interface pointing to the side occupied by the fluid is used; this information is part of the input data.
3. Modify the intersected elements to fit the interface position. How this is done for each of the different eight cases is explained below.

The simplest case is provided by the example in Figure 1 in which only intersections of the type of Case 4 in Figure 3 occur; in this case the adaptation consists in changing the location of the nodes in the inactive side of the intersected elements to the position of the interface as shown in Figure 4 where the repositioning of the nodes is illustrated.

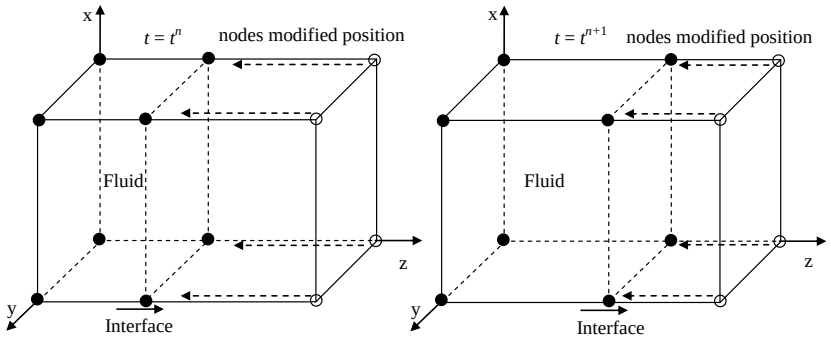


Figure 4: mesh adaptation used to generate the modified mesh in Figure 2.

At the end of these three steps the new adapted mesh has been generated and is used to calculate velocity and pressure. There is not a unique way to complete step 3 and different strategies can be used to do it, this is easier to explain with a two dimensional example as shown in Figure 4; where several possibilities are shown and a preferred strategy discussed.

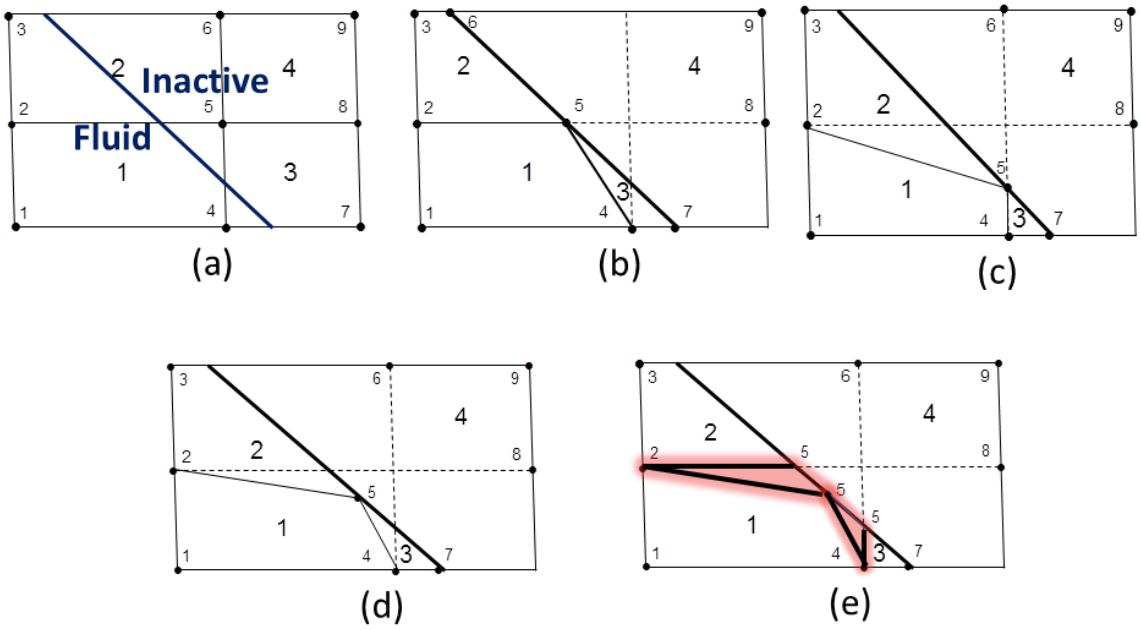


Figure 5: Two-dimensional example to illustrate different mesh adaptation strategies. (a) Four element mesh intersected by an interface; (b) Inactive node moved along a horizontal mesh line; (c) Inactive node moved along a vertical mesh line; (d) Inactive node moved to the center of the intersection segment; (e) A combination of the three previous cases.

A simple rectangular mesh composed of four elements and intersected by an interface is shown in Figure 5(a), elements 1, 2 and 3 are intersected and element 4 is fully inactive. Element 1 has become a pentagon, to integrate over it needs to either be decomposed into a combination of quadrilateral and/or triangular elements, or modified into a single quadrilateral element. The first alternative introduces additional nodes at the interface and

therefore complicates the solution unnecessarily; the second option can be easily implemented by moving the inactive node number 5 to place it on the interface and modifying the geometry of the fluid elements that have node 5 in common. Node 5 can be moved in a number of ways, in Figure 5(b) it has been displaced toward the interface sliding along the mesh line that contains the intersection at the top of the element, in Figure 5(c) it has been displaced along the mesh line intersecting the element on the right hand side, and in Figure 5(d) it has been moved to the midpoint of the interface segment intersecting element 1. All three of these options are good, and all three produce modified quadrilateral and triangular elements that can be quite deformed, the effect of these irregularities in the mesh on the accuracy of the solution is discussed in [32]. In three dimensions the same kind of geometric scenarios develop, but the effect on the neighboring elements due to displacing a node in one of them can be very complex and sometimes not possible to visualize, this motivates the use of a fourth option illustrated in Figure 5(e) where each of the intersected elements is modified individually, hence, when processing element 1 it is modified displacing node 5 as in Figure 5(d), in element 2 node 5 is moved as in Figure 5(b), and in element 3 node 5 is moved as in Figure 5(c). This has the advantage that the shape of the elements remains more uniform, and therefore the approximation error is reduced [40], on the other hand it has the disadvantage that it introduces a local inconformity because a small area is not considered in the integrations these are the shaded areas shown in Figure 5(e); however, this lack of conformity does not violate the patch test and the error remains of order $O(h^2)$ where h is the mesh parameter [40]. Which of these two errors is larger cannot be accurately assessed; however, in three dimensions the last option makes it possible to visualize the deformed geometries for every situation which has not been possible in the other cases and has the added advantage that the modified elements do not become excessively deformed.

To explain how the adaptation process described above in two dimensions is implemented in three dimensions the notation in Figure 6 is used to number the nodes and edges of a trilinear element.

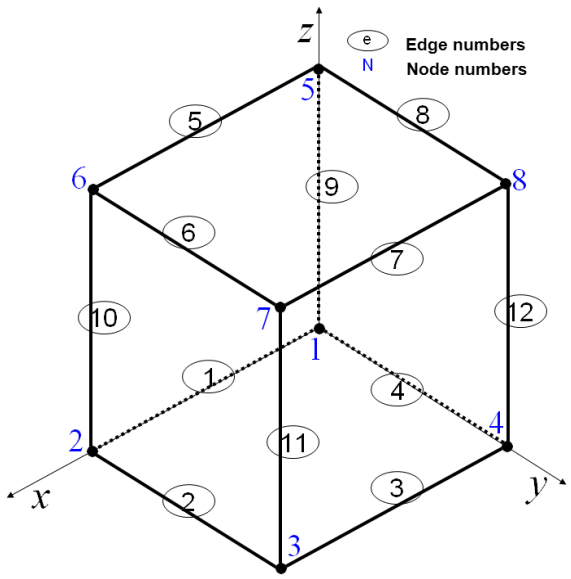


Figure 6: Notation used to describe the element intersections

For element intersection in Cases 1 or 2 of Figure 3, the fluid part is a pyramid or prismatic element respectively and is integrated using an isoparametric form. When an element is of the type of Case 3, it is modified into an isoparametric hexahedron as shown in Figure 7 in the following way: First the element is rotated to the

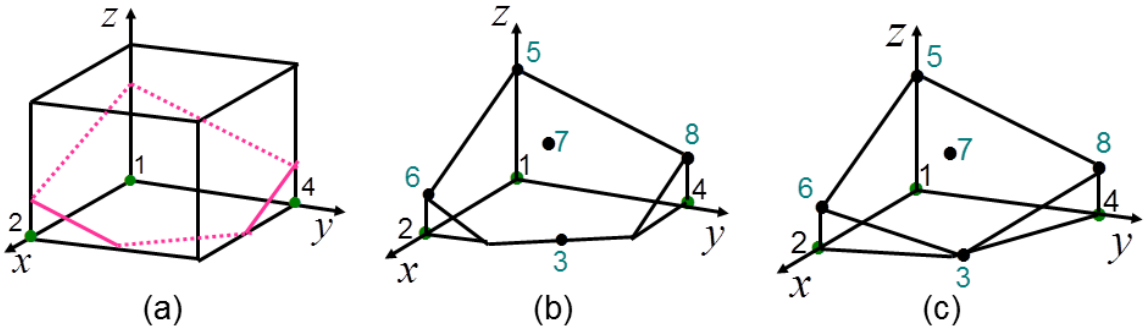


Figure 7: (a) Element intersection of the type of Case 3. (b) Displaced position of nodes 3, 5, 6, 7 and 8. (c) Final modified geometry.

standard position shown in Figure 7(a), where nodes 1, 2 and 4 are the fluid nodes. Next the inactive nodes are repositioned as shown in Figure 7(b), so that node 3 is moved to the midpoint of the intersections in edges 2 and 3; node 5 is moved to the intersection point in edge 9; node 6 is moved to the intersection point in edge 10; node 7 moves to the midpoint between the repositioned nodes 3 and 5 and node 8 is moved to the intersection point in edge 11. The end result is the degenerate hexahedron shown in Figure 7(c), in which nodes 3, 5, 6, 7 and 8 all lay in the interface. However, the integrations over the volume, and the volumes that are generated by cases 5 and 6 as well, are automatically done using the shape functions in conjunction with isoparametric transformations.

Element ntersections of the type of Case 4 result in standard isoparametric hexahedrons that do not require any special treatment. Intersections of the type of Case 5 follow the steps shown in Figure 8; the element is rotated to the standard position shown in Figure 8(a) where nodes 1, 2, 4 and 5 are in

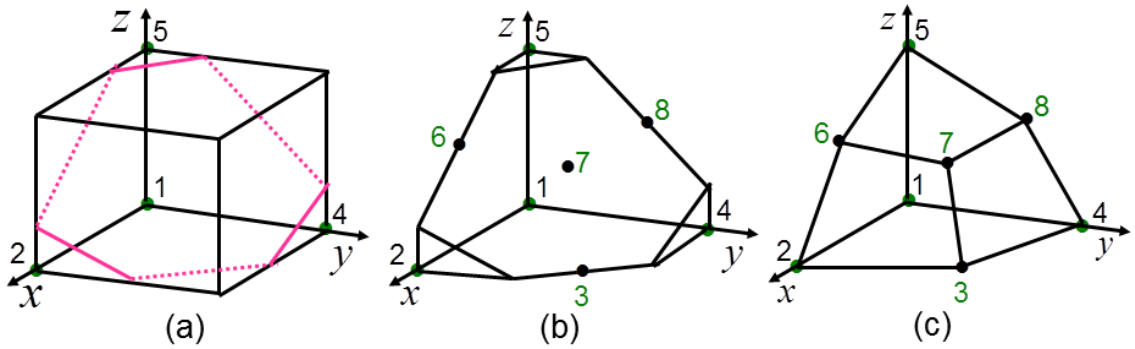


Figure 8: (a) Element intersection of the type of Case 5. (b) Displaced position of nodes 3, 6, 7 and 8. (c) Final modified geometry.

the fluid. In Figure 8(b) the inactive nodes are repositioned, node 3 is moved to the midpoint between the intersections of edges 2 and 3; node 6 to the midpoint of the intersections in edges 5 and 10; node 8 is moved to the midpoint of the intersections in edges 8 and 12 and node 7 is placed at the centroid of nodes 3, 6 and 8. The Nodes 3, 6, 7 and 8 are now all on the interface. The modified element is shown in Figure 8(c)

An element intersection of the type of Case 6 is shown in Figure 9(a). The element is rotated to a position

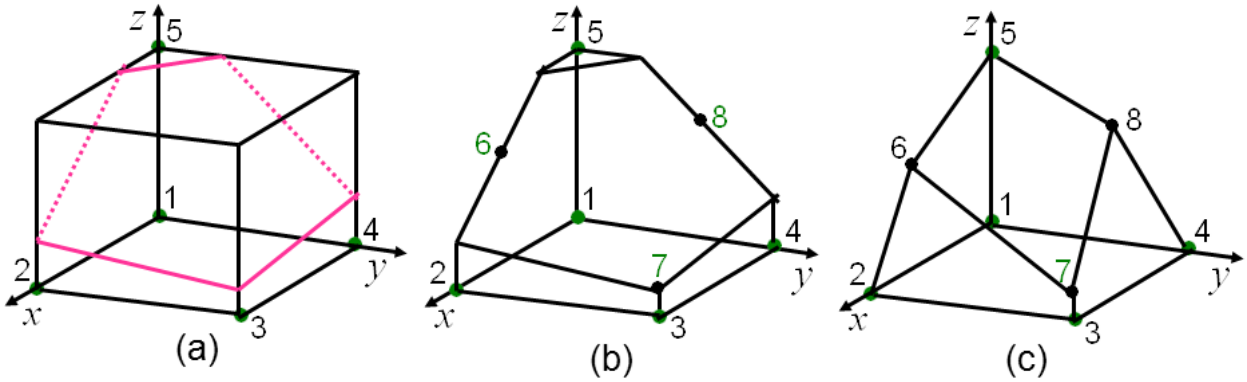


Figure 9: (a) Element intersection of the type of Case 6. (b) Displaced position of nodes 6, 7 and 8. (c) Final modified geometry.

where nodes 1, 2, 3, 4 and 5 are in the fluid .In Figure 9(b) the inactive nodes are repositioned, node 6 is at the midpoint of the intersections of edges 5 and 10; node 7 has been displaced along edge 11to the intersection point on that edge and node 8was moved to the midpoint between the intersections of edges 8 and 12. Figure 9(c) shows the final configuration for the modified element

Figure 10(a) shows an intersection of the type of Case 7 rotated so that nodes 7 and 8 are the

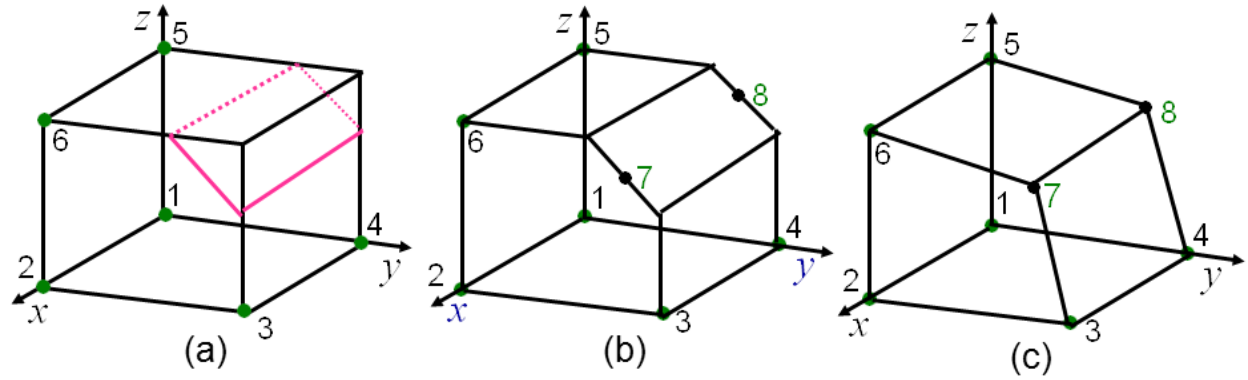


Figure 10: Element intersection of the type of Case 7. (b) Displaced position of nodes 7 and 8. (c) Final modified geometry.

inactive nodes. In Figure 10(b) the displaced position of nodes 7 and 8 to the midpoint between intersected edges 6 and 11 and between edges 8 and 12 respectively is shown, and Figure 10(c) has the final modified geometry.

Finally an element intersection of the type of Case 8 is shown in Figure 11(a) rotated so that the

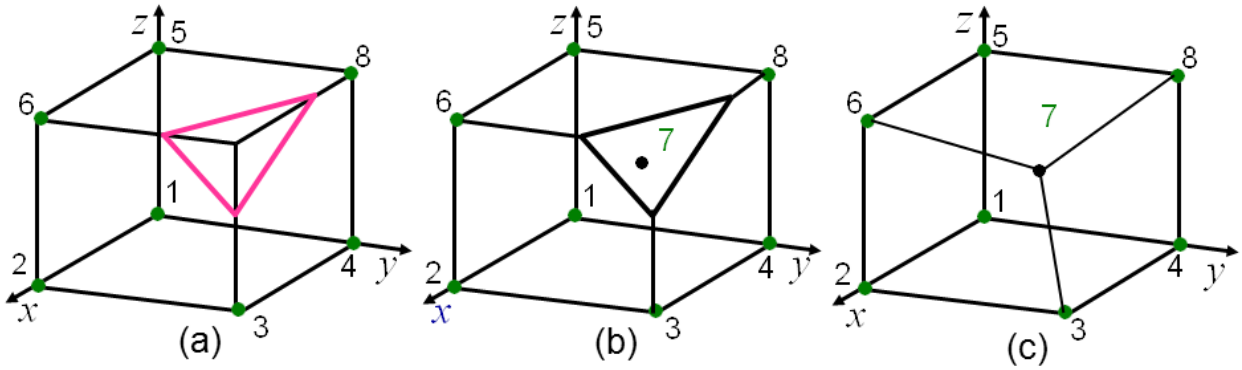


Figure 11: Element intersection of the type of Case 8. (b) Displaced position of nodes 7. (c) Final modified geometry.

only inactive node is node 7. The modified location of node 7 is shown in Figure 11(b) and is the centroid of the triangle determined by the intersections on edges 6, 7 and 11. The final configuration is as in Figure 11(c).

The modification of the intersected elements completes the adaptation process at the current position of the interface. As was explained at the beginning of the section, there is not a unique way to do this, because the nodes are typically common to eight adjacent elements and, as seen in Figure 5(e), depending on the order the element are processed the nodes will be moved to different new locations, therefore some decisions have to be made as to how the adaptation will take place. Three different approaches have been explored in this work, in the first priority is given to sliding the nodes along the intersected edges rather that moving them to a midpoint between intersections. Therefore, once a node has been displaced along an edge it is not allowed to be moved again when considering the other elements that contain it. On the other hand, if the node has been moved to a midpoint between intersections in an element and later a neighboring element requires the node to slide along an intersected edge, the first displacement is discarded, the node is moved according to the current requirement and there it is fixed and not moved again regardless of repositioning requests coming from other elements later. The second approach is the opposite of the first; it gives priority to repositioning the nodes at midpoints over sliding along edges. Both of these approaches result in perfectly conformal adaptations that cover the entire fluid volume and do not have overlaps; however, it is not possible to visualize some of these final configurations and some much distorted elements can be generated. For this reason, in three dimensions the approach explained in two dimensions through Figure 5(e) has been preferred, in which the elements are modified independently. The drawback is that as shown in the figures the final adaptation does not cover the entire fluid domain, small errors in the volume are introduced and the integrals are not exact, it has been proved [40] in the two-dimensional case that the additional error due to inaccuracies in the approximation of the boundary

geometry is $O(h^3)$ where h is the mesh parameter, and the approximation error remains $O(h^2)$; this is not exactly the same situation, but the analysis must be valid for this case also. It has also been proved that the constant of proportionality in the error associated with an isoparametric transformation is directly proportional to the maximum value of the Jacobian of the transformation and to the product of the norms of the transformation and its inverse, and inversely proportional to the minimum value of the Jacobian [40]. As a consequence approximation errors are greatly increased as the elements become much distorted. Which of the methods is better cannot be assessed at the moment, but so far there does not appear to be a significant difference. Examples of simple simulations performed using all three approaches show no significant differences in the velocity components next to the interface.

5. Model Validation

To obtain a partial idea of the accuracy and general behavior of the simulations in the present model the unsteady flow of a viscous fluid between two parallel circular plates that are separating at a prescribed velocity is considered, for which an analytical solution can be obtained [41]

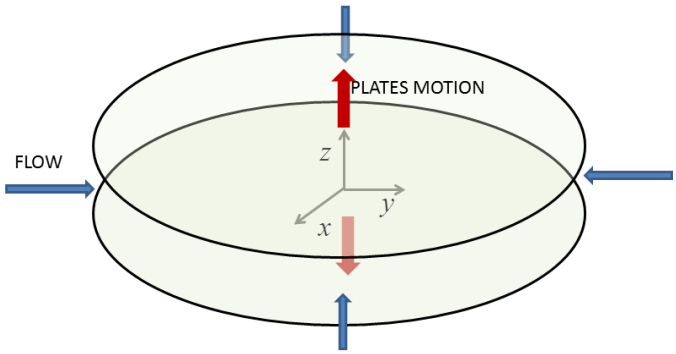


Figure 12: Fluid layer between circular plates separating at a prescribed velocity

If the position and velocity of the upper and lower interfaces are given by $h(t) = \pm h_0 \left(1-\alpha t\right)^{1/2}$ and $\mathrm{w}(t) = \mp \alpha h_0 / 2\left(1-\alpha t\right)^{1/2}$ respectively, where $h_0 \equiv h(0)$ is the initial position of the interface, a similarity solution exists of the form

$$\mathrm{u}(\mathrm{x},\mathrm{y},\mathrm{z},t) = \frac{\alpha x}{4\left(1-\alpha t\right)} f'(\eta) \tag{15}$$

$$\mathrm{v}(\mathrm{x},\mathrm{y},\mathrm{z},t) = \frac{\alpha \mathrm{y}}{4\left(1-\alpha t\right)^{1/2}} f'(\eta) \tag{16}$$

$$\mathrm{w}(\mathrm{x},\mathrm{y},\mathrm{z},t) = \frac{-\alpha h_0}{2\left(1-\alpha t\right)^{1/2}} f(\eta) \tag{17}$$

Where $\eta = \dfrac{z}{h_0(1-\alpha t)^{1/2}}$ is a stretched vertical coordinate, and f is the solution to the ordinary differential equation

$$\eta f''' + 3f'' - ff''' = \frac{1}{S} f'''' \,,\; f(0) = f'(1) = f''(0) = 0,\; f(1) = 1 \tag{18}$$

where $S = \alpha h_0^2 / 2\nu$ and ν is the kinematic viscosity of the fluid. For small values of the parameter S a perturbation solution is given by [41]

$$f(\eta) = f_0(\eta) + S\, f_1(\eta) + S^2 f_2(\eta) \tag{19}$$

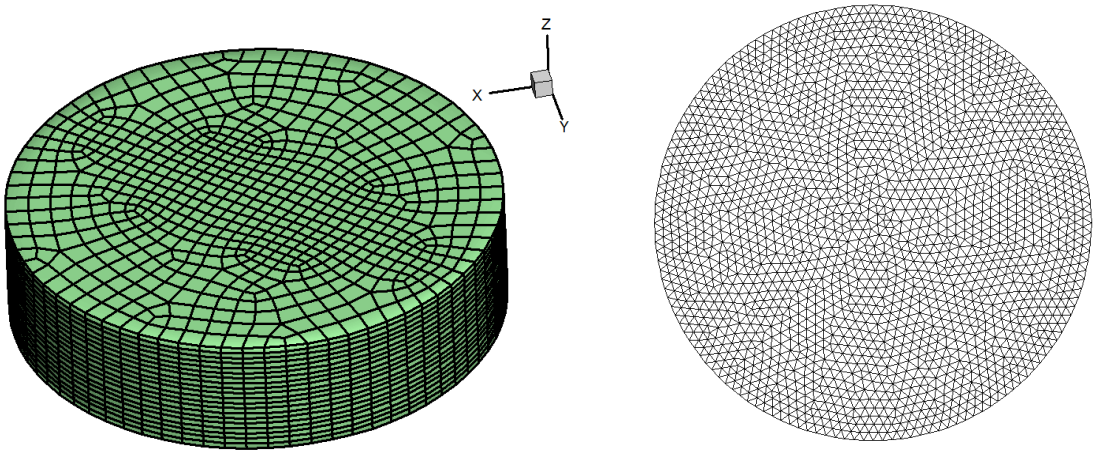
$$f_0(\eta) = \frac{3}{2}\eta - \frac{1}{2}\eta^3 \tag{20}$$

$$f_1(\eta) = -\frac{1}{560}\left(37\eta - 73\eta^3 + 35\eta^5 + \eta^7\right) \tag{21}$$

$$f_2(\eta) = -\frac{1}{140}\left(-\frac{2551}{1848}\eta + \frac{34901}{11088}\eta^3 - \frac{41}{20}\eta^5 + \frac{51}{280}\eta^7 + \frac{7}{72}\eta^9 + \frac{3}{880}\eta^{11}\right) \tag{22}$$

An expression for the pressure solution is also presented in [41]. However, it involves a time dependent integration function that has not been possible to obtain for the purposes of comparison with numerical solutions.

The results of a calculation in which $\alpha = -1.5, h_0 = 0.425$ and the kinematic viscosity ν is chosen so that and $S = -0.1$ are shown below; the domain has diameter $D = 4$ and only the upper half $0 \le z \le 1$ is considered. The mesh of trilinear hexagonal elements containing 14,763 nodes and 13320 elements with 21 nodes in the vertical z-direction is shown in Figure 13 (a). The moving top interface is shown in Figure 13 (b), it has 1,920 maker points and 3694 marker triangles. The computational meshes and the discretizations of moving interfaces have been done using the program CUBIT from Sandia National Laboratory [42]. The velocity boundary conditions are obtained from the analytical solution, and the pressure is set equal to zero at the node (2,0,0). The initial velocity is also taken from the analytical solution.



(a) (b)

Figure 13: Comparison problem, two circular plates moving away at a prescribed velocity: (a)Computational mesh, (b) Moving top boundary mesh

The velocity obtained from calculations utilizing the mesh of Figure 13 and three different time steps $\Delta t = 0.02$, 0.01 and 0.005 are compared with the analytical solution. The results of the calculations show the expected radial flows turning upward from bottom to top, pictures are not informative and are not shown here. To obtain a general idea of the behavior of the error in the domain as a whole, the average relative error in the velocity magnitude is calculated at all active nodes contained in the vertical plane defined by $x = -y$, $x < 0$, that is, the vertical plane at a 45° angle to the x-axis in the fourth quadrant of the x-y-plane. The calculations are performed for the period of time $0 < t \le 2$ at time $t = 2$ the interface is at the position $z = 0.85$. The average error at each time step t_n is defined as

$$\mathrm{E}(t_n) = \left(\frac{\sum_{i=1}^{k} \left[\left(\mathrm{u}_i^* - \mathrm{u}_i^n \right)^2 + \left(v_i^* - v_i^n \right)^2 + \left(w_i^* - w_i^n \right)^2 \right]}{\sum_{i=1}^{k} \left[\left(\mathrm{u}_i^* \right)^2 + \left(v_i^* \right)^2 + \left(w_i^* \right)^2 \right]} \right)^{1/2} \tag{17}$$

The results are shown in Figure 14. The error is reduced at the expected linear rate as a function of time step size and a small jump in the error is observed when the interface crosses a horizontal plane of elements, the magnitude is not significant though and is most probably due to the fact that all nodes in a horizontal line are crossed at the same time.

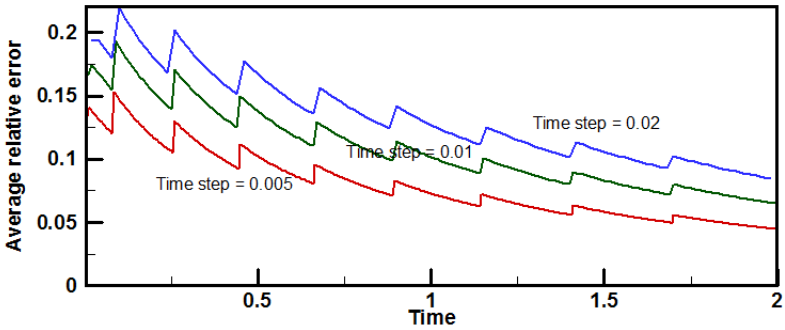
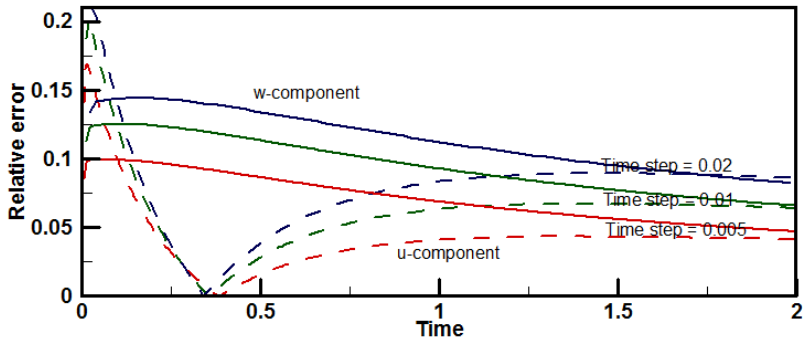
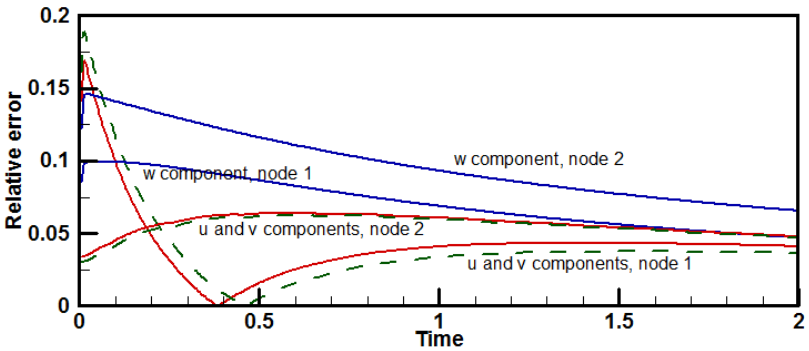


Figure 14: Average relative error at the plane $x = y \leq 0$ for three different time steps as a function of time.

To examine the behavior of the error further two specific nodes located in the plane $x = -y$, $x < 0$ are chosen node 1 at $(x, y) = (0.7071, -0.7071, 0.3)$, on the mid radius $R = 1$ and close to the initial position of the moving interface, and node 2 at $(x, y) = (0.2357, -0.2357, 0.2)$ closer to the center and the bottom of the domain. The relative error for each of the velocity components is calculated at each time step, Figure 15(a) shows the behavior of the error of the velocity components *u*- and *w*- at node 1, the behavior of the *v*-component is practically identical to that of the *u*-component and is not shown. Figure 15(b) shows the behavior of the error in the calculation with $\Delta t = 0.005$ for each of the velocity components at both nodes 1 and 2



(a)



(b)

Figure 15: (a) Relative error as a function of time in the *u*- and *w*- components of velocity at the three different time steps. (b) Relative error as a function of time for each velocity component at nodes 1 and 2 with time step 0.005.

The error is large at the beginning of the calculation, which is probably caused by the initial conditions, but in all cases it goes down and settles with time to close to a constant value, it exhibits a correct behavior as a function of the time step, the *u*- and *v*-components show practically the exact same behavior, but the *w*-component behaves somewhat differently. This is not surprising giving the difference in the mesh in the *z*-direction and the fact that that is the direction of interface motion; the *u*- and *v*- velocity components at node 1 exhibit a rather peculiar behavior, the error becoming extremely small just before $t = 0.5$ and then increasing again, when the calculation approaches $t = 2$ the error in the three components approaches a constant value that for the time step of 0.005 is between 4% and 5%. A much more detailed analysis involving meshes of different size and a full assessment of the analytical solution will give valuable information about the behavior of the present model, but this is out of the scope of this paper. However, the calculations presented here show that the model behaves correctly and exhibits convergence as a function of the time step. In two dimensions spatial convergence of second order has been demonstrated [32,33].

6. Further examples

To illustrate the capabilities of the model a series of examples are given involving different types of geometry and requirements.

(a) Moving one slanted interface:

A slanted interface with sinusoidal motion in a hexahedral channel of non-dimensional length 5 in the z-direction, and a square cross-section of 1 in the x-y plane is simulated to demonstrate the motion of a moving body. The Reynolds number is 20 based on a characteristic length of 1, a characteristic velocity of 1 *m/s* and kinematic viscosity of 0.05 m^2/s. The domain is subdivided by a uniform grid of 6 x 6 x 26 elements. The domain boundaries are assumed to be solid walls except for a slit in the x-z-plane located at $0.0 \le x \le 1.0$, y = 1.0 and $1.0 \le z \le 1.8$ that allows the incompressible fluid to enter and leave the domain. Initially the interface intersects the domain at marker point located at (0.0, 0.0, 2.5), (1.0, 0.0, 2.92262), (0.0, 1.0, 3.07735) and (1.0, 1.0, 3.5). The interface is a plane slanted at an angle of 25 degrees to the z-axis in the x-direction and 30 degrees in the y-direction. The position of the interface in the z - direction as a function of time is given by $z(t) = 0.95\sin\left[\frac{3\pi}{2}(t+1)\right] + 3.45 + y\tan\left(\frac{\pi}{6}\right) + x\sin(\frac{\pi}{7.2})$, that is a stroke with an amplitude of 2.497. The time step used in the calculation is $\Delta t = 0.005$.

In this example seven of the eight intersection cases occur; the only one that does not happen is case 4, which is the most common case in other examples. Therefore here all the different kinds of intersections are tested.

Figure 16 shows snapshots of the flow and interface position at= 0.1when the interface is advancing in the positive z – direction, at t = 0.75 when the interface has just

turned at the end of the stroke and is returning moving towards the right and at t = 1.3when it is very close to the starting point to begin a new cycle.

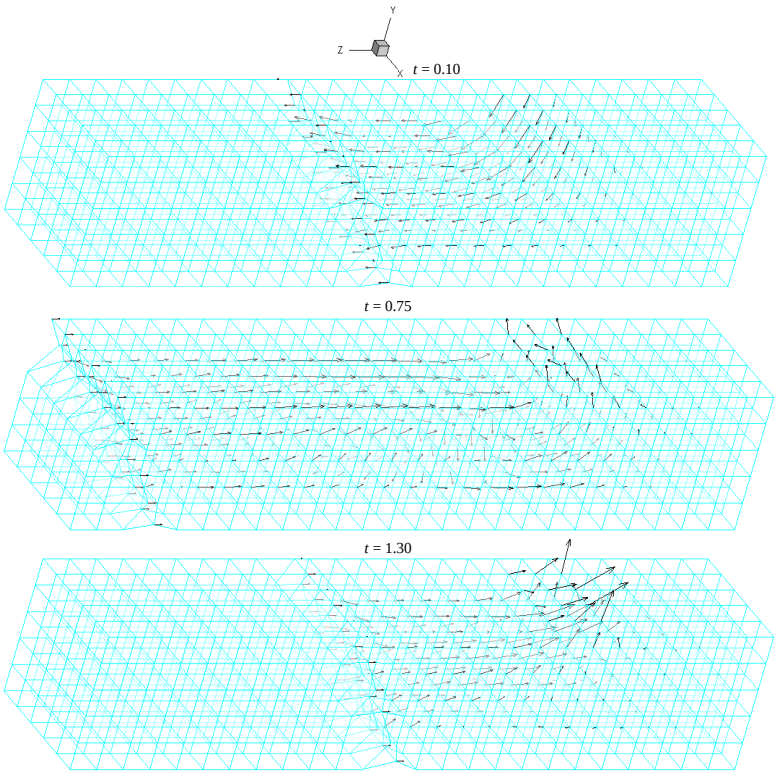


Figure 16: Slanted plane interface under oscillatory motion.

(b) Two slanted interfaces oscillating in opposite directions:

The same conditions and uniform 6 x 6 x 26 elements mesh used in example (a) are used to demonstrate the motion of more than one interface. The boundary conditions are modified closing the slit in the top wall and making two new square openings between 0.2 ≤x, y ≤ 0.8 on the walls at z = 0 and z = 5 to allow incompressible fluid in and out of the computational regions. Two parallel interfaces slanted at an angle of 30 degrees to the y-axis only intersect the domain. The left interface initially intersects the domain at (0.0, 0.0, 2.85), (1.0, 0.0, 2.85), (0.0, 1.0, 3.427) and (1.0, 1.0, 3.427). The right interface initially intersects the domain at (0.0, 0.0, 2.15), (1.0, 0.0, 2.15), (0.0, 1.0, 2.727) and (1.0, 1.0, 2.727). The position of the left interface as a function of time is given by,

$$z(t) = 0.775\sin\left[\frac{3\pi}{2}(t+1)\right] + 3.625 + \mathbf{y}\tan\left(\frac{\pi}{6}\right)$$

and that of the right interface by

$$z(t) = -0.775\sin\left[\frac{3\pi}{2}(t+1)\right] + 1.375 + \mathbf{y}\tan\left(\frac{\pi}{6}\right)$$

the stroke amplitude of each of the interfaces is 2.127.

Figure 17 shows the velocity and interface position at three stages during the first cycle. At t = 0.10 *s* the interfaces are just beginning to move away from each other; at t = 0.75 *s* they have turned the direction of motion and begun to move towards each other; at t = 1.25 *s* they are about to reach their closest point and start a new cycle.

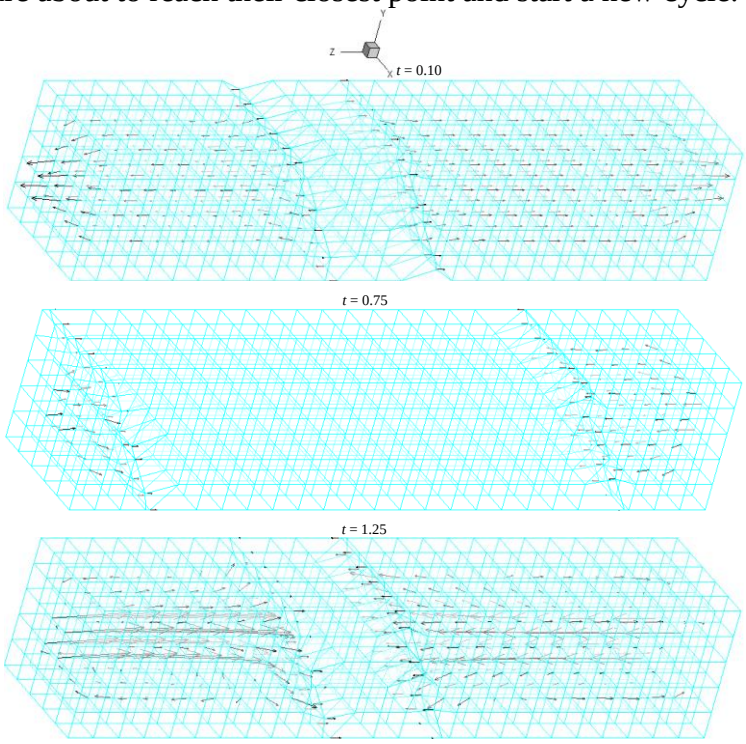


Figure 17: Two slanted interface oscillating in opposite directions

(c) Engine cylinder with bowled piston

The previous examples involved straightforward rectangular geometry and were designed to illustrate how the basic features of the algorithm are capable of modeling a moving interface that intersects a mesh in a general way, but did not address the presence of domain boundaries or interfaces that is introduced in this example. The geometry consists of a cylindrical domain representing an engine cylinder with two valve openings for intake and exhaust of gases, and a piston with a bowl. The cylinder has an inner radius of 5 *cm* and a height of 10.5 *cm*; the piston has a planar ledge with an outer radius of 5 *cm* and an inner radius of 3.75 *cm* where a straight cylinder 1.5 *cm* tall is attached going downward and capped by a plane at the bottom. The cylinder is discretized with a mesh of 7,152 nodes and 6,150 trilinear isoparametric elements and the piston is defined by a grid of linear triangles containing 5,104 markers and 10062 marker triangles. Figure 18 shows the geometry and grid for the piston and Figure 19 that of the cylinder. The cylinder mesh consists of 15 uniformly spaced rows of elements parallel to the *x*-*y* plane; each plane of nodes contains 410 elements. The mesh, although too coarse to expect accurate solutions in modeling simulations, is appropriate to show the effectiveness of the algorithm while keeping the visualization reasonably simple.

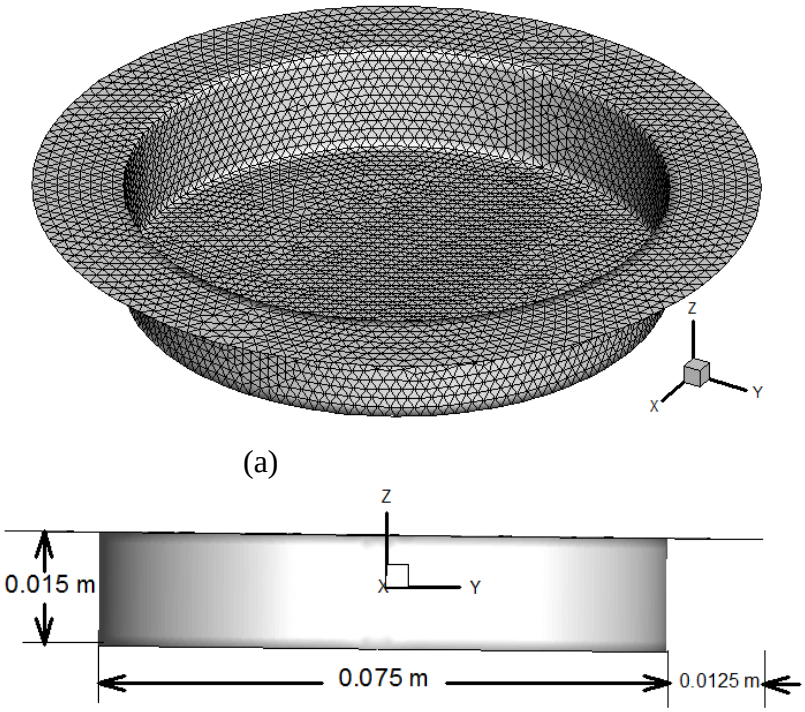
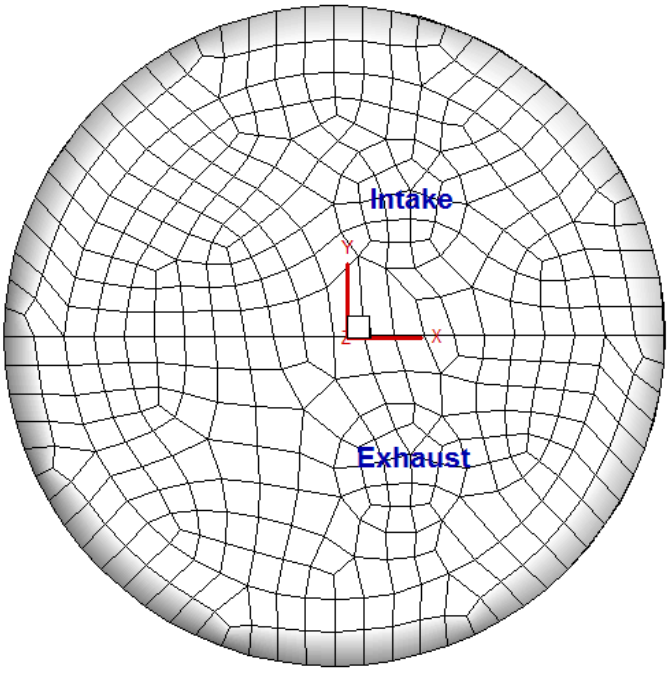
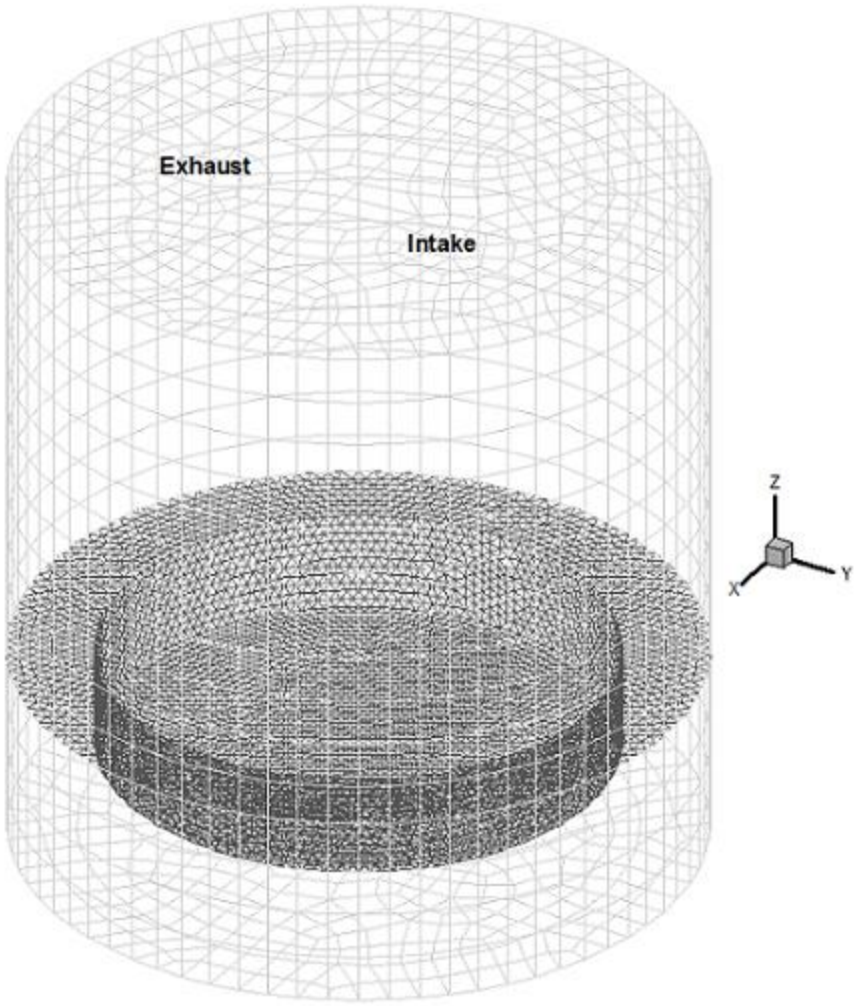


Figure 18: (a) Piston interface defined by a grid of linear triangles. (b) Dimensions of the piston head.

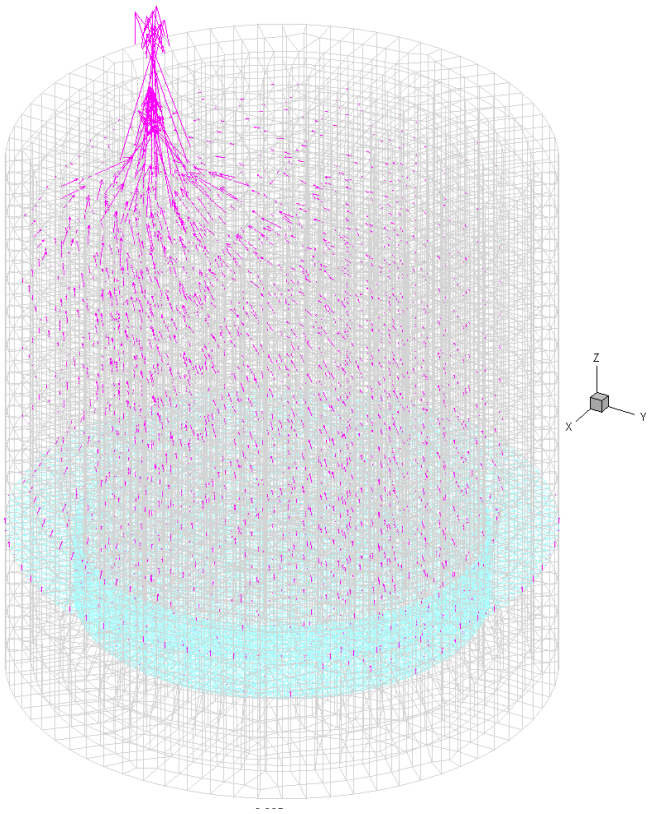




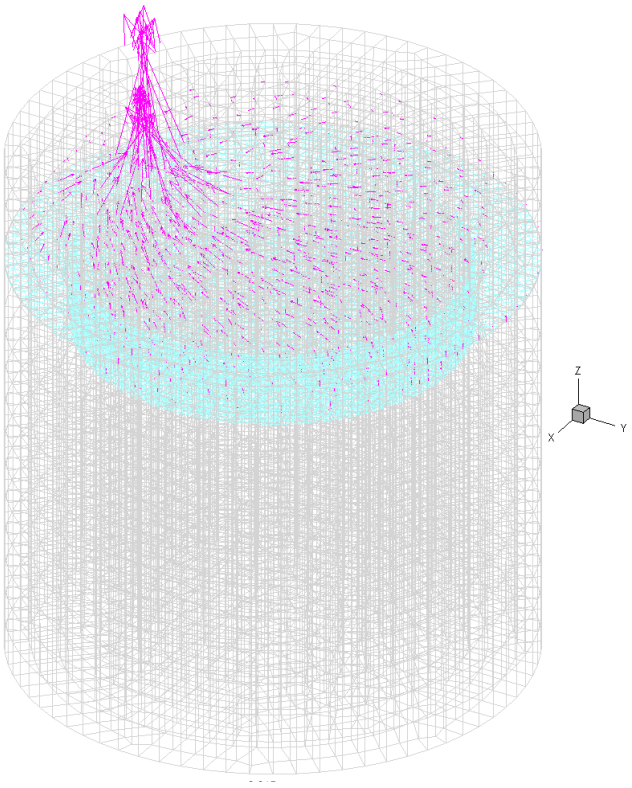
(b)

Figure 19: (a) Two-dimensional horizontal cross-section of the cylinder mesh
Showing the circular intake and exhaust valve openings.
(b) Cylinder – piston configuration showing the outer surfaces of the
cylinder mesh.

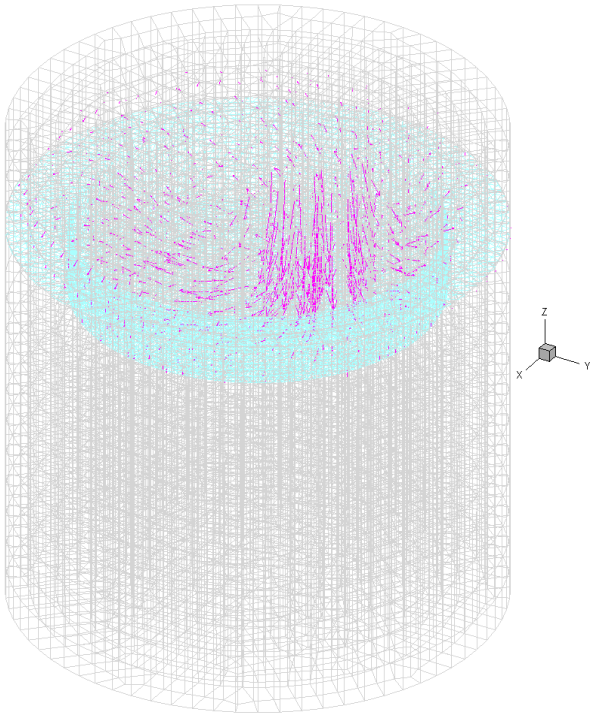
The simulation presented here assumes that the fluid is air, but speeds are in the incompressible range, the reference length and velocity are set to 1.0 and the kinematic viscosity is $\nu = 1.7 \times 10^{-3}$ which results in a Reynolds number $\mathrm{Re} = 588$. The top surface of the piston is initially at $z = 3.25cm$ and it reaches a maximum height of $z = 0.11cm$, the amplitude of the stroke is 7.75 *cm*. The piston is driven according to the function $z_p = 0.03875\left(1 + \sin(50\pi t - \pi/2)\right)$ where z_p is in meters , so the maximum velocity of the piston when $t = 0.01 + 0.02n,\ n = 0,1,2...$ is $w_p = 6.087\ m/s$, and a whole cycle requires 0.04 *s* for completion. Figure 20 shows 4 snapshots during one full cycle of motion.



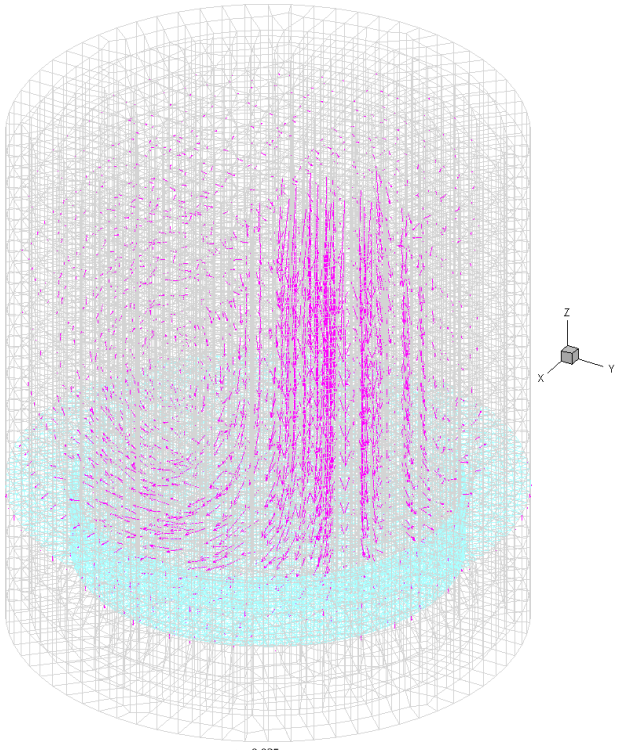
(a)



(b)



(c)



(d)

Figure 20: (a) Piston position and velocity field at $t = 0.005 s$.
(b) Piston position and velocity field at $t = 0.015 s$.
(c) Piston position and velocity field at $t = 0.025 s$.
(d) Piston position and velocity field at $t = 0.035 s$.

At *t* = 0.005 *s* the has advanced 0.113 *cm* and the piston velocity at this point is $4.3 m/s$, the exhaust valve is open and fluid leaving the cylinder; at *t* = 0.015 *s* it is approaching the top, the exhaust valve is still open the piston top is at $z = 9.86 cm$ and its velocity is $4.3 m/s$; at *t* = 0.025 *s* the piston has turned, its position is back to $z = 9.86 cm$, and its velocity is now $-4.3 m/s$; finally at *t* = 0.035 *s* the piston approaches the bottom of the cylinder to start a new cycle, its position is back to $z = 4.38 cm$ and its velocity is $4.26 m/s$. A uniform time step of $10^{-5} s$ was used in the simulation and a full cycle required an average 0.48 seconds CPU in a Dell T5600 workstation.

10. CONCLUSION

In this work, a method based on finite element discretizations that includes moving boundaries or interfaces in CFD calculations has been developed and implemented in three space dimensions. The method belongs to the general family of Arbitrary Lagrangian-Eulerian methods. But differs from previously proposed methods for similar problems in the way the computational mesh is defined and locally adapted as the geometry changes. The present work concentrates strictly on the development of the method to discretize and modify the mesh throughout the calculation and has been applied to the case of laminar incompressible flow at low Reynolds number. The same methodology is currently being extended to high Reynolds number compressible and turbulent flows. The method has been fully tested for accuracy in two dimensions and shown to have second order accuracy in space, in three dimensions a limited number of calculations presented here show the correct error behavior, but additional work is required to properly establish the method's rates of convergence in three dimensions. Additional examples involving slanted interfaces and a realistic cylinder/piston engine assembly show that the method is fully robust and very efficient, and it offers distinct advantages from the point of view of eliminating the use of adaptive mesh generators.

ACKNOWLEDGEMENTS

The DOE's Office of Energy Efficiency and Renewable Energy (EERE) Advanced Combustion Program (Gurpreet Singh) is supporting this effort. Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy (DOE) under contract DE-AC52-06NA25396. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

References

1. D. J. Torres and M. F. Trujillo, 'KIVA-4: an unstructured ALE code for compressible gas flow and sprays'. J. Comp. Phys. **219**, 943-975 (2006)
2. C. D. Rakopoulos and G. C. Mavropoulos, "Study of the steady and transient temperature field and heat flow in the combustion chamber components of a medium speed diesel engine using finite element analyses," *Int. J. Energy Res.*, vol. 20, no. 5, pp. 437–464, 1996.
3. S.L. Yang, B.D. Peschke, and K. Hanjalic, Second-moment closure model for IC engine flow simulations using KIVA code, *Jour. Eng. Gas Turbines & Power*, ASME, 122:355-363 (2000)
4. D. Trescher, Development of an Efficient 3-D CFD Software to simulate and Visualize the Scavenging of a Two-Stroke Engine. *Arch.Comput. Methods Eng* **15**: 67-111 (2008)
5. D. B. Carrington, A fractional step *hp*-adaptive finite element method for turbulent reactive flow. Los Alamos National Laboratory Report, LA-UR-11-00466 (2011)
6. D. B. Carrington, X. Wang, D. W. Pepper, A predictor-corrector split projection method for turbulent reactive flow, *Journal of Computational Thermal Sciences*, Begell House Inc., vol. 5, no. 4, pp. 333-352, 2014
7. C. A. Taylor, T. J. R. Hughes, and C. K. Zarins, "Finite element modeling of blood flow in arteries," *Computer Methods in Applied Mechanics and Engineering*, vol. 158, no. 1-2, pp. 155–196, May 1998.
8. R. van Loon and S. J. Sherwin, "A fluid-structure interaction model of the aortic heart valve," *Journal of Biomechanics*, vol. 39, Supplement 1, p. S293, 2006.
9. S. A. Kock, J. V. Nygaard, N. Eldrup, E.-T. Fründ, A. Klærke, W. P. Paaske, E. Falk, and W. Yong Kim, "Mechanical stresses in carotid plaques using MRI-based fluid-structure interaction models," *Journal of Biomechanics*, vol. 41, no. 8, pp. 1651–1658, 2008.
10. R. Mittal and G. Iaccarino, 'Immersed boundary methods'. Annual Review of Fluid Mechanics **37**, 239-261 (2005).
11. R. Löhner, J. R. Cebral, F. F. Camelli, J. D. Baum and E. L. Mestreau Adaptive Imbedded/Immersed Unstructured Grid Techniques. *Arch.Comput. Methods Eng*. **14**: 279-301 (2007)
12. M. Hamamoto, Y. Ohta, K. Hara, and T. Hisada, "Design of Flexible Wing for Flapping Flight by Fluid-Structure Interaction Analysis," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 2253-2258.
13. K. TAIRA and T. COLONIUS, "Three-dimensional flows around low-aspect ratio flat-plate wings at low Reynolds numbers," *Journal of Fluid Mechanics*, vol. 623, pp. 187–207, 2009.
14. C. W. Hirt, A. A. Amsden and J. L. Cook, An Arbitrary Lagrangian-Eulerian Computing Method for all Flow Speeds. *J. Comp. Phys*. **14**: 227-253 (1974)
15. J. Donea, S. Giuliani and J Halleux, An Arbitrary Lagrangian-Eulerian Finite Elelment Method for Transient Dynamic Fluid-Structure Interactions. *Comp. Methods Appl. Mech. Eng.* **33:** 689-723 (1982)
16. W.Shyy, H. S. Udaykumar, M. M. Rao and R. W. Smith, *Computational Fluid Dynamics with Moving Boundaries*. Taylor and Francis, Philadelphia (1996)

17. W. G. Dettmer and D. Perić, A Fully Implicit Computational Strategy for Strongly Coupled Fluid Solid Interaction. *Arch. Comput. Methods Eng*. **14**: 205-247 (1996)
18. C. Farhat and P. Geuzaine, Design and Analysis of Robust ALE ime-Integrattors for the Solution of Unsteady Flow Problems on MovingGrids. *Comp. Methods Appl. Mech. Engng*. **193**: 4073-4095 (2004)
19. M. S. Gadala, Recent Trends in ALE Formulations and its Applications in Solid Mechanics. *Comp. Methods Appl. Mech. Engng*. **193**: 4247-4275 (2004)
20. R. Codina, G. Houzeaux, H. Coppola-Owen and J. Baiges, 'The fixed-mesh ALE approach for the numerical approximation of flows in moving domains'. J. Comp. Phys. **228**, 1591-1611 (2009)
21. C. Hua, C. Fang, and J. Cheng, "Simulation of fluid-solid interaction on water ditching of an airplane by ALE method," *Journal of Hydrodynamics, Ser. B*, vol. 23, no. 5, pp. 637–642, Oct. 2011.
22. J. U. Brackbill and J. S. Saltzman, Adaptive Zoning for Singular Problems in Two Dimensions. *J. Comp. Phys.* **46**: 342-368 (1982)
23. J. L. Steger and J. A. Benek, On the use of Composite Grid Schemes in Computational Aerodynamics. . *Comp. Methods Appl. Mech. Engng*. **64**: 301-320 (1987)
24. A. Johnson and T. Tezduyar, Mesh Strategies in Parallel Finite Element Computations of Flow Problems with Moving Boundaries and Interfaces. *Comp. Methods Appl. Mech. Engng*. **119**: 73-94 (1994)
25. H. Askes and L. J. Sluys, Remeshing Strategies for Adaptive ALE Analysis of Strain Localization. *Eur. J. Mech. A/Solids* **19**: 447-467 (2000)
26. T. E. Tezduyar, 'Finite element methods for flow problems with moving boundaries and interfaces'. Archives of Computational Methods in Engineering **8**, 83-130 (2001)
27. P. H. Saksono, W. G. Dettmer and D. Perić , An Adaptive Remeshing Strategy for Flows with Moving Boundaries and Fluid-Structure Interaction. *Int. J. Num. Meth. Eng.* **71**: 1009-1050 (2007)
28. D. Juric and G. Tryggvasson, 'A front tracking method for dendritic solidification'. J. Comp. Phys. **123**, 127-148 (1996)
29. P. Zhao and J. C. Heinrich, 'Front-tracking finite element method for dendritic solidification. J. Comp. Phys. **173**, 765-796 (2001)
30. H. Guillard and C. Farhat, 'On the significance of the geometric conservation law for flow computations in moving boundaries'. Comp. Meth. Appl. Mech. Engng. **190**, 1467-1482 (2000).
31. L. Formaggia and F. Nobile, 'Stability analysis of second order time accurate schemes for ALE-FEM computations'. Comp. Meth. Appl. Mech. Engng. **193**, 4097-4116 (2004).
32. V. D. Hatamipour, D. B. Carrington and J. C. Heinrich, 'Accuracy and convergence of Arbitrary Lagrangian-Eulerian finite element simulations based on a fixed mesh'. To be submitted.
33. D. B. Carrington, D. A. Muñoz and J. C. Heinrich, 'Modelling fluid flow in domains containing moving interfaces'. Progress in Computational Fluid Dynamics **14**, 139-150 (2014).

34. S. Badia and R. Codina, 'Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ALE framework'. SIAM J. Numer. Anal. **44**, 2159-2197 (2006)
35. J. L. Guermond, P. Minev and J. Shen, 'An overview of projection methods for incompressible flow'. Comp. Meth. Appl. Mech. Engng. **195**, 6011-6045 (2006).
36. D. Boffi and L. Gastaldi, 'Stability and geometric conservation laws for ALE formulations'. Comp. Meth. Appl. Mech. Engng. **193**, 4717-4739 (2004).
37. L. Quartapelle, *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhauser, Basel (1993).
38. J. C. Heinrich and D. W. Pepper, *Intermediate Finite Element Methods: Heat Transfer and Fluid Flow Applications*. Taylor and Francis, Philadelphia (1999).
39. K-J Bathe, *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey (1982).
40. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Prentice-Hall, Englewood Cliffs, New Jersey (1973).
41. C.-Y. Wang, 'The squeezing of a fluid between two plates'. ASME J. Appl. Mech. **43**, 579-582 (1976).
42. The CUBIT Geometry and Mesh Generation Toolkit, Sandia National Laboratories, https://cubit.sandia.gov/index.html, 2014.