Vol. 8, 2, 83-130 (2001)

Archives of Computational Methods in Engineering
State of the art reviews

Finite Element Methods for Flow Problems with Moving Boundaries and Interfaces

Tayfun E. Tezduyar

Team for Advanced Flow Simulation and Modeling (T*AFSM)
Mechanical Engineering and Materials Science
Rice University - MS 321, 6100 Main Street
Houston, TX 77005, USA

Summary

This paper is an overview of the finite element methods developed by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [http://www.mems.rice.edu/TAFSM/] for computation of flow problems with moving boundaries and interfaces. This class of problems include those with free surfaces, two-fluid interfaces, fluid-object and fluid-structure interactions, and moving mechanical components. The methods developed can be classified into two main categories. The interface-tracking methods are based on the Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation, where the mesh moves to track the interface, with special attention paid to reducing the frequency of remeshing. The interface-capturing methods, typically used for free-surface and two-fluid flows, are based on the stabilized formulation, over non-moving meshes, of both the flow equations and the advection equation governing the time-evolution of an interface function marking the location of the interface. In this category, when it becomes neccessary to increase the accuracy in representing the interface beyond the accuracy provided by the existing mesh resolution around the interface, the Enhanced-Discretization Interface-Capturing Technique (EDICT) can be used to to accomplish that goal. In development of these two classes of methods, we had to keep in mind the requirement that the methods need to be applicable to 3D problems with complex geometries and that the associated large-scale computations need to be carried out on parallel computing platforms. Therefore our parallel implementations of these methods are based on unstructured grids and on both the distributed and shared memory parallel computing approaches. In addition to these two main classes of methods, a number of other ideas and methods have been developed to increase the scope and accuracy of these two classes of methods. The review of all these methods in our presentation here is supplemented by a number numerical examples from parallel computation of complex, 3D flow problems.

1 INTRODUCTION

In this paper we provide an overview of the methods developed in recent years by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [www.mems.rice.edu/TAFSM/] to address the computational challenges involved in simulation of flow problems with moving boundaries and interfaces. Within this general category, the classes of flow problems we identified as areas of computational mechanics where we expect to make an impact include: unsteady flows with interfaces, fluid-object and fluid-structure interactions, airdrop systems, and flows with rapidly-moving mechanical components. The main computational challenge in flows with moving boundaries and interfaces is that the spatial domain occupied by the fluid changes in time, and the formulation must be able to handle this accurately and efficiently. In some classes of problems the location of the boundaries or interfaces is an unknown that needs to be determined as part of the overall solution. The location of the boundary or interface might be unknown within the fluid mechanics problem or through dependence on the solution of the non-fluid part of a multi-physics problem.

Unsteady flows with interfaces can involve two-fluid (such as two different liquids or a liquid and a gas) or free-surface flows. For example, simulation of operational stability of vehicles carrying bulk liquids requires solution of this class of flow problems. This is a class of problems where the location of the boundaries and interfaces is unknown within the fluid mechanics problem, and must be determined together with the solution of the Navier-Stokes equations.

 ISSN: 1134 3060 Received: September 2000

Finite Element Methods for Flow Problems with Moving Boundaries and Interfaces

Tayfun E. Tezduyar
Team for Advanced Flow Simulation and Modeling (T*AFSM)
Mechanical Engineering and Materials Science
Rice University - MS 321, 6100 Main Street
Houston, TX 77005, USA

http://www.mems.rice.edu/TAFSM/

Abstract

This paper is an overview of the finite element methods developed by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [http://www.mems.rice.edu/TAFSM/] for computation of flow problems with moving boundaries and interfaces. This class of problems include those with free surfaces, two-fluid interfaces, fluid-object and fluid-structure interactions, and moving mechanical components. The methods developed can be classified into two main categories. The interface-tracking methods are based on the Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation, where the mesh moves to track the interface, with special attention paid to reducing the frequency of remeshing. The interface-capturing methods, typically used for free-surface and two-fluid flows, are based on the stabilized formulation, over non-moving meshes, of both the flow equations and the advection equation governing the time-evolution of an interface function marking the location of the interface. In this category, when it becomes neccessary to increase the accuracy in representing the interface beyond the accuracy provided by the existing mesh resolution around the interface, the Enhanced-Discretization Interface-Capturing Technique (EDICT) can be used to accomplish that goal. In development of these two classes of methods, we had to keep in mind the requirement that the methods need to be applicable to 3D problems with complex geometries and that the associated large-scale computations need to be carried out on parallel computing platforms. Therefore our parallel implementations of these methods are based on unstructured grids and on both the distributed and shared memory parallel computing approaches. In addition to these two main classes of methods, a number of other ideas and methods have been developed to increase the scope and accuracy of these two classes of methods. The review of all these methods in our presentation here is supplemented by a number numerical examples from parallel computation of complex, 3D flow problems.

1 INTRODUCTION

In this paper we provide an overview of the methods developed in recent years by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [http://www.mems.rice.edu/TAFSM/] to address the computational challenges involved in simulation of flow problems with moving

boundaries and interfaces. Within this general category, the classes of flow problems we identified as areas of computational mechanics where we expect to make an impact include: unsteady flows with interfaces, fluid-object and fluid-structure interactions, airdrop systems, and flows with rapidly-moving mechanical components. The main computational challenge in flows with moving boundaries and interfaces is that the spatial domain occupied by the fluid changes in time, and the formulation must be able to handle this accurately and efficiently. In some classes of problems the location of the boundaries or interfaces is an unknown that needs to be determined as part of the overall solution. The location of the boundary or interface might be unknown within the fluid mechanics problem or through dependence on the solution of the non-fluid part of a multi-physics problem.

Unsteady flows with interfaces can involve two-fluid (such as two different liquids or a liquid and a gas) or free-surface flows. For example, simulation of operational stability of vehicles carrying bulk liquids requires solution of this class of flow problems. This is a class of problems where the location of the boundaries and interfaces is unknown within the fluid mechanics problem, and must be determined together with the solution of the Navier-Stokes equations.

In fluid-object interactions, in addition to the interactions between solid objects and the fluid these objects are moving in, we might have interactions between the objects themselves, such as collisions and groupings. In the modeling of this class of problems, the flow is governed by the Navier-Stokes equations. The 3D dynamics of the objects is governed by the Newton's laws. The fluid forces acting on these particles are calculated from the computed flow field. The motion of the particles is influenced by the fluid forces, and in turn influences the fluid behavior. Therefore the governing equations need to be solved in a coupled fashion. Most of our attention in this class of problems have been focused on fluid-particle interactions, where the particles are of spherical shape.

Fluid-structure interactions involve fluids with moving boundaries and unsteady interfaces between the fluid and the structure. This is somewhat similar to fluid-object interactions. However, now the "objects" are deformable, and these deformations need to be determined coupled with the solution of the Navier-Stokes equations. In this category, we have been mainly focusing on parachute fluid-structure interactions and fluid-structure interactions in interior flows with moving mechanical components. Examples of airdrop systems are: aerodynamic behavior of round and ram-air parachutes, aerodynamic interaction between an aircraft and a paratrooper, and a parachute crossing the wake flow of an aircraft. These simulations involve fluid-object and fluid structure interactions. They also involve aerodynamics of complex shapes, and, in some cases, unsteady long-wake flows generated by such complex objects.

Examples of flows with rapidly-moving mechanical components are flow past a propeller and flow around a helicopter with its main rotor in motion. In this special class of problems, the additional challenge is that the relative motion between the mechanical components involved is too fast to be handled with the general-purpose methods developed for updating the finite element mesh as the spatial domain occupied by the fluid changes in time.

We have developed a number of methods and ideas to support simulation and modeling of the classes of problems described above. The Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) method is the base, general-purpose formulation developed for flow problems with moving boundaries and interfaces. Special- and general-purpose mesh update tech-

niques were developed to be used in conjunction with the DSD/SST formulation. For flow problems with rapidly-moving mechanical components, we developed the Shear-Slip Mesh Update Method (SSMUM). A special DSD/SST formulation has been developed for spatially periodic flows. The Space-Time Contact Technique (STCT) is being proposed for computation of fluid-solid contact problems based on the DSD/SST formulation. For more efficient computation of some special cases of fluid-object interactions, we propose the Fluid-Object Interactions Subcomputation Technique (FOIST). The Enhanced-Discretization Interface-Capturing Technique (EDICT) was developed for more accurate representation of the interfaces computed with interface-capturing techniques. We have developed new, EDICT-based methods for computation of compressible flows with shocks and for computation of vortex flows, and propose an EDICT-based approach for mesh refinement near solid surfaces with boundary layers. We propose the Mixed Interface-Tracking/Interface-Capturing Technique (MITICT) for computation of flow problems that involve both interfaces that can be accurately tracked with a moving mesh method and interfaces that are too complex or unsteady to be tracked and therefore require an interface-capturing technique. The Edge-Tracked Interface Locator Technique (ETILT) is being proposed to enable interface-capturing techniques to have better volume conservation and yield sharper representation of the interfaces. We have developed a special DSD/SST formulation for computation of free-surface flow problems based on the shallow water equations. Iterative solution techniques were developed for solving the large, coupled nonlinear equation systems that need to be solved at every time step of a computation. The Mixed Element-Matrix-Based/Element-Vector-Based Computation Technique (MMVCT) is being proposed to improve the effectiveness of the iterative solution techniques, and the Enhanced-Discretization Successive Update Method (EDSUM) is being proposed to extend these solution techniques to multi-scale computations.

All methods developed are for flow problems involving complex geometries, and all software used in generating the results reported here was developed and implemented on parallel platforms by the T$\star$AFSM. All simulations, except those for testing a new method, were carried out in 3D. Furthermore, all computations reported here were performed on parallel computing platforms.

This overview article is largely based on earlier publications by the T$\star$AFSM, particularly recent overview articles on flow simulation methods for complex flow problems [1, 2]. In Section 2 we review the governing equations used in the computations. Stabilized finite element formulations and different approaches for computation of moving boundaries and interfaces are reviewed in Section 3. The DSD/SST formulation is reviewed in Section 4. The mesh update methods for the DSD/SST formulation are described in Section 5. The SSMUM is described in Section 6, and the DSD/SST formulation for spatially periodic flows is described in Section 7. The STCT and FOIST are introduced in Sections 8 and 9. The EDICT is reviewed in Section 10, with the construction of the function spaces used in the EDICT described in Section 11. Extensions of the EDICT to other classes of problems are described in Section 12. In Sections 13 and 14, we introduce the MITICT and ETILT. Extension of the DSD/SST formulation to shallow water equations is described in Section 15. Section 16 provides an overview of the iterative solution methods and parallel computing platforms used. Related to the iterative solution techniques, in Sections 17 and 18, we introduce the MMVCT and EDSUM. In Section 19 we report several examples of flow simulations, and finish with concluding remarks in Section 20.

2 GOVERNING EQUATIONS

In both compressible and incompressible flow cases, the governing equations used in numerical modeling are the time-dependent Navier-Stokes equations. The space and time domains will be denoted by Ω_t and $(0,T)$, where Γ_t is the boundary of Ω_t. In some cases the spatial domain may change with respect to time, and the subscript t indicates such time-dependence. This will be the case if in the formulation addressing flows with moving boundaries and interfaces the spatial domain is defined to be the part of the space occupied by the fluid(s). The symbols $\rho(\mathbf{x},t)$, $\mathbf{u}(\mathbf{x},t)$, $p(\mathbf{x},t)$ and $e(\mathbf{x},t)$ represent the density, velocity, pressure and the total energy, respectively. The external forces (e.g., the gravity) are represented by $\mathbf{f}(\mathbf{x},t)$.

COMPRESSIBLE FLOWS

The Navier-Stokes equations of compressible flows can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - \frac{\partial \mathbf{E}_i}{\partial x_i} - \mathbf{R} = \mathbf{0} \quad \text{on } \Omega_t \qquad \forall t \in (0,T), \tag{1}$$

where $\mathbf{U} = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho e)$ is the vector of conservation variables, and $\mathbf{F}_i$ and $\mathbf{E}_i$ are, respectively, the Euler and viscous flux vectors, defined as

$$\mathbf{F}_i = \begin{pmatrix} u_i\rho \\ u_i\rho u_1 + \delta_{i1}p \\ u_i\rho u_2 + \delta_{i2}p \\ u_i\rho u_3 + \delta_{i3}p \\ u_i(\rho e + p) \end{pmatrix}, \tag{2}$$

$$\mathbf{E}_i = \begin{pmatrix} 0 \\ T_{i1} \\ T_{i2} \\ T_{i3} \\ -q_i + T_{ik}u_k \end{pmatrix}. \tag{3}$$

Here δ_{ij} are the components of the identity tensor, q_i are the components of the heat flux vector, and T_{ij} are the components of the Newtonian viscous stress tensor:

$$\mathbf{T} = 2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \tag{4}$$

where μ is the dynamic viscosity and $\boldsymbol{\varepsilon}$ is the strain rate tensor, defined as

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{1}{2}((\boldsymbol{\nabla}\mathbf{u}) + (\boldsymbol{\nabla}\mathbf{u})^T). \tag{5}$$

The equation of state used here corresponds to the ideal gas assumption. The term $\mathbf{R}$ represents all other components that might enter the equations, including the external forces.

Equation (1) can further be written in the following form:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}_i\frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial}{\partial x_i}\left(\mathbf{K}_{ij}\frac{\partial \mathbf{U}}{\partial x_j}\right) - \mathbf{R} = \mathbf{0} \quad \text{on } \Omega_t \qquad \forall t \in (0,T), \tag{6}$$

where

$$\mathbf{A}_i = \frac{\partial \mathbf{F}_i}{\partial \mathbf{U}}, \tag{7}$$

$$\mathbf{K}_{ij}\frac{\partial \mathbf{U}}{\partial x_j} = \mathbf{E}_i. \tag{8}$$

Appropriate sets of boundary and initial conditions are assumed to accompany Equation (6).

INCOMPRESSIBLE FLOWS

The Navier-Stokes equations of incompressible flows can be written as

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\mathbf{u} - \mathbf{f}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = \mathbf{0} \quad \text{on } \Omega_t \quad \forall t \in (0,T), \tag{9}$$

$$\boldsymbol{\nabla}\cdot\mathbf{u} = 0 \quad \text{on } \Omega_t \quad \forall t \in (0,T), \tag{10}$$

where ρ is assumed to be constant, and

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mathbf{T}, \tag{11}$$

where $\mathbf{I}$ is the identity tensor. This equation set is completed with an appropriate set of boundary conditions and an initial condition consisting of a divergence-free velocity field specified over the entire domain:

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0, \quad \boldsymbol{\nabla}\cdot\mathbf{u}_0 = 0 \quad \text{on } \Omega_0. \tag{12}$$

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 select 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.

Remark

(1) When we model a liquid-gas interaction, we let Fluid A be the liquid and Fluid B the gas. If we model 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

$$\rho = \phi\rho_A + (1-\phi)\rho_B, \tag{13}$$

$$\mu = \phi\mu_A + (1-\phi)\mu_B. \tag{14}$$

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

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} \phi = 0 \qquad \text{on } \Omega \quad \forall t \in (0, T), \tag{15}$$

Remark

(2) One can also see Equations (13) - (15) as those representing the constitutive law of the fluid system. How accurately this law will be modeled will depend on how accurately the front between $\phi = 1$ and $\phi = 0$ will be represented and advected.

Remark

(3) We will not address here the surface tension effects at the interfaces.

3 INTERFACE-TRACKING AND INTERFACE-CAPTURING METHODS, STABILIZED FINITE ELEMENT FORMULATIONS

In computation of flow problems with moving boundaries and interfaces, depending on the nature of the problem, we can use an interface-tracking or interface-capturing method. An interface-tracking method requires meshes that "track" the interfaces. The mesh needs to be updated as the flow evolves. In an interface-capturing method, the computations are based on fixed spatial domains, where an interface function, such as the one described in Section 2, 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.

The Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation is an interface-tracking method, and was first introduced in [3–5]. In the DSD/SST method the finite element formulation of the problem is written over its associated space-time domain. This automatically takes into account the motion of the boundaries and interfaces. At each time step of a computation, the locations of the boundaries and interfaces are calculated as part of the overall solution. The stabilized space-time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example [6]).

The interface-tracking and interface-capturing methods described in this paper are based on stabilization techniques. Namely, the Streamline-Upwind/Petrov-Galerkin (SUPG) [7–11], Pressure-Stabilizing/Petrov-Galerkin (PSPG) [3,12], and Galerkin/Least-Squares (GLS) [13,14,3] formulations. The SUPG method is one of the earliest and most widely-used stabilized methods. The SUPG formulation for incompressible flows was first introduced in [7,8]. The SUPG formulation for compressible flows, on the other hand, was first introduced, in the context of conservation variables, in a 1983 AIAA paper [9]. Several researches designed and studied SUPG-like methods for compressible flows. For example, the Taylor-Galerkin method, which is described in a 1984 paper [15], is very similar, and under some conditions identical, to one of the SUPG methods introduced in [9]. Other researchers also reported in papers in 1984 and later years SUPG-like methods for compressible flows in the context of conservation variables (see for example [16, 17]). The SUPG formulation for compressible flows was recast in entropy variables and supplemented with a shock-capturing term

(see [18]). It was shown in [10] that, the SUPG formulation introduced in [9], when supplemented with a similar shock-capturing term, is very comparable in accuracy to the one employing entropy variables. In fact, it was shown in [11] for inviscid flows and in [19] for viscous flows that for the 2D comparative test problems computed, the SUPG formulations in conservation and entropy variables yield rather indistinguishable results. The PSPG formulation was introduced in [3] and assures numerical stability while allowing us to use equal-order interpolation functions for velocity and pressure and other unknowns. An earlier version of this stabilized formulation for Stokes flows was reported in [20].

The stabilization techniques described above prevent numerical oscillations and instabilities when the flow involves high Reynolds and/or Mach numbers and strong shocks and boundary layers. In SUPG, PSPG and GLS formulations, the stabilization is accomplished without introducing excessive numerical dissipation (i.e. without "overstabilizing"). Overstabilizing is not always easy to be fully aware of, as the symptoms are not necessarily qualitative. The SUPG, PSPG and GLS formulations were developed with this concern in mind, and perform quite well when the implementation is based on a sound understanding of these methods.

In more recent years, research efforts in flows with moving boundaries and interfaces intensified significantly (see for example [21–28]). A discussion on the geometric conservation properties of various methods developed for moving boundaries and interfaces can be found in [22], which includes a conclusion that the space time formulation leads to solution techniques that inherently satisfy the geometric conservation law.

4 DSD/SST FORMULATION

In the DSD/SST method, the finite element formulation of the governing equations is written over a sequence of N space-time slabs Q_n, where Q_n is the slice of the space-time domain between the time levels t_n and t_{n+1} (see Figure 1). At each time step, the integrations involved

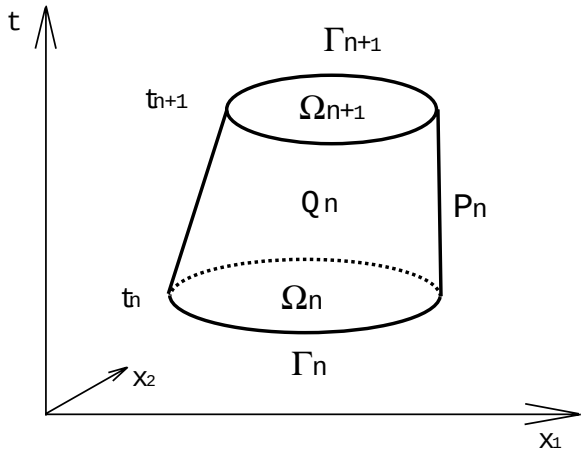

Figure 1. Space-time concept.

in the finite element formulation are performed over Q_n. The finite element interpolation functions are discontinuous across the space-time slabs. In the computations reported here,

we use first-order polynomials as interpolation functions. We use the notation $(\cdot)_n^-$ and $(\cdot)_n^+$ to denote the function values at t_n as approached from below and above respectively. Each Q_n is decomposed into space-time elements Q_n^e, where $e = 1, 2, \ldots, (n_{el})_n$. The subscript n used with n_{el} is to account for the general case in which the number of space-time elements may change from one space-time slab to another.

The DSD/SST formulations for compressible and incompressible flows are based on the same concepts and look very similar. We review them here both for completeness.

DSD/SST FORMULATION FOR COMPRESSIBLE FLOWS

For each slab Q_n, we define appropriate finite-dimensional space-time function spaces $\mathcal{S}_n^h$ and $\mathcal{V}_n^h$ corresponding to the trial solutions and weighting functions, respectively. While the superscript h implies that these are finite-dimensional function spaces, the subscript n implies that corresponding to different space-time slabs we might have different spatial discretizations. The DSD/SST formulation of Equation (6) can then be written as follows: given $(\mathbf{U}^h)_n^-$, find $\mathbf{U}^h \in \mathcal{S}_n^h$ such that $\forall \mathbf{W}^h \in \mathcal{V}_n^h$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{W}^h \cdot \left(\frac{\partial \mathbf{U}^h}{\partial t} + \mathbf{A}_i^h \frac{\partial \mathbf{U}^h}{\partial x_i} \right) dQ + \int_{Q_n} \left(\frac{\partial \mathbf{W}^h}{\partial x_i} \right) \cdot \left(\mathbf{K}_{ij}^h \frac{\partial \mathbf{U}^h}{\partial x_j} \right) dQ \\
&- \int_{Q_n} \mathbf{W}^h \cdot \mathbf{R}^h dQ + \int_{\Omega_n} (\mathbf{W}^h)_n^+ \cdot \left((\mathbf{U}^h)_n^+ - (\mathbf{U}^h)_n^- \right) d\Omega \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \boldsymbol{\tau}_{\text{SUPG}} \, (\mathbf{A}_k^h)^T \left(\frac{\partial \mathbf{W}^h}{\partial x_k} \right) \cdot \left[\frac{\partial \mathbf{U}^h}{\partial t} + \mathbf{A}_i^h \frac{\partial \mathbf{U}^h}{\partial x_i} - \frac{\partial}{\partial x_i} \left(\mathbf{K}_{ij}^h \frac{\partial \mathbf{U}^h}{\partial x_j} \right) - \mathbf{R}^h \right] dQ \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\text{SHOC}} \left(\frac{\partial \mathbf{W}^h}{\partial x_i} \right) \cdot \left(\frac{\partial \mathbf{U}^h}{\partial x_i} \right) dQ = \int_{P_n} \mathbf{W}^h \cdot \boldsymbol{H}^h dP. \qquad (16)
\end{aligned}$$

Here $\boldsymbol{H}^h$ represents the Neumann-type boundary condition, P_n is the lateral boundary of the space-time slab, $\boldsymbol{\tau}_{\text{SUPG}}$ is the SUPG stabilization matrix, and τ_{SHOC} is the scalar shock-capturing parameter. The SUPG stabilization parameter $\boldsymbol{\tau}_{\text{SUPG}}$ originated in [9] as a scalar parameter, and was later modified in [29] to its matrix version used here. The shock-capturing parameter τ_{SHOC} was introduced in [10]. The solution to Equation (16) is obtained sequentially for all space-time slabs $Q_0, Q_1, Q_2, ..., Q_{N-1}$, and the computations start with

$$(\mathbf{U}^h)_0^- = \mathbf{U}_0^h, \qquad (17)$$

where $\mathbf{U}_0$ is the specified initial value of the vector $\mathbf{U}$.

The first four integrals, together with the right-hand-side, represent the time-discontinuous Galerkin formulation of Equation (6), where the fourth integral enforces, weakly, the continuity of the conservation variables in time. The first series of element-level integrals are the SUPG stabilization terms, and the second series are the shock-capturing terms. For problems not involving moving boundaries and interfaces, Equation (16) can be reduced to a semi-discrete formulation by dropping the fourth integral and converting all space-time integrations to spatial integrations.

The trial function spaces for velocity and pressure will be denoted by $(\hat{\mathcal{S}}^h_{\mathbf{u}})_n$ and $(\hat{\mathcal{S}}^h_p)_n$. The weighting function spaces corresponding to momentum equation and incompressibility constraint will be denoted by $(\hat{\mathcal{V}}^h_{\mathbf{u}})_n$ and $(\hat{\mathcal{V}}^h_p)_n$ $(= (\hat{\mathcal{S}}^h_p)_n)$. The DSD/SST formulation of Equations (9) and (10) can be written as follows: given $(\mathbf{u}^h)^-_n$, find $\mathbf{u}^h \in (\hat{\mathcal{S}}^h_{\mathbf{u}})_n$ and $p^h \in (\hat{\mathcal{S}}^h_p)_n$ such that $\forall \mathbf{w}^h \in (\hat{\mathcal{V}}^h_{\mathbf{u}})_n$ and $\forall q^h \in (\hat{\mathcal{V}}^h_p)_n$:

$$
\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
&+ \int_{Q_n} q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}^h)^+_n \cdot \rho \left((\mathbf{u}^h)^+_n - (\mathbf{u}^h)^-_n \right) d\Omega \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \tau_{\mathrm{LSME}} \frac{1}{\rho} \left[\rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h) \right] \\
&\qquad \cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \right] dQ \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q^e_n} \tau_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h \, \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ = \int_{P_n} \mathbf{w}^h \cdot \mathbf{h}^h dP. \qquad (18)
\end{aligned}
$$

Here $\mathbf{h}^h$ represents the Neumann-type boundary condition associated with the momentum equation, and τ_{LSME} and τ_{LSIC} are the stabilization parameters (see [2, 30]). For an earlier, detailed reference on this stabilized formulation see [3].

The solution to Equation (18) is obtained sequentially for $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, and the computations start with

$$
(\mathbf{u}^h)^-_0 = \mathbf{u}^h_0. \qquad (19)
$$

The first four integrals, together with the right-hand-side, represent the time-discontinuous Galerkin formulation of Equations (9)–(10), where the fourth integral enforces, weakly, the continuity of the velocity field in time. The two series of element-level integrals in the formulation are the least-squares stabilization terms corresponding to momentum equation and incompressibility constraint.

For problems not involving moving boundaries and interfaces, Equation (18) can be reduced to a semi-discrete formulation by dropping the fourth integral and the term $\frac{\partial \mathbf{w}^h}{\partial t}$, and by converting all space-time integrations to spatial integrations.

5 MESH UPDATE FOR INTERFACE-TRACKING METHODS

In interface-tracking methods, as the computations proceed, the mesh needs to be updated to accommodate the changes in the spatial domain. It is crucial that this is accomplished as effectively as possible. How the mesh can best 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 as long as it is possible and remeshing (i.e. generating fully or partially a new set of nodes and elements) when the element distortion becomes too high.

Most real-world problems require simulations with complex geometries. A complex geometry typically requires an automatic mesh generator to start with. We developed our own automatic mesh generator to have a number of special features, such as structured layers of elements around solid surfaces and high-speed mesh generation. The automatic, 3D mesh generator we have developed is described in [31]. It has been used very effectively in a number of simulations (for early examples see [32, 33]). This automatic mesh generator has the capability to build structured layers of elements around solid objects with reasonable geometric complexity. With this capability, we can fully control the mesh resolution near solid objects. This feature can be used for more accurate representation of the boundary layers. The mesh generator also has the capability to generate meshes for fluid-object interactions in spatially periodic flows (see [34]).

Automatic mesh generation might become an overwhelming cost especially when the number of elements becomes very large or when frequency of remeshing has to be high. Sometimes special-purpose mesh generators designed for specific problems can be used. Depending on the complexity of the problem, such mesh generators might involve a high initial design cost, but minimal mesh generation cost. We selected this path in a number of our simulations, and were able to overcome the mesh generation issues very effectively (see for example [35]).

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

In some cases where the changes in the shape of the computational domain allow it, a special-purpose mesh moving method can be used in conjunction with a special-purpose mesh generator. In such cases, simulations can be carried out without calling an automatic mesh generator and without solving any additional equations to determine the motion of the mesh. One of the earliest examples of that, 3D parallel computation of sloshing in a vertically vibrating container, can be found in [36].

In general, however, we use an automatic mesh moving scheme [37] to move the nodal points, as governed by the equations of linear elasticity. The motion of the internal nodes is determined by solving these additional equations, with the boundary conditions for these mesh motion equations specified in such a way that they match the normal velocity of the fluid at the interface. Similar mesh moving techniques were used earlier by other researchers (see for example [38]). Mesh moving issues were also addressed in [39] by using a 2D pseudo-structural model based on springs, and more recently in [21] by using improved versions of the same model.

In our mesh moving method based on linear elasticity, the structured layers of elements

generated around solid objects (mentioned above) move "glued" to these solid objects. 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 also results in some cost reduction. But more importantly, the user continues to have full control of the mesh resolution in these layers. For early examples of automatic mesh moving combined with structured layers of elements, see [31, 32].

6 SHEAR-SLIP MESH UPDATE METHOD (SSMUM)

The Shear-Slip Mesh Update Method (SSMUM) was first introduced for computation of flow around two high-speed trains passing each other in a tunnel (see [33]). The challenge was to accurately and efficiently update the meshes used in computations based on the DSD/SST formulation and involving two objects in fast, linear relative motion. In such cases, a special-purpose mesh moving method without remeshing would not work, and an automatic mesh generation method would require remeshing too frequently to have an effective mesh update technique. The idea behind the SSMUM was to restrict the mesh moving and remeshing to a thin layer of elements between the objects in relative motion (see Figure 2). The mesh update at each time step can be accomplished by a "shear" deformation of the elements in this layer, followed by a "slip" in node connectivities (see Figure 3). The slip in the node connectivities, to an extent, un-does the deformation of the elements and results in elements with better shapes than those that were shear-deformed. Because the remeshing consists of simply re-defining the node connectivities, both the projection errors and the mesh generation cost are minimized. The SSMUM can be seen as an alternative to Chimera overset grid technique [40], which requires projection of the solution between portions of the two overlapping grids.

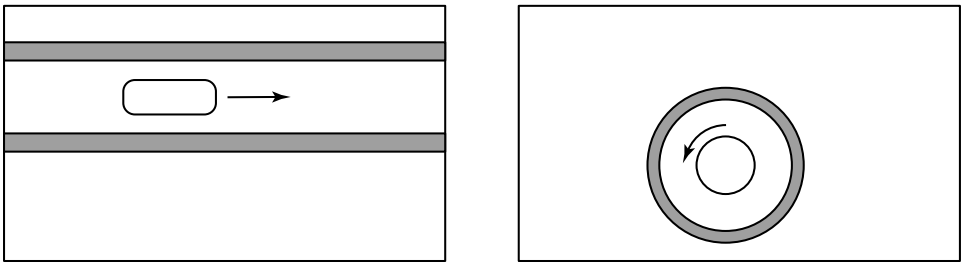

Figure 2. SSMUM concept. Shear-slip layers in linear and rotational relative motions.

For the purpose emphasizing the generality and flexibility of the SSMUM, we point out a number of options.

A) The meshes outside the shear-slip layers can be structured or unstructured. Those meshes most of the time simply undergo rigid-body motion, as special cases, with some of them held fixed. One can exercise all the freedom one would have in generating fixed structured or unstructured meshes, such as generating very thin structured layers of elements around solid objects, combined with unstructured meshes further out.

B) In more general cases, the meshes outside the shear-slip layer can undergo more than just

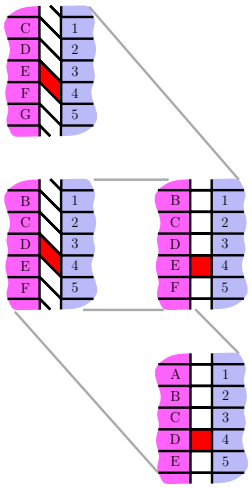

Figure 3. SSMUM concept. Node connectivities during the shear-slip process.

rigid-body motion, and can be updated with special-purpose mesh moving method without remeshing and/or an automatic mesh moving method with tolerable frequency of remeshing.

C) Depending on the thickness of the shear-slip layer and the rate of relative motion, it may not be necessary to have each shear step followed by a slip step. In such cases, multiple shear steps can be taken followed by a slip step, and this would reduce the cost associated with the slip steps.

D) Although the SSMUM has so far been implemented with single layers of elements, in general, the shear-slip zone can be made of multiple layers of elements or even unstructured meshes. This would allow us to reduce the ratio of the number of slip steps to the number of shear steps. In such cases, one needs to consider the balance between the decrease in cost due to decreasing the frequency of the slip steps with the increase in cost due to increased burden of re-defining node connectivities in a more complex shear-slip zone. If the shear-slip zone is made of unstructured meshes, then the shear step would require an automatic mesh moving method. Still, the relative cost associated with this and with redefining connectivities in a complex shear-slip zone would be bounded by the size of this shear-slip mesh zone relative to the total mesh size.

E) Furthermore, when the geometrical and implementational requirements dictate, the shear-slip zones can have shapes that are spatially non-uniform or temporally varying. For example, the shear-slip layer, instead having a disk shape, can have a conical shape. In such more general cases, the shear-slip process can be handled in ways similar to those described in Item D above.

F) Also when the conditions dictate, the SSMUM can be implemented in such a way that the mesh in the shear-slip zone is unstructured in both space and time.

The SSMUM was first implemented for computation of incompressible and compressible flows with objects in linear relative motion, and the results for compressible flow around

the high-speed trains passing each other in a tunnel were reported in [33]. In more recent years, the implementation has been extended to objects in rotational relative motion (see [41, 42]), and we describe some the results from those computations in the section on numerical examples.

7 DSD/SST FORMULATION FOR FLUID-OBJECT INTERACTIONS IN SPATIALLY-PERIODIC FLOWS

In extending the DSD/SST formulation of incompressible flows to computation of fluid-object interactions in spatially-periodic flows (see [34]), we consider a 2D computational domain (see Figure 4). This rectangular domain (width $\times$ height $= L \times H$) is assumed to contain

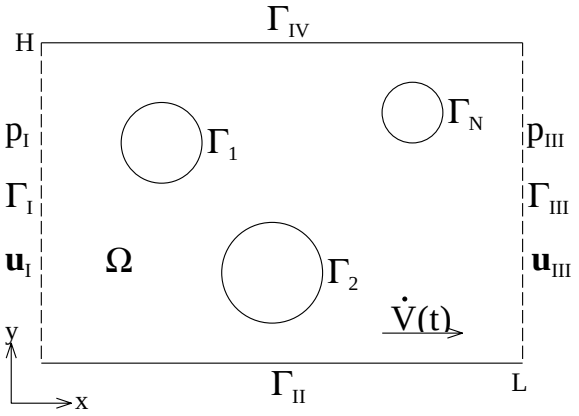

Figure 4. Spatially-periodic computational domain.

N circular objects, with surfaces (inner boundaries for the fluid) Γ_κ, where $\kappa = 1, 2, \ldots, N$. The outer boundaries are denoted by $\Gamma_I, \Gamma_{II}, \Gamma_{III}$, and Γ_{IV}.

The formulation we would like to derive should be applicable to uni-periodic (i.e. periodic in one direction), bi-periodic and tri-periodic flows, and where the total volumetric flow rate in each periodic direction is prescribed. As the first step, the formulation for uni-periodic flows is derived. Then, this is extended to bi-periodic and tri-periodic cases.

First we re-examine Equation (18), and consider only the stress terms and the associated natural boundary conditions along Γ_I and Γ_{III}:

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}) : \boldsymbol{\sigma}(p, \mathbf{u}) dQ = \int_{P_I} \mathbf{w} \cdot \mathbf{h}_I dP + \int_{P_{III}} \mathbf{w} \cdot \mathbf{h}_{III} dP. \tag{20}$$

We have dropped the superscript h and the subscript n to reduce the notational burden during the derivation. The flow is assumed to be periodic in the x_1 direction, and the prescribed total volumetric flow rate in this direction is $\dot{V}$. We define $\mathbf{u}^*$ and $\mathbf{w}^*$ to be the periodic velocity field and the associated weighting function:

$$\mathbf{u}^*|_{P_I} = \mathbf{u}^*|_{P_{III}}, \tag{21}$$

$$\mathbf{w}^*|_{P_I} = \mathbf{w}^*|_{P_{III}}. \tag{22}$$

Then, Equation (20) becomes:

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p, \mathbf{u}^*) dQ = -\int_{P_{III}} (\mathbf{w}^* \cdot \mathbf{e}_1) J dP, \tag{23}$$

where $\mathbf{e}_1$ is the unit vector in the x_1 direction. The term $J(t) = p_{III} - p_I$ represents the pressure jump across the domain in the x_1 direction, and this is an additional unknown corresponding to the constraint imposed by prescribing the total volumetric flow rate in the x_1 direction. This constraint, together with the incompressibility constraint, can be written as

$$\int_Q q(\boldsymbol{\nabla} \cdot \mathbf{u}^*) dQ + \int_I K \left[\dot{V} - \int_{\Gamma_{III}} (\mathbf{u}^* \cdot \mathbf{e}_1) d\Gamma \right] dI = 0, \tag{24}$$

where K is the weighting function corresponding to J, and I represents the time interval (t_n, t_{n+1}).

To simplify the implementation, we introduce a space reduction by decomposing p and q as

$$p = p^* + \frac{J}{L} x_1, \tag{25}$$

$$q = q^* + \frac{K}{L} x_1, \tag{26}$$

where p^* and q^* are continuous across the periodic boundaries, and the discontinuities are represented by J and K.

With this, Equation (23) becomes:

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*) dQ - \int_Q (\boldsymbol{\nabla} \cdot \mathbf{w}^*) \frac{J}{L} x_1 dQ = -\int_{P_{III}} (\mathbf{w}^* \cdot \mathbf{e}_1) J dP. \tag{27}$$

By integrating the second term by parts and further algebraic manipulation, we obtain:

$$\int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*) dQ + \int_Q \frac{J}{L} (\mathbf{w}^* \cdot \mathbf{e}_1) dQ = 0. \tag{28}$$

Also as a consequence of this space reduction, Equation (24) becomes:

$$\int_Q q^* (\boldsymbol{\nabla} \cdot \mathbf{u}^*) dQ + \int_Q \frac{K}{L} x_1 (\boldsymbol{\nabla} \cdot \mathbf{u}^*) dQ + \int_I K \left[\dot{V} - \int_{\Gamma_{III}} (\mathbf{u}^* \cdot \mathbf{e}_1) d\Gamma \right] dI = 0. \tag{29}$$

By integrating the second term by parts and further algebraic manipulation, we obtain for circular or spherical particles:

$$\int_Q q^* (\boldsymbol{\nabla} \cdot \mathbf{u}^*) dQ + \frac{1}{L} \int_I K \left[L\dot{V} + \sum_{\kappa=1}^{N} V_\kappa (U_1)_\kappa - \int_\Omega (\mathbf{u}^* \cdot \mathbf{e}_1) d\Omega \right] dI = 0, \tag{30}$$

where V_κ and $(U_1)_\kappa$ are, respectively, the volume and velocity of sphere κ.

We can now write the complete DSD/SST formulation for fluid-object interactions in spatially-periodic flows as follows: given $(\mathbf{u}^*)_n^-$, find $\mathbf{u}^* \in (S_{\mathbf{u}^*})_n$ and $p^* \in (S_{p^*})_n$ such that $\forall \mathbf{w}^* \in (V_{\mathbf{u}^*})_n$ and $\forall q^* \in (V_{p^*})_n$:

$$\begin{aligned}
&\int_Q \mathbf{w}^* \cdot \rho \left(\frac{\partial \mathbf{u}^*}{\partial t} + \mathbf{u}^* \cdot \boldsymbol{\nabla} \mathbf{u}^* - \mathbf{f} \right) dQ + \int_Q \boldsymbol{\varepsilon}(\mathbf{w}^*) : \boldsymbol{\sigma}(p^*, \mathbf{u}^*) dQ + \frac{1}{L} \int_Q (\mathbf{w}^* \cdot \mathbf{J}) dQ \\
&- \int_{(P_n)_{\mathbf{h}^*}} \mathbf{w}^* \cdot \mathbf{h}^* dP + \int_{Q_n} q^* \boldsymbol{\nabla} \cdot \mathbf{u}^* dQ + \frac{1}{L} \int_I \mathbf{K} \cdot \left[L \dot{\mathbf{V}} + \sum_{\kappa=1}^{N} V_\kappa \mathbf{U}_\kappa - \int_\Omega \mathbf{u}^* d\Omega \right] dI \\
&+ \int_{\Omega_n} (\mathbf{w}^*)_n^+ \cdot \rho \left((\mathbf{u}^*)_n^+ - (\mathbf{u}^*)_n^- \right) d\Omega \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{\tau_{\text{LSME}}}{\rho} \left[\rho \left(\frac{\partial \mathbf{w}^*}{\partial t} + \mathbf{u}^* \cdot \boldsymbol{\nabla} \mathbf{w}^* \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(q^*, \mathbf{w}^*) + \frac{\mathbf{K}}{L} \right] \cdot \\
&\qquad \left[\rho \left(\frac{\partial \mathbf{u}^*}{\partial t} + \mathbf{u}^* \cdot \boldsymbol{\nabla} \mathbf{u}^* - \mathbf{f} \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(p^*, \mathbf{u}^*) + \frac{\mathbf{J}}{L} \right] dQ = 0 \,. \qquad (31)
\end{aligned}$$

This formulation is applicable to uni- bi- or tri-periodic flows, with the understanding that for the flow rate and pressure jump vectors $\dot{\mathbf{V}}$ and $\mathbf{J}$, the components not corresponding to the directions of periodicity will be set to zero.

8 SPACE-TIME CONTACT TECHNIQUE (STCT)

The DSD/SST formulation, combined with the Space-Time Contact Technique (STCT), provides a natural mechanism to handle time-dependent flow problems with contacting and de-contacting surfaces. Formulating this class of problems in the space-time domain can help us develop more effective methods.

Let us imagine a one-dimensional problem, shown in Figure 5, where there is a possibility that between the time levels t_n and t_{n+1} the liquid free surface on the right might be contacting a wall. Let us first perform some intermediate calculations for this space-time slab,

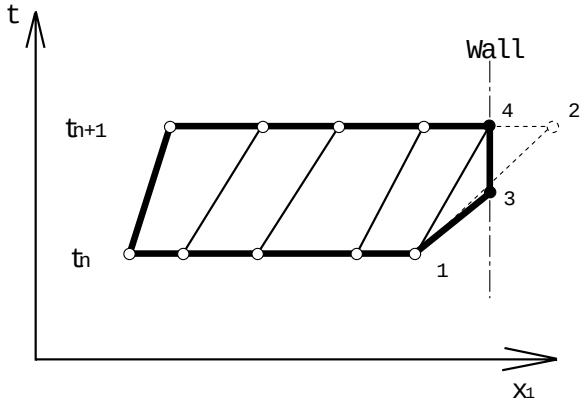

Figure 5. STCT concept in one dimension. Fluid contacting wall.

where we update the positions of the free-surface nodes by assuming that the motion of these

free-surface nodes are not constrained by the wall. Let us say that these calculations show that the new position of Node-1 at t_{n+1} is at location $(x_1)_2$, which is beyond the wall. Next, on the wall, we predict the temporal position of Node-3. Node-3 represents the contact point in the space-time domain. We calculate this predicted value of $t_3 - t_1$ from $(x_1)_3 - (x_1)_1$ and $(u_1)_1^+$. We can now redo the calculations for this modified space-time slab. Although we will be using a predicted value for $t_3 - t_1$, we can see the calculation for this modified space-time slab as one in which $(x_1)_4$ becomes a known, and $t_3 - t_1$ becomes an unknown. This is as opposed to the intermediate calculations, in which $(x_1)_2$ was an unknown, and $t_2 - t_1$ was known. We complete the calculations for this space-time slab by performing a sufficient number of iterations, in which we update $t_3 - t_1$ and $(u_1)_1^+$, as well as p_1^+, p_3, and p_4^-.

In extending this to multi-dimensional cases, we can see the picture as follows. In the intermediate calculations, we would have $t_2 - t_1$ as known and $(x_1)_2$, $(x_2)_2$, and $(x_3)_2$ as unknowns. In the calculations for the modified space-time slab, $(x_1)_4$ would become a known, and we would have $t_3 - t_1$ as unknown, together with $(x_2)_4$ and $(x_3)_4$. Node-4 would symbolically represent more than one node. We will later provide a picture for a simple two-dimensional case.

Let us know now imagine another one-dimensional case, shown in Figure 6, where there is a possibility that between the time levels t_n and t_{n+1} the liquid on the right might be de-contacting the wall. Again, we first perform some intermediate calculations for this

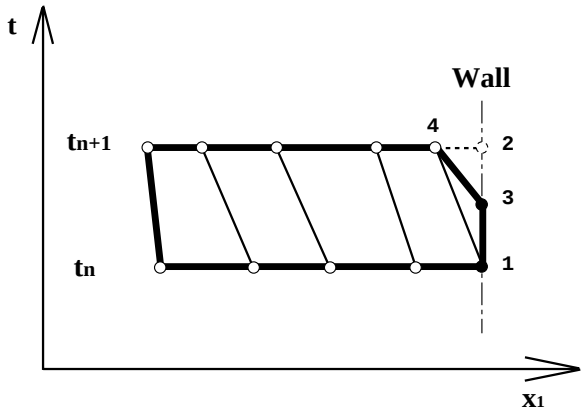

Figure 6. STCT concept in one dimension. Fluid de-contacting wall.

space-time slab. In these calculations we assume that Node-1 stays on the wall and maps to Node-2 at t_{n+1}. During these intermediate calculations, we also predict the liquid pressure on the wall, namely p_1^+ and p_2^-. Next, on the wall, we predict the temporal position of Node-3 by calculating t_3 when the liquid pressure becomes zero (we assume that the viscous stress is negligible). Node-3 represents the de-contact point in the space-time domain. We calculate the predicted value of $t_3 - t_1$ from p_1^+ and p_2^- by seeking the zero of the linear function $p(t)$. We can now redo the calculations for the modified space-time slab. At each iteration of the calculations for this modified space-time slab, we update $(u_1)_4^-$, $(x_1)_4$, p_1^+, p_3, and p_4^-, as well as $t_3 - t_1$ by seeking the zero of the linear function $p(t)$ based on the updated values of p_1^+ and p_3.

Figure 7 shows a simple 2D case where we expect that between the time levels t_n and t_{n+1} Node-1 might be contacting the wall. In principle the calculation process is very similar

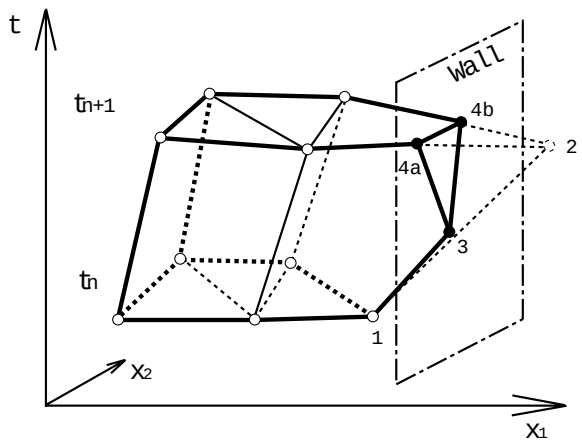

Figure 7. STCT concept in two dimensions. Fluid contacting wall.

to the 1D contact problem. In the intermediate calculations, we have $t_2 - t_1$ as known and $(x_1)_2$, $(x_2)_2$, and $(x_3)_2$ as unknowns. In the calculations for the modified space-time slab, $(x_1)_{4a}$ and $(x_1)_{4b}$ become knowns, and we have $t_3 - t_1$ as unknown, together with $(x_2)_{4a}$ and $(x_2)_{4b}$. We complete the calculations for this space-time slab by performing a sufficient number of iterations, in which we update $t_3 - t_1$, $(x_2)_3$, $(x_2)_{4a}$, $(x_2)_{4b}$, $(u_1)_1^+$, $(u_2)_1^+$, $(u_2)_3$, $(u_2)_{4a}^-$, and $(u_2)_{4b}^-$, as well as p_1^+, p_3, p_{4a}^-, and p_{4b}^-.

We realize that the 2D computations will require a 3D mesh generation in the space-time domain, and the 3D computations will require a 4D mesh generation. However, we also realize that these will be only partial mesh generations, limited to the contact zones.

9 FLUID-OBJECT INTERACTIONS SUBCOMPUTATION TECHNIQUE (FOIST)

The Fluid-Object Interactions Subcomputation Technique (FOIST) is an intermediate level approximation between treating the objects as point masses and using a fully coupled fluid-object interaction formulation. We assume that the nature of the fluid-object interactions, such as the scales involved and flow patterns expected, allows us to take into account only a one-way coupling between the main flow field and the motion of the objects. In other words, we assume that the main flow field influences the motion of the objects, but the presence and motion of the objects do not influence the main flow field. With this assumption, the main flow field can be computed without taking into account any of the smaller-scale objects, and at the same time, the dynamics of the objects can be determined by carrying out flow subcomputations over smaller-scale domains around the objects. The boundary conditions for these domains would be extracted from the main flow field, at locations corresponding to the positions of those boundaries at that instant.

The main flow field would be computed over a fixed mesh. The subcomputation for each object would be carried out over a fixed mesh, and in a coordinate frame attached to that object. In the subcomputations, we take into account the geometry of the objects,

and determine the unsteady flow field around these objects together with the resulting fluid dynamics forces and moments. These forces and moments would be used, while taking into account the instantaneous values of the moments of inertia, to compute the path and orientation of the objects.

Each subcomputation can be carried out as a two-way coupled fluid-object interaction problem without the need for mesh moving. This is because the coordinate frame is attached to the object, and the coupling is implemented by updating the boundary conditions as a function of the orientation of the object, rather than by updating the mesh.

Because the FOIST would typically not involve mesh moving or remeshing, by eliminating the cost associated with those tasks, it would result in a major reduction in the computational cost. The FOIST can be extended to cases where the main flow computation or a subcomputation may require mesh update. This could happen for the main flow, for example, when it involves moving objects that are too large to be handled with the assumptions underlying FOIST. For a subcomputation, this could happen, for example, when the object is undergoing deformations.

Here we also introduce another level of approximation–one that is beyond FOIST, but still with more realistic assumptions than those used in treating the objects as point masses. In this technique that we will call "Beyond- FOIST" (B-FOIST), for each object with a different shape, we would generate a database of fluid dynamics force and moment coefficients. This database would be generated from computations for a set of Reynolds numbers within a range of interest and a set of "basis directions" for the flow velocity. These would typically be unsteady flow computations, but the force and moment coefficients would be determined based on temporal averaging of the results.

With this database, the path and orientation of the objects would be calculated without flow subcomputations. At each instant of the calculation of the path and orientation of an object, the force and moment coefficients needed would be estimated by interpolation from the database of these coefficients. The coefficients corresponding to the Reynolds number and flow velocity directions at an instant would be calculated by a linear or higher-order interpolation with respect to the Reynolds number, and by a directional interpolation with respect to the flow velocity direction. The directional interpolation would use from the database the basis directions nearest to the direction of the flow velocity.

How effective B-FOIST would be for a given problem would depend on the balance between i) the computational cost saved by not doing the flow subcomputations, and ii) the loss of some accuracy and the increased cost associated with generating the database. For example, if the fluid-object interaction problem involves many objects with identical shapes, B-FOIST might prove quite effective, because the database generation would not involve objects with different shapes. In addition, if these objects have one or more symmetry planes or axes, the cost of database generation would further decrease, gaining additional incentive for B-FOIST.

The starting point for the basis directions for the flow velocity would be the six directions identified by (1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1), and (0,0,-1). For the flow direction at an instant, the directional interpolation would use the three nearest of these basis directions. The components or direction cosines of the flow velocity direction in these three nearest directions would then be used in calculating a weighted average of the force and moment coefficients corresponding to these three directions.

To increase the directional resolution of the database, additional basis directions can be defined. The first set of additions would be (1,1,1), (-1,1,1), (1,-1,1), (1,1,-1), (-1,-1,1), (-1,1,-1), (1,-1,-1), and (-1,-1,- 1). The second set of additional directions would be (1,1,0), (-1,1,0), (1,-1,0), (-1,-1,0), (1,0,1), (-1,0,1), (1,0,-1), (-1,0,-1), (0,1,1), (0,-1,1), (0,1,-1), and (0,-1,-1).

If an object has one or more symmetry planes or axes, some of the basis directions become redundant and would be eliminated.

10 ENHANCED-DISCRETIZATION INTERFACE-CAPTURING TECHNIQUE (EDICT)

With interface-tracking methods, sometimes the interface might be too complex or unsteady to track while keeping the frequency of remeshing at an acceptable level. Not being able to reduce the frequency of remeshing in 3D might introduce overwhelming mesh generation and projection costs, making the computations with the interface-tracking method no longer feasible. In such cases, interface-capturing methods, which do not normally require costly mesh update techniques, could be used with the understanding that the interface will not be represented as accurately as we would have with an interface-tracking method (for related discussions see [43]). Not needing a mesh update strategy makes the interface-capturing methods more flexible than the interface-tracking methods. 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 to carry 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 methods 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 [44] with this kind of philosophy. The objective was to enhance the spatial discretization around an interface so that we could have higher accuracy in representing that interface. We start with the basic approach of an interface-capturing technique such as the volume of fluid (VOF) method [45–47]. The Navier-Stokes equations are solved over a non-moving mesh with an interface function serving as a marker identifying the two fluids (see Section 2).

In writing the stabilized finite element formulation for the EDICT (see [43]), the notation we use for representing the finite dimensional function spaces is very similar to the one we used in Section 4. 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. We will give more precise definition of these function spaces in Section 11.

The stabilized formulations of Equations (9), (10), and (15) 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 $\forall \psi^h_{n+1} \in (\mathcal{V}^h_\phi)_{n+1}$:

$$
\begin{aligned}
&\int_\Omega \mathbf{w}_{n+1}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) d\Omega \\
&+ \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}_{n+1}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) d\Omega + \int_\Omega q_{n+1}^h \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega^e} \left(\tau_{\text{SUPG}} \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}_{n+1}^h + \frac{\tau_{\text{PSPG}}}{\rho} \boldsymbol{\nabla} q_{n+1}^h \right) \\
&\qquad \cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \right] d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_{\text{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}_{n+1}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h d\Omega = \int_\Gamma \mathbf{w}_{n+1}^h \cdot \mathbf{h}^h d\Gamma, \qquad (32)
\end{aligned}
$$

$$
\begin{aligned}
&\int_\Omega \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_\phi \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi_{n+1}^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega = 0, \qquad (33)
\end{aligned}
$$

where τ_{SUPG}, τ_{PSPG}, τ_{LSIC} and τ_ϕ are the stabilization parameters:

$$
\begin{aligned}
\tau_{\text{SUPG}} &= \left(\left(\frac{2 \parallel \mathbf{u}^h \parallel}{h} \right)^2 + \left(\frac{4\nu}{h^2} \right)^2 \right)^{-\frac{1}{2}}, \qquad (34) \\
\tau_{\text{PSPG}} &= \tau_{\text{SUPG}}, \qquad (35) \\
\tau_{\text{LSIC}} &= \frac{h}{2} \parallel \mathbf{u}^h \parallel z, \qquad (36) \\
&\qquad \text{where } z = \begin{cases} \left(\frac{Re_u}{3} \right) & Re_u \leq 3 \\ 1 & Re_u > 3 \end{cases}, \\
\tau_\phi &= \frac{h}{2 \parallel \mathbf{u}^h \parallel}, \qquad (37)
\end{aligned}
$$

where $\nu = \mu/\rho$ is the kinematic viscosity and Re_u is the cell Reynolds number.

Remark

(4) In Equation (32), the first three integrals, together with the right-hand-side, represent the Galerkin formulation of Equations (9)-(10). The first series of element-level integrals in the formulation are the SUPG and PSPG stabilization terms. The second series of element-level integrals are the least-squares stabilization terms based on the incompressibility constraint. In Equation (33), the first integral represents the Galerkin formulation of Equation (15), while the series of element-level integrals are the SUPG stabilization terms.

Remark

(5) A new set definitions for the stabilization parameters were introduced in [48]. These new parameters are computed based on the element-level matrices and vectors, which automatically take into account the local length scales, advection field and the Reynolds number.

Remark

(6) In time discretization, the time derivatives, $\partial \mathbf{u}/\partial t$ and $\partial \phi/\partial t$ are represented as follows:

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\mathbf{u}^h_{n+1} - \mathbf{u}^h_n}{\Delta t}, \tag{38}$$

$$\frac{\partial \phi}{\partial t} = \frac{\phi^h_{n+1} - \phi^h_n}{\Delta t}, \tag{39}$$

where Δt is the time step size between time levels n and n+1. In this time discretization, the functions $\mathbf{u}^h$, p^h and ϕ^h are represented as follows:

$$\mathbf{u}^h \leftarrow (1-\alpha)\mathbf{u}^h_n + \alpha \mathbf{u}^h_{n+1}, \tag{40}$$

$$p^h \leftarrow p^h_{n+1}, \tag{41}$$

$$\phi^h \leftarrow (1-\alpha)\phi^h_n + \alpha \phi^h_{n+1}, \tag{42}$$

where α is a time-integration parameter controlling the stability and accuracy of the integration. Normally we set $\alpha = 0.5$.

To have a sharper representation of the interface function ϕ, we incorporate to the formulation a two-step interface-sharpening algorithm [49].

Remark

(7) **Tau-Switching and Tau-Ramping.** A common practice in obtaining steady-state solutions at high Reynolds numbers is to use a time-marching method where we set the parameter $\alpha = 1.0$ and/or ramp-up the Reynolds number from a lower value to its full value. The same approach can be followed in time-marching through a transient period that does not have a physical significance but poses numerical convergence challenges. In that case, at the end of that initial period, we set $\alpha = 0.5$ to recover our time-accuracy, and continue the time-accurate computation for some duration with the full value of the Reynolds number.

Here, as alternative to or in combination with these approaches, we propose to use the stabilization parameters as selective switches or ramping parameters in obtaining steady-state solutions at high Reynolds numbers or in time-marching through immaterial but "difficult" periods. For example, to obtain a steady-state solution, we can switch τ_{SUPG} and τ_{PSPG} to zero for the time-dependent terms in the momentum equation, and ramp-down τ_{SUPG}, τ_{PSPG}, and τ_{LSIC} from some higher values to their formulated values, respectively, for the advective terms, the pressure gradient, and the

incompressibility constraint. During this process, τ_{SUPG} and τ_{PSPG} for the other terms in the governing equations can be left at their formulated values, or could be ramped-up from some lower values to their formulated values. The same approach can be used in time-marching through a transient period with no physical significance but with convergence difficulties. In that case, at the end of that initial period, all stabilization parameters for all the terms in the governing equations would be brought back to their formulated values, and the time-accurate computations would be continued for some duration with these formulated values.

11 CONSTRUCTION OF FUNCTION SPACES FOR EDICT

The basic concept behind the construction of the function spaces used for EDICT is not complicated. However, the description requires certain formalism and involves a number of guidelines. For completeness, we repeat them here from [43]. In constructing the function spaces corresponding to time level n, we start with a base mesh (Mesh-1), with the set of elements and nodal points denoted by ϵ_n^1 and η_n^1. The subscript n implies that Mesh-1 itself might change from one time level to other.

A second-level and more refined mesh (Mesh-2) is constructed over a subset $(\epsilon_n^1)_n^2$ of the these elements. Mesh-2 is generated by patching together the second-level meshes generated over each of the elements in $(\epsilon_n^1)_n^2$ (see Figure 8).

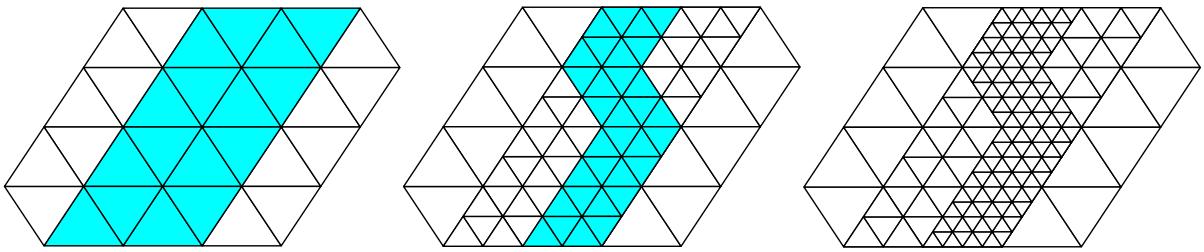

Figure 8. EDICT concept. Multi-level meshes.

Remark

(8) The second subscript n implies that for a given Mesh-1, which elements of this mesh are declared to be in $(\epsilon_n^1)_n^2$ might change from one time level to other. An element which might be declared to be in $(\epsilon_n^1)_n^2$ at some time level, might fall out of it at some other time, and yet come back in again at some time later.

Remark

(9) For each element in ϵ_n^1, there will be a unique second-level mesh. Therefore, if an element is declared to be in $(\epsilon_n^1)_n^2$ for a second time, the refined mesh generated over that element at the earlier declaration can be reused. If an automatic mesh generator is being used to generate these second level meshes, the cost for that mesh generation will be a one-time cost.

The set of elements and nodal points for Mesh-2 are denoted by ϵ_n^2 and η_n^2.

A third-level and even more refined mesh (Mesh-3) is constructed over a subset $(\epsilon_n^2)_n^3$ of the elements in Mesh-2. This will be generated by patching together the third-level meshes generated over each of the elements in $(\epsilon_n^2)_n^3$ (see Figure 8).

Remark

(10) The statements in Remarks 8-9 apply, with the mesh level numbers referred to in Remarks 8-9 shifted up by one.

The set of elements and nodal points for Mesh-3 are denoted by ϵ_n^3 and η_n^3.

Remark

(11) At this time we limit ourselves to linear elements in 2D and 3D, and bilinear and trilinear elements, respectively, in 2D and 3D.

We construct $\mathbf{u}_n^h$ as follows:

$$\mathbf{u}_n^h = \mathbf{u}_n^1 + \mathbf{u}_n^2. \tag{43}$$

The function $\mathbf{u}_n^1$ comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^1, excluding those "surrounded" by the elements in $(\epsilon_n^1)_n^2$. The function $\mathbf{u}_n^1$ also needs to satisfy the Dirichlet-type boundary conditions, except at those nodes that have been surrounded at the boundary of Ω. The function $\mathbf{u}_n^2$ comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^2, excluding those at the boundaries of the zones covered by the elements in ϵ_n^2. However we do include the nodes at the boundary of Ω unless they coincide with the nodes in η_n^1 that have not been surrounded.

We construct p_n^h in exactly the same way, except for recognizing the fact that the references to Dirichlet-type boundary conditions do not apply:

$$p_n^h = p_n^1 + p_n^2. \tag{44}$$

We construct ϕ_n^h with more enhancement:

$$\phi_n^h = \phi_n^1 + \phi_n^2 + \phi_n^3. \tag{45}$$

The function ϕ_n^1 comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^1, excluding those "surrounded" by the elements in $(\epsilon_n^1)_n^2$. The function ϕ_n^1 also needs to satisfy the Dirichlet-type boundary conditions, except at those nodes that have been surrounded at the boundary of Ω. The function ϕ_n^2 comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^2, excluding those at the boundaries of the zones covered by the elements in ϵ_n^2. We also exclude the nodes surrounded by the elements in $(\epsilon_n^2)_n^3$. However we do include the nodes at the boundary of Ω unless they coincide with the nodes in η_n^1 that have not been surrounded. The function ϕ_n^3 comes from a space of functions with the basis set consisting of the shape functions associated with all the nodes in η_n^3, excluding those at the boundaries of the zones covered by the elements in ϵ_n^3. However we do include the nodes at the boundary of Ω unless they coincide with the nodes in η_n^2 that have not been surrounded.

The weighting functions are constructed in a similar fashion:

$$\mathbf{w}_n^h = \mathbf{w}_n^1 + \mathbf{w}_n^2, \tag{46}$$

$$q_n^h = q_n^1 + q_n^2, \tag{47}$$

$$\psi_n^h = \psi_n^1 + \psi_n^2 + \psi_n^3. \tag{48}$$

The components of each weighting function are defined in the same way as we did for the trial functions, except that the weighting functions need to satisfy the homogeneous form of the Dirichlet-type boundary conditions.

Our objective with this enhanced discretization is to capture the interface as accurately as possible by using more refined meshes for the velocity, pressure, and interface function, and possibly even more refined meshes for the interface function. This is done in a dynamic fashion by defining $(\epsilon_n^1)_n^2$ and $(\epsilon_n^2)_n^3$ depending on which elements in ϵ_n^1 and ϵ_n^2 the interface is passing through, and re-define these subsets occasionally to track the interface.

Remark

(12) We update $(\epsilon_n^1)_n^2$ and $(\epsilon_n^2)_n^3$ not every time step but with sufficient frequency to keep the interface within the zones covered by these subsets of elements. How many time steps one can carry out the simulation without re-defining these subsets of elements will depend on, among other things, how "wide" we decide to keep these zones around the interface.

Remark

(13) Whenever we redefine these subsets the mesh generation cost will not be a significant one. If we are using an automatic mesh generator for the second- and third-level meshes, we will be able to use and re-use the meshes which were generated (and stored) the first time these meshes were needed.

Remark

(14) It is possible to eliminate ϕ_n^3 by not choosing to go to a third-level of refinement. It is also possible to design the second- and third-level meshes in such a way that they overlap. One of the advantages in keeping them as non-overlapping meshes is that, by keeping Mesh-2 "wider" than Mesh-3, one can chose to limit the existence (as an unknown) of ϕ to Mesh-2, and therefore solving for it only over the part of the computational domain covered by Mesh-2. With this, we have to make sure that the interface remains in Mesh-2 zone. Since our objective will be to keep the interface in Mesh-3 zone, this would also keep it in Mesh-2 zone, even if the interface occasionally falls slightly out of the Mesh-3 zone.

12 EXTENSIONS OF EDICT TO OTHER CLASSES OF PROBLEMS

Extension of EDICT to other classes of problems was first reported in [50] 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 [51, 52]. In this case, the definition of the interface is extended to mean regions where the vorticity magnitude is larger than a specified value. The finite element functions consist of two components. The first component comes from a base mesh (Mesh-1), generated over the entire domain. A second-level and more refined mesh (Mesh-2) is constructed over a subset of the elements in Mesh-1. Mesh-2 is generated by simply subdividing the elements in that subset. Which elements in Mesh-1 should belong to this subset depends on the magnitude of the vorticity in those elements and varies from time-step to time-step. This version of the EDICT gives us the capability to compute the long-wake flows more accurately, without making the computations too costly.

Here we propose to extend EDICT to computation of flow problems with boundary layers. In this extension, the "interface" means solid surfaces with boundary layers. It is known that in 3D problems with complex geometries and boundary layers, mesh generation might pose 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. However, 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 end up increasing our computational cost excessively or decreasing our numerical accuracy unacceptably. We propose two different ways of using EDICT to increase the mesh refinement in the boundary layers in a desirable fashion.

In the EDICT-Clustered-Mesh-2 approach, Mesh-2 is constructed by patching together clusters of second-level meshes generated over each element of Mesh-1 designated to be one of the "boundary layer elements". Depending on the type of these boundary layer elements in Mesh-1, Mesh-2 could be structured or unstructured, with hexahedral, tetrahedral or triangle-based prismatic elements.

In the EDICT-Layered-Mesh-2 approach, a thin but multi-layered and more refined Mesh-2 is "laid over" the solid surfaces. Depending on the geometric complexity of the solid surfaces and depending on whether we prefer the same type elements as those we used in Mesh-1, the elements in mesh-2 could be hexahedral, tetrahedral or triangle-based prismatic elements.

This 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.

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

The Mixed Interface-Tracking/Interface-Capturing Technique (MITICT) is being introduced primarily for computation of fluid-object interactions with multiple fluids. In particular, the class of applications we are targeting are fluid-particle-gas interaction problems and free-

surface flow of fluid-particle mixtures. However, the MITICT can be applied to a larger class of problems where we find it 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. As before, the interface-tracking technique we would use is the DSD/SST formulation. The interface-capturing technique "rides on" the interface-tracking technique, 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 (see Equations (13), (14), and (15)).

In addition to the DSD/SST formulation of the Navier-Stokes equations, we use the DSD/SST formulation of the advection equation for the interface function:

$$\int_{Q_n} \psi^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) dQ + \int_{\Omega_n} (\psi^h)_n^+ \left((\phi^h)_n^+ - (\phi^h)_n^- \right) d\Omega$$
$$+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_\phi \left(\frac{\partial \psi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi^h \right) \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) dQ = 0. \tag{49}$$

This equation, together with Equation (18), constitutes a mixed interface-tracking/interface-capturing technique that would be effective in accurately tracking the solid-fluid interfaces and capturing 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.

14 EDGE-TRACKED INTERFACE LOCATOR TECHNIQUE (ETILT)

The objective in the development of the Edge-Tracked Interface Locator Technique (ETILT) is 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. Nodes belong to "chunks" of Fluid A or Fluid B. An edge either belongs to a chunk of Fluid A or Fluid B or is an interface edge. Each element is either filled fully by a chunk of Fluid A or Fluid B, or is shared by a chunk of Fluid A and a chunk of Fluid B. If an element is shared like that, the shares are determined by the position of the interface along the edges of that element.

The base finite element formulation is essentially the one described by Equations (32) and (33). Although the ETILT can be used in combination with the EDICT, to make it easier to understand the concept, we can assume that we are working here with the plain, non-EDICT versions of Equations (32) and (33). This simplification consists of dropping the subscript $n+1$ from all weighting functions appearing in the equations.

In ETILT, 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:

$$\rho^h = \phi^{he}\rho_A + (1-\phi^{he})\rho_B, \tag{50}$$

$$\mu^h = \phi^{he}\mu_A + (1-\phi^{he})\mu_B. \tag{51}$$

In marching from time level n to $n+1$, we first calculate ϕ_n^h from ϕ_n^{he} by a least-squares projection:

$$\int_\Omega \psi^h \left(\phi_n^h - \phi_n^{he}\right) d\Omega = 0. \tag{52}$$

To calculate ϕ_{n+1}^h, we use Equation (33). From ϕ_{n+1}^h, we calculate ϕ_{n+1}^{he} by a combination of a least-squares projection:

$$\int_\Omega (\psi_{n+1}^{he})_P \left((\phi_{n+1}^{he})_P - \phi_{n+1}^h\right) d\Omega = 0, \tag{53}$$

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 among the fluid chunks can symbolically be written as

$$VOL\ (\phi_{n+1}^{he}) = VOL\ (\phi_n^{he}). \tag{54}$$

Here the subscript P is used for representing the intermediate values following the projection, but prior to the corrections for volume conservation.

These projections and volume corrections are embedded in our 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.

In the remainder of this section, we provide more explanation of how the projections and volume corrections would be handled at an iteration step taking us from iterative solution i to $i+1$.

A) In determining $(\phi_{n+1}^{he})_P^{i+1}$ from $(\phi_{n+1}^h)^{i+1}$, in the first step of the projection, the position of the interface along each interface edge is calculated. The calculation for an edge might yield for the interface position a value that is not within the range of values representing that edge. This would imply the following consequences.

i) That interface edge does not remain as an interface edge after the projection.
ii) The node at the end of that edge (in the direction of the interface motion) changes from one fluid to another after the projection.
iii) Different edges among those connecting that node to other nodes might be identified as edges expected to be interface edges after the projection. An edge connecting that node to another node would be identified as an interface edge if the other node belongs to a different fluid. If not, it means that a chunk of one of the fluids is merging with another chunk of the same fluid. It might also mean, as a special case, that a chunk of fluid is connecting with itself at another point.

In the second step of the projection, the interface positions would be calculated along the newly-identified interface edges and those predicted to remain as interface edges after the first step of the projection. If additional steps of the projection are required, the same procedure would be followed.

B) After the projection is complete, we need to detect the possible occurrence of mergers between chunks and split of chunks. The mergers can be detected as described earlier when we discussed the options related to identification of interface edges following a projection step. To detect the split of chunks, one way is to go through a sorting process. In this process, for each chunk, we start with one of the nodes belonging to that chunk, identify all the nodes connected to that node with edges belonging to that chunk, do the same for the newly-identified nodes, and continue this recursive process until all the connected nodes are identified.

After this sorting is complete, if we still have some nodes left in that chunk, this would mean that the chunk we are inspecting has been split. The recursive process needs to be repeated for the nodes and edges remaining in that chunk, so that any additional splits that chunk might have undergone are detected.

C) After the process of identifying all the Fluid A and Fluid B chunks is complete, we need to enforce the volume conservation. For each chunk, we compare the volumes corresponding interface locations denoted by $(\phi^{he}_{n+1})^i$ and $(\phi^{he}_{n+1})^{i+1}$. In the cases of mergers and splits, we compare the aggregate volume of a set of chunks corresponding to $(\phi^{he}_{n+1})^i$ and constituting a merger/split group to the aggregate volume of the set of chunks constituting the related merger/split group corresponding to $(\phi^{he}_{n+1})^{i+1}$.

D) The volume conservation for a chunk or a merger/split group would be enforced by inflating or deflating its volume. Let us suppose that multiplying the positive and negative increments along each interface edge by a factor $(1 + x)$ and $(1 - x)$, respectively, results in a volume correction by a factor $(1 + y)$, where y and x are of the same sign. We need to determine the value of x, such that the corresponding value of y is sufficiently close to the volume correction needed. This would be done iteratively, and the convergence of these iterations can be accelerated by calculating the numerical derivative of y with respect to x and using that estimate in updating x at every iteration.

15 DSD/SST FORMULATION FOR SHALLOW WATER EQUATIONS

DSD/SST method can be formulated for shallow water equations in the same way it was formulated for compressible flows in Section 4. This is accomplished by casting the shallow water equations in the forms given by Equations (1) and (6)-(8). To this end, we need to define $\mathbf{U}$, $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{R}$. The coefficient matrices will be the derived quantities.

The shallow water equations (see [53]) can be written as

$$\frac{\partial \zeta}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u}[h + \zeta]) = 0 \quad \text{on } \Omega_t \quad \forall t \in (0, T), \tag{55}$$

where $\mathbf{u}$ is the depth-averaged horizontal velocity, ζ is the water elevation relative to a reference position of the water surface, and h is the water-depth for that reference position.

Defining the total water height as $H = h + \zeta$, we can re-write Equation (55) in the following form:

$$\frac{\partial H}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} H) = 0 \quad \text{on } \Omega_t \quad \forall t \in (0, T). \tag{56}$$

The momentum equations corresponding to the horizontal directions can be written as

$$\frac{\partial (H\mathbf{u})}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} H \mathbf{u}) + gH\boldsymbol{\nabla}(H - h) - 2\nu H \boldsymbol{\nabla} \cdot \boldsymbol{\varepsilon} - \mathbf{S} = \mathbf{0} \quad \text{on } \Omega_t \quad \forall t \in (0, T), \tag{57}$$

where g is the gravitational acceleration, and the strain-rate tensor $\boldsymbol{\varepsilon}$ is defined by Equation (5), with the velocity $\mathbf{u}$ interpreted as the two-dimensional average velocity we have here. The vector $\mathbf{S}$ represents the surface and bottom shear stresses, and is defined as

$$\mathbf{S} = \begin{pmatrix} \left((T_{31})_s - (T_{31})_b\right)/\rho \\ \left((T_{32})_s - (T_{32})_b\right)/\rho \end{pmatrix}, \tag{58}$$

where the subscripts "s" and "b" refer to the surface and bottom respectively. We re-write Equation (57) as

$$\begin{aligned} &\frac{\partial (H\mathbf{u})}{\partial t} + \boldsymbol{\nabla} \cdot (\mathbf{u} H \mathbf{u}) + \boldsymbol{\nabla}\left(\frac{1}{2} g H^2\right) - 2\nu \boldsymbol{\nabla} \cdot (H\boldsymbol{\varepsilon}) \\ &- gH\boldsymbol{\nabla} h + 2\nu \boldsymbol{\nabla} H \cdot \boldsymbol{\varepsilon} - \mathbf{S} = \mathbf{0} \quad \text{on } \Omega_t \quad \forall t \in (0, T). \end{aligned} \tag{59}$$

This equation, together with Equation (56), can be used for defining $\mathbf{U}$, $\mathbf{F}_1$, $\mathbf{F}_2$, $\mathbf{E}_1$, $\mathbf{E}_2$, and $\mathbf{R}$, starting with $\mathbf{U} = (U_1, U_2, U_3) = (H, Hu_1, Hu_2)$. The Euler fluxes are:

$$\mathbf{F}_1 = \begin{pmatrix} u_1 H \\ u_1 H u_1 + \frac{1}{2} g H^2 \\ u_1 H u_2 \end{pmatrix}, \qquad \mathbf{F}_2 = \begin{pmatrix} u_2 H \\ u_2 H u_1 \\ u_2 H u_2 + \frac{1}{2} g H^2 \end{pmatrix}. \tag{60}$$

The viscous fluxes are:

$$\mathbf{E}_1 = \begin{pmatrix} 0 \\ 2\nu H \varepsilon_{11} \\ 2\nu H \varepsilon_{12} \end{pmatrix}, \qquad \mathbf{E}_2 = \begin{pmatrix} 0 \\ 2\nu H \varepsilon_{21} \\ 2\nu H \varepsilon_{22} \end{pmatrix}. \tag{61}$$

The vector $\mathbf{R}$ is:

$$\mathbf{R} = \begin{pmatrix} 0 \\ +gH\left(\partial h/\partial x_1\right) - 2\nu\left[\left(\partial H/\partial x_1\right)\varepsilon_{11} + \left(\partial H/\partial x_2\right)\varepsilon_{21}\right] + S_1 \\ +gH\left(\partial h/\partial x_2\right) - 2\nu\left[\left(\partial H/\partial x_1\right)\varepsilon_{12} + \left(\partial H/\partial x_2\right)\varepsilon_{22}\right] + S_2 \end{pmatrix}. \tag{62}$$

To define $\mathbf{A}_1$ and $\mathbf{A}_2$, we first re-write $\mathbf{F}_1$ and $\mathbf{F}_2$ as follows:

$$\mathbf{F}_1 = \begin{pmatrix} U_2 \\ (U_2/U_1)U_2 + \frac{1}{2} g {U_1}^2 \\ (U_2/U_1)U_3 \end{pmatrix}, \qquad \mathbf{F}_2 = \begin{pmatrix} U_3 \\ (U_3/U_1)U_2 \\ (U_3/U_1)U_3 + \frac{1}{2} g {U_1}^2 \end{pmatrix}. \tag{63}$$

Then, by using Equation (7), we derive $\mathbf{A}_1$ and $\mathbf{A}_2$:

$$\begin{aligned}
\mathbf{A}_1 &= \begin{bmatrix} 0 & 1 & 0 \\ gU_1 - (U_2/U_1)^2 & 2U_2/U_1 & 0 \\ -(U_2/U_1)(U_3/U_1) & U_3/U_1 & U_2/U_1 \end{bmatrix}, \\
\mathbf{A}_2 &= \begin{bmatrix} 0 & 0 & 1 \\ -(U_3/U_1)(U_2/U_1) & U_3/U_1 & U_2/U_1 \\ gU_1 - (U_3/U_1)^2 & 0 & 2U_3/U_1 \end{bmatrix}.
\end{aligned} \tag{64}$$

Similarly, by using Equation (8), we derive $\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{21}$, and $\mathbf{K}_{22}$:

$$\begin{aligned}
\mathbf{K}_{11} &= \begin{pmatrix} 0 & 0 & 0 \\ -2\nu U_2/U_1 & 2\nu & 0 \\ -\nu U_3/U_1 & 0 & \nu \end{pmatrix}, \quad \mathbf{K}_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\nu U_2/U_1 & \nu & 0 \end{pmatrix}, \\
\mathbf{K}_{21} &= \begin{pmatrix} 0 & 0 & 0 \\ -\nu U_3/U_1 & 0 & \nu \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{K}_{22} = \begin{pmatrix} 0 & 0 & 0 \\ -\nu U_2/U_1 & \nu & 0 \\ -\nu U_3/U_1 & 0 & 2\nu \end{pmatrix}.
\end{aligned} \tag{65}$$

With all these definitions, the DSD/SST method given by Equation (16) can now be extended to shallow water equations, and the motion of the water-land interfaces can be taken into account automatically. Beyond this point, the formulation can be seen very much like the DSD/SST formulation we discussed in previous sections. Several of the DSD/SST concepts previously discussed, such as moving interior boundaries, mesh update options, STCT, FOIST, and MITICT, can be extended to this model.

Remark

(15) The MITICT can be extended to shallow water equations to compute the time-evolution of a surface contaminant spreading in large bodies of water such as bays and lakes. These computations would be based on using the DSD/SST formulation (as an interface-tracking technique) for solving the shallow water equations, and using the EDICT for solving the advection equation governing the time-evolution of the contaminant.

16 ITERATIVE SOLUTION METHODS AND PARALLEL COMPUTING

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:

$$\mathbf{N}\left(\mathbf{d}_{n+1}\right) = \mathbf{F}. \tag{66}$$

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 solving a steady-state flow problem (see Remark 7 in Section 10). In such cases, the time steps do not have a physical significance, but are only used in "time-marching" to the steady-state solution.

We solve Equation (66) with the Newton-Raphson method:

$$\left.\frac{\partial \mathbf{N}}{\partial \mathbf{d}}\right|_{\mathbf{d}_{n+1}^i} \left(\Delta \mathbf{d}_{n+1}^i\right) = \mathbf{F} - \mathbf{N}\left(\mathbf{d}_{n+1}^i\right), \tag{67}$$

where i is the step counter for the Newton-Raphson sequence, and $\Delta \mathbf{d}_{n+1}^i$ is the increment computed for $\mathbf{d}_{n+1}^i$. The linear equation system represented by Equation (67) needs to be solved at every step of the Newton-Raphson sequence. We can represent Equation (67) as a linear equation system of the form

$$\mathbf{A}\mathbf{x} \;=\; \mathbf{b}. \tag{68}$$

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:

$$\mathbf{r} \;=\; \mathbf{b} - \mathbf{A}\mathbf{x}. \tag{69}$$

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, or even just element-level vectors. This last strategy is also called a matrix-free technique. After the residual computation, we compute a candidate correction to $\mathbf{x}$ as given by the expression

$$\Delta \mathbf{y} = \mathbf{P}^{-1}\mathbf{r}, \tag{70}$$

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 [54] 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 [55], element-vector-based [55], and sparse-matrix-based [56]. The element-vector-based (matrix-free) methods were successfully used also by other researchers in the context of parallel computations (see for example [57, 58]).

In preconditioning design, we developed some advanced preconditioners such as the Clustered-Element-by-Element (CEBE) preconditioner [59] and the mixed CEBE and Cluster Companion (CC) preconditioner [59]. We have implemented, with quite satisfactory results, the CEBE preconditioner in conjunction with an ILU approximation [56]. 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.

Some of our parallel computations were based on shared-memory platforms, such as multi-processor SGI systems, including a 2-processor SGI ONYX2. Most of our simulations were carried out on distributed-memory platforms. In this category, our earlier computations were based on data-parallel paradigm on a Thinking Machines CM-5. The majority of our more recent computations is based on message-passing paradigm on computing platforms such as a CRAY T3E-1200. More on our parallel implementations can be found in [55].

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

Consider a nonlinear equation system of the kind given by Equation (66), re-written in the following form:

$$\begin{aligned} \mathbf{N}_1\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_1, \\ \mathbf{N}_2\left(\mathbf{d}_1, \mathbf{d}_2\right) &= \mathbf{F}_2, \end{aligned} \tag{71}$$

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 Equation (68) in the form

$$\begin{aligned} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 &= \mathbf{b}_1, \\ \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 &= \mathbf{b}_2, \end{aligned} \tag{72}$$

where

$$\mathbf{A}_{\beta\gamma} = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}. \tag{73}$$

Re-writing Equations (66) and (68) 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 Equation (72) can be written as

$$\begin{aligned} \mathbf{P}_{11}\Delta\mathbf{y}_1 + \mathbf{P}_{12}\Delta\mathbf{y}_2 &= \mathbf{b}_1 - \left(\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2\right), \\ \mathbf{P}_{21}\Delta\mathbf{y}_1 + \mathbf{P}_{22}\Delta\mathbf{y}_2 &= \mathbf{b}_2 - \left(\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2\right), \end{aligned} \tag{74}$$

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 $\mathbf{A}_{11}^e$ and $\mathbf{A}_{22}^e$ 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:

$$\begin{aligned}
(\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2) &= \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{11}\mathbf{x}_1) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_1 \to 0} \left[\frac{\mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2 + \epsilon_1 \mathbf{x}_2) - \mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_1} \right], \\
(\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2) &= \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{22}\mathbf{x}_2) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_2 \to 0} \left[\frac{\mathbf{N}^e_2(\mathbf{d}_1 + \epsilon_2 \mathbf{x}_1, \mathbf{d}_2) - \mathbf{N}^e_2(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_2} \right], \quad (75)
\end{aligned}$$

where A is the finite element assembly operator, and 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:

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{\beta\gamma}\mathbf{x}_\gamma), \quad (76)$$

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_\beta \to 0} \left[\frac{\mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma + \epsilon_\beta \mathbf{x}_\gamma, \ldots) - \mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma, \ldots)}{\epsilon_\beta} \right]. \quad (77)$$

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 follows:

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_\gamma), \quad (78)$$

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.

18 ENHANCED-DISCRETIZATION SUCCESSIVE UPDATE METHOD (EDSUM)

In this section, we introduce a multi-level iteration method for computation of flow behavior at small scales. The Enhanced- Discretization Successive Update Method (EDSUM) 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 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 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 Equation (66) can be re-written as follows:

$$\begin{aligned}
\mathbf{N}_1\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_1, \\
\mathbf{N}_{2E}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2E}, \\
\mathbf{N}_{2F}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2F}, \\
\mathbf{N}_{2I}\ (\mathbf{d}_1,\ \mathbf{d}_{2E},\ \mathbf{d}_{2F},\ \mathbf{d}_{2I}) &= \mathbf{F}_{2I},
\end{aligned} \tag{79}$$

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

This equation system would be solved with the Newton-Raphson method. At each Newton-Raphson step, we would successively update the solution vectors corresponding to each class. We would start with updating the Class-1 nodes, then update the Class-2E, Class-2F, and Class-2I nodes, respectively. This sequence would be repeated as many times as required, and alternating between this sequence and its reverse sequence would be an option.

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.

19 EXAMPLES OF FLOW SIMULATIONS

In this section we present examples of flow simulations carried out by using some of the methods described in the earlier sections. In some cases at high Reynolds numbers, we use a simple turbulence model, where the physical viscosity is augmented by an eddy viscosity as proposed by Smagorinsky (see [60, 61]). All results were obtained by computations on parallel computing platforms. All computations were carried out in 3D, except for two of the EDICT test problems. Each numerical example is described briefly and references are given to our earlier publications for more information.

Flare maneuver of a large ram-air parachute. The Reynolds number in this computation is about 10 million. The initial condition consists of the steady glide configuration of an unconstrained parachute with no flap deflection. The time for the flare maneuver and total flap deflection is obtained from test data. The parachute is treated as a solid body with changing shape. The shape of the parachute during the maneuver is interpolated from the initial and final flap configurations. A special mesh generator/mover was developed. With this the flare maneuver is simulated without any remesh. The DSD/SST formulation leads to 2,258,496 coupled, nonlinear equations that need to be solved at every time step. Figure 9 shows the pressure distribution on the parachute surface during three instants of the flare maneuver. For more on this simulation see [33].

Flow around two high-speed trains in a tunnel. Two high-speed trains are passing each other in a tunnel. Each train has a speed of 100 m/s. The Reynolds number based on the train length is around 67 million. The mesh consists of 101,888 hexahedral elements and 230,982 space-time nodes, and leads to 900,274 coupled, nonlinear equations that need to be solved at every time step. Here the DSD/SST formulation is used together with the SSMUM to handle the mesh while the trains are in rapid, relative motion. Figure 10 shows the trains and pressure distribution at different instants. For more on this simulation see [33].

Flow past a rotating propeller. In this computation, the DSD/SST formulation and SSMUM are applied to simulation of flow past a propeller in a large box. The Reynolds number, based on the upstream flow velocity and the propeller diameter, is approximately 1 million. The rotational speed of the propeller, scaled with the upstream flow velocity and the propeller diameter, is 9.11π. The mesh consists of 305,786 space-time nodes and 939,213 tetrahedral elements. Figure 11 shows on the left the mesh at the outer boundaries and on the outer surface of the shear-slip layer, and on the right the mesh on the propeller surface and on the inner surface of the shear-slip layer. Figure 12 shows cross-sectional views of the mesh and the shear-slip layer, with the side view on the left and the top view on the right. Figure 13 shows on the left, the pressure on the propeller surface and the iso-surface

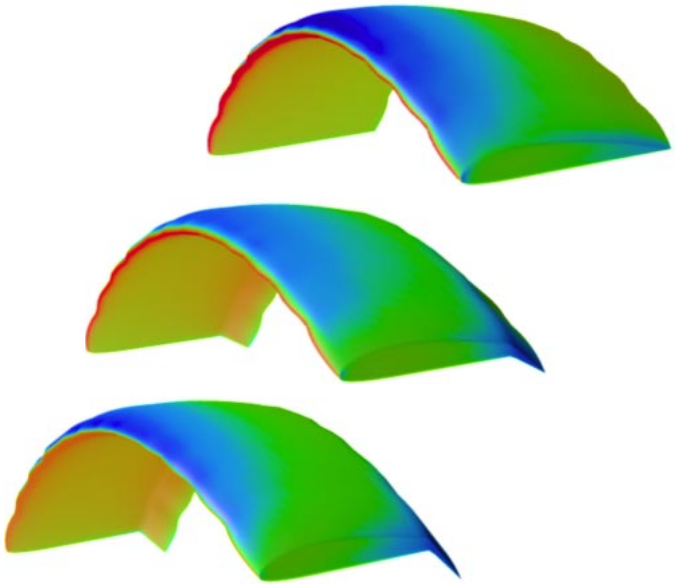

Figure 9. Flare maneuver of a large ram-air parachute. Pressure distribution on the parachute surface during three instants of the flare maneuver.

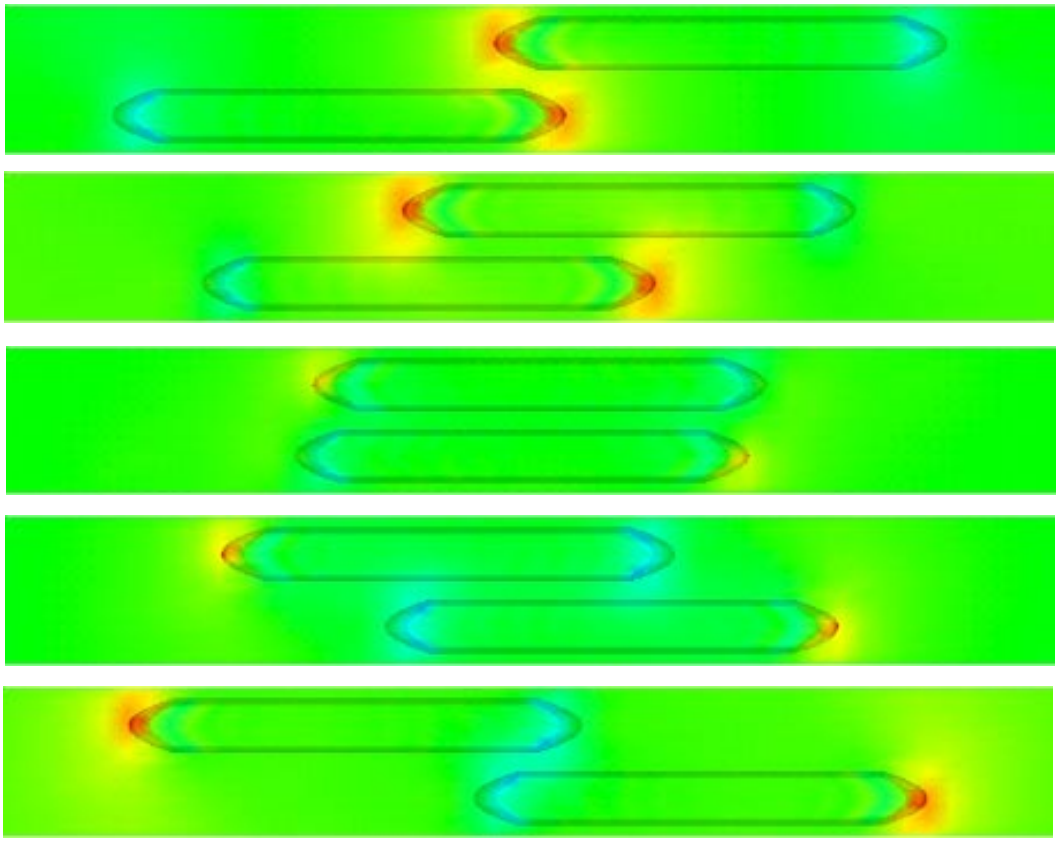

Figure 10. Flow around two high-speed trains in a tunnel. The trains and pressure distribution at different instants.

of -0.025 value of pressure, and on the right, the helicity on the propeller surface and the iso-surface of 0.2 value of helicity. For more on this simulation see [41].

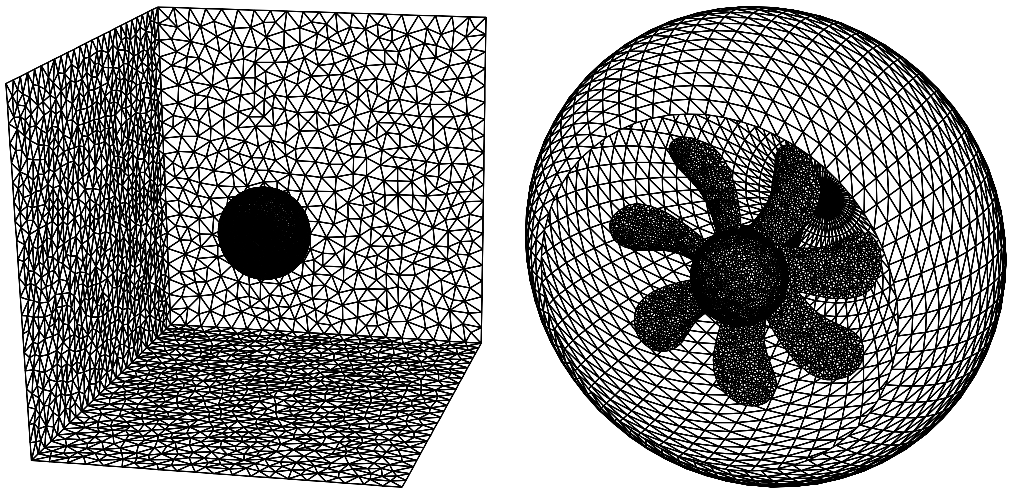

Figure 11. Flow past a rotating propeller. Left: mesh at the outer boundaries and on the outer surface of the shear-slip layer. Right: mesh on the propeller surface and on the inner surface of the shear-slip layer.

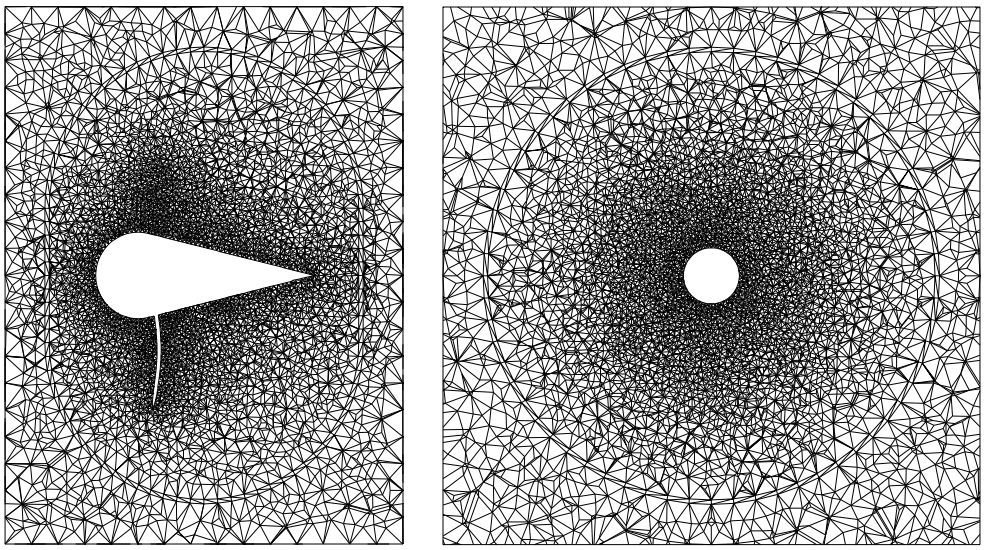

Figure 12. Flow past a rotating propeller. Cross-sectional views of the mesh and the shear-slip layer. Left: side view. Right: top view.

Flow past a helicopter. Here we apply the DSD/SST formulation and SSMUM to computation of flow past a helicopter with its main rotor in motion. In this computation the tail rotor is excluded from the model, although the SSMUM approach could have been used

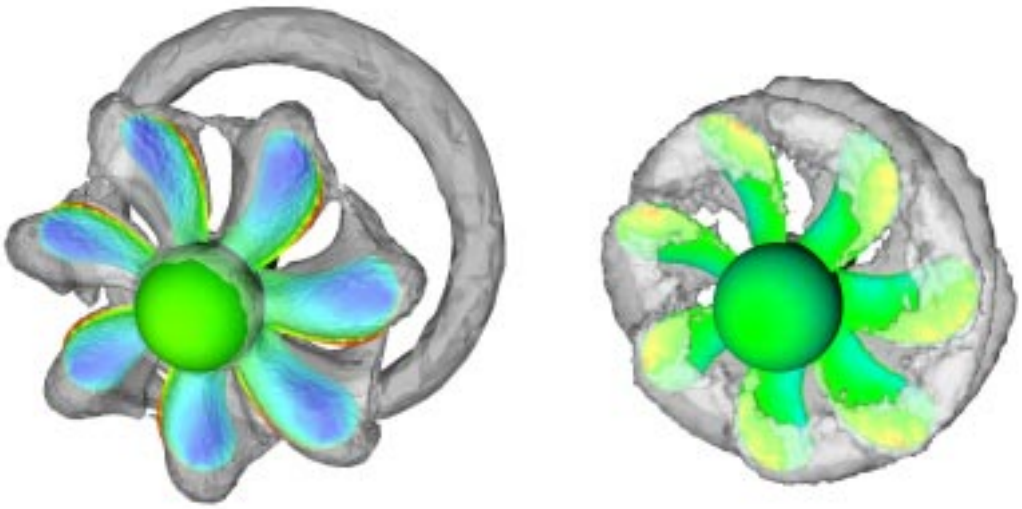

Figure 13. Flow past a rotating propeller. Left: pressure on the propeller surface and the iso-surface of -0.025 value of pressure. Right: helicity on the propeller surface and the iso-surface of 0.2 value of helicity.

to model it too. The mesh consists of 361,434 space-time nodes and 1,096,225 tetrahedral elements. Figure 14 shows the surface meshes for the fuselage, rotor, and the top portion of the inner surface of the shear-slip layer. Figure 15 shows the cross-section of the mesh through the center of the rotor hub. The helicopter is assumed to be in a forward horizontal flight at a speed of 10.0 m/s, with a rotor tip velocity of 200 m/s. Figure 16 shows the air pressure distribution on the surfaces of the fuselage and rotor. For more on this simulation see [42].

Fluid-object interactions with 1000 spheres falling in a liquid-filled tube. The core method for this simulation is the DSD/SST formulation. The methods layered around this include: an efficient distributed-memory implementation of the formulation; fast automatic mesh generation; a mesh update method based on automatic mesh moving with remeshing only as needed; an efficient method for projecting the solution after each remesh; and multi-platform (heterogeneous) computing. Here, while mesh partitioning, flow computations, and mesh movements were performed on a 512-node Thinking Machines CM-5, automatic mesh generation and projection of the solution were accomplished on a 2-processor SGI ONYX2. The two systems communicated over a high-speed network as often as the computation required remeshing. In more recent simulations of this class of problems (see the next numerical example), the CM-5 has been replaced by a CRAY T3E-1200. The spheres, in addition to interacting with the fluid, interact and collide with each other and with the tube wall. The average Reynolds number is around 8. The mesh size is approximately 2.5 million tetrahedral elements, resulting in about 5.5 million coupled, nonlinear equations to be solved every time step. The number of time steps is around 1100 in the simulation. Figure 17 shows the spheres at four different instants during the simulation. The first picture

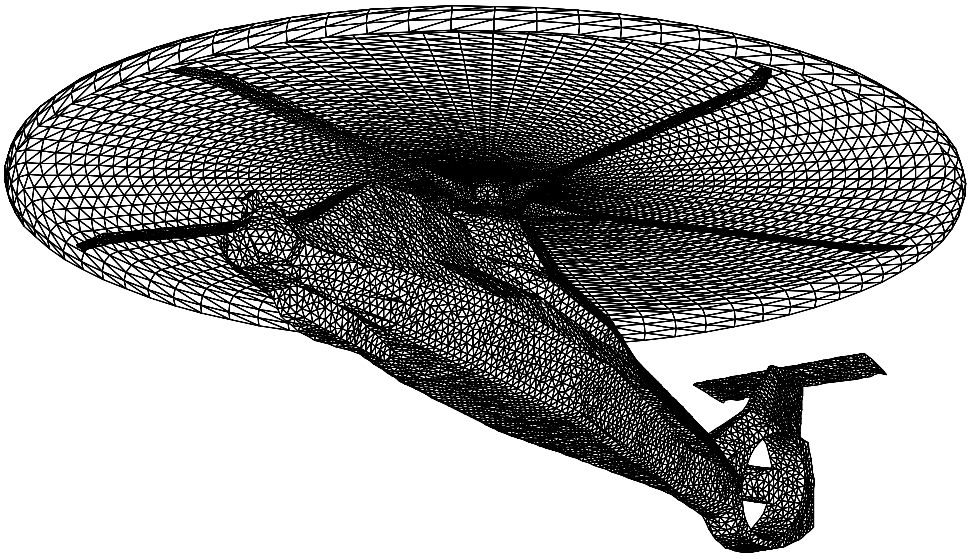

Figure 14. Flow past a helicopter. Surface meshes for the fuselage, rotor, and the top portion of the inner surface of the shear-slip layer.

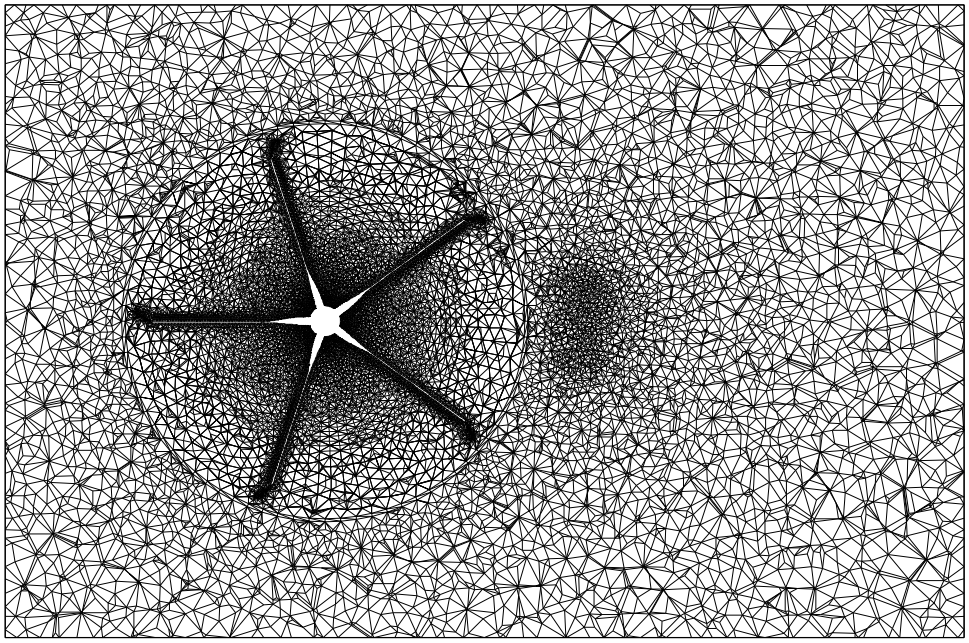

Figure 15. Flow past a helicopter. Cross-section of the mesh through the center of the rotor hub.

shows the initial distribution. The colors are for identification purpose only. For more on this simulation see [62].

Fluid-object interactions in spatially periodic flows. Here we carry out simulation of fluid-object interactions in spatially tri-periodic flows, where the solid objects are spheres with equal radii and fall under the action of gravity. In our study, we keep the volume fraction of the spheres in the fluid-object mixture constant at 26.8%, and investigate how

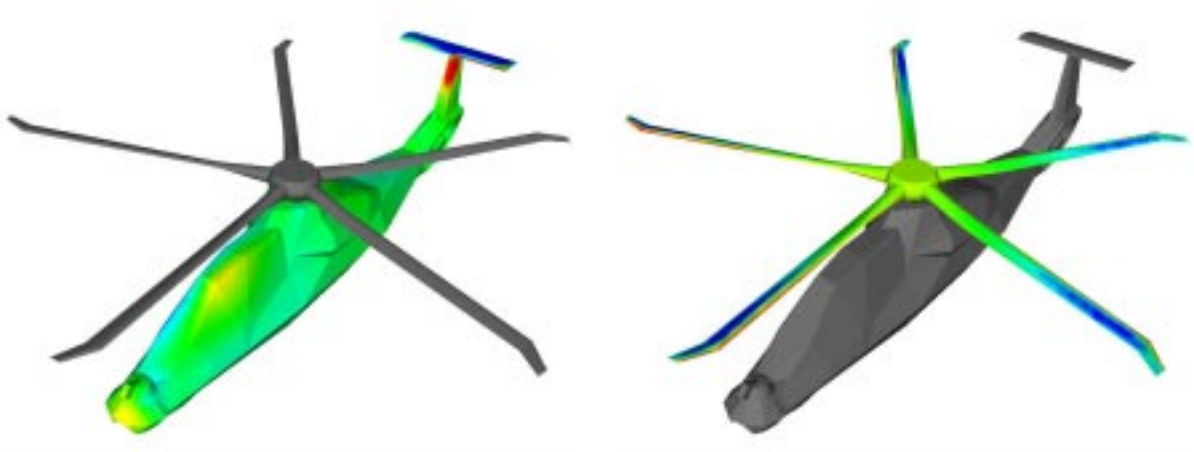

Figure 16. Flow past a helicopter. Air pressure on the surfaces of the fuselage (left) and rotor (right).

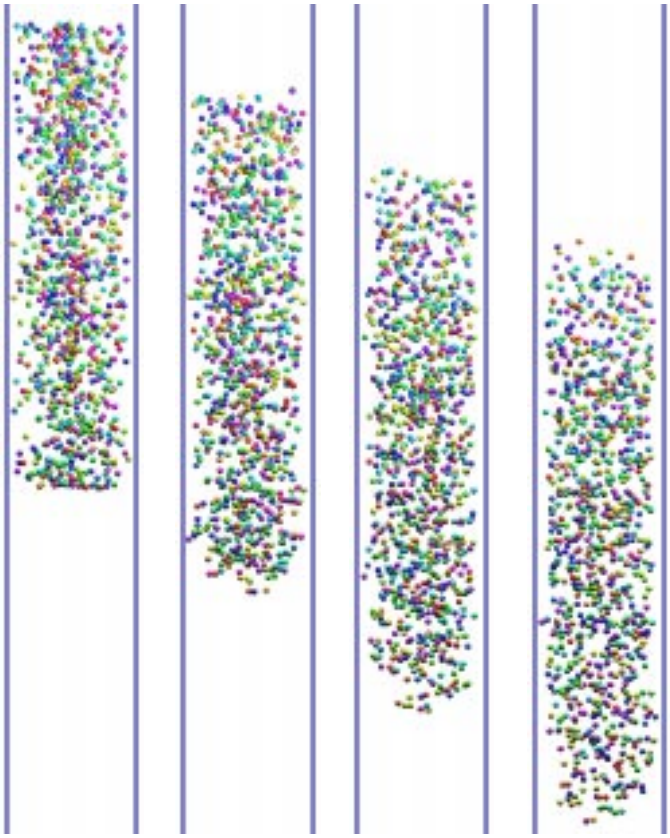

Figure 17. Fluid-object interactions with 1000 spheres falling in a liquid-filled tube. Pictures show at four instants the spheres as they fall.

the mixture behavior varies as the size of the periodic cell is varied to contain more and more spheres. The flow parameters are defined in such a way that a single sphere, at the solid

volume fraction of 26.8%, falls at a terminal Reynolds number of approximately 10. The total volumetric flow rate is set to zero in all three directions, and this is what we expect it to be the case in sedimentation. We computed five cases, where the number of particles in the periodic cell was 1, 8, 27, 64, and 125. The mesh sizes for these five cases were approximately 0.012, 0.120, 0.400, 0.930, and 1.8 million tetrahedral elements, respectively. Here, while the projection of the solution after a remesh, mesh partitioning, flow computations, and mesh movements were performed on a CRAY T3E-1200, the automatic mesh generation was done on a PentiumII based PC. The two systems communicated over a high-speed network as often as the computation required remeshing. The results from these five cases are very similar, even quantitatively, observed in terms of average settling speed, volumetric fluid flow rate, particle drag, and pressure jump values. Figures 18 show at, a vertical cross-section, the velocity vectors. To make the comparison between the results from these five cases easier, except for the last case, we show the velocity vectors over multiple periodic cells. The number of periodic cells assembled together for the cases of 1, 8, 27, 64, and 125 particles in a periodic cell is 5×5, 3×3, 2×2, 2×2, and 1×1, respectively. Figure 19 shows for each case the time history of the average settling speed. One of the notable things is that for all cases the initial grid arrangement of the particles is breaking up at around the same time. For more on this simulation see [34].

Free-surface flow past a bridge support. The Reynolds number based on the upstream velocity is 10 million. The upstream Froude number is 0.564. The mesh consists of 230,480 prism-based space-time elements and 129,382 nodes. The DSD/SST formulation is used with an algebraic mesh update method. The free-surface height is governed by an advection equation and solved with a stabilized formulation. Figure 20 shows, at an instant, the cylinder together with the free-surface color-coded with the velocity magnitude. For more on this simulation see [63].

Flow past the spillway of a dam. The model represents for the dam a 48 feet-wide section of the navigation pass crest and stilling basin. It includes an upstream channel, the spillway crest, and five underwater obstacles intended to dissipate the flow energy. The computation was based on the DSD/SST formulation, with the mesh updated with our automatic mesh moving method. The mesh consists of 139,352 space-time nodes and 396,682 tetrahedral space-time elements. Figure 21 shows the water pressure and streamlines, and the steady shape of the free surface reached in the final stages of the simulation. For more on this simulation see [33].

2D sloshing in a container. A container is filled 2/3 with water and 1/3 with air. It is suddenly subjected to the gravitational acceleration ($g = 9.8$ m/s^2) and a horizontal acceleration of magnitude 0.2g. We compute this problem first with the DSD/SST formulation, where the computational domain is discretized using 6,000 quadrilateral elements and 6,161 nodes. We will refer to this solution as Solution-IT. Next, we compute the problem with the EDICT. The base mesh, Mesh-1, consists of 30,000 triangular elements and 15,251 nodes. Solution-1 is obtained by using the base discretization, where all trial and weighting functions come only from Mesh-1. Solution-2 is obtained by using the EDICT, where all trial and weighting functions come from Mesh-1 $\oplus$ Mesh-2. Solution-3 is obtained by using a more enhanced discretization, where the trial and weighting functions for velocity and pres-

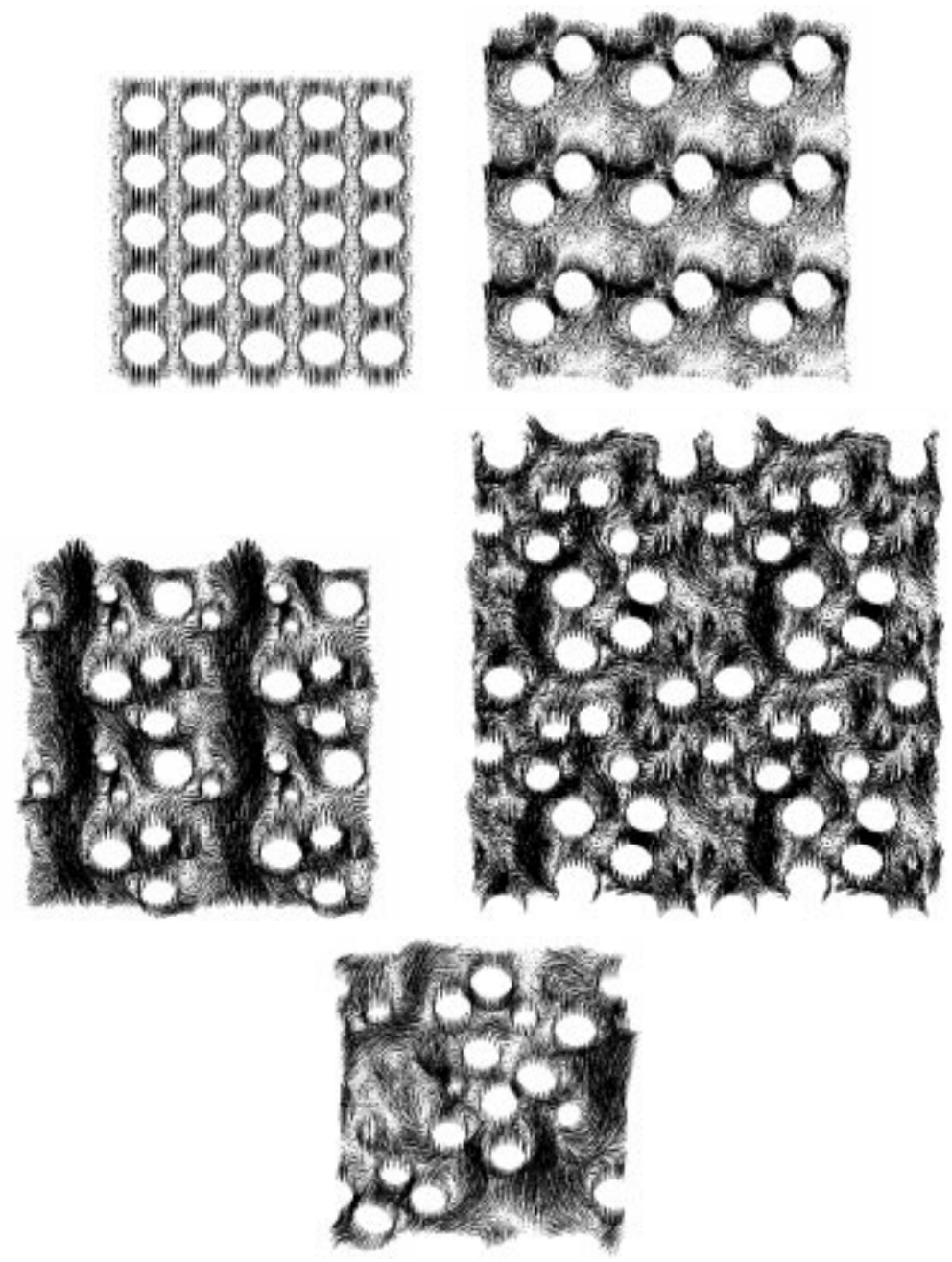

Figure 18. Fluid-object interactions in spatially periodic flows. For the five cases, the velocity vectors at a cross-section. From left to right and top to bottom: 1, 8, 27, 64, and 125 particles/cell.

sure come from Mesh-1 $\oplus$ Mesh-2, and for the interface function from Mesh-1 $\oplus$ Mesh-2 $\oplus$ Mesh-3. Figure 22 shows, on the left, at t = 0.2 s, Mesh-1 together with Mesh-2 and Mesh-3 (both shown on top of Mesh-1); and on the right, the time histories of the horizontal forces exerted on the container for all four solutions. We assume that the target solution is Solution-IT. Solution-1 has significant frequency and amplitude errors. Solution-2 is superior

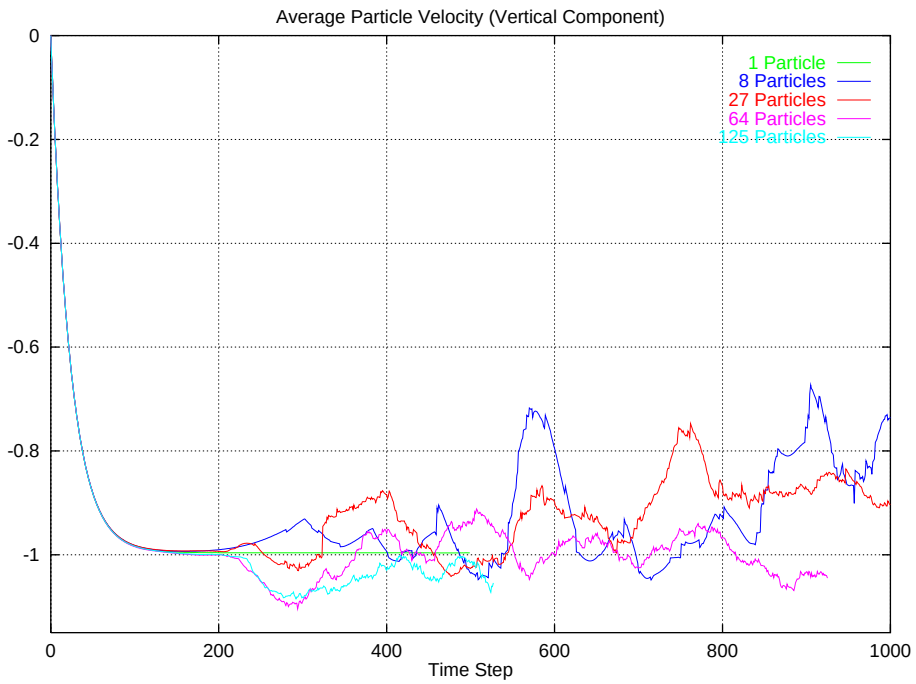

Figure 19. Fluid-object interactions in spatially periodic flows. For each case, the time history of the average settling speed.

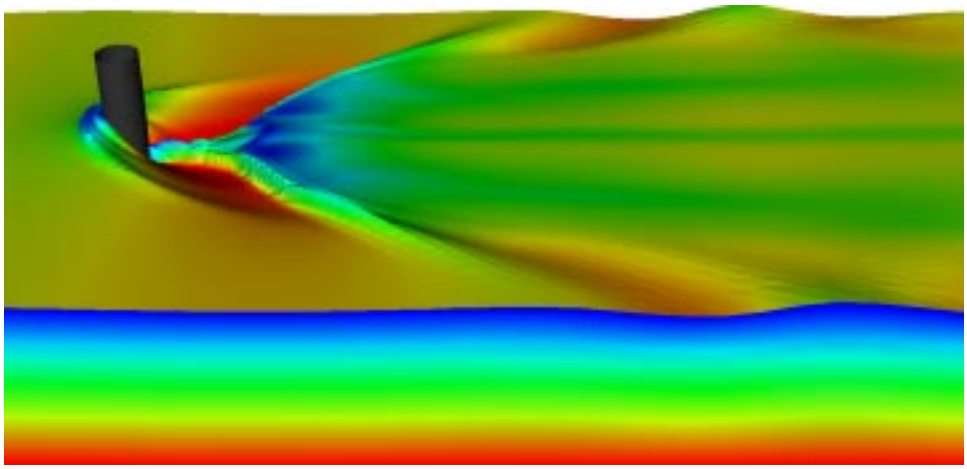

Figure 20. Free-surface flow past a bridge support. The bridge support and the free-surface color-coded with the velocity magnitude.

to Solution-1, and Solution-3 is the one in best agreement with Solution-IT. For more on these simulations see [64].

Liquid-liquid impact. This is an axisymmetric test computation. We have a container in the shape of a circular cylinder. The lower half of the container is filled with a liquid. The upper half is filled with air. At $t = 0.0$ s we start injecting the same liquid through a circular section positioned concentrically at the top of the cylinder. The injection stream has a uniform flow speed. The computation is carried out with the EDICT. The base mesh,

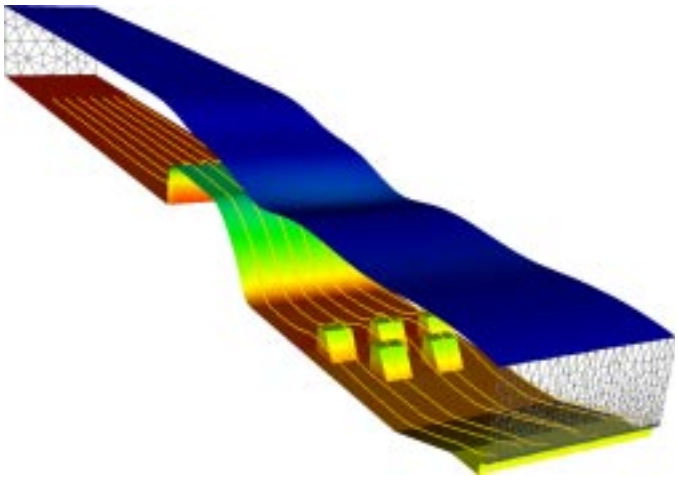

Figure 21. Flow past the spillway of a dam. Water pressure and streamlines, and the steady shape of the free surface.

Mesh-1, consists of 30,000 triangular elements and 15,251 nodes. Mesh-2 and Mesh-3 are generated in the same way as they were generated in the previous test problem. The trial and weighting functions for velocity and pressure come from Mesh-1 $\oplus$ Mesh-2, and for the interface function from Mesh-1 $\oplus$ Mesh-2 $\oplus$ Mesh-3. Figure 23 shows a sequence of air-liquid interactions seen at different instants during the simulation of this problem. The pictures show the injection stream impacting the still liquid, formation of surface waves, and entrapment of air in the liquid. For more on this simulation see [43].

3D sloshing in a container. A container is filled with half water and half air. It is suddenly subjected to the gravitational acceleration, and with both horizontal acceleration components at 0.2g. The base mesh, Mesh-1 consists of $50 \times 30 \times 30$ (45,000) hexahedral elements. For this mesh, at each time step, a coupled, nonlinear equation system with over 236,000 unknowns is solved to obtain the solution. We call this Solution-1. Figure 24 shows for Solution-1, at $t = 0.15$ s and from two different views, Mesh-1 and the air-water interface. Next we compute the problem with the EDICT, with only one level of enhancement, using a Mesh-2 on top of Mesh-1. Mesh-2 has about 85,000 hexahedral elements. For the enhanced discretization, at each time step, a coupled, nonlinear equation system with around 740,000 unknowns is solved to obtain the solution. We call this Solution-2. Figure 25 shows for Solution-2, at $t = 0.15$ s and from two different views, Mesh-2 (on top of Mesh-1) and the air-water interface.

Sloshing in a tanker slowing down on a curved road. The tanker, partially filled with fuel and moving at 30 m/s, starts slowing down at the rate of 0.1g as it enters a curved road with radius of curvature 920 m. The tanker has an elliptical cross section. The dimensions of the minor and major axes are 1.5 m and 2.0 m, respectively. The tanker is partitioned into three compartments, and only one compartment is simulated. The length of each compartment is 1.0 m. The base mesh, Mesh-1 consists of 104,091 nodes and 100,000 hexahedral elements. Mesh-2 contains approximately 180,000 hexahedral elements. Solution-1 is obtained by using the base discretization, where all trial and weighting functions come

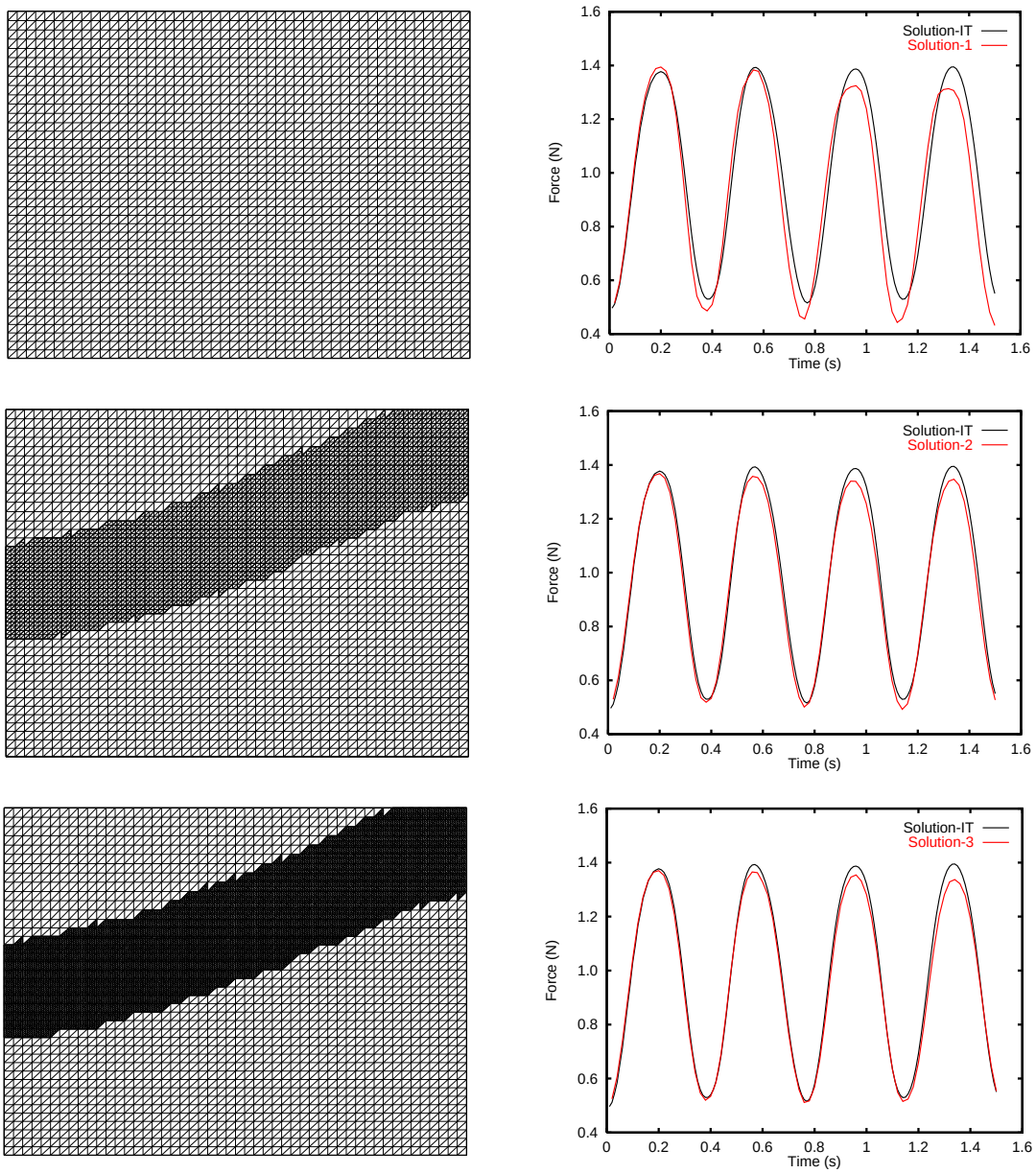

Figure 22. 2D sloshing in a container. Pictures shows, on the left, at t = 0.2 s, Mesh-1 together with Mesh-2 and Mesh-3 (both shown on top of Mesh-1); and on the right, the time histories of the horizontal forces exerted on the container for all four solutions.

only from Mesh-1. Solution-2 is obtained by using the EDICT, where all trial and weighting functions come from Mesh-1 $\oplus$ Mesh-2. Figure 26 shows for Solution-1, at $t = 0.9$ s, Mesh-1 and the air-fuel interface. Figure 27 shows for Solution-2, at $t = 0.9$ s, Mesh-2 (on top of Mesh-1) and the air-fuel interface.

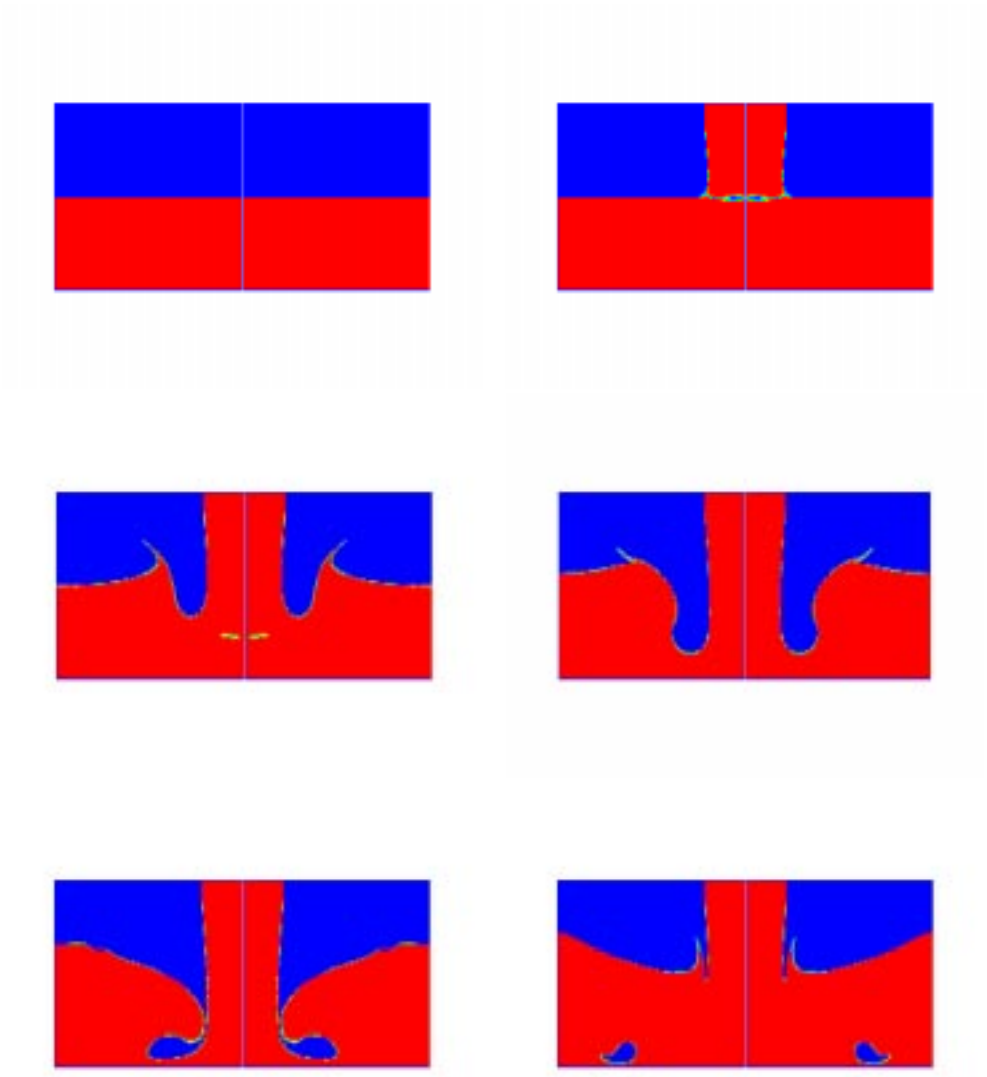

Figure 23. Liquid-liquid impact. Injection stream impacting the still liquid.

20 CONCLUDING REMARKS

We provided an overview of the finite element methods developed in recent years by the Team for Advanced Flow Simulation and Modeling for computation of flow problems with moving boundaries and interfaces. In this category, the classes of problems we were targeting include free-surface and two-fluid flows, fluid-object and fluid-structure interactions, airdrop systems, and flows with rapidly-moving mechanical components. The methods we developed to address these classes of problems can be grouped into two main approaches: interface-tracking techniques and interface-capturing techniques.

The interface-tracking techniques are based on the Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation. In this formulation, the spatial domain occupied by

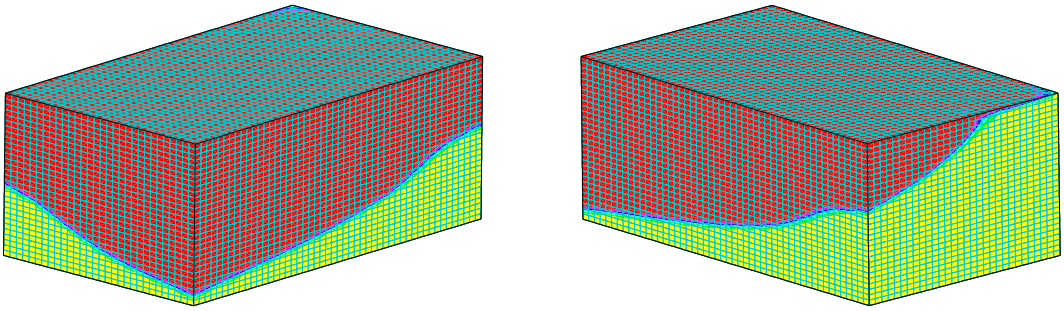

Figure 24. 3D sloshing in a container. Solution-1, at $t = 0.15$ s and from two different views, Mesh-1 and the air-water interface.

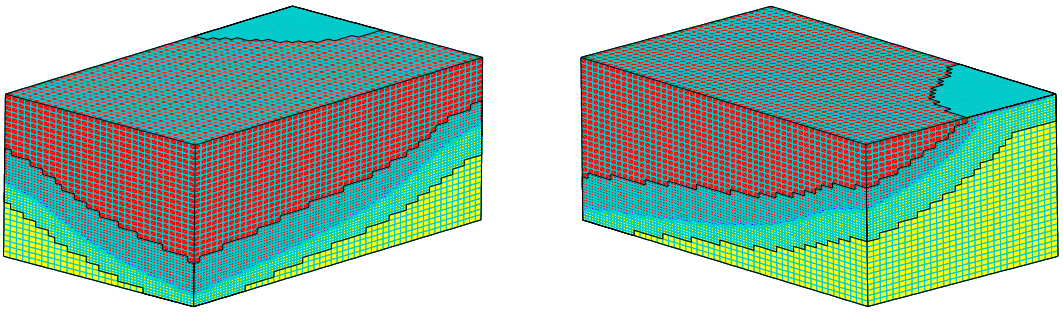

Figure 25. 3D sloshing in a container. Solution-2, at $t = 0.15$ s and from two different views, Mesh-2 (on top of Mesh-1) and the air-water interface.

the fluid changes its shape during the computation to track the interface, and the mesh is continuously updated to handle this change in the spatial domain. In designing the methods needed to update the mesh every time step, we focus on reducing the frequency of remeshing. To this end, we emphasized the development of both the general-purpose and special-purpose mesh update methods.

In developing our interface-capturing techniques, we mainly targeted free-surface and two-fluid flows. These methods are based on stabilized formulation of both the flow equations and the advection equation governing the time-evolution of the interface function marking the location of the interface. These stabilized formulations are discretized over non-moving meshes. We developed the Enhanced-Discretization Interface-Capturing Technique (EDICT) to increase the accuracy in representing the interface. This is for the cases when the accuracy provided by the existing mesh resolution around the interface would not be satisfactory.

The methods we developed are applicable to 3D problems with complex geometries and are suitable for efficient implementation on parallel computers. This makes these methods flexible, accurate, efficient, and powerful. In this paper, we included a large number of

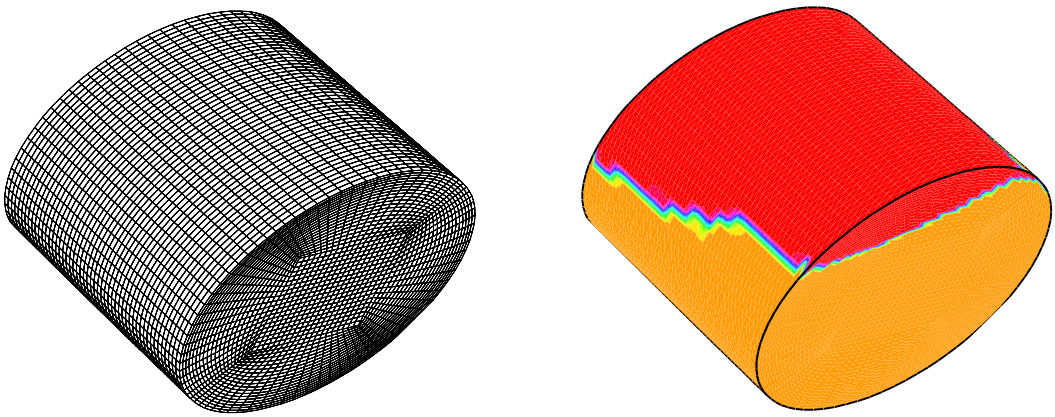

Figure 26. Sloshing in a tanker slowing down on a curved road. Solution-1, at $t = 0.9$ s, Mesh-1 and the air-fuel interface.

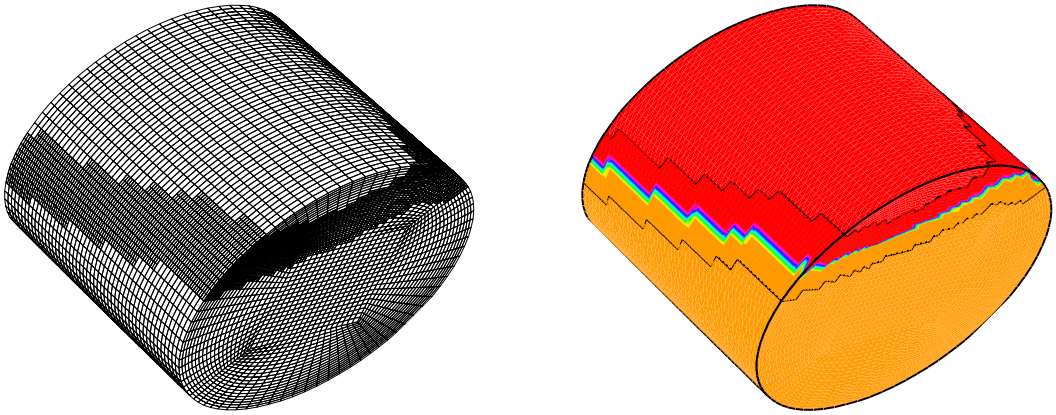

Figure 27. Sloshing in a tanker slowing down on a curved road. Solution-2, at $t = 0.9$ s, Mesh-2 (on top of Mesh-1) and the air-fuel interface.

numerical examples of complex flow problems that we computed on parallel platforms with the methods that have already been implemented for parallel computing. We also included in this paper a number of new ideas and techniques we are proposing to increase the scope and accuracy of these two classes of techniques targeting flows with moving boundaries and interfaces.

ACKNOWLEDGMENT
The work reported in this paper was partially sponsored by NASA JSC (grant no. NAG9-1059), AFOSR (contract no. F49620-98-1-0214), and by the Natick Soldier Center (contract no. DAAD16-00-C-9222). The content does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

REFERENCES

[1] T.E. Tezduyar, "CFD methods for three-dimensional computation of complex flow problems", *Journal of Wind Engineering and Industrial Aerodynamics*, **81** (1999) 97–116.

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

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

[4] T.E. Tezduyar, M. Behr, and J. Liou, "A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: I. The concept and the preliminary numerical tests", *Computer Methods in Applied Mechanics and Engineering*, **94** (1992) 339–351.

[5] T.E. Tezduyar, M. Behr, S. Mittal, and J. Liou, "A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders", *Computer Methods in Applied Mechanics and Engineering*, **94** (1992) 353–371.

[6] 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.

[7] 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.

[8] A.N. Brooks and T.J.R. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations", *Computer Methods in Applied Mechanics and Engineering*, **32** (1982) 199–259.

[9] 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).

[10] G.J. Le Beau and T.E. Tezduyar, "Finite element computation of compressible flows with the SUPG formulation", in *Advances in Finite Element Analysis in Fluid Dynamics*, FED-Vol.123, ASME, New York, (1991) 21–27.

[11] G.J. Le Beau, S.E. Ray, S.K. Aliabadi, and T.E. Tezduyar, "SUPG finite element computation of compressible flows with the entropy and conservation variables formulations", *Computer Methods in Applied Mechanics and Engineering*, **104** (1993) 397–422.

[12] T.E. Tezduyar, S. Mittal, S.E. Ray, and R. Shih, "Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements", *Computer Methods in Applied Mechanics and Engineering*, **95** (1992) 221–242.

[13] 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** (1989) 173–189.

[14] P. Hansbo and A. Szepessy, "A velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations", *Computer Methods in Applied Mechanics and Engineering*, **84** (1990) 175–192.

[15] J. Donea, "A Taylor-Galerkin method for convective transport problems", *International Journal for Numerical Methods in Engineering*, **20** (1984) 101–120.

[16] C. Johnson, U. Navert, and J. Pitkäranta, "Finite element methods for linear hyperbolic problems", *Computer Methods in Applied Mechanics and Engineering*, **45** (1984) 285–312.

[17] C. Johnson and J. Saranen, "Streamline diffusion methods for the incompressible Euler and Navier-Stokes equations", *Mathematics of Computation*, **47** (1986) 1–18.

[18] T.J.R. Hughes, L.P. Franca, and M. Mallet, "A new finite element formulation for computational fluid dynamics: VI. Convergence analysis of the generalized SUPG formulation for linear time-dependent multi-dimensional advective-diffusive systems", *Computer Methods in Applied Mechanics and Engineering*, **63** (1987) 97–112.

[19] S.K. Aliabadi, S.E. Ray, and T.E. Tezduyar, "SUPG finite element computation of compressible flows with the entropy and conservation variables formulations", *Computational Mechanics*, **11** (1993) 300–312.

[20] 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.

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

[22] M. Lesoinne and C. Farhat, "Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations", *Computer Methods in Applied Mechanics and Engineering*, **134** (1996) 71–90.

[23] R. Lohner, C. Yang, and J.D. Baum, "Rigid and flexible store separation simulations using dynamic adaptive unstructured grid technologies", in *Proceedings of the First AFOSR Conference on Dynamic Motion CFD*, New Brunswick, New Jersey, (1996).

[24] K. Stein, R. Benney, V. Kalro, T. Tezduyar, and J. Leonard and M. Accorsi, "Parachute fluid–structure interactions: 3-D computation", in *Proceedings of the 4th Japan-US Symposium on Finite Element Methods in Large-Scale Computational Fluid Dynamics*, Tokyo, Japan, (1998).

[25] J. Garcia, E. Onate, H. Sierra, and S. Idelsohn, "A stabilized numerical method for analysis of ship hydrodynamics", in *Proceedings of ECCOMAS 1998*. J. Wiley, (1998).

[26] E. Onate, S. Idelsohn, C. Sacco, and J. Garcia, "Stabilization of the numerical solution for the free surface wave equation in fluid dynamics", in *Proceedings of ECCOMAS 1998*. J. Wiley, (1998).

[27] E. Onate and J. Garcia, "A methodology for analysis of fluid–structure interaction accounting for free surface waves", in *Proceedings of European Conference on Computational Mechanics*, Munich, Germany, (1999).

[28] M. Cruchaga and E. Onate, "A generalized streamline finite element approach for the analysis of incompressible flow problems including moving surfaces", *Computer Methods in Applied Mechanics and Engineering*, **173** (1999) 241–255.

[29] S.K. Aliabadi and T.E. Tezduyar, "Space–time finite element computation of compressible flows involving moving boundaries and interfaces", *Computer Methods in Applied Mechanics and Engineering*, **107** (1993) 209–223.

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

[31] A.A. Johnson and T.E. Tezduyar, "Parallel computation of incompressible flows with complex geometries", *International Journal for Numerical Methods in Fluids*, **24** (1997) 1321–1340.

[32] A.A. Johnson and T.E. Tezduyar, "Simulation of multiple spheres falling in a liquid-filled tube", *Computer Methods in Applied Mechanics and Engineering*, **134** (1996) 351–373.

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

[34] A. Johnson and T. Tezduyar, "Methods for 3D computation of fluid-object interactions in spatially-periodic flows", *Computer Methods in Applied Mechanics and Engineering*, **190** (2001) 3201–3221.

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

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

[37] 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.

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

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

[40] J.A. Benek, P.G. Buning, and J.L. Steger, "A 3-D Chimera grid embedding technique", Paper 85-1523, AIAA, 1985.

[41] M. Behr and T. Tezduyar, "The Shear-Slip Mesh Update Method", *Computer Methods in Applied Mechanics and Engineering*, **174** (1999) 261–274.

[42] M. Behr and T. Tezduyar, "Shear-slip mesh update in 3D computation of complex flow problems with rotating mechanical components", *Computer Methods in Applied Mechanics and Engineering*, **190** (2001) 3189–3200.

[43] 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.

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

[45] 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.

[46] D. L. Youngs, "Time-dependent multimaterial flow with large fluid distortion", in K. W. Morton and M. J. Baines, editors, *Numerical Methods in Fluid Dynamics*, Notes on Numerical Fluid Mechanics, 273–285, Academic Press, New York, 1984.

[47] C.M. Lemos, "Higher-order schemes for free-surface flows with arbitrary configurations", *International Journal for Numerical Methods in Fluids*, **23** (1996) 545–566.

[48] 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.

[49] S. Aliabadi and T.E. Tezduyar, "Stabilized-Finite-Element/Interface-Capturing Technique for parallel computation of unsteady flows with interfaces", *Computer Methods in Applied Mechanics and Engineering*, **190** (2000) 243–261.

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

[51] T. 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).

[52] T. 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.

[53] R.G. Dean and R.A. Dalrymple, *Water Wave Mechanics for Engineers and Scientists*. Prentice-Hall, 1984.

[54] 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.

[55] T. 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*, Chapter 10, 363–398, John Wiley & Sons, 1997.

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

[57] 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** (1991) 281–304.

[58] 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** (1995) 877–884.

[59] 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** (1992) 27–42.

[60] J. Smagorinsky, "General circulation experiments with the primitive equations", *Monthly Weather Review*, **91** (1963) 99–165.

[61] C. Kato and M. Ikegawa, "Large eddy simulation of unsteady turbulent wake of a circular cylinder using the finite element method", in I. Celik, T. Kobayashi, K.N. Ghia, and J. Kurokawa, editors, *Advances in Numerical Simulation of Turbulent Flows*, FED-Vol.117, ASME, New York, (1991) 49–56.

[62] A.A. Johnson and T.E. Tezduyar, "Advanced mesh generation and update methods for 3D flow simulations", *Computational Mechanics*, **23** (1999) 130–143.

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

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

editors, *Advances in Finite Element Analysis in Fluid Dynamics*, FED-Vol.123, ASME, New York, (1991) 21–27.

[11] G.J. Le Beau, S.E. Ray, S.K. Aliabadi, and T.E. Tezduyar, "SUPG finite element computation of compressible flows with the entropy and conservation variables formulations", *Computer Methods in Applied Mechanics and Engineering*, **104** (1993) 397–422.

[12] T.E. Tezduyar, S. Mittal, S.E. Ray, and R. Shih, "Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements", *Computer Methods in Applied Mechanics and Engineering*, **95** (1992) 221–242.

[13] 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** (1989) 173–189.

[14] P. Hansbo and A. Szepessy, "A velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations", *Computer Methods in Applied Mechanics and Engineering*, **84** (1990) 175–192.

[15] J. Donea, "A Taylor-Galerkin method for convective transport problems", *International Journal for Numerical Methods in Engineering*, **20** (1984) 101–120.

[16] C. Johnson, U. Navert, and J. Pitkäranta, "Finite element methods for linear hyperbolic problems", *Computer Methods in Applied Mechanics and Engineering*, **45** (1984) 285–312.

[17] C. Johnson and J. Saranen, "Streamline diffusion methods for the incompressible Euler and Navier-Stokes equations", *Mathematics of Computation*, **47** (1986) 1–18.

[18] T.J.R. Hughes, L.P. Franca, and M. Mallet, "A new finite element formulation for computational fluid dynamics: VI. Convergence analysis of the generalized SUPG formulation for linear time-dependent multi-dimensional advective-diffusive systems", *Computer Methods in Applied Mechanics and Engineering*, **63** (1987) 97–112.

[19] S. Aliabadi, S.E. Ray, and T.E. Tezduyar, "SUPG finite element computation of compressible flows with the entropy and conservation variables formulations", *Computational Mechanics*, **11** (1993) 300–312.

[20] 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.

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

[22] M. Lesoinne and C. Farhat, "Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations", *Computer Methods in Applied Mechanics and Engineering*, **134** (1996) 71–90.

[23] R. Lohner, C. Yang, and J.D. Baum, "Rigid and flexible store separation simulations using dynamic adaptive unstructured grid technologies", in *Proceedings of the First AFOSR Conference on Dynamic Motion CFD*, New Brunswick, New Jersey, (1996).

[24] K. Stein, R. Benney, V. Kalro, T. Tezduyar, and J. Leonard and M. Accorsi, "Parachute fluid-structure interactions: 3-D computation", in *Proceedings of the 4th Japan-US Symposium on Finite Element Methods in Computational Fluid Dynamics*, Tokyo, Japan, (1998).

[25] J. Garcia, E. Onate, H. Sierra, and S. Idelsohn, "A stabilized numerical method for analysis of ship hydrodynamics", in *Proceedings of ECCOMAS 1998*. J. Wiley, (1998).

[26] E. Onate, S. Idelsohn, C. Sacco, and J. Garcia, "Stabilization of the numerical solution for the free surface wave equation in fluid dynamics", in *Proceedings of ECCOMAS 1998*. J. Wiley, (1998).

[27] E. Onate and J. Garcia, "A methodology for analysis of fluid-structure interaction accounting for free surface waves", in *Proceedings of European Conference on Computational Mechanics*, Munich, Germany, (1999).

[28] M. Cruchaga and E. Onate, "A generalized streamline finite element approach for the analysis of incompressible flow problems including moving surfaces", *Computer Methods in Applied Mechanics and Engineering*, **173** (1999) 241–255.

[29] S. Aliabadi and T.E. Tezduyar, "Space-time finite element computation of compressible flows involving moving boundaries and interfaces", *Computer Methods in Applied Mechanics and Engineering*, **107** (1993) 209–224.

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

[31] A.A. Johnson and T.E. Tezduyar, "Parallel computation of incompressible flows with complex geometries", *International Journal for Numerical Methods in Fluids*, **24** (1997) 1321–1340.

[32] A.A. Johnson and T.E. Tezduyar, "Simulation of multiple spheres falling in a liquid-filled tube", *Computer Methods in Applied Mechanics and Engineering*, **134** (1996) 351–373.

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

[34] A.A. Johnson and T.E. Tezduyar, "Methods for 3D computation of fluid-object interactions in spatially-periodic flows", to appear in *Computer Methods in Applied Mechanics and Engineering*, 2000.

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

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

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

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

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

[40] J.A. Benek, P.G. Buning, and J.L. Steger, "A 3-D Chimera grid embedding technique", Paper 85-1523, AIAA, 1985.

[41] M. Behr and T.E. Tezduyar, "Shear-Slip Mesh Update Method", *Computer Methods in Applied Mechanics and Engineering*, **174** (1999) 261–274.

[42] M. Behr and T.E. Tezduyar, "Shear-slip mesh update in 3D computation of complex flow problems with rotating mechanical components", to appear in *Computer Methods in Applied Mechanics and Engineering*, 2000.

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

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

[45] 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.

[46] D. L. Youngs, "Time-dependent multimaterial flow with large fluid distortion", in K. W. Morton and M. J. Baines, editors, *Numerical Methods in Fluid Dynamics*, Notes on Numerical Fluid Mechanics, 273–285, Academic Press, New York, 1984.

[47] C.M. Lemos, "Higher-order schemes for free-surface flows with arbitrary configurations", *International Journal for Numerical Methods in Fluids*, **23** (1996) 545–566.

[48] T.E. Tezduyar and Y. Osawa, "Finite element stabilization parameters computed from element matrices and vectors", to appear in *Computer Methods in Applied Mechanics and Engineering*, 2000.

[49] S. Aliabadi and T. Tezduyar, "Stabilized-Finite-Element/Interface-Capturing Technique for parallel computation of unsteady flows with interfaces", to appear in *Computer Methods in Applied Mechanics and Engineering*, 2000.

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

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

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

[53] R.G. Dean and R.A. Dalrymple, *Water Wave Mechanics for Engineers and Scientists*. Prentice-Hall, 1984.

[54] 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.

[55] 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*, 363–398, John Wiley & Sons, 1996.

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

[57] 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** (1991) 281–304.

[58] 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** (1995) 877–884.

[59] 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** (1992) 27–42.

[60] J. Smagorinsky, "General circulation experiments with the primitive equations", *Monthly Weather Review*, **91** (1963) 99–165.

[61] C. Kato and M. Ikegawa, "Large eddy simulation of unsteady turbulent wake of a circular cylinder using the finite element method", in I. Celik, T. Kobayashi, K.N. Ghia, and J. Kurokawa, editors, *Advances in Numerical Simulation of Turbulent Flows*, FED-Vol.117, ASME, New York, (1991) 49–56.

[62] A.A. Johnson and T.E. Tezduyar, "Advanced mesh generation and update methods for 3D flow simulations", *Computational Mechanics*, **23** (1999) 130–143.

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

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