Mecánica Computacional Vol. XXI, pp. 116–135
S. R. Idelsohn, V. E. Sonzogni and A. Cardona (Eds.)
Santa Fe–Paraná, Argentina, October 2002

INTERFACE-TRACKING, INTERFACE-CAPTURING AND ENHANCED SOLUTION TECHNIQUES

Tayfun E. Tezduyar*

*Team for Advanced Flow Simulation and Modeling (TAFSM)
Mechanical Engineering, Rice University – MS 321
6100 Main Street, Houston, TX 77005, USA
Email: tezduyar@rice.edu, Web page: http://www.mems.rice.edu/tezduyar/

Key Words: Interface-tracking, Interface-capturing, Enhanced discretization

Abstract. *We describe the interface-tracking and interface-capturing techniques we developed in recent years for computation of flows with moving boundaries and 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 were developed for two-fluid flows. They are based on the stabilized formulation, over typically non-moving meshes, of both the flow equations and an advection equation. The advection equation governs the time-evolution of an interface function marking the interface location. We also describe some of the additional methods developed to increase the scope and accuracy of these two classes of techniques. Among them are the Enhanced-Discretization Interface-Capturing Technique (EDICT), which was developed to increase the accuracy in capturing the interface, and extensions and offshoots of the EDICT.*

1 INTRODUCTION

In computation of flow problems with moving boundaries and 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 interface-tracking and interface-capturing techniques we have developed in recent years (see[1–6]) are based on stabilized formulations. The stabilized methods are the streamline-upwind/Petrov-Galerkin (SUPG),[7] Galerkin/least-squares (GLS),[8] and pressure-stabilizing/Petrov-Galerkin (PSPG)[1] formulations. They 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. Furthermore, this class of stabilized formulations substantially improve the convergence rate in iterative solution of the large, coupled nonlinear equation system that needs to be solved at every time step of a flow computation. Such nonlinear systems are typically solved with the Newton-Raphson method, which involves, at its every iteration step, solution of a large, coupled linear equation system. It is in iterative solution of such linear equation systems that using a good stabilized method makes substantial difference in convergence, and this was pointed out in.[9]

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. 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 be-

ing 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 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[10,11] 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 extensions and offshoots of EDICT.

2 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. The Navier-Stokes equations of incompressible flows can be written on Ω_t and $\forall t \in (0,T)$ as

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

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

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

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

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:

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

The essential and natural boundary conditions for Eq. (1) are represented as

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

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 $\mathbf{u}_0(\mathbf{x})$ 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. 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 $\phi = \{1 \text{ for Fluid A and } 0 \text{ for Fluid B}\}$. The interface between the two fluids is approximated to be at $\phi = 0.5$. In this context, ρ and μ are defined as

$$\rho = \phi\rho_A + (1-\phi)\rho_B, \quad \mu = \phi\mu_A + (1-\phi)\mu_B. \tag{6}$$

The evolution of the interface function ϕ, and therefore the motion of the interface, is governed by a time-dependent advection equation, written on Ω and $\forall t \in (0,T)$ as

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\phi = 0. \tag{7}$$

3 STABILIZED FORMULATIONS

3.1 Advection-Diffusion Equation

Let us consider over a domain Ω with boundary Γ the following time-dependent advection-diffusion equation, written on Ω and $\forall t \in (0,T)$ as

$$\frac{\partial \phi}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\phi - \boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}\phi) = 0, \tag{8}$$

where ϕ represents the quantity being transported (e.g., temperature, concentration, interface function), and ν is the diffusivity. The essential and natural boundary conditions associated with Eq. (8) are represented as

$$\phi = \mathrm{g} \text{ on } \Gamma_{\mathrm{g}}, \qquad \mathbf{n}\cdot\nu\boldsymbol{\nabla}\phi = \mathrm{h} \text{ on } \Gamma_{\mathrm{h}}, \tag{9}$$

A function $\phi_0(\mathbf{x})$ is specified as the initial condition.

Let us assume that we have constructed some suitably-defined finite-dimensional trial solution and test function spaces $\mathcal{S}_\phi^h$ and $\mathcal{V}_\phi^h$. The stabilized finite element formulation of Eq. (8) can then be written as follows: find $\phi^h \in \mathcal{S}_\phi^h$ such that $\forall w^h \in \mathcal{V}_\phi^h$:

$$\begin{aligned}
&\int_\Omega w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\phi^h\right) d\Omega + \int_\Omega \boldsymbol{\nabla} w^h \cdot \nu\boldsymbol{\nabla}\phi^h d\Omega - \int_{\Gamma_{\mathrm{h}}} w^h \mathrm{h}^h d\Gamma \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\mathrm{SUPG}}\mathbf{u}^h\cdot\boldsymbol{\nabla} w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\phi^h - \boldsymbol{\nabla}\cdot\left(\nu\boldsymbol{\nabla}\phi^h\right)\right) d\Omega = 0.
\end{aligned} \tag{10}$$

Here n_{el} is the number of elements, Ω^e is the domain for element e, and τ_{SUPG} is the SUPG stabilization parameter. For various ways of calculating τ_{SUPG}, see.[4,12–15]

3.2 Navier-Stokes Equations of Incompressible Flows

Given Eqs. (1)-(2), let us assume that we have 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 then 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 $q^h \in \mathcal{V}_p^h$:

$$\begin{aligned}
&\int_\Omega \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega - \int_{\Gamma_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h d\Gamma \\
+ \quad &\int_\Omega q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h + \tau_{\mathrm{PSPG}} \nabla q^h] \cdot \\
&\qquad\qquad \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) - \rho \mathbf{f} \right] d\Omega \\
+ \quad &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega \;=\; 0. \qquad (11)
\end{aligned}$$

Here τ_{PSPG} and τ_{LSIC} are the PSPG and LSIC (least-squares on incompressibility constraint) stabilization parameters. For various ways of calculating τ_{SUPG}, τ_{PSPG} and τ_{LSIC}, see.[4,12–15]

4 DSD/SST FINITE ELEMENT 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 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. 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 Dirichlet- and Neumann-type boundary conditions are enforced over $(P_n)_{\mathrm{g}}$ and $(P_n)_{\mathrm{h}}$, 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[1,14,15] 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 $q^h \in (\mathcal{V}^h_p)_n$:

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

where

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

and τ_{SUPG}, τ_{PSPG} and τ_{LSIC} are the stabilization parameters (see[14,15]). 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 DSD/SST formulation see.[1]

5 MESH UPDATE METHODS

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. 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, reducing the cost of automatic mesh generation becomes a major incentive 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.[9]

In general, however, we use an automatic mesh moving scheme[16] 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 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[17]). In our mesh moving method based on linear elasticity, the structured layers of elements generated around solid objects (to fully control the mesh resolution near solid objects and have more accurate representation of the boundary layers) 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.[9] Earlier examples of element layers undergoing rigid-body motion, in combination with deforming structured meshes, can be found in.[1]

In computation of flows with fluid-solid interfaces where the solid is deforming, the motion of the fluid mesh near the interface cannot be represented by a rigid-body motion. Depending on the deformation mode of the solid, we may have to use the automatic mesh moving technique described above. In such cases, presence of very thin fluid elements near the solid surface creates a challenge for the automatic mesh moving technique. In the Solid-Extension Mesh Moving Technique (SEMMT),[3,4,6] we propose to treat those thin fluid elements 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. For mesh moving studies with the SEMMT, see.[18]

6 EDICT

In the EDICT (Enhanced-Discretization Interface-Capturing Technique),[10,11] we start with the basic approach of an interface-capturing technique such as the volume of fluid (VOF) method.[19] 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[11]), 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 formula-

tion. 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 momentum equation, incompressibility constraint and time-dependent advection equation are denoted by $(\mathcal{V}^h_{\mathbf{u}})_n$, $(\mathcal{V}^h_p)_n$ $(= (\mathcal{S}^h_p)_n)$, and $(\mathcal{V}^h_\phi)_n$. 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 $\mathbf{u}^h_n$ and ϕ^h_n, find $\mathbf{u}^h_{n+1} \in (\mathcal{S}^h_{\mathbf{u}})_{n+1}$, $p^h_{n+1} \in (\mathcal{S}^h_p)_{n+1}$, and $\phi^h_{n+1} \in (\mathcal{S}^h_\phi)_{n+1}$, such that, $\forall \mathbf{w}^h_{n+1} \in (\mathcal{V}^h_{\mathbf{u}})_{n+1}$, $\forall q^h_{n+1} \in (\mathcal{V}^h_p)_{n+1}$, and $\forall \psi^h_{n+1} \in (\mathcal{V}^h_\phi)_{n+1}$:

$$
\begin{aligned}
&\int_\Omega \mathbf{w}^h_{n+1} \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h_{n+1}) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega \\
- &\int_{\Gamma_{\mathrm{h}}} \mathbf{w}^h_{n+1} \cdot \mathbf{h}^h d\Gamma + \int_\Omega q^h_{n+1} \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h_{n+1} + \tau_{\mathrm{PSPG}} \nabla q^h_{n+1}] \cdot \left[\mathrm{L}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h_{n+1} \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega = 0,
\end{aligned}
\tag{14}
$$

$$
\begin{aligned}
&\int_\Omega \psi^h_{n+1} \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega \\
+ &\sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_\phi \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi^h_{n+1} \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega = 0.
\end{aligned}
\tag{15}
$$

Here τ_{SUPG}, τ_{PSPG} and τ_ϕ are the stabilization parameters, and τ_ϕ is calculated by applying the definition of τ_{SUPG} to Eq. (15). For ways of calculating τ_{SUPG} and τ_{PSPG}, see.[4,12–15]

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.

7 EXTENSIONS AND OFFSHOOTS OF EDICT

An offshoot of EDICT was first reported in[20] 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.[21,22] In this case, the definition of the interface is extended to mean regions where the vorticity magnitude is larger than a specified value.

Here we describe how we extend EDICT to computation of flow problems with boundary layers. In this offshoot, 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),[3,4,6] we propose two different ways of using the EDICT concept to increase the mesh refinement in the boundary layers in a desirable fashion. In the EDICT-Clustered-Mesh-2 approach,[2–4,6] 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,[2–4,6] 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),[3–6] we propose to use enhanced time-discretization in the context of a space-time formulation. The motivation 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. There would be two ways of formulating EDSTT. In the EDSTT-Single-Mesh (EDSTT-SM) approach,[3–6] 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,[3–6] 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 certain parts of the fluid domain. For example, where the spatial element sizes are small, we may need to use small time steps, so that the element Courant number does not become too large. In computation of two-fluid interface (or free-surface) flows with the DSD/SST method, time-integration of the equation governing the evolution of the interface (i.e. the interface update equation) may require a smaller time step than the one used for the fluid interiors. This requirement might be coming from numerical stability considerations, when time-integration of the interface update equation does not involve any added stabilization terms. In such cases, time-integration with sub-time-stepping on the interface update equation can be based on the EDSTT-SM or EDSTT-MM approaches. As an alternative or complement to these approaches, the sub-time-stepping on the interface update equation can be accomplished with the Interpolated Sub-Time-Stepping Technique (ISTST).

In the ISTST, time-integration of the interface update equation with smaller time steps would be carried out separately from the fluid interiors. The information between the two parts would be exchanged by interpolation. The sub-time-stepping sequence for the interface update, together with the interpolations between the interface and fluid interiors, would be embedded in the iterative solution technique used for the fluid interiors, and would be repeated at every iteration. The iterative solution technique, which is based on the Newton-Raphson method, addresses both the nonlinear and the coupled nature of the set of equations that needs to be solved at each time step of the time-integration of the fluid interiors. When the ISTST is applied to computation of fluid-structure interactions, a separate, "inner" Newton-Raphson sequence would be used at each time step of the sub-time-stepping on the structural dynamics equations.

8 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. Whether we are using a space-time formulation or a semi-discrete formulation, we can represent the equation system that needs to be solved as follows:

$$\mathbf{N}\left(\mathbf{d}_{n+1}\right) = \mathbf{F}. \tag{16}$$

Here $\mathbf{d}_{n+1}$ is the vector of nodal unknowns. In a semi-discrete formulation, this vector contains the unknowns associated with marching from time level n to $n+1$. In a space-time formulation, it contains the unknowns associated with the finite element formulation written for the space-time slab Q_n. The time-marching formulations described earlier

can also be used for computing a steady-state flow. In such cases time does not have a physical significance, but is only used in time-marching to the steady-state solution.

We solve Eq. (16) with the Newton-Raphson method:

$$\left.\frac{\partial \mathbf{N}}{\partial \mathbf{d}}\right|_{\mathbf{d}^i_{n+1}} \left(\Delta \mathbf{d}^i_{n+1}\right) = \mathbf{F} - \mathbf{N}\left(\mathbf{d}^i_{n+1}\right), \tag{17}$$

where i is the step counter for the Newton-Raphson sequence, and $\Delta \mathbf{d}^i_{n+1}$ is the increment computed for $\mathbf{d}^i_{n+1}$. The linear equation system represented by Eq. (17) needs to be solved at every step of the Newton-Raphson sequence. We can represent Eq. (17) as a linear equation system of the form

$$\mathbf{A}\mathbf{x} \;=\; \mathbf{b}. \tag{18}$$

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. At each iteration, we need to compute the residual of this system:

$$\mathbf{r} \;=\; \mathbf{b} - \mathbf{A}\mathbf{x}. \tag{19}$$

This can be achieved in several different ways. The computation can be based on a sparse-matrix storage of $\mathbf{A}$. It can also be based on storing just element-level matrices (element-matrix-based), or even just element-level vectors (element-vector-based). This last strategy is also called the matrix-free technique. After the residual computation, we compute a candidate correction to $\mathbf{x}$ as given by the expression

$$\Delta \mathbf{y} = \mathbf{P}^{-1}\mathbf{r}, \tag{20}$$

where $\mathbf{P}$, the preconditioning matrix, is an approximation to $\mathbf{A}$. $\mathbf{P}$ has to be simple enough to form and factorize efficiently. However, it also has to be sophisticated enough to yield a desirable convergence rate. How to update the solution vector $\mathbf{x}$ by using $\Delta \mathbf{y}$ is also a major subject in iterative solution techniques. Several update methods are available, and we use the GMRES[23] method. We have been focusing our research related to iterative methods mainly on computing the residual $\mathbf{r}$ efficiently and selecting a good preconditioner $\mathbf{P}$. While moving in this direction, we have always been keeping in mind that the iterative solution methods we develop need to be efficiently implemented on parallel computing platforms. For example, the "parallel-ready" methods we designed for the residual computations include those that are element-matrix-based,[24] element-vector-based,[24] and sparse-matrix-based.[25] The element-vector-based methods were successfully used also by other researchers in the context of parallel computations (see for example[26,27]).

In preconditioning design, we developed some advanced preconditioners such as the Clustered-Element-by-Element (CEBE) preconditioner[28] and the mixed CEBE and Cluster Companion (CC) preconditioner.[28] We have implemented, with quite satisfactory results, the CEBE preconditioner in conjunction with an ILU approximation.[25] However,

our typical computations are based on diagonal and nodal-block-diagonal preconditioners. These are very simple preconditioners, but are also very simple to implement on parallel platforms. More on our parallel implementations can be found in.[24]

9 ENHANCED SOLUTION TECHNIQUES

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 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),[3–6] 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. In time-accurate computations of fluid-structure interactions with the EDSTT-SM or EDSTT-MM approaches, the EINST can be used to allow for a larger number of nonlinear iterations for the structure.

In some challenging cases, using a diagonal or nodal-block-diagonal preconditioners 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),[3,4,6] 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. This particular EALST can be summarized by first expressing the assembly process for $\mathbf{A}$ and $\mathbf{P}$ as

$$\mathbf{A} = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{A}^e \tag{21}$$

$$\mathbf{P} = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{P}^e \ , \tag{22}$$

where A is the finite element assembly operator, and then defining $\mathbf{P}^e$ for the elements belonging to the enhanced-approximation and diagonal-approximation groups:

$$\mathbf{P}^e = \begin{cases} \mathbf{A}^e & \textit{for Enhanced Approximation Group} \\ \mathrm{DIAG}(\mathbf{A}^e) & \textit{for Diagonal Approximation Group} \end{cases}, \tag{23}$$

where DIAG represents a diagonal or nodal-block-diagonal approximation operator. We note that in factorizing the submatrices of $\mathbf{P}$ corresponding to the enhanced-approximation group, we can use a direct solution method, or, as an alternative, a second-level iteration sequence. This second-level iteration sequence would have its own preconditioner (possibly a diagonal or nodal-block-diagonal preconditioner) and its own GMRES vector space (possibly shorter than the GMRES vector space used in the first-level iterations).

10 MIXED ELEMENT-MATRIX-BASED/ELEMENT-VECTOR-BASED COMPUTATION TECHNIQUE (MMVCT)

Consider a nonlinear equation system of the kind given by Eq. (16), re-written in the following form:

$$\begin{aligned} \mathbf{N}_1\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_1, \\ \mathbf{N}_2\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_2, \end{aligned} \tag{24}$$

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. Similarly, we re-write Eq. (18) in the form

$$\begin{aligned} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 &= \mathbf{b}_1, \\ \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 &= \mathbf{b}_2, \end{aligned} \tag{25}$$

where

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}. \tag{26}$$

Re-writing Eqs. (16) and (18) in this fashion would help us recognize or investigate the properties associated with the individual blocks of the equation system. It would also help us explore selective treatment of these blocks during the solution process. 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. We would then recognize that computation of the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ poses a significant difficulty and therefore alternative approaches should be explored.

Iterative solution of Eq. (25) can be written as

$$\begin{aligned} \mathbf{P}_{11}\Delta\mathbf{y}_1 + \mathbf{P}_{12}\Delta\mathbf{y}_2 &= \mathbf{b}_1 - \left(\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2\right), \\ \mathbf{P}_{21}\Delta\mathbf{y}_1 + \mathbf{P}_{22}\Delta\mathbf{y}_2 &= \mathbf{b}_2 - \left(\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2\right), \end{aligned} \tag{27}$$

where $\mathbf{P}_{\beta\gamma}$'s represent the blocks of the preconditioning matrix $\mathbf{P}$. Here we focus our attention to computation of the residual vectors on the right hand side, and explore alternative ways for evaluating the matrix-vector products.

Let us suppose that we are able to compute, without a major difficulty, the element-level matrices $\mathbf{A}^e_{11}$ and $\mathbf{A}^e_{22}$ associated with the global matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$, and that we prefer to evaluate $\mathbf{A}_{11}\mathbf{x}_1$ and $\mathbf{A}_{22}\mathbf{x}_2$ by using these element-level matrices. Let us also suppose that calculation of the element-level matrices $\mathbf{A}^e_{12}$ and $\mathbf{A}^e_{21}$ is prohibitively difficult. Reflecting these circumstances, the computations can be carried out by using a mixed element-matrix-based/element-vector-based computation technique:[2]

$$\begin{aligned}
(\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2) &= \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{11}\mathbf{x}_1) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_1 \to 0} \left[\frac{\mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2 + \epsilon_1\mathbf{x}_2) - \mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_1} \right], \\
(\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2) &= \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{22}\mathbf{x}_2) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_2 \to 0} \left[\frac{\mathbf{N}^e_2(\mathbf{d}_1 + \epsilon_2\mathbf{x}_1, \mathbf{d}_2) - \mathbf{N}^e_2(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_2} \right], \qquad (28)
\end{aligned}$$

where ϵ_1 and ϵ_2 are small parameters used in numerical evaluation of the directional derivatives. Here, $\mathbf{A}_{11}\mathbf{x}_1$ and $\mathbf{A}_{22}\mathbf{x}_2$ are evaluated with an element-matrix-based computation technique, while $\mathbf{A}_{12}\mathbf{x}_2$ and $\mathbf{A}_{21}\mathbf{x}_1$ are evaluated with an element-vector-based computation technique.

In extending the mixed element-matrix-based/element-vector-based computation technique described above to a more general framework, evaluation of a matrix-vector product $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ (for $\beta, \gamma = 1, 2, \ldots, N$ and no sum) appearing in a residual vector can be formulated as an intentional choice between the following element-matrix-based and element-vector-based computation techniques:

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{\beta\gamma}\mathbf{x}_\gamma), \qquad (29)$$

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_\beta \to 0} \left[\frac{\mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma + \epsilon_\beta\mathbf{x}_\gamma, \ldots) - \mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma, \ldots)}{\epsilon_\beta} \right]. \qquad (30)$$

Sometimes, computation of the element-level matrices $\mathbf{A}^e_{\beta\gamma}$ might not be prohibitively difficult, but we might still prefer to evaluate $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ with an element-vector-based computation technique. In such cases, instead of an element-vector-based computation technique requiring numerical evaluation of directional derivatives, we might want to use the element-vector-based computation technique described below.

Let us suppose that the nonlinear vector function $\mathbf{N}_\beta$ corresponds to a finite element integral form $\mathbf{B}_\beta(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N)$. Here $\mathbf{W}_\beta$ represents the vector of nodal values associated with the weighting function $\mathbf{w}_\beta$, which generates the nonlinear equation block β. Let us also suppose that we are able to, without a major difficulty, derive the first-order terms in the expansion of $\mathbf{B}_\beta(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N)$ in $\mathbf{u}_\gamma$. Let the finite element integral form $\mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \Delta\mathbf{u}_\gamma)$ represents those first-order terms in $\Delta\mathbf{u}_\gamma$. We note that this

finite element integral form will generate $\frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}$. Consequently, the matrix-vector product $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ can be evaluated as[2]

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}\mathbf{x}_\gamma \quad = \quad \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_\gamma), \tag{31}$$

where, $\mathbf{v}_\gamma$ is a function interpolated from $\mathbf{x}_\gamma$ in the same way that $\mathbf{u}_\gamma$ is interpolated from $\mathbf{d}_\gamma$. This element-vector-based computation technique allows us to evaluate matrix-vector products without dealing with numerical evaluation of directional derivatives. To differentiate between the element-vector-based computation techniques defined by Eqs. (30) and (31), we will call them, respectively, numerical element-vector-based (NEVB) and analytical element-vector-based (AEVB) computation techniques.

11 EDSUM

In this section, we describe a multi-level iteration method for computation of flow behavior at small scales. The Enhanced- Discretization Successive Update Method (EDSUM)[2] is based on the Enhanced-Discretization Interface-Capturing Technique (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, a nonlinear equation system of the

kind given by Eq. (16) can be re-written as follows:

$$\begin{aligned}
\mathbf{N}_1\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_1, \\
\mathbf{N}_{2E}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2E}, \\
\mathbf{N}_{2F}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2F}, \\
\mathbf{N}_{2I}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2I},
\end{aligned} \tag{32}$$

where the subscript "$n+1$" has been dropped to simplify the notation.

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 is shown below, where each class of equations are solved in the order they are written.

$$\begin{aligned}
\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_1^i\right) &= \mathbf{F}_1 - \mathbf{N}_1\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right), \\
\left.\frac{\partial \mathbf{N}_{2E}}{\partial \mathbf{d}_{2E}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2E}^i\right) &= \mathbf{F}_{2E} - \mathbf{N}_{2E}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right), \\
\left.\frac{\partial \mathbf{N}_{2F}}{\partial \mathbf{d}_{2F}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2F}^i\right) &= \mathbf{F}_{2F} - \mathbf{N}_{2F}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right), \\
\left.\frac{\partial \mathbf{N}_{2I}}{\partial \mathbf{d}_{2I}}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2I}^i\right) &= \mathbf{F}_{2I} - \mathbf{N}_{2I}\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^i\right).
\end{aligned} \tag{33}$$

This sequence would be repeated as many times as needed, and, as an option, we could alternate between this sequence and its reverse sequence:

$$\begin{aligned}
\left.\frac{\partial \mathbf{N}_{2I}}{\partial \mathbf{d}_{2I}}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right)} \left(\Delta \mathbf{d}_{2I}^i\right) &= \mathbf{F}_{2I} - \mathbf{N}_{2I}\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^i\right), \\
\left.\frac{\partial \mathbf{N}_{2F}}{\partial \mathbf{d}_{2F}}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^{i+1}\right)} \left(\Delta \mathbf{d}_{2F}^i\right) &= \mathbf{F}_{2F} - \mathbf{N}_{2F}\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^i,\ \mathbf{d}_{2I}^{i+1}\right), \\
\left.\frac{\partial \mathbf{N}_{2E}}{\partial \mathbf{d}_{2E}}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1}\right)} \left(\Delta \mathbf{d}_{2E}^i\right) &= \mathbf{F}_{2E} - \mathbf{N}_{2E}\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^i,\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1}\right), \\
\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1}\right)} \left(\Delta \mathbf{d}_1^i\right) &= \mathbf{F}_1 - \mathbf{N}_1\left(\mathbf{d}_1^i,\ \mathbf{d}_{2E}^{i+1},\ \mathbf{d}_{2F}^{i+1},\ \mathbf{d}_{2I}^{i+1}\right).
\end{aligned} \tag{34}$$

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. (32). To describe this variation, we first write, similar to the way we wrote Eqs. (25) and (26), the linear equation system that needs to be solved:

$$\begin{aligned}
\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12E}\mathbf{x}_{2E} + \mathbf{A}_{12F}\mathbf{x}_{2F} + \mathbf{A}_{12I}\mathbf{x}_{2I} &= \mathbf{b}_1, \\
\mathbf{A}_{2E1}\mathbf{x}_1 + \mathbf{A}_{2E2E}\mathbf{x}_{2E} + \mathbf{A}_{2E2F}\mathbf{x}_{2F} + \mathbf{A}_{2E2I}\mathbf{x}_{2I} &= \mathbf{b}_{2E}, \\
\mathbf{A}_{2F1}\mathbf{x}_1 + \mathbf{A}_{2F2E}\mathbf{x}_{2E} + \mathbf{A}_{2F2F}\mathbf{x}_{2F} + \mathbf{A}_{2F2I}\mathbf{x}_{2I} &= \mathbf{b}_{2F}, \\
\mathbf{A}_{2I1}\mathbf{x}_1 + \mathbf{A}_{2I2E}\mathbf{x}_{2E} + \mathbf{A}_{2I2F}\mathbf{x}_{2F} + \mathbf{A}_{2I2I}\mathbf{x}_{2I} &= \mathbf{b}_{2I},
\end{aligned} \tag{35}$$

where

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}, \tag{36}$$

with β, $\gamma = 1,\ 2E,\ 2F,\ 2I$. Then, for the iterative solution of Eq. (35), we define two preconditioners:

$$\mathbf{P}_{LTOS} = \begin{bmatrix} \mathbf{A}_{11} & 0 & 0 & 0 \\ \mathbf{A}_{2E1} & \mathbf{A}_{2E2E} & 0 & 0 \\ \mathbf{A}_{2F1} & \mathbf{A}_{2F2E} & \mathbf{A}_{2F2F} & 0 \\ \mathbf{A}_{2I1} & \mathbf{A}_{2I2E} & \mathbf{A}_{2I2F} & \mathbf{A}_{2I2I} \end{bmatrix}, \tag{37}$$

$$\mathbf{P}_{STOL} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12E} & \mathbf{A}_{12F} & \mathbf{A}_{12I} \\ 0 & \mathbf{A}_{2E2E} & \mathbf{A}_{2E2F} & \mathbf{A}_{2E2I} \\ 0 & 0 & \mathbf{A}_{2F2F} & \mathbf{A}_{2F2I} \\ 0 & 0 & 0 & \mathbf{A}_{2I2I} \end{bmatrix}. \tag{38}$$

We propose that these two preconditioners are used alternatingly during the inner iterations of the GMRES search method. We note that this mixed preconditioning technique with multi-level discretization is closely related to the mixed CEBE and CC preconditioning technique[28] we referred to in Section 8. To differentiate between the two variations of the EDSUM we described above, we call the nonlinear version, described by Eqs. (33) and (34), EDSUM-N, and the linear version, described by Eqs. (35) - (38), EDSUM-L.

12 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 an advection equation. The advection equation governs the time-evolution of the interface function marking the interface location. We also described in this paper some of the additional methods developed to increase the scope and accuracy of the interface-tracking and interface-capturing techniques. Among these methods are the EDICT, which was developed to increase the accuracy in capturing the interface, and extensions and offshoots of the EDICT, such as the EDMRT, EDSTT, EINST, EALST, and EDSUM.

ACKNOWLEDGMENT

This work was supported by the US Army Natick Soldier Center and NASA Johnson Space Center.

REFERENCES

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

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

[3] T.E. Tezduyar. Finite element interface-tracking and interface-capturing techniques for flows with moving boundaries and interfaces. In *Proceedings of the ASME Symposium on Fluid-Physics and Heat Transfer for Macro- and Micro-Scale Gas-Liquid and Phase-Change Flows (CD-ROM)*, ASME Paper IMECE2001/HTD-24206, New York, New York, (2001). ASME.

[4] T.E. Tezduyar. Stabilized finite element formulations and interface-tracking and interface-capturing techniques for incompressible flows. to appear in *Proceedings of the Workshop on Numerical Simulations of Incompressible Flows*, Half Moon Bay, California, (2002).

[5] T.E. Tezduyar. Computation of moving boundaries and interfaces with interface-tracking and interface-capturing techniques. In *Pre-Conference Proceedings of the Sixth Japan-US International Symposium on Flow Simulation and Modeling*, Fukuoka, Japan, (2002).

[6] 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 (Web Site)*, Vienna, Austria, (2002).

[7] T.J.R. Hughes and A.N. Brooks. A multi-dimensional upwind scheme with no cross-

wind diffusion. In T.J.R. Hughes, editor, *Finite Element Methods for Convection Dominated Flows*, AMD-Vol.34, pages 19–35. ASME, New York, (1979).

[8] T.J.R. Hughes, L.P. Franca, and G.M. Hulbert. A new finite element formulation for computational fluid dynamics: VIII. the Galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, **73**, 173–189 (1989).

[9] T. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, and S. Mittal. Parallel finite-element computation of 3D flows. *IEEE Computer*, **26**(10), 27–36 (October 1993).

[10] 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*, pages 1–6. Japan Society of Mechanical Engineers, (1997).

[11] 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**, 235–248 (1998).

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

[13] T.E. Tezduyar. Adaptive determination of the finite element stabilization parameters. In *Proceedings of the ECCOMAS Computational Fluid Dynamics Conference 2001 (CD-ROM)*, Swansea, Wales, United Kingdom, (2001).

[14] T.E. Tezduyar. Stabilization parameters and element length scales in SUPG and PSPG formulations. In *Book of Abstracts of the 9th International Conference on Numerical Methods and Computational Mechanics*, Miskolc, Hungary, (2002).

[15] T. Tezduyar. Stabilization parameters and local length scales in SUPG and PSPG formulations. In *Proceedings of the Fifth World Congress on Computational Mechanics (Web Site)*, Vienna, Austria, (2002).

[16] 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, pages 7–24, New York, (1992). ASME.

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

[18] K. Stein and T. Tezduyar. Advanced mesh update techniques for problems involving large displacements. In *Proceedings of the Fifth World Congress on Computational Mechanics (Web Site)*, Vienna, Austria, (2002).

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

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

[21] 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).

[22] T.E. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar, and J. McCune. Computational methods for parachute aerodynamics. In M. Hafez, K. Morinishi, and J. Periaux, editors, *Computational Fluid Dynamics for the 21st Century*. Springer, (2001).

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

[24] T.E. Tezduyar, S. Aliabadi, M. Behr, A. Johnson, V. Kalro, and M. Litke. High performance computing techniques for flow simulations. In M. Papadrakakis, editor, *Solving Large-Scale Problems in Mechanics: Parallel Solution Methods in Computational Mechanics*, pages 363–398. John Wiley & Sons, (1996).

[25] V. Kalro and T. Tezduyar. Parallel iterative computational methods for 3D finite element flow simulations. *Computer Assisted Mechanics and Engineering Sciences*, **5**, 173–183 (1998).

[26] Z. Johan, T.J.R. Hughes, and F. Shakib. A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids. *Computer Methods in Applied Mechanics and Engineering*, **87**, 281–304 (1991).

[27] Z. Johan, K.K. Mathur, S.L. Johnsson, and T.J.R. Hughes. A case study in parallel computation: Viscous flow around an Onera M6 wing. *International Journal for Numerical Methods in Fluids*, **21**, 877–884 (1995).

[28] T.E. Tezduyar, M. Behr, S.K. Aliabadi, S. Mittal, and S.E. Ray. A new mixed preconditioning method for finite element computations. *Computer Methods in Applied Mechanics and Engineering*, **99**, 27–42 (1992).