Proceedings of
2001 ASME International Mechanical Engineering Congress and Exposition
November 11-16, 2001, New York, NY

IMECE2001/HTD-24206

FINITE ELEMENT INTERFACE-TRACKING AND INTERFACE-CAPTURING TECHNIQUES FOR FLOWS WITH MOVING BOUNDARIES AND INTERFACES

Tayfun Tezduyar[†]
Team for Advanced Flow Simulation and Modeling (T★AFSM)[‡]
Mechanical Engineering, Rice University
6100 Main Street, Houston, TX 77005, USA

ABSTRACT

We provide an overview of the interface-tracking and interface-capturing techniques we have developed in recent years for computation of flow problems with moving boundaries and interfaces, including two-fluid interfaces. The interface-tracking techniques are based on the Deforming-Spatial-Domain/Stabilized Space-Time formulation, where the mesh moves to track the interface. The interface-capturing techniques, which were developed for two-fluid flows, are based on the stabilized formulation, over non-moving meshes, of both the flow equations and the advection equation governing the time-evolution of an interface function marking the location of the interface. For interface-capturing techniques, to increase the accuracy in representing the interface, the Enhanced-Discretization Interface-Capturing Technique (EDICT) can be used to to accomplish that goal. We also provide and overview of some of the additional ideas developed to increase the scope and accuracy of these two classes of techniques.

INTRODUCTION

In computation of flow problems with moving boundaries and interfaces, including two-fluid interfaces, depending on the complexity of the interface and other aspects of the problem, we can use an interface-tracking or interface-capturing technique. An interface-tracking technique requires meshes that "track" the interfaces. The mesh needs to be updated as the flow evolves. In an interface-capturing technique for two-fluid flows, the computations are based on fixed spatial domains, where an interface function, marking the location of the interface, needs to be computed to "capture" the interface. The interface is captured within the resolution of the finite element mesh covering the area where the interface is. This approach can be seen as a special case of interface representation techniques, where the interface is somehow represented over a non-moving fluid mesh, the main point being that the fluid mesh does not move to "track" the interfaces. A consequence of the mesh not moving to "track" the interface is that for fluid-solid interfaces, independent of how well the interface geometry is represented, the resolution of the boundary layer will be limited by the resolution of the fluid mesh where the interface is.

The Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation,[1] developed for moving boundaries and interfaces, is an interface-tracking technique, where the finite element formulation of the problem is written over its space-time domain. This automatically takes into account the motion of the interfaces. At each time step the locations of the interfaces are calculated as part of the overall solution. As the spatial domain occupied by the fluid changes its shape in time, mesh needs to be updated. In general, this is accomplished by moving the mesh with the motion of the nodes governed by the equations of elasticity, and full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) as needed.

In computation of two fluid-flows (we mean this category to include free-surface flows) with interface-tracking techniques, sometimes the interface might be too complex or unsteady to track while keeping the frequency of remeshing at an acceptable level. Not being able to reduce the frequency of remeshing in 3D might introduce overwhelming mesh generation and projection costs, making the computations with the interface-tracking technique no longer feasible. In such cases, interface-capturing techniques, which do not normally require costly mesh update steps, could be used with the understanding that

[†]James F. Barbour Professor, Fellow
[‡]http://www.mems.rice.edu/TAFSM/

the interface will not be represented as accurately as we would have with an interface-tracking technique. Because they do not require mesh update, the interface-capturing techniques are more flexible than the interface-tracking techniques. However, for comparable levels of spatial discretization, interface-capturing methods yield less accurate representation of the interface. These methods can be used as practical alternatives in carrying out the simulations when compromising the accurate representation of the interfaces becomes less of a concern than facing major difficulties in updating the mesh to track such interfaces. The desire to increase the accuracy of our interface-capturing techniques without adding a major computational cost lead us to seeking techniques with a different kind of "tracking". The Enhanced-Discretization Interface-Capturing Technique (EDICT) was first introduced in[2] to increase accuracy in representing an interface. We will describe the EDICT more in a later section. In later sections, we will also describe some of the additional ideas and methods developed to increase the scope and accuracy of the interface-tracking and interface-capturing techniques.

GOVERNING EQUATIONS

Let $\Omega_t \subset I\!R^{n_{sd}}$ be the spatial fluid mechanics domain with boundary Γ_t at time $t \in (0,T)$, where the subscript t indicates the time-dependence of the spatial domain and its boundary. The Navier-Stokes equations of incompressible flows can be written on Ω_t and $\forall t \in (0,T)$ as

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{f}) - \nabla\cdot\boldsymbol{\sigma} = 0, \quad (1)$$

$$\nabla\cdot\mathbf{u} = 0, \quad (2)$$

where ρ, $\mathbf{u}$ and $\mathbf{f}$ are the density, velocity and the external force, respectively. The stress tensor $\boldsymbol{\sigma}$ is defined as

$$\boldsymbol{\sigma}(p,\mathbf{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\mathbf{u}). \quad (3)$$

Here p, $\mathbf{I}$ and μ are the pressure, identity tensor and the viscosity, respectively. The strain rate tensor $\boldsymbol{\varepsilon}(\mathbf{u})$ is defined as

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}((\nabla\mathbf{u}) + (\nabla\mathbf{u})^T). \quad (4)$$

Both Dirichlet- and Neumann-type boundary conditions are accounted for:

$$\mathbf{u} = \mathbf{g} \text{ on } (\Gamma_t)_g, \qquad \mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h} \text{ on } (\Gamma_t)_h, \quad (5)$$

where $(\Gamma_t)_g$ and $(\Gamma_t)_h$ are complementary subsets of the boundary Γ_t, $\mathbf{n}$ is the unit normal vector, and $\mathbf{g}$ and $\mathbf{h}$ are given functions. A divergence-free velocity field is specified as the initial condition.

If the problem does not involve any moving boundaries or interfaces, the spatial domain does not need to change with respect to time, and the subscript t can be dropped from Ω_t and Γ_t. This might be the case even for flows with moving boundaries and interfaces, if in the formulation used the spatial domain is not defined to be the part of the space occupied by the fluid(s). For example, we can have a fixed spatial domain, and model the fluid-fluid interfaces by assuming that the domain is occupied by two immiscible fluids, A and B, with densities ρ_A and ρ_B and viscosities μ_A and μ_B. When we model liquid-gas interactions, we let Fluid A be the liquid and Fluid B the gas. In modeling a free-surface problem where Fluid B is irrelevant, we assign a sufficiently low density to Fluid B. An interface function ϕ serves as a marker identifying Fluid A and B with the definition $\phi = \{1$ for Fluid A and 0 for Fluid B$\}$. The interface between the two fluids is approximated to be at $\phi = 0.5$. In this context, ρ and μ are defined as

$$\rho = \phi\rho_A + (1-\phi)\rho_B, \quad (6)$$

$$\mu = \phi\mu_A + (1-\phi)\mu_B. \quad (7)$$

The evolution of the interface function ϕ, and therefore the motion of the interface, is governed by a time-dependent advection equation:

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0 \qquad \text{on } \Omega \quad \forall t \in (0,T). \quad (8)$$

DEFORMING-SPATIAL-DOMAIN STABILIZED SPACE-TIME (DSD/SST) FINITE ELEMENT FORMULATION

In the DSD/SST method, the finite element formulation of the governing equations is written over a sequence of N space-time slabs Q_n, where Q_n is the slice of the space-time domain between the time levels t_n and t_{n+1} (see Figure 1). At each time step, the integrations involved in the finite element formulation are performed over Q_n. The space-time finite element interpolation functions are continuous within a space-time slab, but discontinuous from one space-time slab to another. Typically we use first-order polynomials as interpolation functions. The notation $(\cdot)_n^-$ and $(\cdot)_n^+$ denotes the function values at t_n as approached from below and above respectively. Each Q_n is decomposed into space-time elements Q_n^e, where $e = 1,2,\ldots,(n_{el})_n$. The subscript n used with n_{el} is to account for the general case

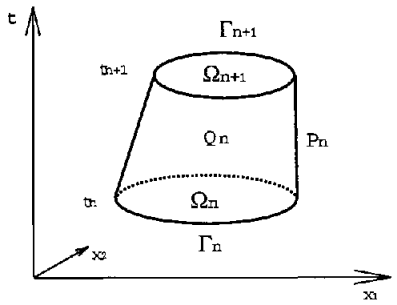

Figure 1. Space-time concept.

in which the number of space-time elements may change from one space-time slab to another. The Dirichlet- and Neumann-type boundary conditions are enforced over $(P_n)_g$ and $(P_n)_h$, the complementary subsets of the lateral boundary of the space-time slab. The finite element trial function spaces $(\mathcal{S}^h_{\mathbf{u}})_n$ for velocity and $(\mathcal{S}^h_p)_n$ for pressure, and the test function spaces $(\mathcal{V}^h_{\mathbf{u}})_n$ and $(\mathcal{V}^h_p)_n$ are defined as

$$(\mathcal{S}^h_{\mathbf{u}})_n = \{\mathbf{u}^h | \mathbf{u}^h \in [H^{1h}(Q_n)]^{n_{sd}}, \mathbf{u}^h \doteq \mathbf{g}^h \text{ on } (P_n)_g\}, \quad (9)$$

$$(\mathcal{V}^h_{\mathbf{u}})_n = \{\mathbf{w}^h | \mathbf{w}^h \in [H^{1h}(Q_n)]^{n_{sd}}, \mathbf{w}^h \doteq \mathbf{0} \text{ on } (P_n)_g\}, \quad (10)$$

$$(\mathcal{S}^h_p)_n = (\mathcal{V}^h_p)_n = \{q^h | q^h \in H^{1h}(Q_n)\}. \quad (11)$$

Here $H^{1h}(Q_n)$ is the finite-dimensional function space over Q_n. Over the space-time element domain, this function space is formed by using first-order polynomials in both space and time.

The DSD/SST formulation is written as follows: given $(\mathbf{u}^h)^-_n$, find $\mathbf{u}^h \in (\mathcal{S}^h_{\mathbf{u}})_n$ and $p^h \in (\mathcal{S}^h_p)_n$ such that $\forall \mathbf{w}^h \in (\mathcal{V}^h_{\mathbf{u}})_n$ and $q^h \in (\mathcal{V}^h_p)_n$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) dQ \\
&+ \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
&- \int_{(P_n)_h} \mathbf{w}^h \cdot \mathbf{h}^h dP \\
&+ \int_{Q_n} q^h \nabla \cdot \mathbf{u}^h dQ \\
&+ \int_{\Omega_n} (\mathbf{w}^h)^+_n \cdot \rho \left((\mathbf{u}^h)^+_n - (\mathbf{u}^h)^-_n \right) d\Omega \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \frac{\tau_{\text{LSME}}}{\rho} \mathbf{L}(q^h, \mathbf{w}^h) \cdot \\
&\qquad \left[\mathbf{L}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] dQ \\
&+ \sum_{e=1}^{n_{el}} \int_{Q^e_n} \tau_{\text{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h dQ \\
&= 0, \quad (12)
\end{aligned}$$

where

$$\mathbf{L}(q^h, \mathbf{w}^h) = \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h), \quad (13)$$

and τ_{LSME} and τ_{LSIC} are the stabilization parameters (see[3,4]). For an earlier, detailed reference on this stabilized formulation see.[1]

This formulation is sequentially applied to all space-time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$. The computation starts with

$$(\mathbf{u}^h)^-_0 = \mathbf{u}_0, \quad \nabla \cdot \mathbf{u}_0 = 0 \quad \text{on } \Omega_0. \quad (14)$$

MESH UPDATE METHODS

In interface-tracking techniques, as the computations proceed, the mesh needs to be updated to accommodate the changes in the spatial domain. How the mesh should be updated depends on several factors, such as the complexity of the interface and overall geometry, how unsteady the interface is, and how the starting mesh was generated. In general, the mesh update could have two components: moving the mesh for as long as it is possible, and full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) when the element distortion becomes too high.

Most real-world problems require simulations with complex geometries. A complex geometry typically requires an automatic mesh generator. We developed an automatic mesh generator to have a number of special features, such as structured layers of elements around solid objects and high-speed mesh generation. This automatic, 3D mesh generator is described in.[5] It has been used very effectively in a number of simulations (for early examples see[6]). It has the capability to build structured layers of elements around solid objects with reasonable geometric complexity. With this, we can fully control the mesh resolution near solid objects and have more accurate representation of the boundary layers.

Automatic mesh generation might become an overwhelming cost especially when the number of elements becomes very large or when frequency of remeshing has to be high. Sometimes special-purpose mesh generators designed for specific problems can

be used. Depending on the complexity of the problem, such mesh generators might involve a high initial design cost, but minimal mesh generation cost. We selected this path in a number of our simulations, and were able to overcome the mesh generation issues very effectively (see for example[7]).

In mesh moving strategies, the only rule the mesh motion needs to follow is that at the interface the normal velocity of the mesh has to match the normal velocity of the fluid. Beyond that, the mesh can be moved in any way desired, with the main objective being to reduce the frequency of remeshing. In 3D simulations, if the remeshing requires calling an automatic mesh generator, the cost of automatic mesh generation becomes a major reason for trying to reduce the frequency of remeshing. Furthermore, when we remesh, we need to project the solution from the old mesh to the new one. This introduces projection errors. Also, in 3D, the computing time consumed by this projection step is not a trivial one. All these factors constitute a strong motivation for designing mesh update strategies which minimize the frequency of remeshing.

In some cases where the changes in the shape of the computational domain allow it, a special-purpose mesh moving method can be used in conjunction with a special-purpose mesh generator. In such cases, simulations can be carried out without calling an automatic mesh generator and without solving any additional equations to determine the motion of the mesh. One of the earliest examples of that, 2D computation of sloshing in a laterally vibrating container, can be found in.[1] Extension of that concept to 3D parallel computation of sloshing in a vertically vibrating container can be found in.[8]

In general, however, we use an automatic mesh moving scheme[9] to move the nodal points, as governed by the equations of linear elasticity. The motion of the internal nodes is determined by solving these additional equations, with the boundary conditions for these mesh motion equations specified in such a way that they match the normal velocity of the fluid at the interface. Similar mesh moving techniques were used earlier by other researchers (see for example[10]). Mesh moving issues were also addressed in[11] by using a 2D pseudo-structural model based on springs, and more recently in[12] by using improved versions of the same model.

In our mesh moving method based on linear elasticity, the structured layers of elements generated around solid objects (mentioned above) move "glued" to these solid objects, undergoing a rigid-body motion. No equations are solved for the motion of the nodes in these layers, because these nodal motions are not governed by the equations of elasticity. This results in some cost reduction. But more importantly, the user has full control of the mesh resolution in these layers. For early examples of automatic mesh moving combined with structured layers of elements undergoing rigid-body motion with solid objects, see.[8] Earlier examples of element layers undergoing rigid-body motion, in combination with deforming structured meshes, can be found in.[1]

In computation of flow problems with fluid-solid interfaces where the solid is deforming, the motion of the fluid mesh near the interface cannot be represented by a simple rigid-body motion. Depending on the deformation mode of the solid, we may have to use the automaic mesh moving technique described earlier in this section. In such cases, presence of very thin fluid elements near the solid surface creates a challenge for the automatic mesh moving technique. In Solid-Extension Mesh Moving Technique (SEMMT), we propose to treat those very thin fluid elements almost like an extension of the solid elements. In the SEMMT, in solving the equations of elasticity governing the motion of the fluid nodes, we assign a much higher rigidity to these thin elements, compared to the other fluid elements. This could be implemented in two ways; we can solve the elasticity equations for the nodes connected to the thin elements separate from the elasticity equations for the other nodes, or together. If we solve them separately, for the thin elements, as boundary conditions at the interface with the other elements, we would use traction-free boundary conditions.

ENHANCED-DISCRETIZATION INTERFACE-CAPTURING TECHNIQUE

In EDICT, we start with the basic approach of an interface-capturing technique such as the volume of fluid (VOF) method.[13] The Navier-Stokes equations are solved over a non-moving mesh together with the time-dependent advection equation governing the evolution of the interface function ϕ. In writing the stabilized finite element formulation for the EDICT (see[14]), the notation we use here for representing the finite-dimensional function spaces is very similar to the notation we used in the section where we described the DSD/SST formulation. The trial function spaces corresponding to velocity, pressure and interface function are denoted, respectively, by $(\mathcal{S}^h_{\mathbf{u}})_n$, $(\mathcal{S}^h_p)_n$, and $(\mathcal{S}^h_\phi)_n$. The weighting function spaces corresponding to the momen-

tum equation, incompressibility constraint and time-dependent advection equation are denoted by $(\mathcal{V}_{\mathbf{u}}^h)_n$, $(\mathcal{V}_p^h)_n$ $(=(\mathcal{S}_p^h)_n)$, and $(\mathcal{V}_\phi^h)_n$. The subscript n in this case allows us to use different spatial discretizations corresponding to different time levels.

The stabilized formulations of the flow and advection equations can be written as follows: given $\mathbf{u}_n^h$ and ϕ_n^h, find $\mathbf{u}_{n+1}^h \in (\mathcal{S}_{\mathbf{u}}^h)_{n+1}$, $p_{n+1}^h \in (\mathcal{S}_p^h)_{n+1}$, and $\phi_{n+1}^h \in (\mathcal{S}_\phi^h)_{n+1}$, such that, $\forall \mathbf{w}_{n+1}^h \in (\mathcal{V}_{\mathbf{u}}^h)_{n+1}$, $\forall q_{n+1}^h \in (\mathcal{V}_p^h)_{n+1}$, and $\forall \psi_{n+1}^h \in (\mathcal{V}_\phi^h)_{n+1}$:

$$
\begin{aligned}
&\int_\Omega \mathbf{w}_{n+1}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) d\Omega \\
+ &\int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}_{n+1}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega \\
- &\int_{\Gamma_h} \mathbf{w}_{n+1}^h \cdot \mathbf{h}^h d\Gamma \\
+ &\int_\Omega q_{n+1}^h \nabla \cdot \mathbf{u}^h d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} [\tau_{\text{SUPG}} \mathbf{u}^h \cdot \nabla \mathbf{w}_{n+1}^h + \frac{\tau_{\text{PSPG}}}{\rho} \nabla q_{n+1}^h] \cdot \\
&\qquad \left[\mathbf{L}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\text{LSIC}} \nabla \cdot \mathbf{w}_{n+1}^h \rho \nabla \cdot \mathbf{u}^h d\Omega \\
= &\ 0, \qquad (15)
\end{aligned}
$$

$$
\begin{aligned}
&\int_\Omega \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_\phi \mathbf{u}^h \cdot \nabla \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) d\Omega \\
= &\ 0. \qquad (16)
\end{aligned}
$$

In this formulation, τ_{SUPG}, τ_{PSPG} and τ_ϕ are the stabilization parameters.[1,3]

To increase the accuracy, we use function spaces corresponding to enhanced discretization at and near the interface. A subset of the elements in the base mesh, Mesh-1, are identified as those at and near the interface. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. The interpolation functions for velocity and pressure will all have two components each: one coming from Mesh-1 and the second one coming from Mesh-2. To further increase the accuracy, we construct a third-level mesh, Mesh-3, for the interface function only. The construction of Mesh-3 from Mesh-2 is very similar to the construction of Mesh-2 from Mesh-1. The interpolation functions for the interface function will have three components, each coming from one of these three meshes. We re-define the subsets over which we build Mesh-2 and Mesh-3 not every time step but with sufficient frequency to keep the interface enveloped in. We need avoid this envelope being too wide or too narrow.

EXTENSIONS OF EDICT TO OTHER CLASSES OF APPLICATIONS

Extension of EDICT to other classes of problems was first reported in[15] for computation of compressible flows with shocks. This extension is based on re-defining the "interface" to mean the shock front. In this approach, at and near the shock fronts, we use enhanced discretization to increase the accuracy in representing those shocks.

Later, the EDICT was extended to computation of vortex flows. The results were first reported in.[16,17] In this case, the definition of the interface is extended to mean regions where the vorticity magnitude is larger than a specified value. The finite element functions consist of two components. The first component comes from a base mesh (Mesh-1), generated over the entire domain. A second-level and more refined mesh (Mesh-2) is constructed over a subset of the elements in Mesh-1. Mesh-2 is generated by simply subdividing the elements in that subset. Which elements in Mesh-1 should belong to this subset depends on the magnitude of the vorticity in those elements and varies from time-step to time-step. This version of the EDICT gives us the capability to compute the long-wake flows more accurately, without making the computations too costly.

Here we propose to extend EDICT to computation of flow problems with boundary layers. In this extension, the "interface" means solid surfaces with boundary layers. In 3D problems with complex geometries and boundary layers, mesh generation poses a serious challenge. This is because accurate resolution of the boundary layer requires elements that are very thin in the direction normal to the solid surface. This needs to be accomplished without having a major increase in mesh refinement also in the tangential directions or creating very distorted elements. Otherwise, we might be increasing the computational cost excessively or decreasing the numerical accuracy unacceptably. In the Enhanced-Discretization Mesh Refinement Technique (EDMRT), we propose two different ways of using the EDICT concept to increase the mesh refinement in the bound-

ary layers in a desirable fashion.

In the EDICT-Clustered-Mesh-2 approach, Mesh-2 is constructed by patching together clusters of second-level meshes generated over each element of Mesh-1 designated to be one of the "boundary layer elements". Depending on the type of these boundary layer elements in Mesh-1, Mesh-2 could be structured or unstructured, with hexahedral, tetrahedral or triangle-based prismatic elements. In the EDICT-Layered-Mesh-2 approach, a thin but multi-layered and more refined Mesh-2 is "laid over" the solid surfaces. Depending on the geometric complexity of the solid surfaces and depending on whether we prefer the same type elements as those we used in Mesh-1, the elements in mesh-2 could be hexahedral, tetrahedral or triangle-based prismatic elements.

The EDMRT, as an EDICT-based boundary layer mesh refinement strategy, would allow us accomplish our objective without facing the implementational difficulties associated with elements having variable number of nodes.

In the Enhanced-Discretization Space-Time Technique (EDSTT), we propose to use enhanced time-discretization in the context of a space-time formulation. The motivation behind this is to have a flexible way of carrying out time-accurate computations of fluid-structure interactions where we find it necessary to use smaller time steps for the structural dynamics part of the problem. There would be two ways of formulating EDSTT. In the EDSTT-Single-Mesh (EDSTT-SM) approach, a single space-time mesh, unstructured both in space and time, would be used to enhance the time-discretization in regions of the fluid domain near the structure. This, in general, might require a fully unstructured 4D mesh generation. In the EDSTT-Multi-Mesh (EDSTT-MM) approach, multiple space-time meshes, all structured in time, would be used to enhance the time-discretization in regions of the fluid domain near the structure. In a way, this would be the space-time version of the EDMRT. This approach would not require a fully unstructured 4D mesh generation, and therefore would not pose a mesh generation difficulty. In general, EDSTT can be used in time-accurate computations where we require smaller time steps in some parts of the fluid domain (for example, where the spatial element sizes are small or where there is a fluid-fluid interface).

Whether we are using a space-time formulation or a semi-discrete formulation, at every time step of the computation, we need to solve a coupled, nonlinear equation system, and we use the Newton-Raphson method for this purpose. Sometimes, some parts of the computational domain may offer more of a challenge for the Newton-Raphson method than the others. This might happen, for example, at the fluid-solid interface in a fluid-structure interaction problem, and in such cases the nonlinear convergence might become even a bigger challenge if the structure is going through some sort of buckling or wrinkling. It might also happen at a fluid-fluid interface, for example, if the interface is very unsteady. In the Enhanced-Iteration Nonlinear Solution Technique (EINST), as a variation of the Newton-Raphson method, we propose to use sub-iterations in the parts of the domain where we are facing a nonlinear convergence challenge. This could be implemented, for example, by identifying the nodes of the zones where we need enhanced iterations, and performing multiple iterations for those nodes for each iteration we perform for all other nodes.

A coupled, linear equation system needs to be solved at every step of the Newton-Raphson sequence. In the class of computations we typically carry out, this equation system would be too large to solve with a direct method. Therefore we solve it iteratively. In these iterations, we use a preconditioning matrix, which is essentially an approximation to the original matrix of the coupled, linear equation system. Because of its simplicity and parallel computation efficiency, in most cases we use a diagonal matrix as the approximation. In some challenging cases, this simple approach might not lead to a satisfactory level of convergence at some locations, in the parts of the domain posing the challenge. This might happen, for example, in a fluid-structure interaction problem, where the structure or the fluid zones near the structure might be suffering from convergence problems, the situation might become worse if the structure is going through buckling or wrinkling. It might also happen at a fluid-fluid interface. We might also face this difficulty in the SEMMT described in the section on mesh update methods, if the elasticity equations for the nodes connected to the thin elements are solved together with the elasticity equations for the other nodes. In the Enhanced-Approximation Linear Solution Technique (EALST), we propose to use stronger approximations for the parts of the domain where we are facing convergence challenges. This could be implemented, for example, by identifying the elements covering the zones where we need enhanced approximation, and reflecting this in defining the element-level constituents of the approximation matrix. For example, for the elements that need stronger approximations, we can use as the element-level approximation matrix the full element-

level matrix, while for all other elements we use a diagonal element-level matrix.

MIXED INTERFACE-TRACKING/ INTERFACE-CAPTURING TECHNIQUE

In computation of flow problems with fluid-solid interfaces, an interface-tracking technique, where the fluid mesh moves to track the interface, would allow us to have full control of the resolution of the fluid mesh in the boundary layers. With an interface-capturing technique (or an interface representation technique in the more general case), on the other hand, independent of how well the interface geometry is represented, the resolution of the fluid mesh in the boundary layer will be limited by the resolution of the fluid mesh where the interface is. In computation of flow problems with fluid-fluid interfaces where the interface is too complex or unsteady to track while keeping the remeshing frequency under control, interface-capturing techniques, with enhanced-discretization as needed, could be used as more flexible alternatives. Sometimes we may need to solve flow problems with both fluid-solid interfaces and complex or unsteady fluid-fluid interfaces.

MITICT was introduced in,[18] primarily for fluid-object interactions with multiple fluids. The class of applications we were targeting were fluid-particle-gas interactions and free-surface flow of fluid-particle mixtures. However, the MITICT can be applied to a larger class of problems, where it is more effective to use an interface-tracking technique to track the solid-fluid interfaces and an interface-capturing technique to capture the fluid-fluid interfaces. The interface-tracking technique is the DSD/SST formulation (but could as well be the Arbitrary Lagrangian-Eulerian method or other moving mesh methods). The interface-capturing technique rides on this, and is based on solving over a moving mesh, in addition to the Navier-Stokes equations, the advection equation governing the time-evolution of the interface function. The additional DSD/SST formulation is for this advection equation:

$$\int_{Q_n} \psi^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) dQ + \int_{\Omega_n} (\psi^h)_n^+ \left((\phi^h)_n^+ - (\phi^h)_n^- \right) d\Omega + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_\phi \left(\frac{\partial \psi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \psi^h \right) \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) dQ = 0. \quad (17)$$

This equation, together with Equation (12), constitute a mixed interface-tracking/interface-capturing technique that would track the solid-fluid interfaces and capture the fluid-fluid interfaces that would be too complex or unsteady to track with a moving mesh. The interface-capturing part of MITICT can be upgraded to the EDICT formulation for more accurate representation of the interfaces captured.

The MITICT can also be used for computation of fluid-structure interactions with multiple fluids or for flows with mechanical components moving in a mixture of two fluids. In more general cases, the MITICT can be used for classes of problems that involve both interfaces that can be accurately tracked with a moving mesh method and interfaces that are too complex or unsteady to be tracked and therefore require an interface-capturing technique.

EDGE-TRACKED INTERFACE LOCATOR TECHNIQUE

The Edge-Tracked Interface Locator Technique (ETILT) was introduced in,[18] to have an interface-capturing technique with better volume conservation properties and sharper representation of the interfaces. To this end, we first define a second finite-dimensional representation of the interface function, namely ϕ^{he}. With ϕ^{he}, interfaces are represented as collection of positions along element edges crossed by the interfaces. Nodes belong to "chunks" of Fluid A or Fluid B. An edge either belongs to a chunk of Fluid A or Fluid B or is an interface edge. Each element is either filled fully by a chunk of Fluid A or Fluid B, or is shared by a chunk of Fluid A and a chunk of Fluid B. If an element is shared like that, the shares are determined by the position of the interface along the edges of that element. The base finite element formulation is essentially the one described by Equations (15) and (16). Although the ETILT can be used in combination with the EDICT, we assume that we are working here with the plain, non-EDICT versions of Equations (15) and (16).

At each time step, given $\mathbf{u}_n^h$ and ϕ_n^{he}, we determine $\mathbf{u}_{n+1}^h$, p_{n+1}^h, and ϕ_{n+1}^{he}. The definitions of ρ and μ are modified to use the edge-based representation of the interface function: $\rho^h = \phi^{he}\rho_A + (1-\phi^{he})\rho_B$, $\mu^h = \phi^{he}\mu_A + (1-\phi^{he})\mu_B$. In marching from time level n to $n+1$, we first calculate ϕ_n^h from ϕ_n^{he} by a least-squares projection:

$$\int_\Omega \psi^h \left(\phi_n^h - \phi_n^{he} \right) d\Omega = 0. \quad (18)$$

To calculate ϕ^h_{n+1}, we use Equation (16). From ϕ^h_{n+1}, we calculate ϕ^{he}_{n+1} by a combination of a least-squares projection:

$$\int_\Omega (\psi^{he}_{n+1})_P \left((\phi^{he}_{n+1})_P - \phi^h_{n+1}\right) d\Omega = 0, \qquad (19)$$

and corrections to enforce volume conservation for all chunks of Fluid A and Fluid B, taking into account the mergers between the chunks and the split of chunks. This volume conservation condition can symbolically be written as $VOL\,(\phi^{he}_{n+1}) = VOL\,(\phi^{he}_n)$. Here the subscript P is used for representing the intermediate values following the projection, but prior to the corrections for volume conservation. These projections and volume corrections are embedded in the iterative solution technique, and are carried out at each iteration (see[18]).

NUMERICAL EXAMPLES

Free-Surface Flow Past a Bridge Support

The free-stream Reynolds and Froude numbers are 10 million and 0.564. The mesh has 230,480 prism-based space-time elements. The DSD/SST formulation is used with algebraic mesh update. Figure 2 shows, at an instant, the cylinder together with the free-surface color-coded with the velocity magnitude. For more on this simulation see.[19]

2D Sloshing in a Container

A container partially filled with water is subjected to a horizontal acceleration of 0.2g. We first compute with the DSD/SST formulation, with 6,000 quadrilateral elements. We refer to this as Solution-1T. Solution-1 is obtained by using Mesh-1, with 30,000 triangular elements. Solution-2 is obtained with the EDICT, where all functions come from Mesh-1 $\oplus$ Mesh-2. Solution-3 is obtained with the functions for velocity and pressure coming from Mesh-1 $\oplus$ Mesh-2, and for the interface function from Mesh-1 $\oplus$ Mesh-2 $\oplus$ Mesh-3. Figure 3 shows shows the results. For more details see.[20]

CONCLUDING REMARKS

In this paper, we provided an overview of the stabilized finite element interface-tracking and interface-capturing techniques we have developed in recent years for computation of flow problems with moving boundaries and interfaces. The interface-tracking techniques are based on the DSD/SST formulation, where the mesh moves to track the interface. The interface-capturing techniques, which were developed for two-fluid flows, are based on the stabilized formulation, over non-moving meshes, of both the flow equations and the advection equation governing the time-evolution of an interface function marking the location of the interface. For interface-capturing techniques, the EDICT increases the accuracy in representing the interface. The MITICT was developed for the classes of problems that involve both interfaces that can be accurately tracked with a moving mesh method and interfaces that are too complex or unsteady to be tracked and therefore require an interface-capturing technique. The ETILT was developed to improve the interface-capturing techniques with better volume conservation properties and sharper representation of the interfaces.

ACKNOWLEDGEMENT

This work was supported by NASA JSC (grant no. NAG9-1059), AFOSR (contract no. F49620-98-1-0214), and by the Natick Soldier Center (contract no. DAAD16-00-C-9222).

References

[1] T.E. Tezduyar, "Stabilized finite element formulations for incompressible flow computations", *Advances in Applied Mechanics*, **28** (1991) 1–44. ⟵ (1992)

[2] T.E. Tezduyar, S. Aliabadi, and M. Behr, "Enhanced-Discretization Interface-Capturing Technique", in Y. Matsumoto and A. Prosperetti, editors, *Proceedings of the ISAC '97 High Performance Computing on Multiphase Flows*, 1–6, Japan Society of Mechanical Engineers, 1997.

[3] T.E. Tezduyar and Y. Osawa, "Methods for parallel computation of complex flow problems", *Parallel Computing*, **25** (1999) 2039–2066.

[4] M. Behr and T.E. Tezduyar, "Finite element solution strategies for large-scale flow simulations", *Computer Methods in Applied Mechanics and Engineering*, **112** (1994) 3–24.

[5] A.A. Johnson and T.E. Tezduyar, "Parallel computation of incompressible flows with complex geometries", *International Journal for Nu-*

merical Methods in Fluids, **24** (1997) 1321–1340.

[6] T.E. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro, and M. Litke, "Flow simulation and high performance computing", *Computational Mechanics*, **18** (1996) 397–412.

[7] S. Mittal and T.E. Tezduyar, "Massively parallel finite element computation of incompressible flows involving fluid-body interactions", *Computer Methods in Applied Mechanics and Engineering*, **112** (1994) 253–282.

[8] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, and S. Mittal, "Parallel finite-element computation of 3D flows", ~~*IEEE*~~ *Computer*, **26** (1993) 27–36.

[9] T.E. Tezduyar, M. Behr, S. Mittal, and A.A. Johnson, "Computation of unsteady incompressible flows with the finite element methods – space-time formulations, iterative strategies and massively parallel implementations", in P. Smolinski, W.K. Liu, G. Hulbert, and K. Tamma, editors, *New Methods in Transient Analysis*, AMD-Vol.143, ASME, New York, (1992) 7–24.

[10] D.R. Lynch, "Wakes in liquid-liquid systems", *Journal of Computational Physics*, **47** (1982) 387–411.

[11] J.T. Batina, "Unsteady Euler airfoil solutions using unstructured dynamics meshes", in *Proceedings of AIAA 27th Aerospace Sciences Meeting*, AIAA Paper 89-0115, Reno, Nevada, (1989).

[12] C. Farhat, M. Lesoinne, and N. Maman, "Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution", *International Journal for Numerical Methods in Fluids*, **21** (1995) 807–835.

[13] C. W. Hirt and B. D. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries", *Journal of Computational Physics*, **39** (1981) 201–225.

[14] T.E. Tezduyar, S. Aliabadi, and M. Behr, "Enhanced-Discretization Interface-Capturing Technique (EDICT) for computation of unsteady flows with interfaces", *Computer Methods in Applied Mechanics and Engineering*, **155** (1998) 235–248.

[15] S. Mittal, S. Aliabadi, and T.E. Tezduyar, "Parallel computation of unsteady compressible flows with the EDICT", *Computational Mechanics*, **23** (1999) 151–157.

[16] T.E. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar, and J. McCune, "Numerical methods for computer assisted analysis of parachute mechanics", in *Proceedings of 8th Conference on Numerical Methods in Continuum Mechanics (CD-ROM)*, Liptovsky Jan, Slovakia, (2000).

[17] T.E. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar, and J. McCune, "Computational methods for parachute aerodynamics", to appear in *Proceedings of Computational Fluid Dynamics for the 21st Century*, Kyoto, Japan, 2000. ⟵ See update U1 below.

[18] T.E. Tezduyar, "Finite element methods for flow problems with moving boundaries and interfaces", ~~to appear~~ in *Archives of Computational Methods in Engineering*, 2001. ⟵ **8** (2001) 83-130.

[19] I. Güler, M. Behr, and T.E. Tezduyar, "Parallel finite element computation of free-surface flows", *Computational Mechanics*, **23** (1999) 117–123.

[20] T.E. Tezduyar and S. Aliabadi, "EDICT for computation of unsteady flows with interfaces", in *Modeling and Simulation Based Engineering (eds. S. Atluri and P. O'Donoghue), Proceedings of International Conference on Computational Engineering Science*, Atlanta, Georgia, (1998).

Publication Update

U1. T. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar and J. McCune, "Computational Methods for Parachute Aerodynamics", *Computational Fluid Dynamics for the 21st Century* (eds. M. Hafez, K. Morinishi and J. Periaux), Springer (2001).

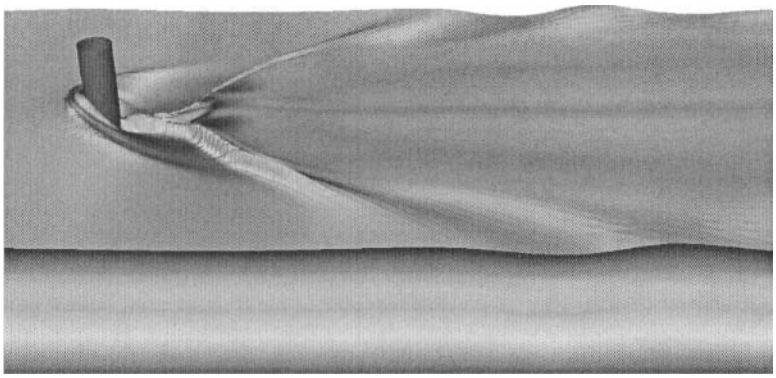

Figure 2. Free-surface flow past a bridge support. The bridge support and the free-surface color-coded with the velocity magnitude.

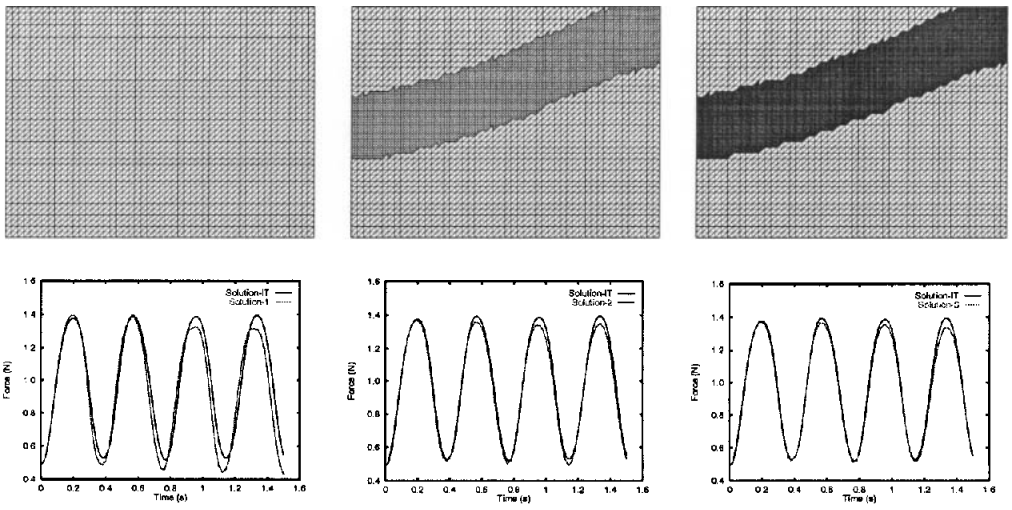

Figure 3. 2D sloshing in a container. Pictures shows, at the top, at t = 0.2 s, Mesh-1 together with Mesh-2 and Mesh-3 (both shown on top of Mesh-1); and at the bottom, the time histories of the horizontal forces exerted on the container.