# COMPUTATIONAL METHODS FOR ENVIRONMENTAL FLUID MECHANICS

**Tayfun Tezduyar** – tezduyar@rice.edu
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

**Abstract.** *This is a review of some of the methods developed by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [*http://www.mems.rice.edu/TAFSM/*] in the recent past. These methods enable us to address certain classes of problems in environmental fluid mechanics, such as unsteady flows with interfaces and air circulation and contaminant dispersion. The methods we developed for two fluid-interfaces are in both interface tracking (moving mesh) and interface-capturing (non-moving mesh) categories, and which one is more effective to use depends on the nature of the class of problems being addressed. Advanced mesh moving methods that reduce the frequency of remeshing have been developed for mesh update in interface-tracking techniques. In the case of the interface-capturing approach, special methods have been developed to increase the accuracy in representing the interface. These methods have been designed and implemented for parallel computing, and therefore are suitable for simulation of complex, large-scale problems. The numerical examples we highlight here cover problems from air circulation and contaminant dispersion, flows past river dams, and free-surface flows with complex and very unsteady free-surface configurations.*

**Keywords:** *Two-fluid interfaces; Interface tracking; Interface capturing.*

## 1. INTRODUCTION

In the recent past, the Team for Advanced Flow Simulation and Modeling (T⋆AFSM) [http://www.mems.rice.edu/TAFSM/] developed a number of computational methods that enable us to address certain classes of applications in environmental fluid mechanics. These applications include those in the categories of unsteady flows with interfaces and air circulation and contaminant dispersion.

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 the water falling down over a river dam is a challenging application in this class of problems. The main challenge here is that the location of the free-surface is also an unknown and must be determined together with the solution of the Navier-Stokes equations. This class of problems is a subset of a larger class flow problems with moving boundaries and interfaces, where the location of these boundaries or interfaces is an unknown that needs to be computed as part of the overall solution.

An example of air circulation and contaminant dispersion problems is how air circulation in a subway station can be simulated and how a contaminant introduced in that subway station spreads under those air circulation conditions. This class of simulations involve solution of the Navier-Stokes equations interior or exterior to some complex geometries. After the flow field is determined, using the velocity field computed, a time-dependent advection-diffusion equation is solved to compute the time-evolution of the passive contaminant.

The methods developed to support simulation and modeling of the classes of problems described above include: special numerical stabilization methods, methods for moving boundaries and interfaces, advanced mesh update methods, iterative solution techniques for large nonlinear equation systems that need to be solved at every time step of a computation, and parallel implementations. All methods developed are for flow problems involving complex geometries, and all software was developed and implemented on parallel platforms by the T$\star$AFSM.

What method we propose to use for a specific class of problems depends on the nature of that class of problems. For general flow problems with free surfaces or two-fluid interfaces, stabilized finite element interface-capturing techniques (see [1]) can be used effectively in cases where the geometry is complex and the interface is also complex and unsteady. The Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation (see [2]), on the other hand, can be be used effectively with relatively coarser meshes if the interface is not as complex or as unsteady. For comparable mesh refinement, the DSD/SST formulation results in more accurate representation of the interface compared to an interface-capturing technique (ICT). On the other hand, an ICT is more flexible and applicable to a larger class of problems compared to the DSD/SST formulation. The Enhanced-Discretization Interface-Capturing Technique (EDICT) [1] can attain increased accuracy at the interface while maintaining flexibility. The contaminant dispersion problems are simulated by solving, in addition to the flow equations, the time-dependent advection-diffusion equation governing the transport of the contaminant. The advection-diffusion equation is solved with a stabilized formulation. This formulation can be supplemented with the EDICT for more accurate computation of the contaminant-air interfaces.

## 2. GOVERNING EQUATIONS

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

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}\mathbf{u} - \mathbf{f}) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} = 0 \quad \text{on } \Omega_t \quad \forall t \in (0,T), \tag{1}$$

$$\boldsymbol{\nabla}\cdot\mathbf{u} = 0 \quad \text{on } \Omega_t \quad \forall t \in (0,T), \tag{2}$$

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

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

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

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

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

$$\begin{aligned} \mathbf{u} &= \mathbf{g} \text{ on } (\Gamma_\mathrm{t})_\mathrm{g}, \\ \mathbf{n}\cdot\boldsymbol{\sigma} &= \mathbf{h} \text{ on } (\Gamma_\mathrm{t})_\mathrm{h}. \end{aligned} \tag{5}$$

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

If the problem does not involve any moving boundaries or interfaces, the spatial domain does not need to change with respect to time, and the subscript $t$ can be dropped from $\Omega_t$ and $\Gamma_t$. 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 $\rho_A$ and $\rho_B$ and viscosities $\mu_A$ and $\mu_B$. With this approach, we can also model a liquid-gas interaction by letting, for example, Fluid A be the liquid and Fluid B the gas. If we are modeling a free-surface problem where Fluid B is irrelevant, we simply assign to Fluid B a sufficiently low density.

In these models, an interface function $\phi$ 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, $\rho$ and $\mu$ are defined as

$$\rho = \phi\rho_A + (1-\phi)\rho_B, \tag{6}$$

$$\mu = \phi\mu_A + (1-\phi)\mu_B. \tag{7}$$

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

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

## 3. INTERFACE-TRACKING AND INTERFACE-CAPTURING

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 which "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 stabilized finite element interface-capturing techniques described in [1] fall into this category of methods.

The Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation is an interface-tracking method, and was first introduced in [2–4]. 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 interface-tracking and interface-capturing methods described in this paper are based on stabilization techniques. These stabilization techniques are the streamline-upwind/Petrov-Galerkin (SUPG) [5], pressure-stabilizing/Petrov-Galerkin (PSPG) [2], and Galerkin/least-squares (GLS) [6] formulations. The PSPG formulation assures numerical stability while allowing us to use equal-order interpolation functions for velocity and pressure and other unknowns.

## 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 $t_{n+1}$. At each time step, the integrations involved in 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 trial function spaces for velocity and pressure will be denoted by $(\hat{\mathcal{S}}_{\mathbf{u}}^h)_n$ and $(\hat{\mathcal{S}}_p^h)_n$. The weighting function spaces corresponding to momentum equation and incompressibility constraint will be denoted by $(\hat{\mathcal{V}}_{\mathbf{u}}^h)_n$ and $(\hat{\mathcal{V}}_p^h)_n$ $(= (\hat{\mathcal{S}}_p^h)_n)$. The DSD/SST formulation of Equations (1) and (2) can be written as follows: given $(\mathbf{u}^h)_n^-$, find $\mathbf{u}^h \in (\hat{\mathcal{S}}_{\mathbf{u}}^h)_n$ and $p^h \in (\hat{\mathcal{S}}_p^h)_n$ such that $\forall \mathbf{w}^h \in (\hat{\mathcal{V}}_{\mathbf{u}}^h)_n$ and $\forall q^h \in (\hat{\mathcal{V}}_p^h)_n$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left( \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
&+ \int_{Q_n} q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho \left( (\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^- \right) d\Omega \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\text{LSME}} \frac{1}{\rho} \left[ \rho \left( \frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h) \right] \\
&\qquad \cdot \left[ \rho \left( \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f} \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \right] dQ \\
&+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\text{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h \, \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ = \int_{P_n} \mathbf{w}^h \cdot \mathbf{h}^h dP.
\end{aligned} \tag{9}$$

Here $\tau_{\text{LSME}}$ and $\tau_{\text{LSIC}}$ are the stabilization parameters.

The solution is obtained sequentially for all space-time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, and the computations start with

$$(\mathbf{u}^h)_0^- = \mathbf{u}_0^h. \tag{10}$$

The first four integrals, together with the right-hand-side, represent the time-discontinuous Galerkin formulation of (1)–(2), 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.

## 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 essential 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 much 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. Automatic mesh generation might become an overwhelming cost especially when the number of elements become 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.

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 general, we use an automatic mesh moving scheme [7] 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. The structured layers of elements generated around solid objects 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.

## 6. 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 [1]). 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 introduced in [1] 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 [8]. The Navier-Stokes equations are solved over a non-moving mesh with an interface function serving as a marker identifying the two fluids.

In writing the stabilized finite element formulation for the EDICT (see [1]), 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.

The stabilized formulations of Equations (1), (2), and (8) can be written as follows: given $\mathbf{u}^h_n$ and $\phi^h_n$, find $\mathbf{u}^h_{n+1} \in (\mathcal{S}^h_{\mathbf{u}})_{n+1}$, $p^h_{n+1} \in (\mathcal{S}^h_p)_{n+1}$, and $\phi^h_{n+1} \in (\mathcal{S}^h_\phi)_{n+1}$, such that, $\forall \mathbf{w}^h_{n+1} \in (\mathcal{V}^h_{\mathbf{u}})_{n+1}$, $\forall q^h_{n+1} \in (\mathcal{V}^h_p)_{n+1}$, and $\forall \psi^h_{n+1} \in (\mathcal{V}^h_\phi)_{n+1}$:

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

$$\begin{aligned}
&\int_\Omega \psi^h_{n+1} \left( \frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_{\Omega^e} \tau_\phi \mathbf{u}^h \cdot \boldsymbol{\nabla} \psi^h_{n+1} \left( \frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \phi^h \right) d\Omega = 0, \qquad (12)
\end{aligned}$$

where $\tau_{\text{SUPG}}$, $\tau_{\text{PSPG}}$, $\tau_{\text{LSIC}}$ and $\tau_\phi$ are the stabilization parameters:

$$
\begin{aligned}
\tau_{\mathrm{SUPG}} &= \left( \left( \frac{2 \parallel \mathbf{u}^h \parallel}{h} \right)^2 + \left( \frac{4\nu}{h^2} \right)^2 \right)^{-\frac{1}{2}}, && (13)\\
\tau_{\mathrm{PSPG}} &= \tau_{\mathrm{SUPG}}, && (14)\\
\tau_{\mathrm{CONT}} &= \frac{h}{2} \parallel \mathbf{u}^h \parallel z, && (15)\\
& \quad \text{where } z = \begin{cases} \left(\frac{Re_u}{3}\right) & Re_u \leq 3 \\ 1 & Re_u > 3 \end{cases}, \\
\tau_{\phi} &= \frac{h}{2 \parallel \mathbf{u}^h \parallel}, && (16)
\end{aligned}
$$

where $Re_u$ is the cell Reynolds number.

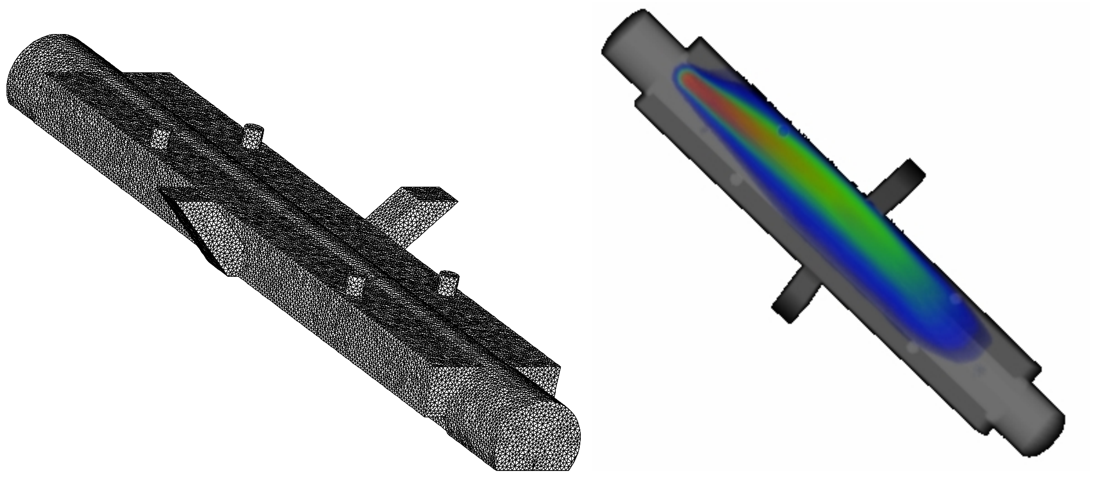
In Equation (11), the first three integrals, together with the right-hand-side, represent the Galerkin formulation of (1)-(2). The first series of element-level integrals in the formulation are the SUPG and PSPG stabilization terms. The second series of element-level integrals are the least-squares stabilization terms based on the incompressibility constraint. In Equation (12), the first integral represents the Galerkin formulation of (8), while the series of element-level integrals are the SUPG stabilization terms.

To increase the accuracy in representing the interface, we use function spaces corresponding to enhanced discretization at and near the interface. A subset of the elements in the base mesh, Mesh-1, are identified as those at and near the interface. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. For each element in this subset there will be a unique second-level mesh. If an automatic mesh generator is used to generate that, the mesh will be generated only once and stored, to be used later if that element needs a second-level mesh again. The trial and weighting functions for velocity and pressure will all have two components each: one coming from Mesh-1 and the second one coming from Mesh-2. To further increase the accuracy in representing the interface, we construct a third-level mesh, Mesh-3, for the interface function only. This is done by identifying a subset of the elements in Mesh-2 as those at and very near the interface. The construction of Mesh-3 from Mesh-2 will be very similar to the construction of Mesh-2 from Mesh-1. The trial and weighting functions for the interface function will have three components, each coming from one of these three meshes.

We re-define the subsets over which we build Mesh-2 and Mesh-3 not every time step but with sufficient frequency to keep the interface within the zones covered by these subsets of elements.

## 7. NUMERICAL EXAMPLES

**Contaminant dispersion in a model subway station.** The subway station has two entrances on each side and four vents located on the upper surface. This 3D parallel computation is carried out in two stages. First, the Navier-Stokes equations are solved to obtain the flow velocity field. This velocity field is used in the second stage in the time-dependent contaminant advection-diffusion equation to obtain time-evolution of the concentration of the contaminant. The contaminant is released from a point source with constant strength. The unstructured mesh used in this simulation consists of 187,612 nodes and 1,116,992 tetrahedral elements. The steady-state solution of the flow equations is obtained by solving over 0.65 million coupled, nonlinear equations at every pseudo-time step. For the contaminant dispersion, at every time step, we solve a linear system with more than 0.15 million equations. Figure 1 shows shows the mesh and the contaminant concentration at an instant. For more on this simulation see [9].



Mesh Contaminant Conc

Figure 1. Contaminant dispersion in a model subway station.

**Flow past the spillway of a dam.** The model represents a 48 feet-wide section of the navigation pass crest and stilling basin. It includes a long upstream channel, the spillway crest, and a set of underwater obstacles designed to dissipate the flow energy. This 3D parallel computation is based on the DSD/SST formulation. The mesh is updated using our automatic mesh moving method. The mesh consists of 139,352 space-time nodes and 396,682 tetrahedral space-time elements. Figure 2 shows the water pressure and streamlines, and the steady shape of the free surface achieved in the final stages of the simulation. For more on this simulation see [9].

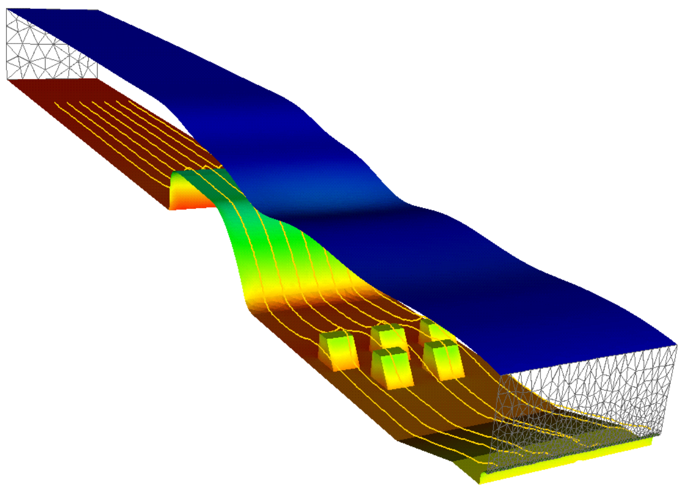


Figure 2. Flow past the spillway of a dam. Water pressure and streamlines, and the steady shape of the free surface.

**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, 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 3 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 [1].

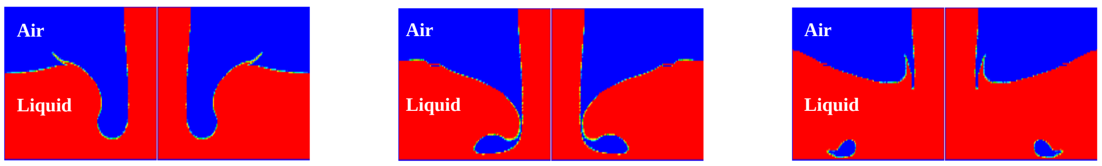


Figure 3. Liquid-liquid impact. Injection stream impacting the still liquid.

## REFERENCES

[1] 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.

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

[3] 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 tests", *Computer Methods in Applied Mechanics and Engineering*, **94** (1992) 339–351.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.