COMPUTATIONAL MECHANICS
WCCM VI in conjunction with APCOM'04, Sept. 5-10, 2004, Beijing, China

Proceedings of the 6th World Congress on Computational Mechanics, Beijing, China, CD-ROM (2004).

Interface-Tracking and Interface-Capturing Techniques for Computation of Moving Boundaries and Interfaces

Tayfun E. Tezduyar

Mechanical Engineering, Rice University - MS 321, Houston, Texas, U.S.A.

e-mail: tezduyar@rice.edu

Abstract: As a category of challenging flow problems, flows with moving boundaries and interfaces, includes fluid–particle, fluid–object and fluid–structure interactions; free-surface and two-fluid flows; and flows with moving mechanical components. To address the challenges involved in computation of this category of problems, we developed a number of interface-tracking and interface-capturing techniques. Both classes of techniques are based on stabilized formulations. The interface-tracking techniques are based on the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation, where the mesh moves to track the interface. The interface-capturing techniques, developed primarily for free-surface and two-fluid interface flows, are formulated typically over non-moving meshes, using an advection equation in addition to the flow equations. The advection equation governs the evolution of an interface function that marks the location of the interface. We also describe some of the methods we developed to increase the scope and accuracy of these two classes of techniques.

Key words: Moving boundaries and interfaces, Interface-tracking, Interface-capturing, Enhanced discretization and solution.

1 INTRODUCTION

Interface-tracking and interface-capturing techniques are widely used in computation of flow problems with moving boundaries and interfaces. This category of problems includes fluid–particle, fluid–object and fluid–structure interactions; free-surface and two-fluid flows; and flows with moving mechanical components. These problems offer many computational challenges. An interface-tracking technique requires meshes that "track" the interfaces and are 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, is computed to "capture" the interface. The interface is captured within the resolution of the finite element mesh covering the area where the interface is. The interface-tracking and interface-capturing techniques we developed (see [1, 2, 3, 4]) are based on stabilized formulations. The stabilized methods are the streamline-upwind/Petrov-Galerkin (SUPG) [5, 6] and pressure-stabilizing/Petrov-Galerkin (PSPG) [1] formulations. An earlier version of the pressure-stabilizing formulation for Stokes flows was reported in [7]. These stabilized formulations prevent numerical oscillations and other instabilities in solving problems with high Reynolds and/or Mach numbers and shocks and strong boundary layers, as well as when using equal-order interpolation functions for velocity and pressure and other

unknowns. This class of stabilized formulations also substantially improve the convergence rate in iterative solution of the large, coupled nonlinear equation system involved.

The Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation [1, 2] is an interface-tracking technique. In this technique, the finite element formulation of a problem is written over its space–time domain. As the spatial domain occupied by the fluid changes in time, the mesh is updated. In general we do that by using an automatic mesh moving method [8, 2] we developed. In this method, the motion of the nodes is governed by the equations of elasticity. Full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) is carried out as needed. The stabilized space–time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example [9]).

In computation of two-fluid flows, in some cases the interface might be too complex 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 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 governing equations and core methods are described in Sections 2–4. Methods developed to increase the scope and accuracy of the core methods are described in Sections 5–10.

2 GOVERNING EQUATIONS

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

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

$$\boldsymbol{\nabla}\cdot\mathbf{u} = 0, \tag{2}$$

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

$$\boldsymbol{\sigma}(p,\mathbf{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \qquad \boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}\left((\boldsymbol{\nabla}\mathbf{u}) + (\boldsymbol{\nabla}\mathbf{u})^T\right). \tag{3}$$

Here p is the pressure, $\mathbf{I}$ is the identity tensor, $\mu = \rho\nu$ is the viscosity, ν is the kinematic viscosity, and $\boldsymbol{\varepsilon}(\mathbf{u})$ is the strain-rate tensor. The essential and natural boundary conditions for Eq. (1) are represented as

$$\mathbf{u} = \mathbf{g} \text{ on } (\Gamma_t)_g, \qquad \mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h} \text{ on } (\Gamma_t)_h, \tag{4}$$

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 $\mathbf{u}_0(\mathbf{x})$ is specified as the initial condition.

If there are no moving boundaries or interfaces, the spatial domain does not need to change in 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 the formulation is not based on defining the spatial domain to be the part of the space occupied by the fluid(s). For example, fluid–fluid interfaces can be modeled over a fixed spatial domain 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. A free-surface problem can be modeled as a special case where Fluid B is irrelevant and assigned a sufficiently low density. An interface function ϕ serves as the marker identifying Fluids A and B with the definition $\phi = \{1 \text{ for Fluid A and } 0 \text{ for Fluid B}\}$. The interface between the two fluids is approximated to be at $\phi = 0.5$. In this context, ρ and μ are defined as $\rho = \phi\rho_A + (1-\phi)\rho_B$ and $\mu = \phi\mu_A + (1-\phi)\mu_B$. The evolution of ϕ, and consequently the motion of the interface, is governed by a time-dependent advection equation, written on Ω and $\forall t \in (0,T)$ as

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} \phi = 0. \tag{5}$$

In conjunction with this equation, ϕ is specified at inflow boundaries, and a function $\phi_0(\mathbf{x})$ is given as the initial condition.

3 STABILIZED SEMI-DISCRETE FORMULATION

Given Eqs. (1)–(2), we form some suitably-defined finite-dimensional trial solution and test function spaces for velocity and pressure: $\mathcal{S}_{\mathbf{u}}^h$, $\mathcal{V}_{\mathbf{u}}^h$, $\mathcal{S}_p^h$ and $\mathcal{V}_p^h = \mathcal{S}_p^h$. The stabilized finite element formulation of Eqs. (1)–(2) can be written as follows: find $\mathbf{u}^h \in \mathcal{S}_{\mathbf{u}}^h$ and $p^h \in \mathcal{S}_p^h$ such that $\forall \mathbf{w}^h \in \mathcal{V}_{\mathbf{u}}^h$ and $q^h \in \mathcal{V}_p^h$:

$$\begin{aligned}
&\int_\Omega \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega - \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h d\Gamma \\
+ &\int_\Omega q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h + \tau_{\mathrm{PSPG}} \nabla q^h \right] \cdot \left[\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \nu_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega \;=\; 0,
\end{aligned} \tag{6}$$

where

$$\text{Ł}(q^h, \mathbf{w}^h) = \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h). \tag{7}$$

Here τ_{SUPG}, τ_{PSPG} and ν_{LSIC} are the SUPG, PSPG and LSIC (least-squares on incompressibility constraint) stabilization parameters. For ways of calculating τ_{SUPG}, τ_{PSPG} and ν_{LSIC}, see [10, 11, 4].

4 DEFORMING-SPATIAL-DOMAIN/STABILIZED SPACE–TIME (DSD/SST) FORMULATION

In the DSD/SST method [1], 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}. At each time step the integrations 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. The notation $(\cdot)_n^-$ and

$(\cdot)_n^+$ denotes the function values at t_n as approached from below and above. Each Q_n is decomposed into elements Q_n^e, where $e = 1, 2, \ldots, (n_{el})_n$. The subscript n used with n_{el} is for the general case in which the number of space–time elements may change from one space–time slab to another. The essential and natural boundary conditions are enforced over $(P_n)_{\rm g}$ and $(P_n)_{\rm h}$, the complementary subsets of the lateral boundary of the space–time slab. The finite element trial function spaces $(\mathcal{S}_{\mathbf{u}}^h)_n$ for velocity and $(\mathcal{S}_p^h)_n$ for pressure, and the test function spaces $(\mathcal{V}_{\mathbf{u}}^h)_n$ and $(\mathcal{V}_p^h)_n = (\mathcal{S}_p^h)_n$ are defined by using, over Q_n, first-order polynomials in both space and time. The DSD/SST formulation is written as follows: given $(\mathbf{u}^h)_n^-$, find $\mathbf{u}^h \in (\mathcal{S}_{\mathbf{u}}^h)_n$ and $p^h \in (\mathcal{S}_p^h)_n$ such that $\forall \mathbf{w}^h \in (\mathcal{V}_{\mathbf{u}}^h)_n$ and $q^h \in (\mathcal{V}_p^h)_n$:

$$
\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
- &\int_{(P_n)_{\rm h}} \mathbf{w}^h \cdot \mathbf{h}^h dP + \int_{Q_n} q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho \left((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^- \right) d\Omega \\
+ &\sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{1}{\rho} \left[\tau_{\rm SUPG} \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) + \tau_{\rm PSPG} \nabla q^h \right] \cdot \left[\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] dQ \\
+ &\sum_{e=1}^{n_{el}} \int_{Q_n^e} \nu_{\rm LSIC} \boldsymbol{\nabla} \cdot \mathbf{w}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ = 0. \qquad (8)
\end{aligned}
$$

This formulation is applied to all space–time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, starting with $(\mathbf{u}^h)_0^- = \mathbf{u}_0$. For an earlier, detailed reference on the formulation see [1].

How the mesh is updated as the spatial domain occupied by the fluid changes in time depends on several factors. These factors include the complexity of the interface and overall geometry, how unsteady the interface is, and how the starting mesh was generated. Detailed descriptions of various mesh update techniques we developed can be found in [2, 3, 4]. These include an automatic mesh moving method, techniques to reduce the frequency of remeshing, and a mesh update method for handling solid objects (or surfaces) in fast linear or rotational relative motion. Also included are the techniques for handling the structured layers of elements generated around solid or deformable solid objects (to fully control the mesh resolution near solid objects and have more accurate representation of the boundary layers).

5 ENHANCED-DISCRETIZATION INTERFACE-CAPTURING TECHNIQUE (EDICT)

In the EDICT [12, 2], 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. The trial function spaces corresponding to velocity, pressure, and interface function are denoted, respectively, by $(\mathcal{S}_{\mathbf{u}}^h)_n$, $(\mathcal{S}_p^h)_n$, and $(\mathcal{S}_\phi^h)_n$. The weighting function spaces corresponding to the momentum 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 $\forall \psi_{n+1}^h \in (\mathcal{V}_\phi^h)_{n+1}$:

$$
\begin{aligned}
& \int_\Omega \mathbf{w}_{n+1}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}_{n+1}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega \\
- & \int_{\Gamma_{\mathrm{h}}} \mathbf{w}_{n+1}^h \cdot \mathbf{h}^h d\Gamma + \int_\Omega q_{n+1}^h \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega \\
+ & \sum_{e=1}^{n_{el}} \int_{\Omega^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}_{n+1}^h + \tau_{\mathrm{PSPG}} \nabla q_{n+1}^h \right] \cdot \left[\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] d\Omega \\
+ & \sum_{e=1}^{n_{el}} \int_{\Omega^e} \nu_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}_{n+1}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega = 0, \qquad (9)
\end{aligned}
$$

$$
\begin{aligned}
& \int_\Omega \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega \\
+ & \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_\phi \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega = 0. \qquad (10)
\end{aligned}
$$

Here τ_ϕ is calculated by applying the definition of τ_{SUPG} to Eq. (10).

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 to avoid this envelope being too wide or too narrow.

6 MIXED INTERFACE-TRACKING/INTERFACE-CAPTURING TECHNIQUE (MITICT)

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 locator technique in the more general case), on the other hand, independent of how accurately the interface is located, 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.

The MITICT [2, 3, 4] was introduced 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 the advection equation:

$$\int_{Q_n} \psi^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) dQ + \int_{\Omega_n} (\psi^h)_n^+ \left((\phi^h)_n^+ - (\phi^h)_n^- \right) d\Omega$$
$$+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_\phi \left(\frac{\partial \psi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi^h \right) \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) dQ = 0. \tag{11}$$

This equation, together with Eq. (8), 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.

We propose the MITICT-L as a version of the MITICT with limited mesh moving. In the MITICT-L, mesh moving would be limited to sufficiently wide regions of the computational domain enveloping the fluid–solid interfaces. These regions would be wide enough so that we can keep the mesh deformation, frequency of remeshing, and frequency of re-defining the mesh-moving regions at reasonable levels. By not moving the mesh outside of these regions, we would significantly reduce the computational cost for solving the mesh-moving equations. By using only semi-discrete formulations outside of the mesh-moving regions, we would also avoid the computational cost associated with using space–time formulations. Appropriate interface (matching) conditions would be used where two differently treated regions meet.

7 FLUID–SOLID INTERFACE LOCATOR TECHNIQUE (FSILT)

If the Reynolds number is not high enough to be concerned about the mesh resolution near the solid surfaces, then there is no strong reason for moving the mesh for the purpose of controlling the mesh resolution where those interfaces are. Depending on the nature of the problem, such flow simulations can still be carried out with the MITICT or MITICT-L. Not having high-resolution meshes near the fluid–solid interfaces would reduce the mesh moving burden in terms of the number of equations to be solved and frequency of remeshing. As an alternative to this approach, we propose the FSILT. In the FSILT, we propose to carry out the fluid–solid interface computations over non-moving meshes. We propose to accomplish this by modifying the left-hand-side of Eq. (6) as follows:

$$(LHS\ of\ Eq.\ (6)) - \int_{\Gamma_{\text{FS}}} \mathbf{w}^h \cdot \mathbf{J}_{FS}^h \, d\Gamma = 0, \tag{12}$$

where $\Gamma_{\rm FS}$ is the fluid–solid interface, which is discretized by the structural interface mesh, and $\mathbf{J}^h_{FS}$ represents the interface stresses acting on the fluid at the fluid–solid interface. This would require projection between the non-moving fluid mesh and the moving structural interface mesh. The structural motion would be governed by the applicable structural mechanics equations (examples: rigid- or deformable-body mechanics equations, and membrane or shell equations), taking into account the interface stresses, $-\mathbf{J}^h_{FS}$, acting on the structure. In conjunction with the additional unknown $\mathbf{J}^h_{FS}$, we add on the following constraint equation:

$$\int_{\Gamma_{\rm FS}} \mathbf{K}^h_{FS} \cdot \left(\mathbf{u}^h_F - \mathbf{u}^h_S\right)\, d\Gamma = 0, \tag{13}$$

where $\mathbf{u}^h_F$ is the fluid velocity evaluated on $\Gamma_{\rm FS}$, $\mathbf{u}^h_S$ is the structural displacement rate, and $\mathbf{K}^h_{FS}$ is the test function (variation of $\mathbf{J}^h_{FS}$). We propose two possible ways to approach this constraint problem. In the penalty formulation approach, we write

$$\mathbf{J}^h_{FS} = -\lambda_{FS}\big(\mathbf{u}^h_F - \mathbf{u}^h_S\big), \tag{14}$$

where λ_{FS} is a penalty parameter. In the stabilized formulation approach, we modify Eq. (13) as follows:

$$\int_{\Gamma_{\rm FS}} \mathbf{K}^h_{FS} \cdot \left(\mathbf{u}^h_F - \mathbf{u}^h_S\right)\, d\Gamma$$
$$+ \int_{\Gamma_{\rm FS}} \tau_{\rm FS}\, \frac{1}{\rho_M L_{FS}}\, \mathbf{K}^h_{FS} \cdot \left[\mathbf{J}^h_{FS} - \left(\mathbf{n}^h_A \cdot \boldsymbol{\sigma}^h_A + \mathbf{n}^h_B \cdot \boldsymbol{\sigma}^h_B\right)\right]\, d\Gamma = 0, \tag{15}$$

where we assume that the structure is $(n_{sd}-1)$-dimensional (e.g. membrane or shell) and that we have two different fluids, Fluid A and Fluid B, on each side of the structure. Here $\rho_M = max(\rho_A, \rho_B)$, L_{FS} is a global length scale for the fluid–solid interface, and the stabilization parameter is defined as $\tau_{\rm FS} = \left((\tau_{\rm FS1})^{-r} + (\tau_{\rm FS3})^{-r}\right)^{-1/r}$ (typically $r = 2$), with

$$\tau_{\rm FS1} = \frac{h_{\rm FS1}}{2\ u_{\rm cha}}\ , \qquad \tau_{\rm FS3} = \frac{(h_{\rm FS3})^2}{4\ (\mu_M/\rho_M)}\ , \tag{16}$$

where $h_{\rm FS1}$ and $h_{\rm FS3}$ are appropriate local length scales, and $\mu_M = max(\mu_A, \mu_B)$. For other stabilized formulations with Lagrange multipliers on the boundary see [14, 15].

We propose to use the underlying concepts of the FSILT also to take into account the surface tension effects in computation of free-surface and two-fluid interface flows with interface-capturing techniques. We propose to accomplish this by adopting Eq. (12) for use in the context of Eqs. (9) and (10) in their non-EDICT form. For that, the left-hand-side of Eq. (9) would be modified as follows:

$$(LHS\ of\ Eq.\ (9)) - \int_{\Gamma_{\rm ST}} \mathbf{w}^h \cdot \mathbf{J}^h_{ST}\, d\Gamma = 0, \tag{17}$$

where $\Gamma_{\rm ST}$ is the two-fluid interface (or free surface), discretized by a separate $(n_{sd}-1)$-dimensional mesh, and $\mathbf{J}^h_{ST}$ represents the surface tension forces acting on the fluid at the interface. This would require projection between the non-moving fluid mesh and the moving interface mesh. The interface motion would continue to be governed by Eq. (10).

In fluid–structure interactions where the structure is $(n_{sd}-1)$-dimensional, and in free-surface and two-fluid interface flows with surface tension, we need to allow the fluid pressure be discontinuous across the interface. As a rudimentary fix, in the elements crossed by the

interface, we propose to interpolate the pressure at each side of the interface by using nodal values only from that side. For each side, we first calculate in some way the "extended" values for the nodes at the other side. We then interpolate the pressure by using the real nodal values from that side and the "extended" nodal values from the other side.

We propose the FSILT-ED (FSILT - Extended Domain) as a version of the FSILT where the pressure interpolation is improved by enhancing the finite element function spaces around the interface. In the FSILT-ED, the domain for the fluid at one side of the interface is extended to the other side. As density and viscosity, in the extended domain for Fluid A, we use $\epsilon\rho_A$ and $\epsilon\mu_A$, and in extended domain for Fluid B, $\epsilon\rho_B$ and $\epsilon\mu_B$. Here ϵ is a very small parameter that would make the density and viscosity values negligible and the fluid dynamics practically irrelevant. How much the extended domain goes beyond the interface would depend on how frequently we would like to re-define the extended domain. For elements crossed by the interface, the elements are assembled once for Fluid A and and once for Fluid B. We do the same for elements that are not crossed by the interface but are in both a real and an extended domain. In a fluid–structure interaction problem, if we have a single fluid, then we use the terminology Side A and Side B instead of Fluid A and Fluid B, but the approach remains the same.

With the enhanced functions spaces, we split Eq. (12) into two equations. One equation would be for Fluid A (or Side A), where $\mathbf{J}^h_{FS}$ is replaced with $(\mathbf{J}^h_{FS})_A$, and the other equation for Fluid B (or Side B), where $\mathbf{J}^h_{FS}$ is replaced with $(\mathbf{J}^h_{FS})_B$. The interface stresses acting on the fluid at each side would then be added to calculate the total interface stresses: $\mathbf{J}^h_{FS} = (\mathbf{J}^h_{FS})_A + (\mathbf{J}^h_{FS})_B$. Similarly, Eqs. (13), (14) and (15) would be split into two sets of equations. In the set for Fluid A, $\mathbf{K}^h_{FS}$, $\mathbf{u}^h_F$ and $\mathbf{J}^h_{FS}$ would be replaced with $(\mathbf{K}^h_{FS})_A$, $(\mathbf{u}^h_F)_A$ and $(\mathbf{J}^h_{FS})_A$, and in the set for Fluid B, with $(\mathbf{K}^h_{FS})_B$, $(\mathbf{u}^h_F)_B$ and $(\mathbf{J}^h_{FS})_B$. In the version of Eq. (15) for Fluid A, the subscript B would signify the extended domain part of Fluid A. In the version for Fluid B, the subscript A would signify the extended domain part of Fluid B.

We propose the FSILT-EDP (FSILT - Extended Domain for Pressure) as a version of the FSILT where the pressure interpolation is improved by enhancing only the pressure function space around the interface. In this version Eqs. (12), (13), (14) and (15) would not be split into two sets of equations. For Eqs. (12) and (15), for elements crossed by the interface, at each side of the interface we interpolate the pressure by using the real nodal values from that side and the "extended" nodal values from the other side. For the part of Eq. (12) corresponding to the incompressibility constraint, for elements crossed by the interface, integration over each side of the interface generates what we assemble to the equations associated with the fluid at that side. For an element that is not crossed by the interface but is in both a real and an extended domain for pressure, integration over the extended domain, after multiplication by ϵ, generates what we assemble to the equations associated with the fluid that is not present in that element. For such an element, over the extended domain the pressure is interpolated by using the nodal values associated with the fluid that is not present in that element.

We also propose to use the underlying concepts of the FSILT-EDP to take into account the surface tension effects in computation of free-surface and two-fluid interface flows with interface-capturing techniques. The underlying concepts would be used in the context of Eq. (17) and Eq. (10) in its non-EDICT form.

8 EDGE-TRACKED INTERFACE LOCATOR TECHNIQUE (ETILT)

The ETILT [2, 3, 4] was introduced 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}. The added superscript "e" indicates that this is an edge-based representation. With ϕ^{he}, interfaces are represented as collection of positions along element edges crossed by the interfaces (i.e., along the "interface edges"). 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 Eqs. (9) and (10). 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 Eqs. (9) and (10).

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:

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

To calculate ϕ_{n+1}^h, we use Eq. (10). From ϕ_{n+1}^h, we calculate ϕ_{n+1}^{he} by a combination of a least-squares projection:

$$\int_\Omega (\psi_{n+1}^{he})_P \left((\phi_{n+1}^{he})_P - \phi_{n+1}^h\right) d\Omega = 0, \tag{19}$$

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 $VOL\ (\phi_{n+1}^{he}) = VOL\ (\phi_n^{he})$. Here the subscript P is used for representing the intermediate values following the projection, but prior to the corrections for volume conservation. It can be shown that the projection given by Eq. (19) can be interpreted as locating the interface along the interface edges at positions where $\phi_{n+1}^h = 1/2$.

As an alternative way for computing ϕ_n^h from ϕ_n^{he}, we propose to solve the equation

$$\int_{\Omega_{INT}} \psi_n^h \left(\phi_n^h - \phi_n^{he}\right) d\Omega \ + \ \sum_{k=1}^{n_{ie}} \psi_n^h(\mathbf{x}_k)\ \lambda_{PEN} \left(\phi_n^h(\mathbf{x}_k) - 1/2\right) = 0, \tag{20}$$

where n_{ie} is the number of interface edges, $\mathbf{x}_k$ is the coordinate of the interface location along the k^{th} interface edge, λ_{PEN} is a penalty parameter, and Ω_{INT} is the solution domain. This domain is the union of all the elements containing at least one node where the value of ϕ_n^h is unknown. We can assume ϕ_n^h to be unknown only at the nodes of the interface edges, with known values $\phi_n^h = 1$ (for Fluid A) and $\phi_n^h = 0$ (for Fluid B) at all other nodes. We can also augment the number of nodes where ϕ_n^h is unknown and thus enlarge the solution domain. This can be done all the way to the point where $\Omega_{INT} = \Omega$. As another alternative, in Eq. (20) we can replace the least-squares projection term with a slope-minimization term:

$$\int_{\Omega_{INT}} \boldsymbol{\nabla}\psi_n^h \cdot \boldsymbol{\nabla}\phi_n^h \ d\Omega \ + \ \sum_{k=1}^{n_{ie}} \psi_n^h(\mathbf{x}_k)\ \lambda_{PEN} \left(\phi_n^h(\mathbf{x}_k) - 1/2\right) = 0. \tag{21}$$

A 1D version of the way of computing ϕ_n^h from ϕ_n^{he} can be formulated by minimizing $(\phi_n^h - \phi_n^{he})^2$ along "chains" of interface edges:

$$\int_{S_{INT}} \psi_n^h \left(\phi_n^h - \phi_n^{he}\right) ds \; + \; \sum_{k=1}^{n_{ie}} \psi_n^h(\mathbf{x}_k) \; \lambda_{PEN} \left(\phi_n^h(\mathbf{x}_k) - 1/2\right) = 0, \tag{22}$$

where S_{INT} is the collection of all chains of interface edges, and s is the integration coordinate along the interface edges. This is, of course, a simpler formulation, and much of the equations for the unknown nodal values will be uncoupled.

These projections and volume corrections are embedded in the iterative solution technique, and are carried out at each iteration. The iterative solution technique, which is based on the Newton–Raphson method, addresses both the nonlinear and coupled nature of the set of equations that need to be solved at each time step. More explanation of how the projections and volume corrections would be handled at a nonlinear iteration step can be found in [2, 3, 4].

9 ITERATIVE SOLUTION METHODS

The finite element formulations reviewed in the earlier sections fall into two categories: a space–time formulation with moving meshes or a semi-discrete formulation with non-moving meshes. Full discretizations of these formulations lead to coupled, nonlinear equation systems that need to be solved at every time step of the simulation. In a form that is partitioned with respect to the models represented, these nonlinear equations can be written as follows:

$$\begin{aligned} \mathbf{N}_1\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_1, \\ \mathbf{N}_2\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_2, \end{aligned} \tag{23}$$

where $\mathbf{d}_1$ and $\mathbf{d}_2$ are the vectors of nodal unknowns corresponding to unknown functions $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. For example, in the context of a coupled fluid–structure interaction problem, $\mathbf{u}_1$ and $\mathbf{u}_2$ might be representing the fluid and structure unknowns, respectively. In solving these equations with a Newton–Raphson sequence, at every step of the sequence we solve the following linear equation system:

$$\begin{aligned} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 &= \mathbf{b}_1, \\ \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 &= \mathbf{b}_2, \end{aligned} \tag{24}$$

where $\mathbf{b}_1$ and $\mathbf{b}_2$ are the residuals of the nonlinear equation system, $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the correction increments computed for $\mathbf{d}_1$ and $\mathbf{d}_2$, and

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}. \tag{25}$$

In fluid–structure interactions, computation of the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ might pose a significant challenge. A mixed analytical/numerical element-vector-based (AEVB/NEVB) computation technique was introduced to address this challenge, and the description of this technique can be found in [2, 4]. An overview of the more general iterative solution techniques used in our computations can also be found in [2, 4]. The overview includes preconditioning techniques used in solving the linear equation system given by Eq. (24) and the techniques for computation of the residuals of the linear equation system.

In fluid–structure interaction computations where the structure is light, in the absence of taking into account the coupling blocks $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$, we propose a short cut approach

for improving the convergence of the coupling iterations. In this approach, to reduce "over-correcting" (i.e. "over-incrementing") the structural displacements during the coupling iterations, we artificially increase the structural mass contribution to the matrix block corresponding to the structural mechanics equations and unknowns. With the understanding that subscript 2 denotes the structure, this would be equivalent to artificially increasing the mass matrix contribution to $\mathbf{A}_{22}$. This is achieved without altering $\mathbf{b}_1$ or $\mathbf{b}_2$ (i.e. $\mathbf{F}_1 - \mathbf{N}_1\ (\mathbf{d}_1, \mathbf{d}_2)$ or $\mathbf{F}_2 - \mathbf{N}_2\ (\mathbf{d}_1, \mathbf{d}_2)$), and therefore when the coupling iterations converge, they converge to the solution of the problem with the correct structural mass.

In fluid–structure interaction computations with light and thin structures (such as membranes), it might be desirable to eliminate the higher spatial modes of the structural response normal to the membrane. We propose to accomplish that by adding to the finite element formulation of the structural mechanics problem a "Directional-Inertia Stabilizing Mass (DISM)" term, which we define as

$$S_{\text{DISM}} = \sum_{e=1}^{n_{el}} \int_{(\Omega^s)^e} \mathbf{w} \cdot (\eta\ \rho^s\ \mathbf{nn}) \cdot \left(\frac{d^2 \mathbf{y}^h}{dt^2} \right) d\Omega^s\ , \tag{26}$$

where Ω^s is the membrane domain, $\mathbf{y}^h$ is the displacement, ρ^s is the material density, $\mathbf{n}$ is the unit vector normal to the membrane, and η is a non-dimensional measure of the curvature in $\left(\frac{d^2 \mathbf{y}^h}{dt^2} \right)$. As a possible alternative to the DISM term, we propose a "Directional-Damping Stabilization (DDS)" term defined as

$$S_{\text{DDS}} = \sum_{e=1}^{n_{el}} \int_{(\Omega^s)^e} \mathbf{w} \cdot \left(\xi^h\ \omega_{\text{int}}^h\ \rho^s\ \mathbf{nn} \right) \cdot \left(\frac{d \mathbf{y}^h}{dt} \right) d\Omega^s\ , \tag{27}$$

where ω_{int}^h is an intrinsic frequency and ξ^h is a non-dimensional measure of the curvature in $\left(\frac{d \mathbf{y}^h}{dt} \right)$. While we view these admittedly ad hoc techniques as short cut stabilization approaches, we also propose the "Fluid–Structure Interactions Mixed Structural Modeling (FSIMSM)" as a more rigorous approach. In FSIMSM, a mixture of different models (such as membrane, shell and continuum elements) would be used for representing the structure, depending on the nature of its deformation modes. For example, parachute computations would normally be based on using membrane and cable elements to model the parachute structure. In the FSIMSM approach, in regions of the structure where wrinkling or other instabilities are experienced or expected, the model would convert to one that is based on using shell and beam elements. This would bring bending rigidity to where it is needed. Appropriate interface (matching) conditions would be used where two different models meet. The FSIMSM can be implemented in a static or dynamic way. In the static way, the model to be used for each structural element would be determined based on what we know about the FSI problem in advance. In the dynamic way, regions experiencing instabilities during the computations would convert to models based on shell and beam elements. Regional models would not be re-defined every time step. They would be re-defined frequently enough to have a safe coverage of the regions experiencing instabilities.

10 THE ENHANCED-DISCRETIZATION SUCCESSIVE UPDATE METHOD (EDSUM)

In this section, we describe a multi-level iteration method for computation of flow behavior at small scales. The EDSUM [2, 16, 4] is based on the EDICT. Although it might be possible

to identify zones where the enhanced discretization could be limited to, we need to think about and develop methods required for cases where the enhanced discretization is needed everywhere in the problem domain to accurately compute flows at smaller scales. In that case the enhanced discretization would be more wide-spread than before, and possibly required for the entire domain. Therefore an efficient solution approach would be needed to solve, at every time step, a very large, coupled nonlinear equation system generated by the multi-level discretization approach.

Such large, coupled nonlinear equation systems involve four classes of nodes. Class-1 consists of all the Mesh-1 nodes. These nodes are connected to each other through the Mesh-1 elements. Class-2E consists of the Mesh-2 edge nodes (but excluding those coinciding with the Mesh-1 nodes). The edge nodes associated with different edges are not connected (except those at each side of an edge, but we could possibly neglect that a side node might be connected to the side nodes of the adjacent edges). Nodes within an edge are connected through Mesh-2 elements. Class-2F contains the Mesh-2 face nodes (but excluding those on the edges). The face nodes associated with different faces are not connected (except those at sides of a face, but we could possibly neglect that those side nodes might be connected to the side nodes of the adjacent face). Nodes within a face are connected through Mesh-2 elements. Class-2I nodes are the Mesh-2 interior nodes. The interior nodes associated with different clusters of Mesh-2 elements are not connected. Nodes within a cluster are connected through Mesh-2 elements.

Based on this multi-level decomposition concept, the nonlinear equation system we generate can be re-written as follows:

$$\begin{aligned}
\mathbf{N}_1\left(\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}\right) &= \mathbf{F}_1,\\
\mathbf{N}_{2E}\left(\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}\right) &= \mathbf{F}_{2E},\\
\mathbf{N}_{2F}\left(\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}\right) &= \mathbf{F}_{2F},\\
\mathbf{N}_{2I}\left(\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}\right) &= \mathbf{F}_{2I}.
\end{aligned} \tag{28}$$

This equation system would be solved with an approximate Newton–Raphson method. At each nonlinear iteration step, we would successively update the solution vectors corresponding to each class. While updating each class, we would use the most recent values of the solution vectors in calculating the vectors $\mathbf{N}_1$, $\mathbf{N}_{2E}$, $\mathbf{N}_{2F}$, and $\mathbf{N}_{2I}$ and their derivatives with respect to the solution vectors. We would start with updating the Class-1 nodes, then update the Class-2E, Class-2F, and Class-2I nodes, respectively. The process, for an iteration step taking us from iterative solution i to $i+1$, is shown below, where each class of equations are solved in the order they are written.

$$\begin{aligned}
\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_1^i\right) &= \mathbf{F}_1 - \mathbf{N}_1\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right),\\
\left.\frac{\partial \mathbf{N}_{2E}}{\partial \mathbf{d}_{2E}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2E}^i\right) &= \mathbf{F}_{2E} - \mathbf{N}_{2E}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right),\\
\left.\frac{\partial \mathbf{N}_{2F}}{\partial \mathbf{d}_{2F}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2F}^i\right) &= \mathbf{F}_{2F} - \mathbf{N}_{2F}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right),\\
\left.\frac{\partial \mathbf{N}_{2I}}{\partial \mathbf{d}_{2I}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2I}^i\right) &= \mathbf{F}_{2I} - \mathbf{N}_{2I}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i\right).
\end{aligned} \tag{29}$$

This sequence would be repeated as many times as needed, and, as an option, we could alternate between this sequence and its reverse sequence (see [16, 4])

Updating the solution vector corresponding to each class would also require solution of a large equation system. These equations systems would each be solved iteratively, with an effective preconditioner, a reliable search technique, and parallel implementation. It is important to note that the bulk of the computational cost would be for Class-1 and Class-2I. While the Class-1 nodes would be partitioned to different processors of the parallel computer, for the remaining classes, nodes in each edge, face or interior cluster would be assigned to the same processor. Therefore, solution of each edge, face or interior cluster would be local. If the size of each interior cluster becomes too large, then nodes for a given cluster can also be distributed across different processors, or a third level of mesh refinement can be introduced to make the enhanced discretization a tri-level kind.

A variation of the EDSUM could be used for the iterative solution of the linear equation system that needs to be solved at every step of a (full) Newton–Raphson method applied to Eq. (28). To describe this variation, we first write the linear equation system that needs to be solved:

$$\begin{aligned}
\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12E}\mathbf{x}_{2E} + \mathbf{A}_{12F}\mathbf{x}_{2F} + \mathbf{A}_{12I}\mathbf{x}_{2I} &= \mathbf{b}_1, \\
\mathbf{A}_{2E1}\mathbf{x}_1 + \mathbf{A}_{2E2E}\mathbf{x}_{2E} + \mathbf{A}_{2E2F}\mathbf{x}_{2F} + \mathbf{A}_{2E2I}\mathbf{x}_{2I} &= \mathbf{b}_{2E}, \\
\mathbf{A}_{2F1}\mathbf{x}_1 + \mathbf{A}_{2F2E}\mathbf{x}_{2E} + \mathbf{A}_{2F2F}\mathbf{x}_{2F} + \mathbf{A}_{2F2I}\mathbf{x}_{2I} &= \mathbf{b}_{2F}, \\
\mathbf{A}_{2I1}\mathbf{x}_1 + \mathbf{A}_{2I2E}\mathbf{x}_{2E} + \mathbf{A}_{2I2F}\mathbf{x}_{2F} + \mathbf{A}_{2I2I}\mathbf{x}_{2I} &= \mathbf{b}_{2I},
\end{aligned} \tag{30}$$

where

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}, \tag{31}$$

with β, $\gamma = 1$, $2E$, $2F$, $2I$. Then, for the iterative solution of Eq. (30), in conjunction with GMRES search [17], we propose that the following three preconditioners are used in sequence during the inner iterations:

$$\mathbf{P}_L = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & 0 \\ 0 & \mathrm{DIAG}\,(\mathbf{A}_{2E2E}) & 0 & 0 \\ 0 & 0 & \mathrm{DIAG}\,(\mathbf{A}_{2F2F}) & 0 \\ 0 & 0 & 0 & \mathrm{DIAG}\,(\mathbf{A}_{2I2I}) \end{bmatrix}, \tag{32}$$

$$\mathbf{P}_{SETOI} = \begin{bmatrix} \mathrm{DIAG}\,(\mathbf{A}_{11}) & 0 & 0 & 0 \\ 0 & \mathbf{A}_{2E2E} & 0 & 0 \\ 0 & \mathbf{A}_{2F2E} & \mathbf{A}_{2F2F} & 0 \\ 0 & \mathbf{A}_{2I2E} & \mathbf{A}_{2I2F} & \mathbf{A}_{2I2I} \end{bmatrix}, \tag{33}$$

$$\mathbf{P}_{SITOE} = \begin{bmatrix} \mathrm{DIAG}\,(\mathbf{A}_{11}) & 0 & 0 & 0 \\ 0 & \mathbf{A}_{2E2E} & \mathbf{A}_{2E2F} & \mathbf{A}_{2E2I} \\ 0 & 0 & \mathbf{A}_{2F2F} & \mathbf{A}_{2F2I} \\ 0 & 0 & 0 & \mathbf{A}_{2I2I} \end{bmatrix}. \tag{34}$$

As possible sequences, we propose ($\mathbf{P}_L$, $\mathbf{P}_{SETOI}$, $\mathbf{P}_{SITOE}$, ..., $\mathbf{P}_L$, $\mathbf{P}_{SETOI}$, $\mathbf{P}_{SITOE}$), as well as ($\mathbf{P}_L$, $\mathbf{P}_{SETOI}$, ..., $\mathbf{P}_L$, $\mathbf{P}_{SETOI}$) and ($\mathbf{P}_L$, $\mathbf{P}_{SITOE}$, ..., $\mathbf{P}_L$, $\mathbf{P}_{SITOE}$). As a somewhat downgraded version of $\mathbf{P}_L$, we can use a preconditioner that is equivalent to not updating

$\mathbf{x}_{2E}$, $\mathbf{x}_{2F}$, and $\mathbf{x}_{2I}$, instead of updating them by using DIAG $(\mathbf{A}_{2E2E})$, DIAG $(\mathbf{A}_{2F2F})$, and DIAG $(\mathbf{A}_{2I2I})$. Similarly, as downgraded versions of $\mathbf{P}_{SETOI}$ and $\mathbf{P}_{SITOE}$, we can use preconditioners that are equivalent to not updating $\mathbf{x}_1$, instead of updating it by using DIAG $(\mathbf{A}_{11})$. For additional preconditioners see [16, 4].

To differentiate between the two variations of the EDSUM we described in this section, we call the nonlinear version, described by Eq. (29), EDSUM-N, and the linear version, described by Eqs. (30) – (34), EDSUM-L.

The EDSUM provides a natural framework for resourceful and selective application of stabilized formulations to multi-scale computations and subgrid-scale modeling. Along these lines we propose the "Enhanced-Discretization Selective Stabilization Procedure (EDSSP)". In the EDSSP, finite element equations generating different blocks of the nonlinear equation system given by Eq. (28) would be based on different stabilized formulations. Level-1 equations (generating the first block) would be based on a stabilized formulation more suitable for flow behavior at larger scales, and the Level-2 equations (generating the second, third and fourth blocks) would be based on a stabilized formulation more suitable for flow behavior at smaller scales. As a special version of the EDSSP, we propose to use with the Level-1 equations only the SUPG and PSPG stabilizations, and with the Level-2 equations use additionally the DCDD stabilization [11, 4].

We propose the EDSUM-B as an approximate version of the EDSUM. In the EDSUM-B, we propose to solve the Level-2 equations less frequently than the Level-1 equations. For example, instead of performing the same number of iterations to solve all four blocks of Eq. (28), we can perform one iteration for the Level-2 blocks for every two iterations we perform for the Level-1 block. This would mean that the small-scale data used in solving the large-scale equations is updated at every other iteration. As another example of EDSUM-B, we can use for the Level-2 equations a more dissipative time-integration algorithm and a time-step size that is twice the time-step size we use for the Level-1 equations. This would mean that the small-scale data used in solving the large-scale equations is updated at every other time step. Although the EDSUM-B is a more approximate technique compared to the EDSUM, it will still be superior in accuracy compared to carrying out the computation with the Level-1 discretization alone.

11 EXAMPLES OF FLOW SIMULATIONS AND TEST COMPUTATIONS

11.1 Aerodynamic interactions of two parachutes

Two US Army T–10 parachutes, at descent speeds of 22 ft/s, are undergoing aerodynamic interactions as one enters the wake of the other one. The reference frame is attached to the lower parachute, which is assumed to be rigid. The upper parachute is allowed to move relative to the lower one and deform. The DSD/SST formulation is used with the automatic mesh moving method. In this test computation the $\frac{\partial \mathbf{w}^h}{\partial t}$ term in Eq. (8) has been dropped. The motion and deformation of the upper parachute is governed by the membrane and cable equations, which are solved together with the fluid mechanics equations. Figure 1 shows, during time period 0.0 to 3.5 s, the vorticity field and how the upper parachute moves closer to the lower one and deforms. For more on this computation and the fluid–structure interaction technique used, see [18, 19].

11.2 Parachute soft-landing dynamics

Soft landing with the aid of a retraction device reduces the landing impact for payloads delivered with parachutes. In this example, for a T–10 parachute, a pneumatic muscle actuator

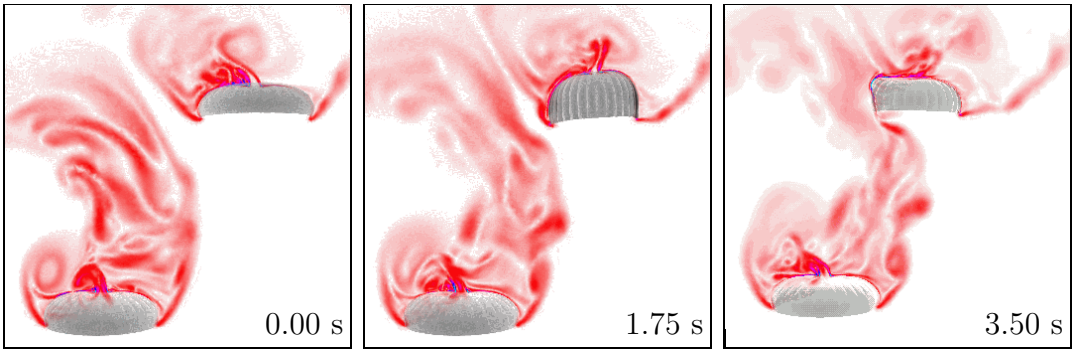

Figure 1: Aerodynamic interactions of two parachutes. Vorticity during time period 0.0 to 3.5 s. For more on this computation see [18, 19].

(PMA) placed between the suspension lines and the payload serves at the retraction device by causing rapid contraction just before landing. The DSD/SST formulation is used with the automatic mesh moving method. In this test computation the $\frac{\partial \mathbf{w}^h}{\partial t}$ term in Eq. (8) has been dropped. The motion and deformation of the parachute is governed by the membrane and cable equations, which are solved together with the fluid mechanics equations. Figure 2 shows the payload trajectory. For more on soft-landing simulations and the methods used,

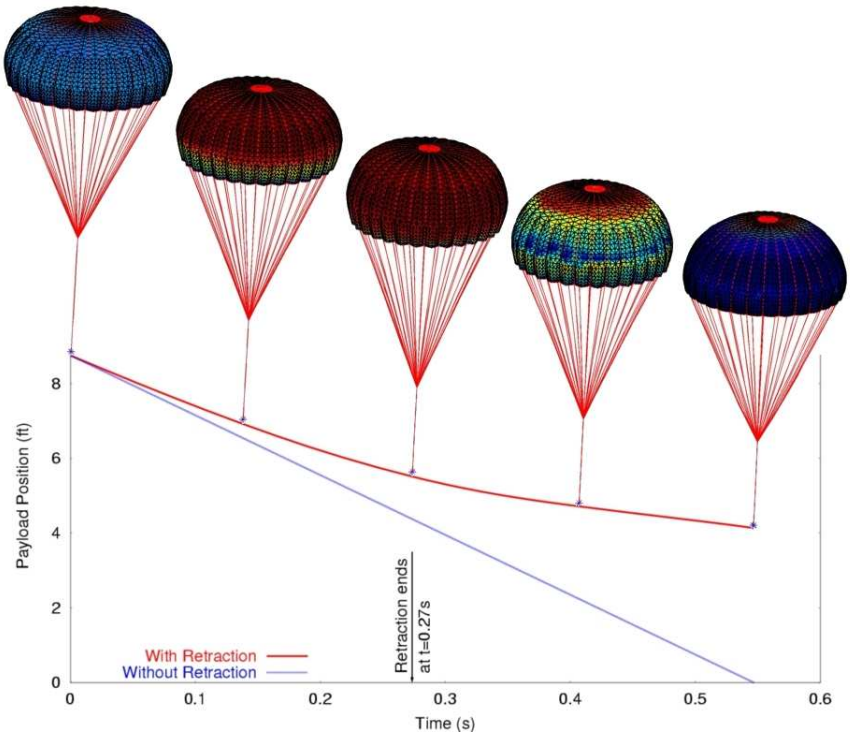

Figure 2: Parachute soft-landing dynamics. Payload trajectory for soft landing of a T–10 parachute during and immediately after retraction. The straight line is the trajectory that the payload would have had without the retraction. The parachutes displayed illustrate the deformations of the canopy and the cables. The length scale used in displaying the parachutes is not the same as it is for the trajectory graph. For more on this simulation, see [20].

see [20].

11.3 Performance evaluation of the Standard and EDSUM function spaces for a steady advection problem

In this 2D test problem, EDICT and non-EDICT versions of the stabilized finite element formulation of an advection equation are used for computations based on standard-discretization ("Standard") and enhanced-discretization ("EDSUM"), respectively. The objective is to investigate if the EDSUM function space is inherently superior in iterative computations to the Standard function space. The domain is a square and the discretization has 97×97 nodes. For EDSUM computations the enhanced-discretization zone covers the entire domain, and the 97×97 discretization is achieved by the combination of Mesh-1 with 25×25 nodes and Mesh-2 with 97×97 nodes. We use diagonal preconditioners with both function spaces, and keep the number of inner and outer GMRES iterations the same. An essential boundary condition in the form of a cosine hill imposed at an internal line. We perform 20 inner GMRES iterations (per outer GMRES iteration), and compare the Standard and EDSUM solutions at the end of the 4th, 5th, and 6th outer GMRES iterations. The solutions are shown in Figure 3. We can clearly see that the EDSUM convergence is superior to the Standard convergence. For more on this test computation, other test computations with EDSUM, and tests with different preconditioners, see [21].

12 CONCLUDING REMARKS

We highlighted some of techniques we developed to address the challenges involved in computation of flows with moving boundaries and interfaces. This category of problems include fluid–particle, fluid–object and fluid–structure interactions; free-surface and two-fluid flows; and flows with moving mechanical components. The methods we developed can be grouped into two classes: interface-tracking and interface-capturing techniques. Both classes of techniques are based on stabilized formulations. The core method in the interface-tracking approach is the DSD/SST formulation, where the mesh moves to track the interface. The core method for the interface-capturing techniques is the SUPG/PSPG formulation of both the flow equations and the advection equation governing the time-evolution of the interface function. We also described some of the additional methods we developed to increase the scope and accuracy of these two classes of techniques. With numerical examples and test computations, we showed how some of the techniques described work.

Acknowledgments: This work was supported by the US Army Natick Soldier Center, NASA JSC, and National Science Foundation.

REFERENCES

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

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

[3] T. Tezduyar, "Interface-tracking and interface-capturing techniques for computation of moving boundaries and interfaces", in *Proceedings of the Fifth World Congress on Computational Mechanics*, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81513, Vienna, Austria, (2002).

[4] T.E. Tezduyar, "Finite element methods for fluid dynamics with moving boundaries and interfaces", in E. Stein, R. De Borst, and T.J.R. Hughes, editors, *Encyclopedia of Computational Mechanics*, Volume 3: Fluids, Chapter 17, John Wiley & Sons, 2004.

[5] T.J.R. Hughes and A.N. Brooks, "A multi-dimensional upwind scheme with no crosswind diffusion", in T.J.R. Hughes, editor, *Finite Element Methods for Convection Dominated Flows*, AMD-Vol.34, 19–35, ASME, New York, 1979.

[6] T.E. Tezduyar and T.J.R. Hughes, "Finite element formulations for convection dominated flows with particular emphasis on the compressible Euler equations", in *Proceedings of AIAA 21st Aerospace Sciences Meeting*, AIAA Paper 83-0125, Reno, Nevada, (1983).

[7] T.J.R. Hughes, L.P. Franca, and M. Balestra, "A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška–Brezzi condition: A stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations", *Computer Methods in Applied Mechanics and Engineering*, **59** (1986) 85–99.

[8] 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 *New Methods in Transient Analysis*, PVP-Vol.246/AMD-Vol.143, ASME, New York, (1992) 7–24.

[9] T.J.R. Hughes and G.M. Hulbert, "Space–time finite element methods for elastodynamics: formulations and error estimates", *Computer Methods in Applied Mechanics and Engineering*, **66** (1988) 339–363.

[10] T.E. Tezduyar and Y. Osawa, "Finite element stabilization parameters computed from element matrices and vectors", *Computer Methods in Applied Mechanics and Engineering*, **190** (2000) 411–430.

[11] T.E. Tezduyar, "Computation of moving boundaries and interfaces and stabilization parameters", *International Journal for Numerical Methods in Fluids*, **43** (2003) 555–575.

[12] T. 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.

[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] H.J.C. Barbosa and T.J.R. Hughes, "The finite element method with Lagrange multipliers on the boundary: Circumventing the Babuska–Brezzi condition", *Computer Methods in Applied Mechanics and Engineering*, **85** (1991) 109–128.

[15] H.J.C. Barbosa and T.J.R. Hughes, "Circumventing the Babuska–Brezzi condition in mixed finite element approximations of elliptic variational inequalities", *Computer Methods in Applied Mechanics and Engineering*, **97** (1992) 193–210.

[16] T.E. Tezduyar, "Interface-tracking, interface-capturing and enhanced solution techniques", in *Proceedings of the First South-American Congress on Computational Mechanics (CD-ROM)*, Santa Fe–Parana, Argentina, (2002).

[17] Y. Saad and M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM Journal of Scientific and Statistical Computing*, **7** (1986) 856–869.

[18] K. Stein, T. Tezduyar, V. Kumar, S. Sathe, R. Benney, E. Thornburg, C. Kyle, and T. Nonoshita, "Aerodynamic interactions between parachute canopies", *Journal of Applied Mechanics*, **70** (2003) 50–57.

[19] K. Stein, T. Tezduyar, and R. Benney, "Computational methods for modeling parachute systems", *Computing in Science and Engineering*, **5** (2003) 39–46.

[20] T.E. Tezduyar, S. Sathe, R. Keedy, and K. Stein, "Space–time techniques for finite element computation of flows with moving boundaries and interfaces", in S. Gallegos, I. Herrera, S. Botello, F. Zarate, and G. Ayala, editors, *Proceedings of the III International Congress on Numerical Methods in Engineering and Applied Science*, CD-ROM, 2004.

[21] T.E. Tezduyar and S. Sathe, "Enhanced-discretization successive update method (EDSUM)", *International Journal for Numerical Methods in Fluids*, **47** (2005) 633–654.

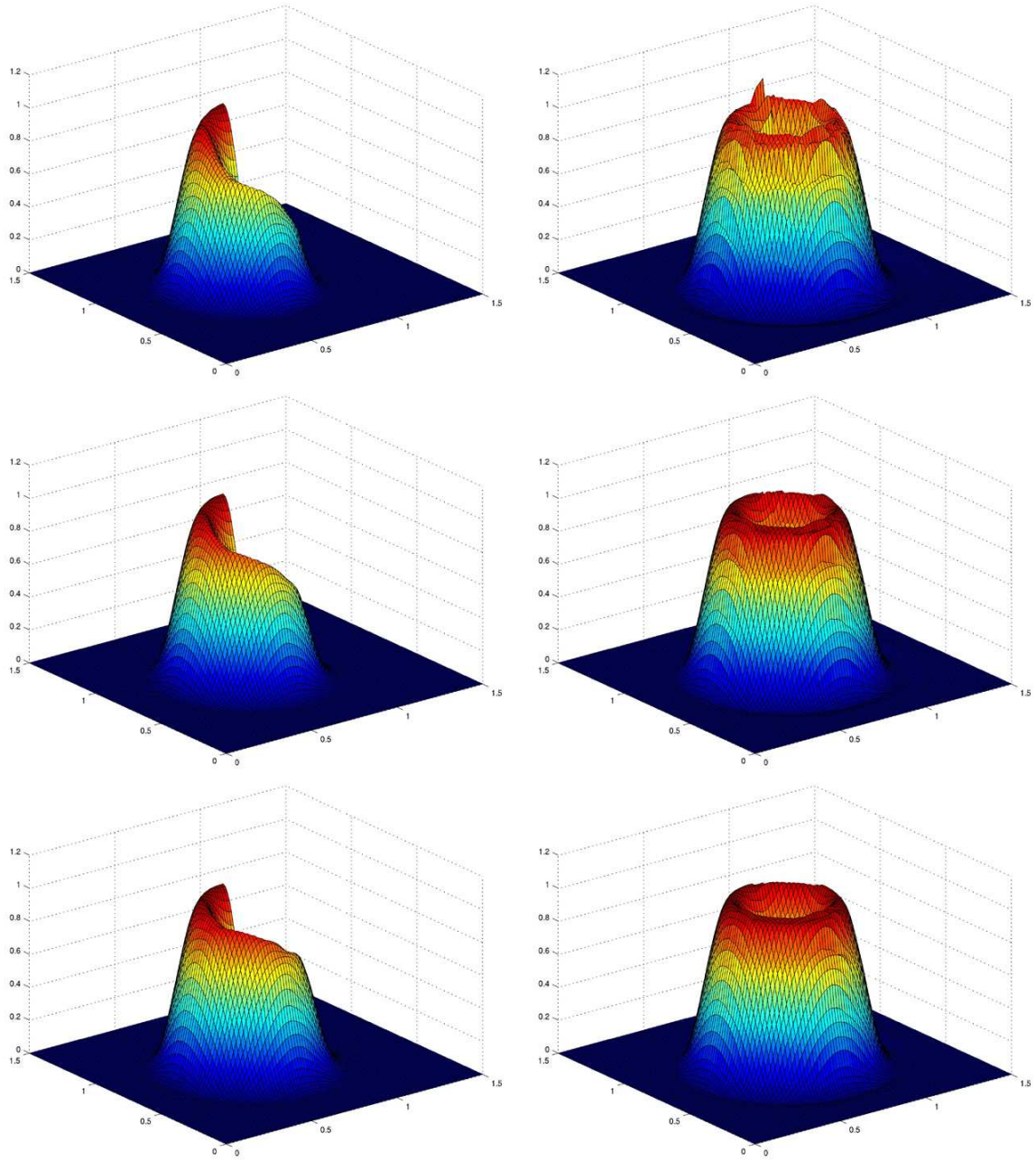

Figure 3: Performance evaluation of the Standard and EDSUM function spaces for a steady advection problem. Solutions obtained with the Standard (left) and EDSUM (right) function spaces, after 4 (top), 5 (middle), and 6 (bottom) outer GMRES iterations. For more on this test computation, see [21].