INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **00**:1–6 *Prepared using* fldauth.cls *[Version: 2002/09/18 v1.01]*

Algorithms for Interface Treatment and Load Computation in Embedded Boundary Methods for Fluid and Fluid-Structure Interaction Problems

K. Wang[3], A. Rallu[2], J-F. Gerbeau[*4] and C. Farhat[*1,2,3]

[1] *Department of Aeronautics and Astronautics*
[2] *Department of Mechanical Engineering*
[3] *Institute for Computational & Mathematical Engineering*
Stanford University, Stanford, CA 94305, U.S.A
[4] *INRIA Paris-Rocquencourt, 78153 Le Chesnay Cedex, France*

SUMMARY

Embedded boundary methods for CFD (Computational Fluid Dynamics) simplify a number of issues. These range from meshing the fluid domain, to designing and implementing Eulerian-based algorithms for fluid-structure applications featuring large structural motions and/or deformations. Unfortunately, embedded boundary methods also complicate other issues such as the treatment of the wall boundary conditions in general, and fluid-structure transmission conditions in particular. This paper focuses on this aspect of the problem in the context of compressible flows, the finite volume method for the fluid, and the finite element method for the structure. First, it presents a numerical method for treating simultaneously the fluid pressure and velocity conditions on static and dynamic embedded interfaces. This method is based on the exact solution of local, one-dimensional, fluid-structure Riemann problems. Next, it describes two consistent and conservative approaches for computing the flow-induced loads on rigid and flexible embedded structures. The first approach reconstructs the interfaces within the CFD solver. The second one represents them as zero level sets, and works instead with surrogate fluid/structure interfaces. For example, the surrogate interfaces obtained simply by joining contiguous segments of the boundary surfaces of the fluid control volumes that are the closest to the zero level sets are explored in this work. All numerical algorithms presented in this paper are applicable with any embedding CFD mesh, whether it is structured or unstructured. Their performance is illustrated by their application to the solution of three-dimensional fluid-structure interaction problems associated with the fields of aeronautics and underwater implosion. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: ALE; Eulerian; finite volume method; fluid-structure interaction; immersed boundary method; level sets

[*]This co-author contributed to this work while visiting the Department of Mechanical Engineering at Stanford University
[*]Correspondence to: Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, U.S.A.

Contract/grant sponsor: Office of Naval Research; contract/grant number: N00014-06-1-0505, N00014-09-C-015

Contract/grant sponsor: Army Research Laboratory; contract/grant number: W911NF-07-2-0027

Received August 3, 2011
Revised August 3, 2011

1. INTRODUCTION

Methods for computing flows on non body-fitted CFD (Computational Fluid Dynamics) grids in which wet surfaces of various obstacles (the interfaces) are embedded are gaining popularity in many scientific and engineering applications under different names. These include, among others, the immersed boundary [1], embedded boundary [2], fictitious domain [3], and Cartesian [4] methods. All of these and related methods are collectively referred to in this paper as embedded boundary methods. They can be attractive because they simplify a number of issues ranging from meshing the fluid domain, to formulating and implementing Eulerian-based algorithms for difficult fluid-structure applications such as those involving very large structural motions and deformations [5], or topological changes [6], and for which alternative Arbitrary Lagrangian-Eulerian (ALE) algorithms [7, 8, 9] are unfeasible. However, because they typically operate on non body-fitted grids, embedded boundary methods also tend to complicate other issues such as the treatment of wall boundary conditions in general, and fluid-structure transmission conditions in particular. Indeed, recent developments in embedded boundary methods have focused mostly on these two issues, albeit primarily on the treatment of the velocity wall boundary condition for incompressible viscous flows past rigid and motionless obstacles (for example, see the review paper [10]). In this context, recently proposed algorithms for interface treatment have focused either on some form of interpolation [11] with particular attention to numerical stability [12] or higher-order accuracy [11, 13, 14], or on the concept of a ghost cell [15, 16], some variant of the penalty method [17], and the mirroring technique [18].

For fluid-structure applications, two transmission conditions must be dealt with at the intersection of the embedded structural surface and embedding fluid mesh. The first one is the slip or no-slip condition, depending on whether the flow is inviscid or viscous. The semi-discretization or discretization of this condition is similar to that of its counterpart for flows past rigid and motionless obstacles. For this reason, virtually all methods mentioned above for the treatment of wall boundary conditions in embedded boundary methods can be applied for this purpose. The second transmission condition expresses equilibrium at a fluid-structure interface between the fluid and structural surface tractions. In practice, it leads to the computation of the generalized and/or total flow-induced load on the wet surface of the structure. For embedded boundary methods, this computation shares with standard lift and drag computations the same difficulty of integrating the pressure and viscous tractions of the flow on a surface that is not explicitly represented in the computational fluid model. Typically, this issue has been addressed in the literature separately from that associated with the semi-discretization of the first transmission condition, albeit using in many cases similar techniques based on interpolation and/or extrapolation, with or without resorting to the explicit computation of the intersection of the embedded interfaces and embedding mesh [19, 17].

Focusing on compressible flows, this paper contributes a new approach for the treatment of fluid-wall interfaces for both purely fluid and fluid-structure applications. The proposed approach is a departure from the methods outlined above and related published works in that it treats the velocity and pressure boundary conditions on the embedded interfaces simultaneously, rather than disjointly. Furthermore, instead of relying for this purpose exclusively on interpolation or extrapolation, the proposed method enforces the appropriate value of the fluid velocity at a wall and recovers the value of the fluid pressure at this wall via the exact solution of local, one-dimensional, fluid-structure Riemann problems. This paper also presents two consistent and conservative methodologies for evaluating flow-induced forces and moments on rigid and flexible embedded interfaces. One of them is based on the local reconstruction of the embedded discrete interfaces. The other one is based on the level set concept. It is particularly attractive because it rigorously allows the substitution of an embedded discrete interface by a simpler surrogate, which simplifies computations. All of these proposed algorithms are applicable independently from the type and topology of the embedding CFD grid. They are described here in the context of inviscid flows and the finite volume (FV) method. However, the underlying ideas are equally applicable to viscous flows and the finite element (FE) method.

The remainder of this paper is organized as follows. Section 2 specifies the context and governing equations of the problem summarized above, and formulates the objectives of this paper. Section 3 presents a method based on the exact solution of local, one-dimensional, fluid-structure Riemann problems for treating simultaneously the velocity and pressure wall boundary conditions in fluid and fluid-structure interaction problems. Section 4 presents a numerical scheme for computing the generalized and total flow-induced forces and moments acting on embedded fluid-structure interfaces that relies on a local and geometrically accurate reconstruction of these interfaces within the CFD solver. Alternatively, Section 5 proposes a computational framework based on the level set concept that allows in principle the rigorous evaluation of these forces and moments on surrogates of the embedded interfaces, in view of simplifying software development and minimizing computational overhead. It also proposes a specific surrogate for a given fluid/structure discrete interface that trades higher geometrical fidelity for computational simplicity. Both algorithms presented in Section 4 and Section 5 are consistent in the sense that they preserve some important property of the exact computation of a flow-induced load, and conservative in the sense that the sum of the virtual works of the fluid and structural interface forces vanishes. In Section 6, the performance of all algorithms proposed in this paper is first assessed in details for an academic problem, then highlighted for three-dimensional fluid-structure interaction problems associated with the fields of aeronautics and underwater implosion. Finally, conclusions are offered in Section 7.

2. PROBLEM CONTEXT AND FORMULATION

2.1. Governing equations

In this work, the compressible flow in a domain of interest $\Omega_F \subset \mathbb{R}^3$ with a wall boundary surface Σ_w is assumed to be inviscid and governed by the Euler equations. These can be written in vector and conservation form as

$$\frac{\partial W}{\partial t} + \vec{\nabla} \cdot \vec{\mathscr{F}}(W) = 0, \qquad \text{in } \Omega_F \tag{1}$$

where

$$W = (\rho,\, \rho v_x,\, \rho v_y,\, \rho v_z,\, E)^T, \quad \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)^T, \qquad \vec{\mathscr{F}}(W) = (\mathscr{F}_x(W),\, \mathscr{F}_y(W),\, \mathscr{F}_z(W))^T, \tag{2}$$

$$\mathscr{F}_x = \begin{pmatrix} \rho v_x \\ p + \rho v_x^2 \\ \rho v_x v_y \\ \rho v_x v_z \\ v_x(E+p) \end{pmatrix}, \qquad \mathscr{F}_y = \begin{pmatrix} \rho v_y \\ \rho v_x v_y \\ p + \rho v_y^2 \\ \rho v_y v_z \\ v_y(E+p) \end{pmatrix}, \qquad \mathscr{F}_z = \begin{pmatrix} \rho v_z \\ \rho v_x v_z \\ \rho v_y v_z \\ p + \rho v_z^2 \\ v_z(E+p) \end{pmatrix}, \tag{3}$$

ρ denotes the fluid density, E is its total energy per unit volume and is given by $E = \rho e + \frac{1}{2}\rho(v_x^2 + v_y^2 + v_z^2)$, where e denotes the internal energy per unit mass, p denotes the fluid pressure, and $\vec{v} = (v_x,\ v_y,\ v_z)$ is its velocity vector. For simplicity, but without any loss of generality, the equation of state (EOS) of the fluid is assumed to be that of the ideal gas — that is, $p = (\gamma - 1)\rho e$, where γ is the adiabatic index.

If the wall boundary surface is fixed in time, it is denoted by Σ_w°. In this case, the slip boundary condition satisfied by the fluid velocity vector can be written as

$$\vec{v} \cdot \vec{n}_w^\circ = 0 \qquad \text{on } \Sigma_w^\circ, \tag{4}$$

where $\vec{n}_w^\circ$ denotes the unit normal to Σ_w°.

On the other hand, if the wall boundary surface or a subset of it is associated with the wet surface of a moving rigid body or a flexible dynamic structure Ω_S, Σ_w is decomposed as follows

$$\Sigma_w = \Sigma_w^\circ \bigcup \Sigma_w^t, \tag{5}$$

where Σ_w° and Σ_w^t denote the fixed and time-dependent components of Σ_w, respectively. In this case, the equations of dynamic equilibrium of the moving body and/or flexible dynamic structure are written in compact form as

$$\rho_S \frac{\partial^2 u_j}{\partial t^2} = \frac{\partial}{\partial x_i}\left(\sigma_{ij} + \sigma_{im}\frac{\partial u_j}{\partial x_m}\right) + b_j \qquad \text{in } \Omega_S, \quad j = 1,\ 2,\ 3, \tag{6}$$

where the subscripts i, j, and m varying between 1 and 3 designate the coordinate system (x, y, z), $\vec{u}$ is the displacement vector field of the moving body or flexible structure, σ denotes the second Piola-Kirchhoff stress tensor, and $\vec{b}$ denotes the vector of body forces acting in Ω_S. Given a structural material and its constitutive law, the resulting fluid-structure interaction problem is governed by Eq. (1), Eq. (6), and the two transmission conditions

$$\left(\vec{v} - \frac{\partial \vec{u}}{\partial t}\right) \cdot \vec{n}_w^t \quad = \quad 0 \qquad \text{on } \Sigma_w^t \subset \Sigma_w, \tag{7}$$

and

$$\left(\sigma_{ij} + \sigma_{im}\frac{\partial u_j}{\partial x_m} + p\delta_{ij}\right) n_{w_i}^t - \mathscr{T}_j \quad = \quad 0 \qquad \text{on } \Sigma_w^t \subset \Sigma_w, \quad j = 1,\ 2,\ 3, \tag{8}$$

where δ_{ij} denotes the Kronecker delta, $\vec{n}_w^t = \vec{n}_w^t(t)$ is the outward unit normal to $\Sigma_w^t = \Sigma_w^t(t)$, and $\mathscr{T}_j$ denotes the tractions due to external forces whose origin is not due to the flow.

If $\Omega_S = \Omega_S(t)$ is a moving rigid body, σ is set to zero in Eq. (6) and Eq. (8).

2.2. *Semi-discretization*

The governing fluid equations (1) are semi-discretized here by a classical FV method. The basic steps of this method are outlined below, in order to introduce a notation and some concepts that are beneficial to Section 3.

Let $\mathscr{D}_h$ denote a standard discretization of the flow domain of interest Ω_F, where h designates the maximal length of the edges of this discretization. For every vertex $V_i \in \mathscr{D}_h$, $i = 1, \cdots, N_V$, a cell or control volume C_i is constructed. For example, if $\mathscr{D}_h$ consists of hexahedra, C_i is defined as the union of the sub-hexahedra resulting from the subdivision by means of the median planes of each hexahedron of $\mathscr{D}_h$ having V_i as a vertex (see Figure 1). The boundary surface of C_i is denoted by ∂C_i, and the unit outward normal to ∂C_i is denoted by $\vec{n}_i = (n_{i_x}, n_{i_y}, n_{i_z})$. The union of all of the control volumes defines a dual discretization of $\mathscr{D}_h$ verifying

$$\bigcup_{i=1}^{N_V} C_i = \mathscr{D}_h. \tag{9}$$

Using the standard characteristic function associated with a control volume C_i, a standard variational approach, and integration by parts, Eq. (1) can be transformed into its weaker form

$$\begin{aligned} \int_{C_i} \frac{\partial W_h}{\partial t}\, d\Omega \quad &+ \quad \sum_{j\in K(i)} \int_{\partial C_{ij}} \vec{\mathscr{F}}(W_h)\cdot\vec{n}_{ij}\, d\Sigma \quad <1> \\ &+ \quad \int_{\partial C_i \cap \Sigma_E} \vec{\mathscr{F}}(W_h)\cdot\vec{n}_E\, d\Sigma \quad <2> \\ &+ \quad \int_{\partial C_i \cap \Sigma_\infty} \vec{\mathscr{F}}(W_h)\cdot\vec{n}_i\, d\Sigma \quad <3> \\ &= 0, \end{aligned} \tag{10}$$

where W_h denotes the approximation of the fluid state vector W in a semi-discrete space, $K(i)$ denotes the set of neighboring vertices of V_i, ∂C_{ij} denotes a segment of ∂C_i that is defined below, $\vec{n}_{ij}$ is the unit outward normal to ∂C_{ij}, Σ_E is the discrete approximation of Σ_w, $\vec{n}_E$ is the unit outward normal to Σ_E, and Σ_∞ denotes the far-field boundary of the flow (if it exists). This weaker form reveals that in practice, the computations are performed in a one-dimensional manner, essentially by evaluating fluxes along normal directions to boundaries of the control volumes. For this purpose, ∂C_i is split in control volume boundary facets ∂C_{ij} connecting the centroids of the hexahedra having V_i and V_j as common vertices (Figure 1), and term $< 1 >$ in (10) is approximated as follows

$$\sum_{j\in K(i)} \int_{\partial C_{ij}} \vec{\mathscr{F}}(W_h)\cdot\vec{n}_{ij}\, d\Sigma = \sum_{j\in K(i)} \Phi_{\mathscr{F}_{ij}}(W_i, W_j, \mathrm{EOS}, \vec{n}_{ij}). \tag{11}$$

Here, $\Phi_{\mathscr{F}_{ij}}$ is a numerical flux function associated, for example, with a first-order upwind scheme such as Roe's approximate Riemann solver [20] — or more typically, with a second-order extension based on the MUSCL (Monotonic Upwind Scheme Conservation Law) [22] — and W_i and W_j denote the values of W_h at the vertices V_i and V_j, respectively. Term $< 3 >$ in (10) is usually approximated by a far-field boundary technique such as the non-reflective version of the flux-splitting of Steger and Warming [21]. Finally, the computation of term $< 2 >$ in (10) involves the first transmission condition (7) and therefore constitutes one of the two main objectives of this paper.

On the other hand, the structural equations of dynamic equilibrium (6) are semi-discretized by the FE method. This leads to

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{f}^{int}(\mathbf{u},\dot{\mathbf{u}}) = \mathbf{f}^F(\mathbf{w}) + \mathbf{f}^{ext}, \tag{12}$$

where $\mathbf{M}$ denotes the FE mass matrix and is symmetric positive definite, $\mathbf{u}$ denotes the vector of structural displacements, $\mathbf{f}^{int}$, $\mathbf{f}^{ext}$, and $\mathbf{f}^F$ denote the vectors of internal, external, and flow-induced forces, respectively, and a dot designates a time derivative.

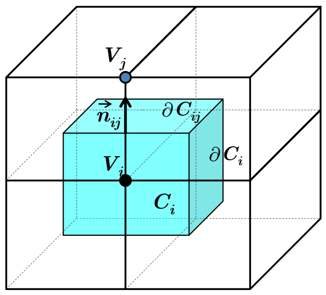

Figure 1. Definition of a control volume C_i, its boundary surface ∂C_i, and a facet ∂C_{ij} of ∂C_i (view of half entities for an hexahedral discretization).

2.3. *Context and objectives*

The first objective of this paper is to propose a numerical method for discretizing the first transmission condition (7) — and as a particular case the slip boundary condition (4) — in the context of embedded boundary methods and static and dynamic rigid or flexible embedded interfaces. The second objective is to present, in the same context, two alternative approaches for discretizing the second transmission condition (8) and computing the generalized and total flow-induced forces and moments on both rigid bodies and flexible structures. Before characterizing this specific context, conventional methods for (semi)-discretizing the transmission conditions (7) and (8) on body-fitted fluid meshes are first outlined.

For a body-fitted CFD grid, $\mathscr{D}_h$ includes the surface discretizations of the wall boundaries Σ_w° and Σ_w^t, and therefore the surface discretization Σ_E of the entire wall boundary surface Σ_w. These geometrical discretizations are denoted here by Σ_F°, Σ_F^t, and $\Sigma_F = \Sigma_E$, respectively (see Figure 2). The corresponding unit outward normals are denoted by $\vec{n}_F^\circ$, $\vec{n}_F^t$, and $\vec{n}_F$, respectively. The counterparts of Σ_F^t and $\vec{n}_F^t$ in the geometrical discretization of the structural domain Ω_S are denoted by Σ_S and $\vec{n}_S$, respectively. In this case, using the definitions given in (3), term $< 2 >$ in (10) can be evaluated as follows

$$\int_{\partial C_i \cap \Sigma_E} \vec{\mathscr{F}}(W_h) \cdot \vec{n}_E \, d\Sigma = \int_{\partial C_i \cap \Sigma_F} \begin{pmatrix} \rho_h \vec{v}_h \cdot \vec{n}_F \\ p_h n_{F_x} + \rho_h v_{h_x} \vec{v}_h \cdot \vec{n}_F \\ p_h n_{F_y} + \rho_h v_{h_y} \vec{v}_h \cdot \vec{n}_F \\ p_h n_{F_z} + \rho_h v_{h_z} \vec{v}_h \cdot \vec{n}_F \\ (E_h + p_h) \vec{v}_h \cdot \vec{n}_F \end{pmatrix} d\Sigma. \tag{13}$$

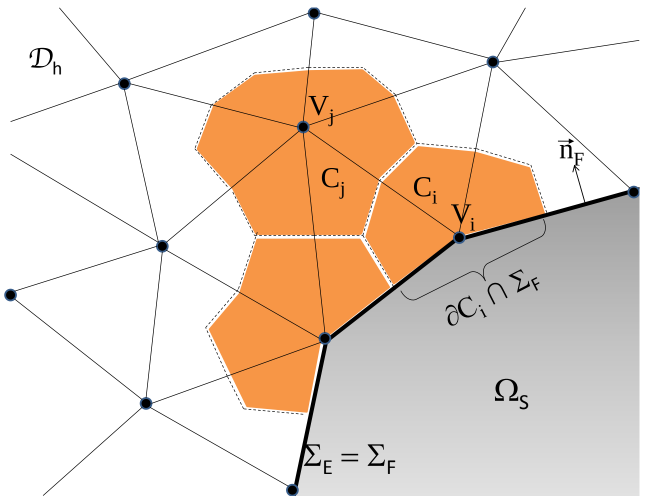

Figure 2. Spatial discretization of a two-dimensional fluid computational domain using a body-fitted CFD grid with triangular elements.
.

Substituting Eq. (7) and Eq. (5) into Eq. (13) above leads to

$$\int_{\partial C_i \cap \Sigma_E} \vec{\mathscr{F}}(W_h)\cdot\vec{n}_E\, d\Sigma = \int_{\partial C_i \cap \Sigma_F^\circ} \begin{pmatrix} 0 \\ p_h n^\circ_{F_x} \\ p_h n^\circ_{F_y} \\ p_h n^\circ_{F_z} \\ 0 \end{pmatrix} d\Sigma + \int_{\partial C_i \cap \Sigma_F^t} \begin{pmatrix} \rho_h \dfrac{\partial \vec{u}_h}{\partial t}\cdot\vec{n}^t_F \\ p_h n^t_{F_x} + \rho_h v_{h_x}\dfrac{\partial \vec{u}_h}{\partial t}\cdot\vec{n}^t_F \\ p_h n^t_{F_y} + \rho_h v_{h_y}\dfrac{\partial \vec{u}_h}{\partial t}\cdot\vec{n}^t_F \\ p_h n^t_{F_z} + \rho_h v_{h_z}\dfrac{\partial \vec{u}_h}{\partial t}\cdot\vec{n}^t_F \\ (E_h + p_h)\dfrac{\partial \vec{u}_h}{\partial t}\cdot\vec{n}^t_F \end{pmatrix} d\Sigma. \qquad (14)$$

Eq. (14) above outlines a simple and popular method for enforcing wall boundary conditions on body-fitted fluid grids. In the presence of a moving wall boundary surface Σ^t_w, the velocity $\dfrac{\partial \vec{u}_h}{\partial t}$ is determined from the dynamics of the wall, whether it is rigid or flexible.

The discretization of the second transmission condition (8) on body-fitted CFD grids often amounts to computing a generalized flow-induced load of the form

$$\mathbf{f}^F_i = \sum_j c_{ij}\int_{\Sigma^t_{F_j}} (-p_h)\vec{n}^t_F D_j\, d\Sigma \qquad \text{or} \qquad \mathbf{f}^F_i = \sum_j c_{ij}\int_{\Sigma_{S_j}} p_h \vec{n}_S D_j\, d\Sigma, \qquad (15)$$

where a bold font instead of an arrow is used to designate the discrete aspect of a vector quantity consistently with the notation of Eq. (12), $\mathbf{f}^F_i$ is the value at node V_i of the FE structural model of interest of the vector of flow-induced forces $\mathbf{f}^F$ (12), the c_{ij} are some scalar coefficients, and the D_j are some shape functions with support on subsets $\Sigma^t_{F_j}$ of Σ^t_F or subsets Σ_{S_j} of Σ_S.

Whether the wall boundary surface or a subset of it is associated with the wet surface of a dynamic structure or not, as long as the CFD grid remains body-fitted during a numerical simulation, there is only one issue worth mentioning about the numerical evaluation of the second term in the right hand-side of Eq. (14), and that of the generalized load vector (15) and/or corresponding resultant over the entire wall boundary surface. This issue consists of the treatment of the frequent case where the surface discretizations Σ_F^t and Σ_S are non-matching, and has been amply discussed in the literature (for example, see [23, 24, 25]).

For a non body-fitted CFD grid however (for example, see Figure 3), $\mathscr{D}_h$ does not contain at any time (in principle) the surface discretization Σ_E — that is, neither the discretization Σ_F° of Σ_w° nor the discretization Σ_F^t of Σ_w^t. Therefore, different approaches are required in this case for evaluating the numerical flux (14) and generalized load vector (15).

Unlike previously published alternatives, the method proposed in this paper for computing the numerical flux (14) in an embedded boundary method treats the velocity and pressure boundary conditions on the embedded interfaces Σ_w° and Σ_w^t simultaneously, rather than disjointly. Furthermore, this method is not based exclusively on interpolation or extrapolation, and is independent of the type and topology of the discretization $\mathscr{D}_h$ of the fluid domain. However, it is applicable to compressible flow problems only.

Two consistent and conservative approaches are also proposed in this paper for computing in an embedded boundary method the generalized load vector (15). One of them is based on the local reconstruction of embedded discrete interfaces. The other one is based on the level set concept. It rigorously allows the substitution of any embedded discrete interface by a surrogate one that simplifies computations. Both approaches are independent from the proposed method for enforcing the velocity condition (7) on an embedded discrete interface and recovering there the value of the fluid pressure. However, they are discussed in details in this paper in conjunction with this method only. Furthermore, both of these load computation approaches are fully described in the context of inviscid flows and the FV method. Nevertheless, the underlying ideas are equally applicable to viscous flows and the FE method.

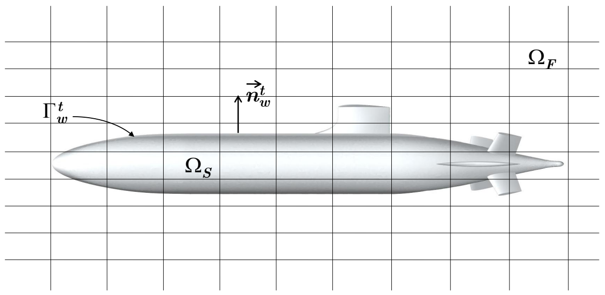

Figure 3. Non body-fitted fluid grid and embedded mockup submarine.

3. EXACT FLUID-STRUCTURE RIEMANN SOLVER FOR THE TREATMENT OF THE FLUID-WALL PRESSURE AND VELOCITY

In [26], a FV method based on the exact solution of local, one-dimensional, two-phase Riemann problems was proposed for the solution of compressible multi-fluid problems. Here, this method is adapted for the solution of fluid-structure interaction problems. More specifically, it is tailored to the enforcement of the velocity transmission condition (7) at a wall boundary surface and the recovery of the value of the fluid pressure at this boundary surface.

3.1. Algorithm for treating a fluid-structure interface

Let the superscripts L and R designate the grid points on the left and right sides of a small region of a discrete interface Σ_E embedded in a given Eulerian CFD grid $\mathscr{D}_h$. If V_{i^L} denotes a first-layer grid point on the left side of Σ_E, then V_{i^R} denotes a vertex on its right side that is connected to V_{i^R} by an edge $V_{i^L}V_{i^R}$ traversing Σ_E (see Figure 4). Let also $M_{i^L i^R}$ denote the point where $V_{i^L}V_{i^R}$ intersects the control volume boundary facet

$$\partial C_{i^L i^R} = C_{i^L} \cap C_{i^R}. \tag{16}$$

Consider first the case where the flow occurs only on one side of the region of interest of Σ_E — for example, the left side. Then, the following four-step algorithm is proposed for treating a fluid-structure interface in an embedded boundary method.

<u>ALGORITHM 1</u>

For each control volume C_{i^L} and for each edge $V_{i^L}V_{i^R}$ intersecting the embedded discrete interface Σ_E:

1. If the considered region of Σ_E corresponds to a dynamic embedded interface, compute the velocity of Σ_E at the point $M_{i^L i^R}$, $\dot{\mathbf{u}}_M = \dot{\mathbf{u}}(M_{i^L i^R})$, by interpolation, extrapolation, or a combination of interpolation and extrapolation of the discrete velocity field $\dot{\mathbf{u}}$ obtained from the solution of the discrete structural equations of dynamic equilibrium (12). If on the other hand the considered region of Σ_E corresponds to a static embedded interface, set $\dot{\mathbf{u}}_M \cdot \vec{n}_{E_M} = 0$, where $\vec{n}_{E_M} = \vec{n}_E(M_{i^L i^R})$ is the unit outward normal to Σ_E at the point $M_{i^L i^R}$.
2. Assume that at the point $M_{i^L i^R}$, the control volume boundary facet $\partial C_{i^L i^R}$ (16) and the embedded discrete interface Σ_E coincide (ASSUMPTION 1). Construct and solve analytically at $M_{i^L j^R}$ the following "one-sided", one-dimensional Riemann problem

$$\left\{ \begin{array}{rcl} \dfrac{\partial \tilde{W}}{\partial t} + \dfrac{\partial \vec{\mathscr{F}}}{\partial s}(\tilde{W}) & = & 0 \\ \tilde{W}(s,0) & = & \tilde{W}_{i^L},\ \text{if } s \geq 0 \\ v(v_0 t, t) & = & v_0,\ \forall\, 0 \leq t \leq \Delta t \end{array} \right. \tag{17}$$

where

$$\tilde{W} = \tilde{W}(s,t) = (\rho,\ \rho v,\ E)^T. \tag{18}$$

Here, $\tilde{W}$ is the conservative state vector of the one-dimensional flow along the opposite direction to $\vec{n}_{i^L i^R}$, s is the abscissa along this direction and has its origin at the point $M_{i^L i^R}$ (see Figure 4), and the flux function $\vec{\mathscr{F}}$ is defined by

$$\vec{\mathscr{F}}(\tilde{W}) = \left(\rho v,\ \rho v^2 + p,\ (E+p)v\right)^T. \tag{19}$$

The initial condition $\tilde{W}_{i^L}$ is obtained from W_{i^L} as follows

$$\tilde{W}_{i^L} = \begin{bmatrix} \rho_{i^L} \\ v_{i^L}^{\perp} \\ v_{i^L}^{\perp}(E_{i^L}^{\perp} + p_{i^L}) \end{bmatrix}, \tag{20}$$

where $v_{i^L}^{\perp} = \vec{v}_{i^L} \cdot \vec{n}_{i^L i^R}$ is the normal component of the fluid velocity at V_{i^L} and $E_{i^L}^{\perp} = \rho_{i^L} e_{i^L} + \frac{1}{2}(v_{i^L}^{\perp})^2$. Finally,

$$v_0 = \dot{\mathbf{u}}_M \cdot \vec{n}_{E_M} \tag{21}$$

is the normal component of the structural velocity at the point $M_{i^L i^R}$ and is assumed to be constant for $0 \leq t \leq \Delta t$, where Δt denotes the computational time-step. Therefore, as far as this one-dimensional problem is concerned, the fluid-structure interface is located at $s_0(t) = v_0 t$ at any time $t \in [0,\ \Delta t]$.

The exact solution of the above one-sided, one-dimensional Riemann problem contains a constant (in time) state at the fluid-structure interface which is denoted here by

$$W^* = (\rho^*,\ \rho^* v^*,\ E^*)^T. \tag{22}$$

The corresponding pressure is denoted by p^*.

3. Recover the state vector of the three-dimensional flow at $M_{i^L i^R}$, namely W_M^L, as follows

$$\begin{aligned} \rho_M^L &= \rho^* \\ \vec{v}_M^L &= v^* \vec{n}_{E_M} + \left(\vec{v}_{i^L} - (\vec{v}_{i^L} \cdot \vec{n}_{E_M})\vec{n}_{E_M}\right) \\ p_M^L &= p^* \\ e_M^L &= \frac{p_M^L}{(\gamma-1)\rho_M^L} \\ E_M^L &= \rho_M^L e_M^L + \frac{1}{2}\vec{v}_M^L \cdot \vec{v}_M^L \end{aligned}$$

4. Evaluate each component of term $<1>$ in (10) as follows

$$\int_{\partial C_{i^L i^R}} \vec{\mathscr{F}}(W_h) \cdot \vec{n}_{i^L i^R}\, d\Sigma = \Phi_{\mathscr{F}_{i^L i^R}}(W_{i^L}, W_M^L, \mathrm{EOS}^L, \vec{n}_{i^L i^R}). \tag{23}$$

The one-sided Riemann problem (17) is a left fluid-structure Riemann problem. Its exact solution $\tilde{W}(s,t)$ is self-similar — that is, it verifies $\tilde{W}(s,t) = \tilde{W}(s/t)$. It is given in APPENDIX A in order to keep this paper as self-contained as possible.

If the flow occurs on both left and right sides of the considered region of Σ_E as in the case of embedded thin structures, ALGORITHM 1 is applied twice: (1) once as described above to compute for each control volume C_{i^L} and each edge $V_{i^L}V_{i^R}$ intersecting Σ_E the numerical flux $\Phi_{\mathscr{F}_{i^L i^R}}(W_{i^L}, W_M^L, \mathrm{EOS}^L, \vec{n}_{i^L i^R})$ and the pressure value p_M^L, and (2) a second time for each control volume

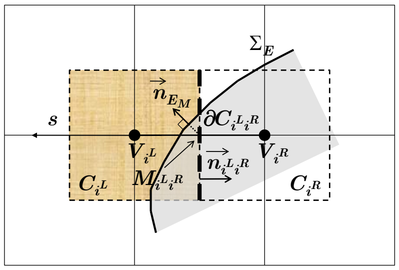

Figure 4. Two control volumes on the left and right sides of a region of an embedded discrete interface (two-dimensional case, quadrilateral mesh).

C_{i^R} and each edge $V_{i^R}V_{i^L}$ intersecting Σ_E with the left one-dimensional fluid-structure Riemann problem replaced by its right counterpart to compute instead $\Phi_{\mathscr{F}_{i^R i^L}}(W_{i^R}, W_M^R, \text{EOS}^R, \vec{n}_{i^R i^L})$ and p_M^R.

Remark 1. It is noted that ASSUMPTION 1 described above introduces a first-order geometric (and therefore spatial) error in the computation of the numerical flux function across an embedded interface and the recovery of the value of the pressure field on that surface.

3.2. Some implementational details

ALGORITHM 1 can be seamlessly adapted to any time-integration scheme. Its implementation in a conventional FV-based CFD solver is straightforward, provided that computational tools are identified for: (a) determining the status of a grid point as defined below, and (b) computing the intersection of the CFD grid and the embedded discrete interfaces of interest. These two issues can be coupled and therefore the order in which they are discussed does not matter.

Here, the status of a grid point V_i is defined as an integer that is negative when V_i is inside an obstacle, or otherwise non negative and equal to the integer identifier of the EOS governing the medium containing that grid point. Several fast computational technologies can be applied for determining and updating the status of a grid point. These include, for example, the well known concept of a (local) signed distance function, and the bounding box hierarchies and flood-fill algorithms that are often encountered in computational graphics for accelerating the computational process. Ray casting can be used for computing the intersection points I_k of the given CFD grid and embedded discrete interfaces (for example, see [27] and references therein).

Once the status of the CFD grid points have been determined and the intersection points I_k have been computed, the implementation of ALGORITHM 1 in a given FV-based CFD solver can be performed essentially by modifying the flux computation loop as follows. For each control volume boundary facet ∂C_{ij}, the status of the vertices V_i and V_j are first inspected. If both vertices are found to have the same non negative status, both vertices are in the same fluid medium: in this case, the numerical fux function $\Phi_{\mathscr{F}_{ij}}$ (11) is computed as usual, using the EOS of the fluid containing these two grid points. If both vertices are found to have the same negative status, both vertices are inside the obstacle and therefore no flux is computed across the boundary facet ∂C_{ij}. If V_i and V_j are found to have different status and only one of them is negative, one of these two vertices is in a fluid medium and the other inside the obstacle: in this case, $\Phi_{\mathscr{F}_{ij}}$ is computed as in Eq. (23). On the other hand, if V_i and V_j are found to have different status that are both non negative, these two grid points are separated by a thin structure (for example, a shell): in this case, two flux functions $\Phi_{\mathscr{F}_{ij}}$ are computed, each as in Eq. (23).

4. A NUMERICAL ALGORITHM FOR LOAD COMPUTATION BASED ON THE LOCAL RECONSTRUCTION OF EMBEDDED INTERFACES

As noted in Section 2.3, the discretization of the second transmission condition (8) on body-fitted CFD grids typically amounts to computing a generalized flow-induced load of the form given in (15) and/or its resultant. However, as illustrated in Figure 3, for a non body-fitted CFD grid, $\mathscr{D}_h$ does not contain in general the surface discretizations Σ_F° and Σ_F^{t} of the wall boundaries Σ_w° and Σ_w^{t}, respectively. Consequently, the CFD solver based on an embedded boundary method does not have explicit representations of the surfaces where the flow-induced load needs to be computed. On the other hand, the FE structural model contains a discrete representation Σ_S of flexible wall boundaries but the structural solver does not have direct access to the pressure field computed by the flow solver. For these reasons, an approach different than that outlined in Section 2.3 is required in this case for evaluating the generalized load vector (15) and its resultant. For this purpose, a first approach is presented here. This approach relies on the local reconstruction within the CFD solver of the embedded discrete interfaces of interest.

Let Σ_E denote an embedded discrete interface consisting of elements τ_q of arbitrary shape

$$\Sigma_E = \bigcup_{q=1}^{N_\tau} \tau_q. \tag{24}$$

For a flexible wall boundary, Σ_E can be in practice Σ_S or another discrete representation of this wall boundary characterized by a different (for example, finer) mesh resolution. A key observation, particularly in the context of this work, is that the evaluation of the fluid state vector at an intersection of the embedded discrete interface and the CFD grid is simpler than its evaluation at a vertex of that surface. Indeed, at the intersections of Σ_E and $\mathscr{D}_h$, W can be — and in this work, is already — determined from the solution of local, one-dimensional, one-sided Riemann problems as explained in Section 3. On the other hand, the evaluation of W at the vertices of Σ_E requires relatively complex interpolations, particularly in three dimensions. This observation is behind the specific approach described next for reconstructing an embedded discrete interface within a given CFD grid.

4.1. Algorithm for reconstructing an embedded discrete interface

After the points I_k defined by the intersections of Σ_E and the edges (both in two and three dimensions) of $\mathscr{D}_h$ have been computed — for example, using ray casting as already mentioned in Section 3 — ALGORITHM 2 outlined below reconstructs the embedded discrete interface Σ_E as the union $\widetilde{\Sigma}_E$ of a collection of *triangles* $\tilde{\tau}_q$ obtained by connecting the points I_k — that is,

$$\widetilde{\Sigma}_E = \bigcup_{q=1}^{N_{\tilde{\tau}}} \tilde{\tau}_q. \tag{25}$$

The choice of a triangle for $\widetilde{\tau}_q$ is based on the fact that a tetrahedron, prism, pyramid, or hexahedron — all of which can be encountered in a CFD grid — usually intersects a surface in at least three non colinear points, and the simplest approach for connecting more than three points is to use triangles.

<u>ALGORITHM 2</u>

For each element $\mathscr{D}_h^e$ in $\mathscr{D}_h$:

1. Inspect the status of the vertices of $\mathscr{D}_h^e$.
2. If all status are positive or all of them are negative, $\mathscr{D}_h^e$ does not cross Σ_E.
3. Otherwise, $\mathscr{D}_h^e$ crosses Σ_E and contributes one or more elements $\widetilde{\tau}_q$ to $\tilde{\Sigma}_E$ that are constructed as follows.
   (a) Initialize a list $\mathscr{L}^e$ of intersection points.
   (b) Loop on the edges of $\mathscr{D}_h^e$: for each edge intersecting Σ_E, identify the intersection point I_k and add it to $\mathscr{L}^e$.
   (c) Connect all points of $\mathscr{L}^e$ to form one or more elements $\tilde{\tau}_q$.

As illustrated in Figure 5 which focuses on the two-dimensional case for the sake of clarity, the reconstructed discrete interface $\widetilde{\Sigma}_E$ is in general a geometrically accurate approximation of the embedded discrete interface Σ_E, provided that the CFD grid has at least the same mesh resolution as Σ_E, which is usually the case.

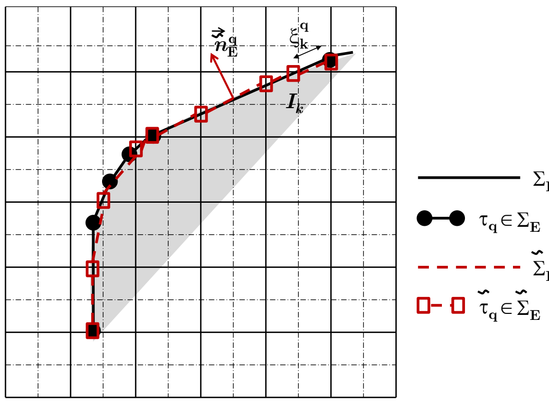

Figure 5. Reconstruction of the embedded discrete interface (two-dimensional case, quadrilateral mesh).

4.2. Algorithm for computing the generalized and total flow-induced load vectors

For simplicity, but without any loss of generality, assume next that the embedded discrete interface Σ_E is the discretization of the wet surface of the structure of interest, Σ_S. For each computed intersection point I_k, the id of the intersecting element of Σ_E, τ_q, and the natural coordinates of I_k in τ_q, ξ_k^q (see Figure 5), are collected. This information can be exploited to effectively interpolate at each point I_k the displacement and velocity of the reconstructed discrete interface $\widetilde{\Sigma}_E$ as follows

$$\mathbf{u}(I_k) = \sum_j N_j^q(\xi_k^q)\mathbf{u}_j \qquad \text{and} \qquad \dot{\mathbf{u}}(I_k) = \sum_j N_j^q(\xi_k^q)\dot{\mathbf{u}}_j, \tag{26}$$

where N_j^q is the FE shape function associated with node V_j^q of the element (or face) τ_q containing the point I_k, the summation in (26) is carried over all relevant shape functions of τ_q, these shape functions satisfy the partition of unity property, and $\mathbf{u}_j$ and $\dot{\mathbf{u}}_j$ are the displacement and velocity of Σ_E at node V_j^q of element (or face) τ_q, respectively.

Let p_k and $\mathbf{v}_k$ denote the computed values of the fluid pressure and velocity vector at the interface points I_k. In this work, these values are computed by solving local, one-dimensional, one-sided Riemann problems as explained in Section 3. On the reconstructed discrete interface $\widetilde{\Sigma}_E$, the fluid pressure and velocity fields can be approximated at each point $\vec{x} \in \tilde{\tau}_q$ as follows

$$p_h(\vec{x}) = \sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)p_k, \qquad \text{and} \qquad \vec{v}(\vec{x}) = \sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)\mathbf{v}_k, \tag{27}$$

where $\tilde{N}_k^q$ is the classical FE shape function associated with node I_k^q of the triangle $\tilde{\tau}_q$, and $\eta(\vec{x})$ are the natural coordinates of the point $\vec{x}$ in the element $\tilde{\tau}_q$ containing it.

Discretizing the first transmission condition (7) on $\widetilde{\Sigma}_E$ yields

$$\mathbf{v}_k \vec{n}_E^q = \dot{\mathbf{u}}(I_k)\vec{n}_E^q, \tag{28}$$

where $\vec{n}_E^q$ is the normal to an element $\tilde{\tau}_q$ of $\tilde{\Sigma}_E$ connected to I_k.

Let N_E denote the total number of vertices in Σ_E. Following the methodology developed in [25] for computing the generalized flow-induced load vector in the context of the ALE framework where the CFD grid remains body-fitted at all time-instances, the generalized flow-induced load vector at node V_i of the FE structural model, $\mathbf{f}_i^F$, is computed here by applying the virtual power principle at the fluid/structure interface. In this case, this interface is approximated by the reconstructed discrete interface $\widetilde{\Sigma}_E$. Using the δ symbol to denote a virtual quantity, Eq. (27), Eq. (28) and Eq. (26), this principle can be written as

$$\begin{aligned}
\sum_{i=1}^{N_E} \mathbf{f}_i^F \delta\dot{\mathbf{u}}_i &= -\sum_{\tilde{\tau}_q} \int_{\tilde{\tau}_q} \big(-p_h(\vec{x})\big)\vec{n}_E^q \delta\vec{v}_h(\vec{x})\, d\tau \\
&= \sum_{\tilde{\tau}_q} \int_{\tilde{\tau}_q} \left[\sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)p_k\vec{n}_E^q \sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)\delta\mathbf{v}_k\right] d\tau \\
&= \sum_{\tilde{\tau}_q} \int_{\tilde{\tau}_q} \left[\sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)p_k\vec{n}_E^q \sum_{k=1}^{3} \tilde{N}_k^q\big(\eta(\vec{x})\big)\Big[\sum_j N_j^q(\xi_k^q)\delta\dot{\mathbf{u}}_j\Big]\right] d\tau.
\end{aligned} \tag{29}$$

Eq. (29) above and the partition of unity property of the shape functions N_i^q and $\tilde{N}_k^q$ lead to the following algorithm for computing the generalized flow-induced load vector at node V_i of the given FE structural model and its resultant.

ALGORITHM 3

$$\mathbf{f}_i^F = \sum_{\substack{\tilde{\tau}_q / \exists I_k \in \tilde{\tau}_q \\ and \\ I_k \in (\tau_q \ni V_i)}} \int_{\tilde{\tau}_q} \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) p_k \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) N_i^q(\xi_k^q)\, d\tau, \tag{30}$$

and

$$\begin{aligned}
\mathbf{f}^F &= \sum_{i=1}^{N_E} \mathbf{f}_i^F = \sum_{i=1}^{N_E} \sum_{\substack{\tilde{\tau}_q / \exists I_k \in \tilde{\tau}_q \\ and \\ I_k \in (\tau_q \ni V_i)}} \int_{\tilde{\tau}_q} \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) p_k \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) N_i^q(\xi_k^q)\, d\tau \\
&= \sum_{\tilde{\tau}_q \in \tilde{\Sigma}_E} \int_{\tilde{\tau}_q} \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) p_k \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) \sum_{i=1}^{N_E} N_i^q(\xi_k^q)\, d\tau \\
&= \sum_{\tilde{\tau}_q \in \tilde{\Sigma}_E} \int_{\tilde{\tau}_q} \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) p_k \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x}))\, d\tau \\
&= \sum_{\tilde{\tau}_q \in \tilde{\Sigma}_E} \int_{\tilde{\tau}_q} \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) p_k \vec{\tilde{n}}_E^q\, d\tau \qquad \text{(as expected).}
\end{aligned} \tag{31}$$

Remark 2. If the embedded discrete interface Σ_E does not coincide with the discretization of the wet surface of the structure of interest, Σ_S, the derivation of ALGORITHM 3 presented above is extended as follows. First, the projection of Σ_E onto Σ_S is computed only once in a pre-processing step, and the natural coordinates of each projected point of Σ_E in the element of Σ_S containing it are collected. Then, each nodal displacement $\mathbf{u}_i$ or velocity $\dot{\mathbf{u}}_i$ appearing in Eq. (26) is interpreted as a nodal displacement or velocity of the embedded interface Σ_E, respectively, and is computed by interpolation of the nodal displacement and velocities of Σ_S using the aforementioned projections and natural coordinates. This leads to a generalized load vector of the form given in (30) but where the result $\mathbf{f}_i^F$ has the physical meaning of the flow-induced , finite element load on vertex V_i of Σ_E and not Σ_S. Therefore, an additional step is performed to redistribute $\mathbf{f}_i^F$ on the nodes of the FE structural model associated with Σ_S. This additional step is carried out using the same virtual power principle used for deriving ALGORITHM 3, but the natural coordinates of the points of projection of Σ_E onto Σ_S.

4.3. Conservation and consistency properties

ALGORITHM 3 proposed above for computing the generalized flow-induced load vector is conservative in the sense that the total work produced by the action of the fluid on the structure at the reconstructed fluid/structure interface $\tilde{\Sigma}_E$ is equal and opposite to the total work produced by the reaction of the structure on the fluid at $\tilde{\Sigma}_E$.

Furthermore, for a constant pressure field $p = p^\star$, it follows from Eq. (30) and the partition of unity property of the shape functions N_i^q and $\tilde{N}_k^q$ that the resultant flow-induced force vector is

$$\begin{aligned}
\sum_{i=1}^{N_E} \mathbf{f}_i^F &= p^\star \sum_{i=1}^{N_E} \sum_{\substack{\tilde{\tau}_q / \exists I_k \in \tilde{\tau}_q \\ \textit{and} \\ I_k \in (\tau_q \ni V_i)}} \int_{\tilde{\tau}_q} \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) N_i^q(\xi_k^q)\, d\tau \\
&= p^\star \sum_{\tilde{\tau}_q \in \widetilde{\Sigma}_E} \int_{\tilde{\tau}_q} \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x})) \sum_{i=1}^{N_E} N_i^q(\xi_k^q)\, d\tau \\
&= p^\star \sum_{\tilde{\tau}_q \in \widetilde{\Sigma}_E} \int_{\tilde{\tau}_q} \vec{\tilde{n}}_E^q \sum_{k=1}^{3} \tilde{N}_k^q(\eta(\vec{x}))\, d\tau \\
&= p^\star \sum_{\tilde{\tau}_q \in \widetilde{\Sigma}_E} \int_{\tilde{\tau}_q} \vec{\tilde{n}}_E^q\, d\tau. \qquad (32)
\end{aligned}$$

Hence, for a constant pressure field $p = p^\star$ and a closed embedded discrete interface Σ_E,

$$\sum_{i=1}^{N_E} \mathbf{f}_i^F = p^\star \sum_{\tilde{\tau}_q \in \widetilde{\Sigma}_E} \int_{\tilde{\tau}_q} \vec{\tilde{n}}_E^q\, d\tau = 0 \qquad (33)$$

as it should be. Therefore, ALGORITHM 3 for computing the generalized flow-induced load vector is also consistent in the sense that it preserves the vanishing property of the exact integration of a constant pressure field over a closed surface.

4.4. Accuracy analysis

In principle, ALGORITHM 3 proposed in this paper for load computation in an embedded boundary method is *locally* third-order space-accurate and *globally* second-order space-accurate. In other words, it delivers a generalized flow-induced force distribution (30) that is third-order accurate in space, and a generalized flow-induced resultant (31) that is second-order accurate in space. However, because of ASSUMPTION 1 which introduces a first-order geometric error in both the computation of the numerical flux function across an embedded interface and the pressure field on this interface, ALGORITHM 3 delivers a load computation scheme that is locally second-order space-accurate and globally first-order space-accurate. Hence, improving the global accuracy of this algorithm requires developing first a higher-order extension of ALGORITHM 1 proposed in this paper for treating a fluid-structure interface in an embedded boundary method.

4.5. Practical implementation

In practice, $\widetilde{\Sigma}_E$ needs be reconstructed only locally, and ALGORITHM 2 and ALGORITHM 3 can be combined and implemented as follows for computing the generalized load vector (30).

For each element $\mathscr{D}_h^e$ in $\mathscr{D}_h$:

1. Inspect the status of the vertices of $\mathscr{D}_h^e$.

2. If all status are positive or all of them are negative, $\mathscr{D}_h^e$ does not cross Σ_E.
3. Otherwise, $\mathscr{D}_h^e$ crosses Σ_E and contributes one or more elements $\widetilde{\tau}_q$ to $\tilde{\Sigma}_E$ that are constructed as follows.
   (a) Initialize a list $\mathscr{L}^e$ of intersection points.
   (b) Loop on the edges of $\mathscr{D}_h^e$: for each edge intersecting Σ_E, identify the intersection point I_k and add it to $\mathscr{L}^e$.
   (c) Connect all points of $\mathscr{L}^e$ to form one or more elements $\tilde{\tau}_q$.
   (d) Compute the contributions of the elements $\tilde{\tau}_q$ to the generalized flow-induced load vectors $\mathbf{f}_i^F$ (30), for all $V_i \in \left(\tau_q \ni (I_k^q \in \tilde{\tau}_q)\right)$

Hence, if $\Sigma_E = \Sigma_S$, the generalized load vector (30) is computed by the flow solver and communicated to the structural analyzer. However, if $\Sigma_E \neq \Sigma_S$, the flow solver computes the equivalent (in the sense of the virtual work) nodal load on the embedded interface Σ_E and communicates them to the structural analyzer which redistributes them on Σ_S, as explained in *Remark 2* of Section 4.2.

5. A RECONSTRUCTION-FREE COMPUTATIONAL FRAMEWORK FOR LOAD COMPUTATION

The first approach for load computation on embedded interfaces presented in Section 4 is practical even when the embedding CFD grid is unstructured, or made of other elements besides hexahedra, thanks to the widely available computational geometry tools, particularly from the computational graphics community. As demonstrated in Section 6, it even delivers a good CPU performance, despite the fact that it relies on an explicit but local reconstruction of the embedded discrete interface. Nevertheless, a second approach for load computation on embedded discrete interfaces is presented here. This alternative approach bypasses any reconstruction and some of the complex computational geometry operations it entails.

5.1. Surrogate embedded interface and projection on the zero level set

As already stated in Section 2.3, the computation of a generalized flow-induced load vector or its resultant is a relatively simple post-processing task in a CFD computation on a body-fitted grid because at this stage of a flow analysis, the pressure field and all other entities constituting the integrands of the integrals to be computed during this post-processing step are readily available on the spatial domain of integration. Unfortunately, this is not the case in an embedded boundary method, because a typical embedded discrete interface does not necessarily go everywhere through grid points or cell centers of the computational domain. Hence, the idea here is to ease the computation of a generalized flow-induced load vector or its resultant in an embedded boundary method by: (1) choosing a "convenient" surrogate interface $\widehat{\Sigma}_E$ where all data forming the integrands of the integrals to be evaluated for load computation are either readily available at this stage of a flow analysis or easy to obtain, and (2) convert any integral over the real discrete interface Σ_E into an equivalent integral over its surrogate $\widehat{\Sigma}_E$ or any other representation $\widetilde{\Sigma}_E$ of $\widehat{\Sigma}_E$ where load computation is even easier. These two components of the idea developed here are discussed below, in reverse order.

Let $\vec{x} = (x_1, x_2, x_3) \in \Omega_F$ denote a point in the computational fluid domain, and let $\phi(\vec{x})$ denote a signed distance from $\vec{x}$ to the embedded discrete interface Σ_E. Hence, Σ_E is characterized by $\phi = 0$ for $\vec{x} \in \Sigma_E$, and two regions of Ω_F characterized by $\phi > 0$ and $\phi < 0$ are either occupied by two different fluid media or by a fluid and a structure. The level set function ϕ is introduced here for two purposes: to identify the edges V_iV_j that are crossed at a given time by an embedded interface, in which case $\phi(V_i) \times \phi(V_j) < 0$, and to compute the unit outward normal to Σ_E at a point $\vec{x}$,

$$\vec{n}_E(\vec{x}) = \nabla_x \phi(\vec{x}), \qquad \text{where} \qquad \|\nabla_x \phi(\vec{x})\|_2 = 1. \tag{34}$$

Let $\widehat{\Sigma}_E$ denote a surrogate of the embedded discrete interface Σ_E that is convenient in the sense defined above and further clarified later in this Section. Let also π denote the function which maps $\widehat{\Sigma}_E$ onto Σ_E — that is,

$$\pi: \quad \vec{x} = (x_1, x_2, x_3) \in \widehat{\Sigma}_E \longrightarrow \vec{y} = (y_1, y_2, y_3) = \pi(\vec{x}) \in \Sigma_E. \tag{35}$$

In an ideal setting such as that graphically depicted in case (a) of Figure 6, π is a one-to-one mapping that can be expressed as

$$\pi(\vec{x}) = \vec{x} - \phi(\vec{x}) \nabla_x \phi(\vec{x}). \tag{36}$$

Hence, π is a mapping which projects $\widehat{\Sigma}_E$ on the zero level set Σ_E.

If $\widehat{\Sigma}_E$ is parameterized by $\vec{\zeta} = (\zeta_1, \zeta_2)$, where ζ_1 and ζ_2 are, for example, some natural coordinates in a reference configuration $\widetilde{\Sigma}_E$ of a point in $\widehat{\Sigma}_E$, let θ denote the function which maps $\widetilde{\Sigma}_E$ onto $\widehat{\Sigma}_E$ — that is,

$$\theta: \quad \vec{\zeta} = (\zeta_1, \zeta_2) \in \widetilde{\Sigma}_E \longrightarrow x = (x_1, x_2, x_3) = \theta(\vec{\zeta}) \in \widehat{\Sigma}_E. \tag{37}$$

Let also

$$G = \nabla_\zeta \theta = \left[\frac{\partial \theta_i}{\partial \zeta_j} \right]_{i=1,\cdots,3; j=1,2} \qquad \text{and} \qquad F = \nabla_x \pi = \left[\frac{\partial \pi_i}{\partial x_j} \right]_{i,j=1,\cdots,3}. \tag{38}$$

The computation of the matrix G is standard in any FE code. On the other hand, the level set function can be used to compute the matrix F, assuming that ϕ is twice differentiable. Indeed,

$$\frac{\partial \pi_i}{\partial x_j}(\vec{x}) = \delta_{ij} - \frac{\partial \phi}{\partial x_j}(\vec{x}) \frac{\partial \phi}{\partial x_i}(\vec{x}) - \phi(\vec{x}) \frac{\partial^2 \phi}{\partial x_j \partial x_i}(\vec{x}), \tag{39}$$

where δ_{ij} denotes the Kronecker delta, and using Eq. (34) and the notation

$$H(\vec{x}) = \left[\frac{\partial^2 \phi}{\partial x_j x_i}(\vec{x}) \right]_{i,j=1,\cdots,3}, \tag{40}$$

F can be written as

$$F = \nabla_x \pi = I - \vec{n}_E(\vec{x}) \otimes \vec{n}_E(\vec{x}) - \phi(\vec{x}) H(\vec{x}), \tag{41}$$

where I is the identity matrix.

Consider now the integral over an embedded discrete interface Σ_E of a vector function $\vec{f}(\vec{y})$ such as that involved in the computation of a generalized flow-induced load vector or its resultant. From the mappings introduced above, it follows that

$$\int_{\Sigma_E} p(\vec{y})\, d\Sigma = \int_{\widehat{\Sigma}_E} p\big(\pi(\vec{x})\big) \sqrt{\det[F^T F]}\, d\widehat{\Sigma}, \tag{42}$$

$$\int_{\widehat{\Sigma}_E} p(\vec{x})\, d\Sigma = \int_{\widetilde{\Sigma}_E} p\left(\theta(\vec{\zeta})\right) \sqrt{\det[G^T G]}\, d\zeta_1\, d\zeta_2, \tag{43}$$

and therefore

$$\int_{\Sigma_E} p(\vec{y})\, d\Sigma = \int_{\widetilde{\Sigma}_E} p\left(\pi\left(\theta(\vec{\zeta})\right)\right) \sqrt{\det[G^T F^T F G]}\, d\zeta_1\, d\zeta_2. \tag{44}$$

The result (42) above reveals that a surrogate embedded discrete interface $\widehat{\Sigma}_E$ is convenient if: (a) $p\big(\pi(\vec{x})\big)$ is either readily available on this interface or can be obtained there by simple computational means, (b) the associated mapping π and corresponding gradient matrix F can be efficiently evaluated on this interface, and of course, (c) Σ_E is explicitly contained in the discretization $\mathscr{D}_h$. Even when such a surrogate interface is identified, it may be even simpler to perform the load computation on a reference configuration $\widetilde{\Sigma}_E$ for $\widehat{\Sigma}_E$, in which case the result (44) is used instead of its counterpart (42).

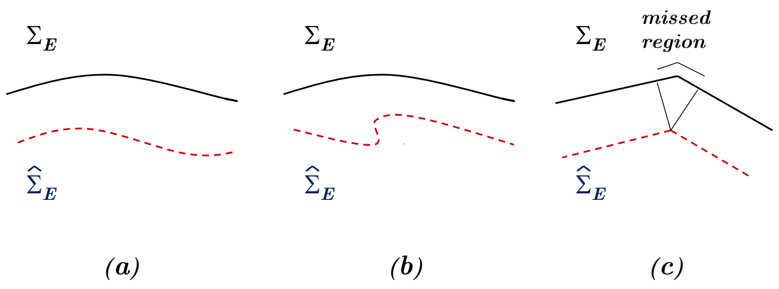

Figure 6. Three typical situations arising from the choice of a surrogate embedded interface $\widehat{\Sigma}_E$: (a) an ideal situation, (b) a situation where π is not a one-to-one mapping, and (c) a situation where the variation of the normal to $\widehat{\Sigma}_E$ is non smooth and leads to loss of accuracy.

The following academic example illustrates the application of the result (44) derived above. In this example, Σ_E is a triangle τ and $\widehat{\Sigma}_E$ is chosen as the projection of Σ_E onto a plane that does not contain τ. Hence in this case, $\widehat{\Sigma}_E$ is also a triangle $\hat{\tau}$ whose vertices are denoted here by $V_i, i = 1, \cdots, 3$, $\pi(\vec{x})$ is a one-to-one mapping and more specifically a linear function, and therefore $H(\vec{x}) = 0$ and

$$F = \tilde{F} = I - n_E \otimes n_E \Longrightarrow \det[G^T F^T F G] = \det[G^T \tilde{F}^T \tilde{F} G] = \det[G^T \tilde{F} G], \tag{45}$$

where

$$G^T \tilde{F} G = G^T (I - \vec{n}_E \otimes \vec{n}_E) G = G^T G - (G^T \vec{n}_E) \otimes (G^T \vec{n}_E). \tag{46}$$

Let $\vec{g}_1 = \overrightarrow{V_1 V_2}$ and $\vec{g}_2 = \overrightarrow{V_1 V_3}$, let $\tilde{\tau}$ denote the reference right triangle with two edges of unit length and a hypotenuse of length = $\sqrt{2}$, and let $\lambda_1(\vec{\zeta}) = 1 - \zeta_1 - \zeta_2$, $\lambda_2(\vec{\zeta}) = \zeta_1$, and $\lambda_3(\vec{\zeta}) = \zeta_2$ be the usual linear shape functions associated with a triangle such as $\tilde{\tau}$. In this case,

$$\theta(\vec{\zeta}) = \sum_{i=1}^{3} \lambda_i(\vec{\zeta}) A_i, \qquad \frac{\partial \theta}{\partial \zeta_j} = g_j,\ j = 1, 2, \qquad \text{and} \qquad G = (g_1\ g_2). \tag{47}$$

If $|\hat{\tau}|$ denotes the area of the triangle $\hat{\tau}$ and $\vec{n}_{\hat{\tau}}$ the normal to $\hat{\tau}$, then

$$|\hat{\tau}| = \frac{\|g_1 \times g_2\|}{2} \tag{48}$$

and

$$\sqrt{\det(G^T \tilde{F}^T \tilde{F} G)} \quad = \quad 2|\hat{\tau}||\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|. \tag{49}$$

From Eq. (44), it follows that for this academic example,

$$\int_\tau \vec{f}(\vec{y})\, d\Sigma = 2|\hat{\tau}| \int_{\hat{\tau}} \vec{f}\Big(\pi\big(\theta(\zeta)\big)\Big) |\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|\, d\zeta_1\, d\zeta_2. \tag{50}$$

In particular for $\vec{f}(y) = 1$, the identity (50) above gives

$$\int_\tau d\Sigma = 2|\hat{\tau}| \int_{\hat{\tau}} |\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|\, d\zeta_1\, d\zeta_2, \tag{51}$$

which can also be written as

$$|\tau| = |\hat{\tau}||\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|, \qquad \text{or equivalently,} \qquad |\hat{\tau}| = \frac{|\tau|}{|\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|}, \tag{52}$$

which is the well-known formula for the exact evaluation of the area of the projection of a triangle onto a plane. Hence, the academic example presented above highlights the potential of the results (42)–(44) for load computation on a surrogate discrete interface.

However, given a discretization $\mathscr{D}_h$ of the fluid domain of interest, finding a surrogate interface $\widehat{\Sigma}_E$ that is convenient in the sense defined above and for which a one-to-one mapping π exists is not always possible. For example, Figure 6, case (b), graphically depicts a situation where a convenient surrogate embedded discrete interface does not lead to a projector π that is a one-to-one mapping. Furthermore, the computation of the gradient matrix F (41) can require computational and implementational efforts that are comparable to those of the reconstruction of Σ_E discussed in Section 4. For these reasons, the practical exploitation of the results (42)–(44) calls for introducing some approximations in their computation. These are suggested by the spatial and temporal characteristics of the underlying CFD method, among others.

In the context of this work, the FV method chosen in Section 2.2 for illustrating the ideas presented in this paper, and the flux-based method proposed in Section 3 for enforcing the velocity condition (7) on an embedded discrete interface and recovering there the value of the fluid pressure, suggest choosing as a surrogate fluid/structure interface

$$\widehat{\Sigma}_E = \bigcup_{(l,m)/\phi_l \phi_m < 0} \partial C_{lm}, \tag{53}$$

where $\phi_l = \phi(\vec{V}_l)$, $\phi_m = \phi(\vec{V}_m)$, and $V_l V_m$ is the edge associated with (or traversing) the boundary facet ∂C_{lm} (see Figure 7). Indeed, this interface is practical as it is made of control volume boundary facets ∂C_{lm} that are explicitly built in a FV code operating on dual control volumes. After ALGORITHM 1 has been used for enforcing the velocity condition (7) on $\widehat{\Sigma}_E$, the value of the fluid pressure p_k at the intersection of each facet C_{lm} and its associated edge $V_l V_m$ is automatically obtained using ASSUMPTION 1. Hence, p is known in this case on both $\widehat{\Sigma}_E$ and Σ_E. Therefore, it is chosen here to approximate $p\big(\pi(\vec{x})\big)$ by $p(\vec{x})$ — that is, to approximate $\pi(\vec{x})$ by $\vec{x}$ and therefore F by I. This gives

$$\int_{\Sigma_E} p(\vec{y})\, d\Sigma \approx \int_{\widehat{\Sigma}_E} p(\vec{x})\, d\widehat{\Sigma}. \tag{54}$$

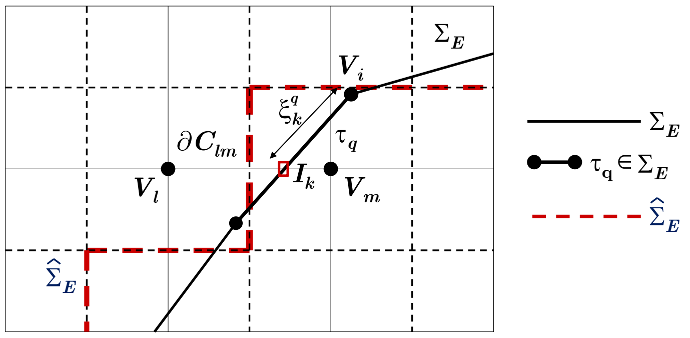

Figure 7. Surrogate embedded discrete interface $\widehat{\Sigma}_E$ in the context of a finite volume method with dual control volumes (two-dimensional case, quadrilateral mesh).

5.2. *Conservative and consistent algorithm for load computation*

Following the same conservative approach based on virtual work adopted in ALGORITHM 3 for computing the generalized flow-induced load vector, choosing $\widehat{\Sigma}_E$ given in (53) as a surrogate fluid/structure interface, choosing also to approximate $\pi(\vec{x})$ by $\vec{x}$ and therefore F by I to obtain the approximation (54), and recalling the partition of unity property of the shape functions N_i^q leads to the following conservative algorithm for computing the distribution and resultant of the generalized flow-induced load.

<u>ALGORITHM 4</u>

$$\mathbf{f}_i^F = \sum_{\tau_q \ni V_i} \sum_{\substack{(l,m)/ \\ (\phi_l \phi_m < 0 \ \& \ I_k \in \tau_q)}} \int_{\partial C_{lm}} p_k \vec{n}_{lm} N_i^q(\xi_k^q)\, d\tau, \tag{55}$$

and

$$\begin{aligned} \mathbf{f}^F &= \sum_{i=1}^{N_E} \mathbf{f}_i^F = \sum_{\tau_q \in \Sigma_E} \sum_{\substack{(l,m)/ \\ (\phi_l \phi_m < 0 \ \& \ I_k \in \tau_q)}} \int_{\partial C_{lm}} p_k \vec{n}_{lm} \sum_{V_i \in \tau_q} N_i^q(\xi_k^q)\, d\tau \\ &= \sum_{(l,m)/\phi_l \phi_m < 0} p_k \int_{\partial C_{lm}} \vec{n}_{lm}\, d\tau. \end{aligned} \tag{56}$$

From Eq. (56) above and the partition of unity property of the shape functions N_i^q, it follows that for a constant pressure $p = p^\star$,

$$\sum_{i=1}^{N_E} \mathbf{f}_i^F = p^\star \sum_{(l,m)/\phi_l \phi_m < 0} \int_{\partial C_{lm}} \vec{n}_{lm}\, d\tau. \tag{57}$$

From the definition (53), it follows that if Σ_E is closed, then $\widehat{\Sigma}_E$ is also closed. Hence, for a constant pressure and a closed embedded surface, the resultant (56) is equal to zero. This proves that ALGORITHM 4 for computing the generalized flow-induced load vector is also consistent in the sense of preserving the vanishing property of the exact integration of a constant pressure field over a closed surface.

5.3. Accuracy analysis

The following observations are noteworthy:

- Except when π happens to be the identity mapping, the approximation $p\big(\pi(\vec{x})\big) \approx p(\vec{x})$ is a first-order spatial approximation of the pressure on the embedded interface $\widehat{\Sigma}_E$ (or equivalently on $\widetilde{\Sigma}_E$).
- The resulting approximation (54) is locally second-order space-accurate, and therefore as accurate, if not more accurate, than the numerical solution delivered at an embedded interface by a typical embedded boundary method.
- Attempting to "improve" the spatial accuracy of the approximation (54) by incorporating in it a correction factor of the form $|\vec{n}_{\hat{\tau}} \cdot \vec{n}_E|$ in order to account for the effect of the projection of $\widehat{\Sigma}_E$ onto Σ_E would be counterproductive. To begin, as shown in Eq. (49) pertaining to the academic example of Section 5.1, such a correction factor can be traced to the exact expression (36) of the mapping π and therefore is inconsistent with the approximation $\pi(\vec{x}) \approx \vec{x}$. Furthermore, as shown in Figure 6, case (c), when $\widehat{\Sigma}_E$ is not sufficiently smooth as in the case of the surrogate interface (53) illustrated in Figure 7, projecting $\widehat{\Sigma}_E$ onto Σ_E results in missing some regions of Σ_E during the integration process. Not only this degrades spatial accuracy, but it also leads to a load computation scheme that does not preserve the vanishing property of the exact integration of a constant pressure field over a closed surface (see Section 5.2).

It follows that in general, ALGORITHM 4 proposed in this paper for load computation in an embedded boundary method is locally second-order space-accurate, and globally first-order space-accurate. In other words, it delivers a generalized flow-induced force distribution (55) that is second-order accurate in space, and a generalized flow-induced resultant (56) that is first-order accurate in space. In the particular case where π is the identity mapping — that is, when the proposed surrogate interface is identical to the real interface — ALGORITHM 4 becomes locally third-order space-accurate and globally second-order space-accurate.

6. APPLICATIONS AND PERFORMANCE ASSESSMENTS

The numerical algorithms presented in this paper for interface treatment and load computation in embedded boundary methods were implemented in the flow solver AERO-F [9, 28] which also incorporates an ALE computational framework. In this Section, they are illustrated by their application to two dynamic fluid-structure interaction problems in the fields of aeronautics and underwater implosion. The first problem is characterized by a structure whose thinness challenges the robustness of ray-based intersection algorithms. The second problem is characterized by shock waves and large structural deformations. For both problems, reference AERO-F ALE solutions are also computed for the purpose of verification. This requires however modifying slightly the second problem so that it can be solved by an ALE-based CFD method. The computed reference solutions are reliable as the ALE computational framework of AERO-F has been successfully verified and validated for many applications in aeronautics [9, 28] and underwater implosion [26, 5].

AERO-F's embedded boundary method is based on the same second-order spatial discretization of AERO-F's ALE method. In the case of AERO-F's embedded boundary method, the semi-discrete equations governing the coupled fluid-structure interaction problem are time-integrated using a variant of the second-order time-accurate explicit-explicit staggered solution procedure developed in [5]. However in the case of AERO-F's ALE method, the governing semi-discrete equations are time-integrated using the second-order implicit (fluid) - explicit (structure) coupling scheme also developed in [5]. In addition, the numerical algorithms proposed in this paper are applied to the solution of a two-dimensional manufactured problem in order to verify the associated theoretical results presented in this paper.

Whereas most embedded boundary methods are designed to operate on structured parallelepiped meshes or right rectangular prismatic grids, the majority of the computations reported here are performed on unstructured tetrahedral meshes. This is essentially for two reasons: (1) AERO-F is an unstructured grid flow solver, and (2) this demonstrates the versatility of the proposed algorithms.

It is also noted that all computations reported in this Section are performed in double-precision arithmetic on a massively parallel Linux Cluster system.

6.1. Verification of the accuracy analysis for idealized and realistic mesh configurations

In Section 4.4, it was concluded that in the ideal situation where ASSUMPTION 1 is satisfied, ALGORITHM 3 for load computation equipped with ALGORITHM 2 for surface reconstruction is locally third-order space-accurate and globally second-order space-accurate; in the realistic situation where ASSUMPTION 1 is not satisfied, ALGORITHM 3 is locally second-order space-accurate and globally first-order space-accurate. On the other hand, it was concluded in Section 5.3 that ALGORITHM 4 is in principle second-order space-accurate locally and first-order space-accurate globally; however, it becomes locally third-order space-accurate and globally second-order space-accurate when the surrogate interface (53) becomes identical to real interface. In this section, these theoretical accuracy results are numerically verified for the two-dimensional manufactured problem described below.

Let $\Omega = [-2,2] \times [-2,2]$ denote a square domain where the pressure field is assumed to vary as follows

$$p(x,y) = \sin(x+y), \tag{58}$$

where x and y are measured in an orthonormal frame whose origin is at the center of Ω. Assume that a square ribbon — playing here the role of a closed interface — with a side-length $l = 2$ is embedded in Ω so that the center of the area it covers coincides with the center of Ω. The total pressure-induced force on this interface is given by the integral of $p(x,y)$ on the perimeter of this interface and therefore is equal to

$$\mathbf{f}^F = \begin{pmatrix} 2(1-\cos 2) \\ 2(1-\cos 2) \end{pmatrix}. \tag{59}$$

To verify the global convergence properties of ALGORITHM 3 (equipped with ALGORITHM 2) and ALGORITHM 4 applied to the approximation of the above total force, two different sets of discretizations of Ω are constructed. The first set, S1, consists of a series of increasingly refined uniform quadrilateral meshes constructed so that ASSUMPTION 1 is always satisfied and the surrogate interface (53) is identical to the real interface (the ideal case) — that is, the perimeter of the embedded interface always coincides everywhere with control volume facets of the dual mesh. For example, Figure 8 shows the coarsest mesh in the set S1, together with the reconstructed interface according to ALGORITHM 2, and the surrogate interface according to (53). The second set of meshes, S2, is a set of arbitrary triangular meshes for which ASSUMPTION 1 is not satisfied and the surrogate interface (53) is different from the real one (the realistic case). The coarsest of these meshes is shown in Figure 9 together with the reconstructed and surrogate surfaces according to ALGORITHM 2 and definition (53), respectively.

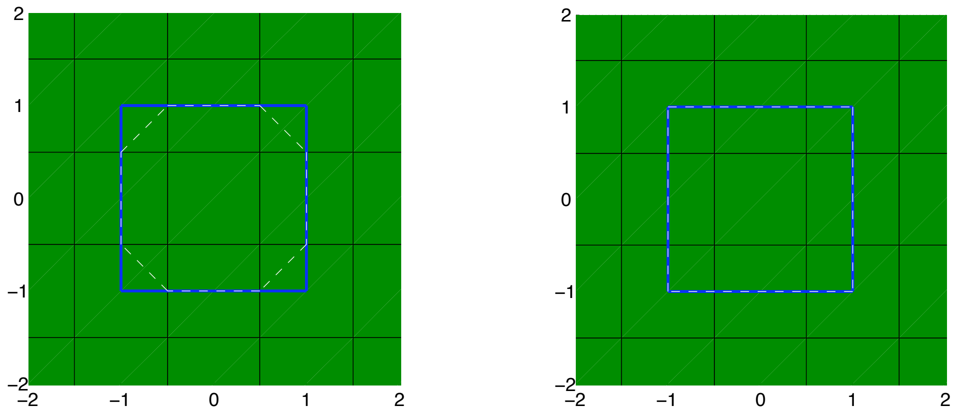

Figure 8. Ideal case: coarsest mesh in set S1 (uniform, quadrilateral, satisfying ASSUMPTION 1) and reconstructed (left) and surrogate (right) interfaces.

Figure 10 and Figure 11 report in log-log format the variations with the element mesh size of the absolute value of the relative error between the approximate and exact evaluations of the total force $\mathbf{f}^F$ for the ideal and realistic cases, respectively.

For the ideal case, the results shown in Figure 10 reveal that:

- ALGORITHM 3 (equipped with ALGORITHM 2 for interface reconstruction) and ALGORITHM 4 deliver the expected second-order accuracy.
- ALGORITHM 4 delivers a significantly better accuracy than ALGORITHM 3. This is because in this ideal case, the proposed surrogate interface (53) coincides everywhere with the real interface (see Figure 8).

For the realistic case, the variations of the relative error shown in Figure 11 are irregular because of the irregular nature of the considered set of increasingly refined meshes S2. Nevertheless, they reveal that:

- ALGORITHM 3 (equipped with ALGORITHM 2 for interface reconstruction) delivers on average

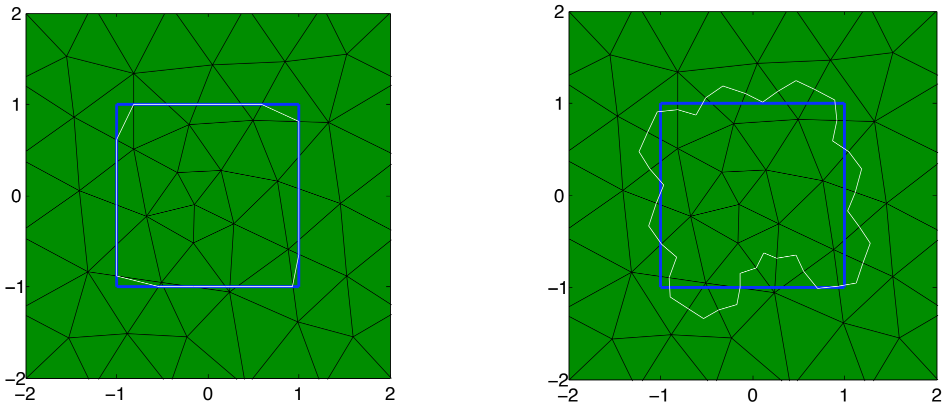

Figure 9. Realistic case: coarsest mesh in set S2 (arbitrary, triangular, not satisfying ASSUMPTION 1) and reconstructed (left) and surrogate (right) interfaces.

a convergence rate equal to 1.98, which is significantly higher than the theoretically determined rate of 1.

- ALGORITHM 4 delivers on average 1.33, which is slightly higher than the theoretically determined rate of 1.
- ALGORITHM 3 delivers a significantly better accuracy than ALGORITHM 4. This is because in the realistic case, the reconstruction of the interface performed by ALGORITHM 2 can be expected to be in general a more accurate representation of this interface than the surrogate (53) (see Figure 9). However, it should be emphasized that even in this case, both ALGORITHM 3 and ALGORITHM 4 deliver an excellent accuracy given a reasonable mesh resolution.

In summary, all numerical results obtained for the academic problem described above verify the theoretical results presented in this paper, except in one case where a higher-order of accuracy is obtained numerically.

6.2. Verification for a transient subsonic flow past a heaving rigid wing

Here, the problem of computing the unsteady airflow past a rigid wing in heaving motion is considered. The wing has a root chordlength $L_c = 22.0$ in, a semi-span $L_s = 30.0$ in, a tip chordlength $L_t = 14.5$ in, and a quarter-chord sweep angle of 45^{deg}. Its panel aspect ratio is equal to 1.65 and its taper ratio is equal to 0.66. Its airfoil section is the NACA 65A004.

The wing is set in a harmonic heaving motion characterized by the amplitude $h_a = 0.05$ in and the frequency $h_f = 500$ Hz. The free-stream conditions (Mach number, angle of attack, density, and pressure) are set to $M_\infty = 0.3$, $\alpha_\infty = 0^{\text{deg}}$, $\rho_\infty = 9.357255 \times 10^{-8}$ (lb/in^4).s^2, and $p_\infty = 14.5$ psi, respectively.

In the coordinate system whose origin is at the leading edge of the root section of the wing, and x and y directions are along its chord and span, respectively, the chosen computational fluid domain can

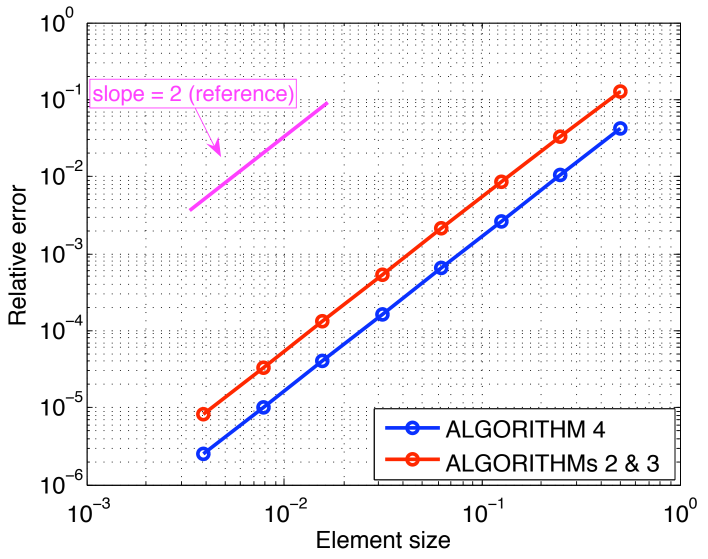

Figure 10. Ideal case: performance of ALGORITHM 3 and ALGORITHM 4 for load computation.

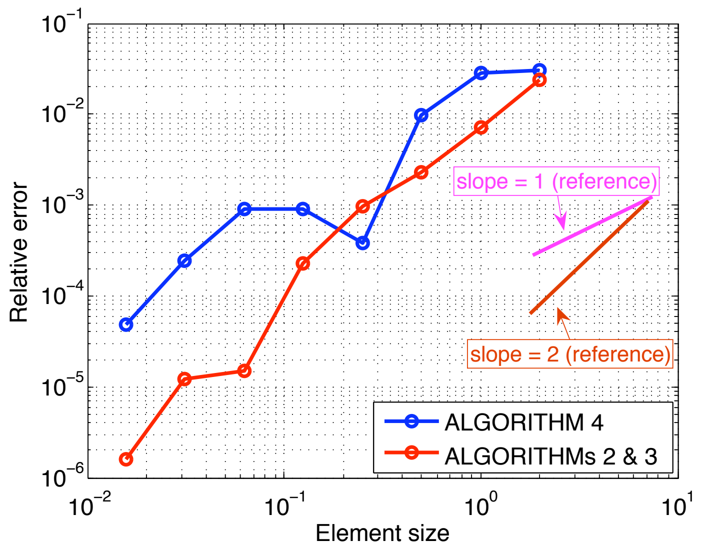

Figure 11. Realistic case: performance of ALGORITHM 3 and ALGORITHM 4 for load computation.

be described as the rectangular box extending from $x = -500.0$ in to $x = 500.0$ in, $z = -500.0$ in to $z = 500.0$ in, and $y = 0.0$ in to $y = 500$ in.

Two CFD grids are generated: (1) a body-fitted grid G1 with 546,712 tetrahedra and 101,623 grid points for the computation of an ALE reference solution for this problem (Figure 12), and (2) a non body-fitted grid G2 with 609,576 tetrahedra and 105,030 grid points for computing an Eulerian solution for this problem using an embedded boundary method for CFD (Figure 13). Two surface meshes of the wing are also generated for the purpose of being embedded in grid G2. The first one (E1) contains $20,721$ grid points and $41,438$ triangles (Figure 14). The second one (E2) is much coarser. It contains only 48 grid points and 86 triangles (Figure 15).

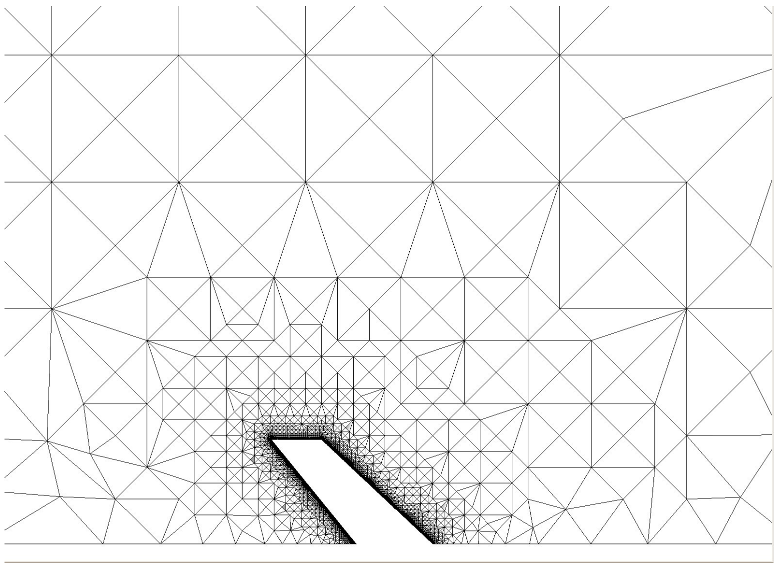
Figure 12. G1: a body-fitted fluid grid for the ALE simulation of the unsteady flow past a rigid wing in harmonic heaving motion (cutview at $z = 0$).

Symmetry boundary conditions are applied in the plane $y = 0$ containing the root of the wing. Non-reflecting boundary conditions are applied at the remaining boundaries of the external computational fluid domain.

Five numerical simulations are performed:

1. Using the CFD grid $G2$, the embedded discrete surface $E1$, and AERO-F's embedded boundary method equipped with ALGORITHM 2 for interface reconstruction and ALGORITHM 3 for load computation.
2. Using the CFD grid $G2$, the embedded discrete surface $E1$, and AERO-F's embedded boundary method equipped with ALGORITHM 4 for load computation.
3. Using the CFD grid $G2$, the embedded discrete surface $E2$, and AERO-F's embedded boundary method equipped with ALGORITHM 2 for interface reconstruction and ALGORITHM 3 for load computation.
4. Using the CFD grid $G2$, the embedded discrete surface $E2$, and AERO-F's embedded boundary method equipped with ALGORITHM 4 for load computation.

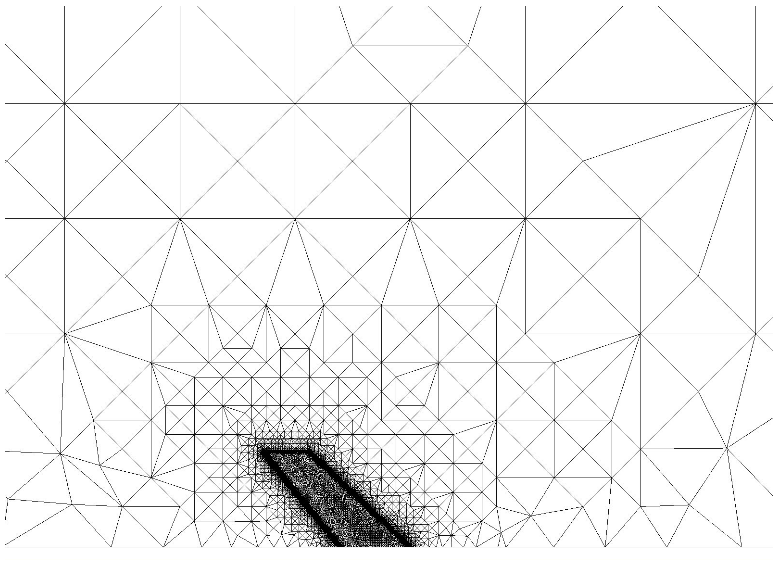

Figure 13. G2: a non body-fitted fluid grid for the Eulerian simulation using an embedded boundary method of the unsteady flow past a rigid wing in harmonic heaving motion (cutview at $z = 0$).

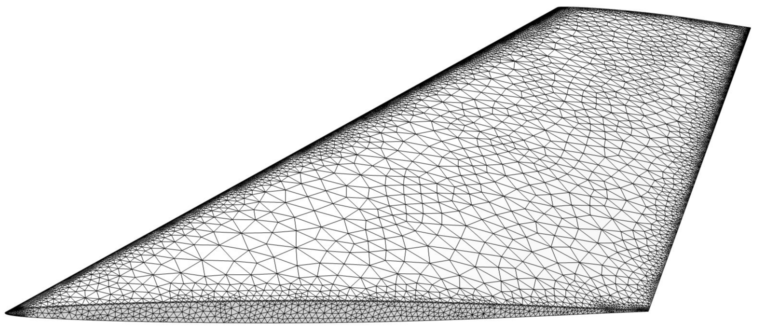

Figure 14. E1: Embedded discrete surface of the wing with 20,721 grid points and 41,438 triangles.

5. Using the CFD grid $G1$ and AERO-F's ALE computational framework.

All five simulations are initialized with a uniform flow corresponding to the free-stream conditions specified above.

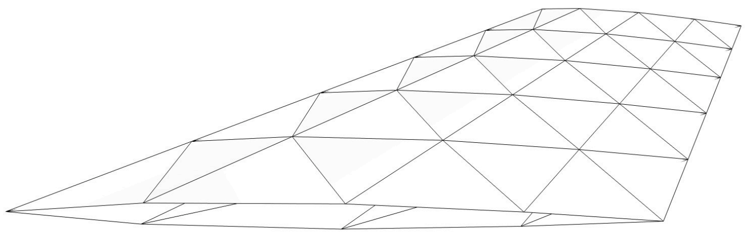

Figure 15. E2: Embedded discrete surface of the wing with 48 grid points and 86 triangles.

It is noted here that the ALE computational framework is ideal for solving the rather simple, but frequently encountered, unsteady CFD problem considered here. Indeed, because the wing is rigid, updating in time the motion of grid G1 is trivial and computationally inexpensive. In comparison, an embedded boundary method is computationally inefficient for this class of problems because of the repeated grid intersections it entails. This disadventage is particularly accentuated when the mesh of the embedded surface is fine. On the other hand, the problem considered here is an excellent verification problem as the thinness of the wing can challenge the robustness of a computational "intersector" — more specifically, its aptitude for finding two intersection points, one on the upper surface of the wing and one on its lower surface, for the typical cast ray. Hence, the main purpose of this application problem is to verify the numerical algorithms for interface treatment and load computation proposed in this paper.

The obtained time-histories of the lift are reported in Figure 16 (simulations 1, 2, and 5) and Figure 17 (simulations 3, 4, and 5) for the first five periods of oscillation (0 s $\leq t \leq 0.01$ s). The reader can observe that all five simulations predict almost the same lift time-histories. Hence, at least for this problem, ALGORITHM 3 and ALGORITHM 4 for load computation deliver, as expected, the same accuracy.

Table I reports the CPU timings on 32 cores associated with all five simulations outlined above. The following observations and comments are noteworthy:

(1) There are two reasons why the CPU time elapsed in flux computations is higher (by about 18%) for the embedded boundary method than for the ALE method: (a) grid G2 has a few more elements and grid points than G1 does, and these are located inside the wing, and (b) this CPU time includes that associated with the solutions of the fluid-structure Riemann problems performed by ALGORITHM 1 during the treatment of the interface conditions.
(2) As anticipated, for this problem, the ALE mesh motion update is virtually cost-free for the reason explained earlier in this Section.
(3) For simulations 1–4 performed using the embedded boundary method, the CPU time consumed by the computation of grid intersections is shown to be significantly affected by the mesh resolution of the embedded discrete surface. This is because the complexity of the intersection

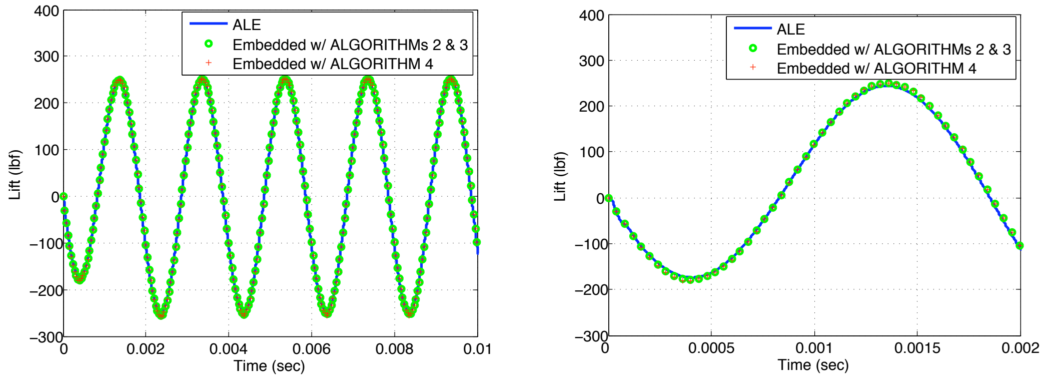

Figure 16. Time-history of the lift generated by the heaving rigid wing as predicted by simulations 1, 2, and 5.

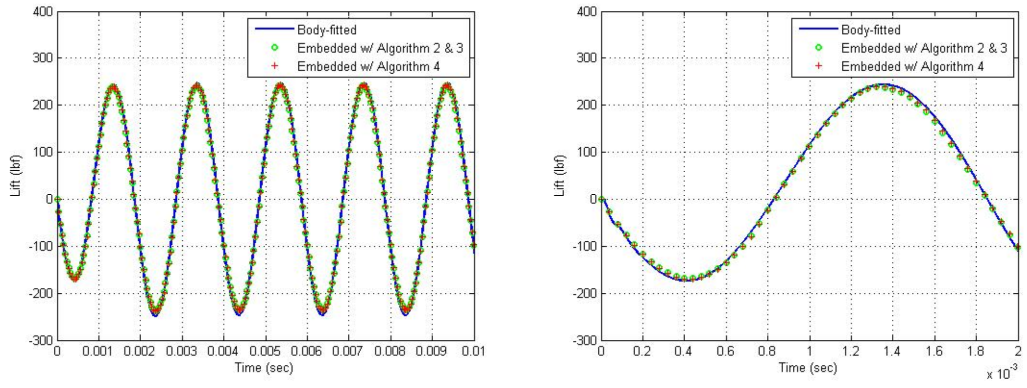

Figure 17. Time-history of the lift generated by the heaving rigid wing as predicted by simulations 3, 4, and 5.

algorithm grows as $O(N \log n)$, where N and n denote the number of edges in grid G2 and number of triangles of the embedded surface mesh that are candidate for intersection, respectively. Given that the computational complexity of an explicit flow solver typically varies as $O(N)$, the computational overhead induced by intersections is bound to represent a serious percentage of the total CPU time.

(4) The combination of ALGORITHM 2 and ALGORITHM 3 for load computation is only slightly more computationally expensive than the alternative ALGORITHM 4, despite the fact that ALGORITHM 2 reconstructs the embedded surface.

(5) For this problem, Figure 17 shows that using the coarse embedded discrete surface E2 delivers essentially the same accuracy as its finer counterpart E1, and Table I shows that this reduces the total CPU time by a factor greater than four. In particular, when using E2, the CPU overhead associated with computing intersections is only about 15% of the total CPU.

Table I. CPU performance on 32 cores of the simulation of the unsteady flow past a heaving rigid wing during one period of oscillations.

	Simulation 1	Simulation 2	Simulation 3	Simulation 4	Simulation 5
Flow computations	1,558 s	1,558 s	1,558 s	1,556 s	1,779 s
Finite volume fluxes	1,064 s	1,063 s	1,075 s	1,073 s	870 s
Mesh metrics update	0 s	0 s	0 s	0 s	472 s
Others	494 s	495 s	483 s	483 s	437 s
Mesh motion update	0 s	0 s	0 s	0 s	7 s
Intersection computations	5,940 s	5,917 s	320 s	318 s	0 s
Load computations	468 s	216 s	372 s	129 s	0 s
Total simulation time	9,258 s	8,987 s	2,343 s	2,101 s	1,895 s

6.3. *Verification for the dynamic collapse of a cylindrical shell submerged in water*

Finally, an underwater implosion problem is considered here as a verification test case for fluid-structure interaction problems characterized by shock waves and large structural deformations. In this problem which has been the subject of an experiment at the University of Texas at Austin, a short aluminum cylinder of length $L = 3$ in, circular cross-section with external diameter $D = 1.4995$ in, and thickness $h_S = 0.0277$ in is submerged in a rigid water tank. It is closed at both ends with two rigid caps and filled with air at the atmospheric pressure $p_i = 14.5$ psi (see Figure 18). The cylinder is maintained at the center of the tank by a set of rigid bars attached to the rigid tank. It is surrounded by pressure sensors that are positioned at approximately the same radial distance $d = 2.5$ in to the center of the cylinder.

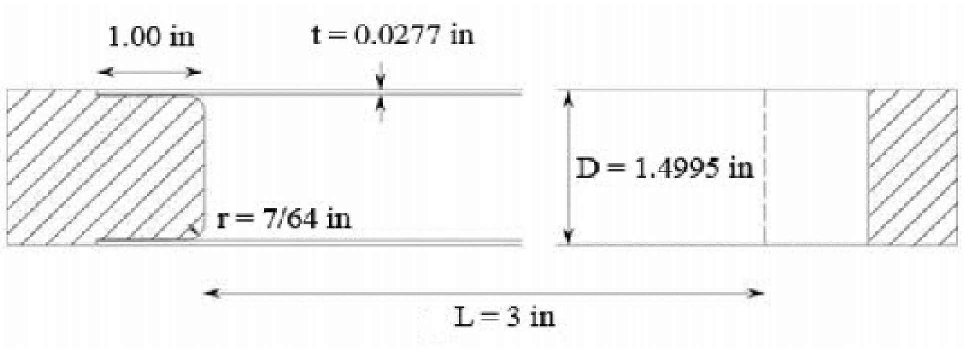

Figure 18. Submerged cylindrical implodable with end caps designated by stripes (courtesy of Stelios Kyriakides).

Initially, the water is at rest ($v_o = 0$ in/s). Its pressure is $p_o = 14.5$ psi and its density is $\rho_o^0 = 9.357255 \times 10^{-5}$ (lb/in^4).s^2. At $t = 0$, p_o is increased to $p_o^0 = 729.5$ psi, at which point the cylinder collapses and cracks open at its end caps, but not along its main body (see Figure 19).

To simulate the above experiment numerically, only half of the cylinder (lengthwise) is modeled.

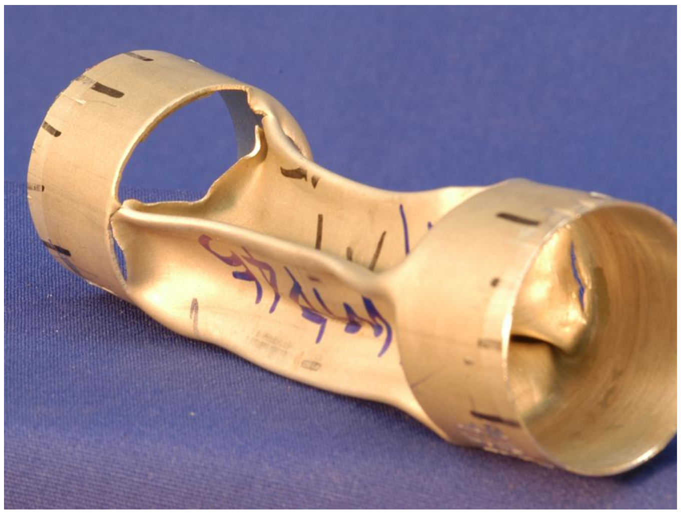

Figure 19. Collapsed cylinder with cracks at end caps (courtesy of Stelios Kyriakides).

Its aluminum material is represented as a nonlinear elasto-plastic medium with a Young modulus $E = 1.008 \times 10^7$ psi, a Poisson ratio $\nu = 0.3$, a density $\rho_S = 2.599 \times 10^{-4}$ (lb/in^4).s^2, a yield stress equal to 4.008×10^4 psi, and a hardening modulus equal to 9.2×10^4 psi. The cylinder itself is discretized by a finite element model with 756 four-noded shell elements.

Because of the ultrahigh compressions involved in this problem, water is modeled as a stiffened gas whose equation of state can be written as

$$(\gamma - 1)\rho e = p + \gamma\pi,$$

where e denotes the internal energy per unit mass, and γ and π are two constants that are set here to $\gamma = 4.4$ and $\pi = 8.7 \times 10^4$ psi.

The external computational fluid domain is chosen as the rectangular box extending from 0 in to 16.44 in in the x direction originating at the center of one end cap and coinciding with the axis of the cylinder, and from -20 in to 20 in in both the y and z directions. The (half) cylinder model is placed at the center of this computational domain, in the y-z plane. It extends from $x = 0$ in to $x = 2.5$ in.

In principle, the internal computational fluid domain delimited by the wall boundary of the cylinder and the symmetry plane should be discretized even for the ALE computation in order to capture the pressure variations during the collapse of the structure. However the large deformations of the structure would severely squeeze an interior ALE fluid mesh to the point where no mesh motion solver is able to handle it, particularly near collapse time. This issue emphasizes why an embedded boundary method is in general more suitable than an ALE one for the solution of a fluid-structure problem characterized by large deformations, contact, topology changes, and other geometrical complications. Hence, in order to be able to generate an ALE reference solution for this problem, the pressure inside the internal computational domain is fixed here at 14.5 psi. Without this assumption or some similar simplification, an ALE method would typically fail to complete the simulation of the experiment considered herein, whereas an embedded boundary method equipped with the interface treatment and load computation algorithms proposed in this paper would successfully complete it.

Two CFD grids are generated: (1) a body-fitted grid G3 with 12,027,679 tetrahedra and 2,034,067 grid points suitable for computing an ALE solution for this problem (Figure 20), and (2) a simpler, non body-fitted grid G4 with 19,642,615 tetrahedra and 3,294,197 grid points suitable for computing an Eulerian solution for this problem using an embedded boundary method (Figure 21). A surface mesh of the cylindrical implodable with 1,596 triangles and 841 grid points is also generated and embedded in grid G4.

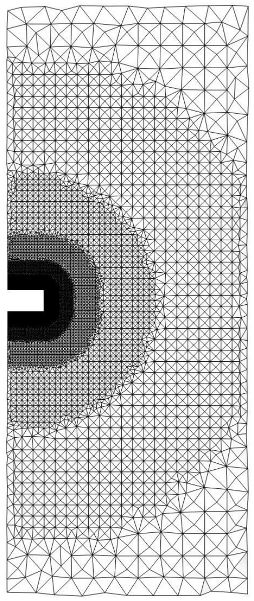

Figure 20. G3: a body-fitted fluid grid for the ALE solution of an underwater implosion problem (cutview at $z = 0$).

Symmetry boundary conditions are applied to both the fluid and structural models at the transversal plane passing through the middle cross section of the cylinder. Zero displacement and velocity boundary conditions are applied at its end caps. Non-reflecting boundary conditions are applied at the remaining boundaries of the external computational fluid domain. To trigger the collapse of the cylindrical implodable, a geometric imperfection is introduced in its structural model. More specifically, the circular cross-section of the cylinder is replaced by an ellipse described by

$$r = r_0\big(1 - 0.02\cos 4\theta\big),$$

where (r, θ) are the polar coordinates of a point on the ellipse and r_0 is the radius of the true circular cross-section.

Two numerical simulations are performed: one using AERO-F's embedded boundary method equipped with ALGORITHM 2 for interface reconstruction and ALGORITHM 3 for load computation, and one using AERO-F's ALE computational framework. The initial state of the surrounding water is uniformly

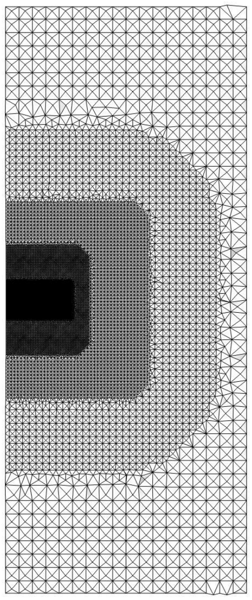

Figure 21. G4: a non body-fitted fluid grid for the Eulerian solution by an embedded boundary method of an underwater implosion problem (cutview at $z = 0$).

set to $v_o^0 = 0$ in/s, $\rho_o^0 = 9.357255 \times 10^{-5}$ (lb/in^4).s^2, and $p_o^0 = 700.0$ psi. Then, p_o^0 is statically increased from its atmospheric value to the collapse value of 729.5 psi. Time-integration of the coupled fluid-structure system is started only when the collapse pressure is reached — that is, at $t = 4.25 \times 10^{-5}$ s. It is carried out until $T = 1.0 \ \times 10^{-3}$ s.

Figure 22 shows the shapes at $T = 1.0 \ \times 10^{-3}$ s of the collapsed cylinder predicted by AERO-F's embedded boundary and ALE methods. The reader can observe that both shapes are nearly identical. They are also similar to the experimental collapsed shape shown in Figure 19. Figure 23 reports the pressure signals predicted by both aforementioned CFD methods at the sensor located on the symmetry plane of the computational domain, 2.5 in away from the central axis along the longitudinal direction of the cylinder. Again, the reader can observe that AERO-F's embedded boundary and ALE methods predict essentially the same results. The pressure drops after approximately 2.6×10^{-4} s by roughly 127 psi, then rises by about 180 psi, before the cylinder collapses and gets into self-contact around $t = 5.3 \times 10^{-4}$ s. At this point, a very sharp pressure rise to 997.7 psi can be observed. This second rise is followed by a broader and higher pressure peak of $1,064$ psi, before the pressure drops in an oscillatory manner and begins to fluctuate around the initial hydrostatic pressure.

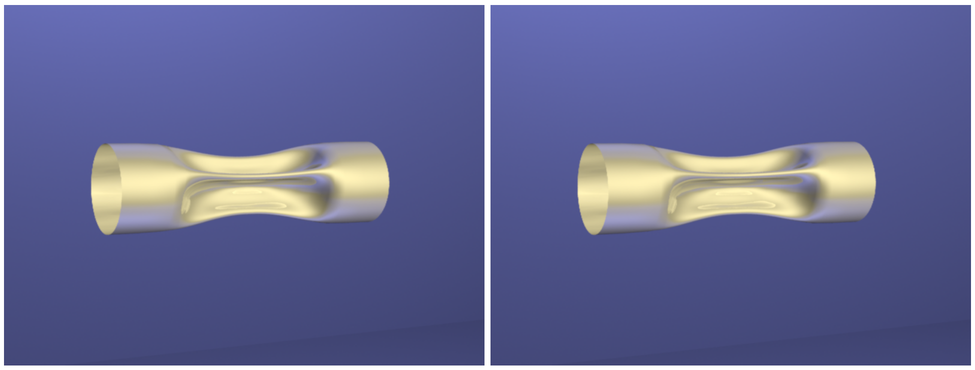

Figure 22. Deformed shapes at $T = 1.0 \times 10^{-3}$ s predicted by AERO-F's (1) embedded boundary method equipped with ALGORITHM 2 for interface reconstruction and ALGORITHM 3 for load computation (left), and (2) ALE method (right).

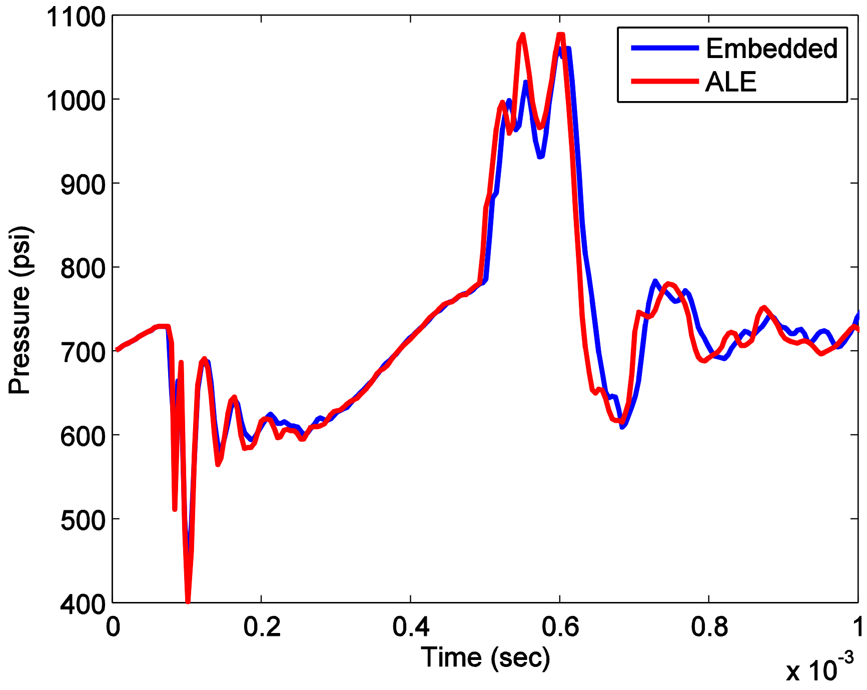

Figure 23. Pressure signals at a sensor location predicted by AERO-F's (1) embedded boundary method equipped with ALGORITHM 2 for interface reconstruction and ALGORITHM 3 for load computation, and (2) ALE method.

7. CONCLUSIONS

Focusing on the context of compressible flows and embedded boundary methods for Computational Fluid Dynamics (CFD), this paper contributes numerical algorithms for treating fluid-wall and fluid/structure interfaces and computing flow-induced loads on such interfaces. More specifically, it proposes an approach for treating simultaneously the fluid pressure and velocity conditions on static and dynamic embedded interfaces that is based on the exact solution of local, one-dimensional, fluid-structure Riemann problems. It also presents two consistent and conservative approaches for computing the flow-induced forces and moments on rigid and flexible embedded structures. The first one reconstructs locally the fluid/structure interfaces. The second one bypasses this reconstruction and chooses instead to work with surrogate fluid/structure interfaces. All of these numerical algorithms are equally applicable to structured and unstructured embedding meshes. The results obtained for their application to the solution of realistic three-dimensional fluid-structure interaction problems associated with the fields of aeronautics and underwater implosion demonstrate their accuracy, robustness, and potential for solving complex fluid-structure interaction problems characterized by shock waves, large structural deformations, and topology changes.

ACKNOWLEDGEMENTS

The authors acknowledge partial support by the Office of Naval Research under Grant N00014-06-1-0505 and Grant N00014-09-C-015, and partial support by the Army Research Laboratory through the Army High Performance Computing Research Center (AHPCRC) under Cooperative Agreement W911NF-07-2-0027. The content of this publication does not necessarily reflect the position of policy of neither the Office of Naval Research nor the U.S. Army and Government, and no official endorsement should be inferred.

APPENDIX A: EXACT SOLUTION OF THE ONE-SIDED, ONE-DIMENSIONAL, FLUID-STRUCTURE RIEMANN PROBLEM

Using the same notation as in Section 3, except for dropping the superscripts L, R and $\sim$ to simplify notation, the one-sided, one-dimensional, fluid-structure Riemann problem (17) is re-written here as

$$\left\{ \begin{array}{rcl} \dfrac{\partial W}{\partial t} + \dfrac{\partial \vec{\mathscr{F}}}{\partial s}(W) & = & 0 \\ W(s,0) & = & W_i, \text{ if } s \geq 0 \\ v(v_0 t, t) & = & v_0, \; \forall\, 0 \leq t \leq \Delta t, \end{array} \right. \tag{60}$$

where W denotes the state of the fluid located on one side of the given obstacle $\big($the left side in Figure (4)$\big)$, W_i denotes the initial state of that fluid and is assumed to be uniform, s is the abscissa defined in Section 3 and graphically depicted in Figure 4, $v(0,t)$ is the instantaneous velocity of the fluid at $s = 0$, and

$$v_0 = \dot{\mathbf{u}}_M \cdot \vec{n}_{E_M} \tag{61}$$

is assumed to be constant in the time-interval $[0, \Delta t]$.

The solution of the above Riemann problem is given by two constant states separated by a nonlinear wave that can be either a shock wave or a rarefaction fan. One constant state, W_M, is at the fluid/structure interface and is characterized by a fluid velocity equal to v_0. The other constant state is on the other side of the nonlinear wave and is equal to the initial uniform state W_i. These two constant states are related by the Rankine-Hugoniot jump conditions when the nonlinear wave that separates them is a shock wave, and by the isentropic laws for rarefaction when the nonlinear wave is a rarefaction. Hence, the above problem can be reduced to an explicit expression of the velocity at the fluid/structure interface, v_M, as a function of the pressure at this material interface, p_M, of the form

$$v_M \quad = \quad v_i + \mathscr{R}(p_M; p_i, \rho_i), \tag{62}$$

where $\mathscr{R}$ is a vector function that depends on the structure of the wave solution, and the semi-column ";" separates the unknown variable from the known quantities.

When the nonlinear wave is a shock wave, $\mathscr{R}$ is given by

$$\mathscr{R}(p_M; p_i, \rho_i) = \left(\sqrt{\frac{a_i}{p_M + b_i}}\right)(p_M - p_i), \tag{63}$$

where

$$a_i = \frac{2}{(\gamma+1)\rho_i} \qquad \text{and} \qquad b_i = \left(\frac{\gamma-1}{\gamma+1}\right)p_i. \tag{64}$$

From the non-penetration condition — that is, the continuity of the normal component of the velocity field at the fluid/structure interface — and the Rankine-Hugoniot conditions, it follows that

$$\begin{aligned} v_M &= v_0 = \dot{\mathbf{u}}_M \cdot \vec{n}_{E_M} \\ p_M &= p_i + \frac{(\gamma+1)\rho_i(v_0 - v_i)^2}{4}\left(1 + \sqrt{1 + \left(\frac{16}{(\gamma+1)^2}\right)\left(\frac{\gamma p_i}{\rho_i}\right)\left(\frac{1}{(v_0 - v_i)^2}\right)}\right) \\ \rho_M &= \rho_i \left(\frac{p_M}{p_i + \left(\frac{\gamma-1}{\gamma+1}\right)}\right)\left(\frac{1}{1 + \left(\frac{\gamma-1}{\gamma+1}\right)\left(\frac{p_M}{p_i}\right)}\right), \end{aligned}$$

which completes in this case the exact solution of problem (60).

On the other hand in the presence of a rarefaction fan, $\mathscr{R}$ is given by

$$\mathscr{R}(p_M; p_i, \rho_i) = \left(\frac{2c_i}{\gamma - 1}\right)\left(\left(\frac{p_M}{p_i}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right), \tag{65}$$

where c denotes the speed of sound. Finally, from the non-penetration condition and the isentropic relations, if follows that

$$\begin{aligned} v_M &= v_0 \\ p_M &= p_i \left(1 + \left(\frac{\gamma-1}{2}\right)\left(\frac{v_0 - v_i}{c_i}\right)\right)^{\frac{2\gamma}{\gamma-1}} \\ \rho_M &= \rho_i \left(\frac{p_M}{p_i}\right)^{\frac{1}{\gamma}}, \end{aligned}$$

which completes the exact solution of problem (60).

REFERENCES

1. Peskin CS. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics* 1972; **10**:252-271.
2. Kreiss HO, Petersson A. A second-order accurate embedded boundary method for the wave equation with Dirichlet data. *SIAM Journal of Scientific Computation* 2006; **27**:1141–1167.
3. Glowinski R, Pan TW, Kearsley AJ, Periaux J. Numerical simulation and optimal shape for viscous flow by a fictitious domain method. *International Journal for Numerical Methods in Fluids* 2005; **20**:695–711.
4. Johansen H, Colella P. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics* 1998; **147**:60–85.
5. Farhat C, Rallu A, Wang K, Belytschko T. Robust and provably second-order explicit-explicit and implicit-explicit staggered time-integrators for highly nonlinear fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering*, *(in press)*
6. Farhat C, Maute K, Argrow B, Nikbay M. A shape optimization methodology for reducing the sonic boom initial pressure rise. *AIAA Journal of Aircraft* 2007; **45**:1007–1018.
7. Farhat C, Geuzaine P, Grandmont C. The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids. *Journal of Computational Physics* 2001; **174**: 669–694.
8. Farhat C, Lesoinne M, Maman N. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids* 1995; **21**:807–835.
9. Farhat C, Geuzaine P, Brown G. Application of a three-field nonlinear fluidstructure formulation to the prediction of the aeroelastic parameters of an F-16 fighter. *Computers & Fluids* 2003; **32**:3–29.
10. Mittal R, Iaccarino G. Immersed bounday methods. *Annual Review of Fluid Mechanics* 2005; **37**:239–261.
11. Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J. Combined-immersed boundary finite difference methods for three-dimensional complex flow simulations. *Journal of Computational Physics* 2000; **161**:35–60.
12. Kim J, Kim D, Choi H. An immersed-boundary finite volume method for simulations of flow in complex geometries. *Journal of Computational Physics* 2001; **171**:132–150.
13. Glimanov A, Sotiropoulos F. A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. *Journal of Computational Physics* 2005; **207**:457–492.
14. Choi JI, Oberoi RC, Edwards JR, Rosati JA. An immersed boundary method for complex incompressible flows. *Journal of Computational Physics* 2007; **224**:757–784.
15. Tseng YH, Ferziger JH. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics* 2003; **192**:593–623.
16. Berthelsen PA, Faltinsen OM. A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *Journal of Computational Physics* 2008; **227**:4354–4397.
17. Lohner R, JBaum JD, Mestreau EL, Sharov D, Charman C, Pelessone D. Adaptive embedded unstructured grid methods. *AIAA-03-1116*, 2003.
18. Mark A, van Wachem BGM. Derivation and validation of a novel implicit second-order accurate immersed boundary method. *Journal of Computational Physics* 2008; **227**:6660–6680.
19. Fedkiw R. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *Journal of Computational Physics* 2002; **175**:200–224.
20. Roe PL. Approximate Riemann solvers, parameters vectors and difference schemes. *Journal of Computational Physics* 1981; **43**:357–371.
21. Steger J, Warming RF. Flux vector splitting for the inviscid gas dynamic with applications to finite-difference methods. *Journal of Computational Physics* 1981; **40**:263–293.
22. Van Leer B. Towards the ultimate conservative difference scheme V: a second-order sequel to Goudonov's method. *Journal of Computational Physics* 1979; **32**:361–370.
23. Harder RL, Desmarais RN. Interpolation using surface splines. *Journal of Aircraft* 1972; **9**:189–191.
24. Cebral JR, Lohner R. Conservative load projection and tracking for fluid-structure problems. *AIAA Journal* 1997; **35**:687–692.
25. Farhat C, Lesoinne M, LeTallec P. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer Methods in Applied Mechanics* 1998; **157**:95–114.
26. Farhat C, Rallu A, Shankaran S. A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions. *Journal of Computational Physics* 2008; **227**:7674–7700.
27. Guendelman E, Selle A, Losasso F, Fedkiw R. Coupling water and smoke to thin deformable and rigid shells. *SIGGRAPH* 2005; ACM TOG **24**:973-981.
28. Geuzaine P, Brown G, Harris C, Farhat C. Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions. *AIAA Journal* 2003; **41**:363–371.