Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 193 (2004) 1385–1401
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Enhanced-discretization space–time technique (EDSTT)

Tayfun E. Tezduyar *, Sunil Sathe

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

Received 7 January 2003; received in revised form 20 August 2003; accepted 9 December 2003

Abstract

The enhanced-discretization space–time technique (EDSTT) was developed for the purpose of being able to, in the context of a space–time formulation, enhance the time-discretization in regions of the fluid domain requiring smaller time steps. Such requirements are often encountered in time-accurate computations of fluid–structure interactions, where the time-step size required by the structural dynamics part is smaller, and carrying out the entire computation with that time-step size would be too inefficient for the fluid dynamics part. In the EDSTT-single-mesh (EDSTT-SM) approach, a single space–time mesh, unstructured both in space and time, would be used to enhance the time-discretization in regions requiring smaller time steps. In the EDSTT-multi-mesh (EDSTT-MM) approach, we complement the space–time concept of the deforming-spatial-domain/stabilized space–time (DSD/SST) formulation with the multi-mesh concept of the enhanced-discretization interface-capturing technique (EDICT). In applications to fluid–structure interactions, the structural dynamics modeling is based on a single space–time mesh and the fluid dynamics modeling is based on two space–time meshes. The structural dynamics interface nodes in the space–time domain also belong to the second fluid mesh, which accommodates the time-step requirement of the structural dynamics. We apply the EDSTT-SM and EDSTT-MM approaches to a number of test problems to demonstrate how these methods work and why they would be desirable to use in time-accurate computations.

Keywords: Flow simulation; Space–time formulation; Enhanced-discretization technique; Moving boundaries and interfaces; Fluid–structure interactions

1. Introduction

The deforming-spatial-domain/stabilized space–time (DSD/SST) formulation [1] was developed for moving boundaries and interfaces, including flows with free-surfaces and two-fluid interfaces, flows with moving mechanical components, and fluid–object and fluid–structure interactions. In the DSD/SST formulation the finite element formulation of the problem is written over its space–time domain. At each time step the locations of the interfaces are calculated as part of the overall solution. As the spatial domain

* Corresponding author. Tel.: +1-713-348-6051; fax: +1-713-348-5423.
E-mail address: tezduyar@rice.edu (T.E. Tezduyar).
URL: http://www.mems.rice.edu/TAFSM/.

0045-7825/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2003.12.029

occupied by the fluid changes its shape in time, the mesh needs to be updated. In general, this is accomplished by moving the mesh with the motion of the nodes governed by the equations of elasticity [2], and full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) as needed. It needs to be pointed out that the stabilized space–time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example [3]). Stabilized space–time finite element formulations for moving boundaries and interfaces have been the focus of a number of research activities, including method development [4], modeling of complex fluid–structure interaction problems [5], as well as analysis and evaluation [6]. A discussion on the geometric conservation properties of various methods developed for moving boundaries and interfaces can be found in [6], which includes a conclusion that the space–time formulation leads to solution techniques that inherently satisfy the geometric conservation law.

The DSD/SST formulation is an interface-tracking technique. An interface-tracking technique requires meshes that "track" the interfaces. The mesh needs to be updated as the flow evolves. The mesh update consists of moving the mesh for as long as possible and remeshing as frequently as needed. In computation of two-fluid flows with interface-tracking techniques, sometimes the interface might be too complex or unsteady to track while keeping the frequency of remeshing at an acceptable level. In such cases, interface-capturing techniques, which do not normally require costly mesh update steps, could be used with the understanding that the interface will not be represented as accurately as we would have with an interface-tracking technique. In an interface-capturing technique the computations are based on fixed spatial domains, where an interface function, marking the location of the interface, needs to be computed to "capture" the interface. The interface is captured within the resolution of the finite element mesh covering the area where the interface is.

The interface-tracking and interface-capturing techniques we have developed in recent years (see [1,7–9]) are based on stabilized formulations. The stabilized methods are the streamline-upwind/Petrov–Galerkin (SUPG) [10,11] and pressure-stabilizing/Petrov–Galerkin (PSPG) [1] formulations. An earlier version of the pressure-stabilizing formulation for Stokes flows was reported in [12]. These stabilized formulations prevent numerical oscillations and other instabilities in solving problems with high Reynolds and/or Mach numbers and shocks and strong boundary layers, as well as when using equal-order interpolation functions for velocity and pressure and other unknowns. Furthermore, this class of stabilized formulations substantially improve the convergence rate in iterative solution of the large, coupled nonlinear equation system that needs to be solved at every time step of a flow computation.

The enhanced-discretization interface-capturing technique (EDICT) was introduced in [13] to increase accuracy in representing an interface. This is accomplished by using 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 or near the interface. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. The interpolation functions for velocity and pressure will all have two components each: one coming from Mesh-1 and the second one coming from Mesh-2. To further increase the accuracy, we construct a third-level mesh, Mesh-3, for the interface function only. The construction of Mesh-3 from Mesh-2 is very similar to the construction of Mesh-2 from Mesh-1. The interpolation functions for the interface function will have three components, each coming from one of these three meshes. We re-define the subsets over which we build Mesh-2 and Mesh-3 not every time step but with sufficient frequency to keep the interface enveloped in. We need to avoid this envelope being too wide or too narrow. Functionally, the EDICT would let us have the benefits one would draw from adaptive local mesh refinement. The EDICT would allow us to accomplish this objective without facing the implementational difficulties associated with elements having variable number of nodes. It would allow us to accomplish this objective even when the required increase in mesh refinement is large and locally abrupt.

The EDICT was followed by a number of extensions and offshoots. One example, which was first reported in [14], is the offshoot 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. More accurate representation of the shock is accomplished while avoiding local mesh refinement involving elements with variable number of nodes or global mesh refinement involving large increases in computational cost. Another example is the extension to computation of vortex flows, with results reported in [15]. In this extension the regions where the vorticity magnitude is larger than a specified value become the zones of enhanced discretization.

In the enhanced-discretization space–time technique (EDSTT), which was introduced in [8,9], we use enhanced time-discretization in the context of a space–time formulation. The original motivation behind the development of the EDSTT was to have a flexible way of carrying out time-accurate computations of fluid–structure interactions where we find it necessary to use smaller time steps for the structural dynamics part. Carrying out a fluid–structure interaction computation with a single time-step size determined by what the fluid dynamics part requires would often not yield acceptable temporal accuracy for the structural dynamics part. Computation with a single time-step size determined by the structural dynamics requirment, on the other hand, would often be too inefficient for the fluid dynamics, which typically has one higher spatial dimension than the structural dynamics. In general, EDSTT can be used in time-accurate computations where we require smaller time steps in certain parts of the fluid domain. For example, where the spatial element sizes are small, we may need to use small time steps, so that the element Courant number does not become too large. In computation of two-fluid interface (or free-surface) flows with the DSD/SST method, as another example, time-integration of the equation governing the evolution of the interface (i.e. the interface update equation) may require a smaller time step than the one used for the fluid interiors. This requirement might be coming from numerical stability considerations, when time-integration of the interface update equation does not involve any added stabilization terms.

There are two ways of formulating the EDSTT. In the EDSTT-single-mesh (EDSTT-SM) approach [8,9], a single space–time mesh, unstructured both in space and time, would be used to enhance the time-discretization in regions of the fluid domain where we require smaller time steps. This, in general, might require a fully unstructured 4D mesh generation. The EDSTT-multi-mesh (EDSTT-MM) approach [8,9] is based on combining the DSD/SST formulation and the EDICT multi-mesh concept. In this approach, multiple space–time meshes, all structured in time, would be used to enhance the time-discretization in regions where we require smaller time steps. The EDSTT-MM approach would not require a fully unstructured 4D mesh generation, and therefore would not pose a mesh generation difficulty.

In Section 2 we describe the governing equations, and in Section 3 we summarize the stabilized semi-discrete formulations for the advection–diffusion equation and the Navier–Stokes equations of incompressible flows. The DSD/SST formulation is briefly described in Section 4. The EDICT and the construction of function spaces for the EDICT are described in Sections 5 and 6. The EDSTT-SM and EDSTT-MM approaches are illustrated in Section 7. The numerical examples are given in Section 8 and the concluding remarks in Section 9.

2. Governing equations

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

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

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

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

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

Here p is the pressure, $\mathbf{I}$ is the identity tensor, $\mu = \rho v$ is the viscosity, v is the kinematic viscosity, and $\boldsymbol{\varepsilon}(\mathbf{u})$ is the strain-rate tensor:

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

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

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

where $(\Gamma_t)_g$ and $(\Gamma_t)_h$ are complementary subsets of the boundary Γ_t, $\mathbf{n}$ is the unit normal vector, and $\mathbf{g}$ and $\mathbf{h}$ are given functions. A divergence-free velocity field $\mathbf{u}_0(\mathbf{x})$ is specified as the initial condition.

If there are no 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 the formulation is not based on defining the spatial domain to be the part of the space occupied by the fluid(s). For example, fluid–fluid interfaces can be modeled over a fixed spatial domain by assuming that the domain is occupied by two immiscible fluids, A and B, with densities ρ_{A} and ρ_{B} and viscosities μ_{A} and μ_{B}. In this approach, a free-surface problem can be modeled as a special case where Fluid B is irrelevant and is assigned a sufficiently low density. An interface function ϕ serves as the marker identifying Fluid A and B with the definition $\phi = \{1$ for Fluid A and 0 for Fluid B$\}$. The interface between the two fluids is approximated to be at $\phi = 0.5$. In this context, ρ and μ are defined as

$$\rho = \phi\rho_{\mathrm{A}} + (1-\phi)\rho_{\mathrm{B}}, \quad \mu = \phi\mu_{\mathrm{A}} + (1-\phi)\mu_{\mathrm{B}}. \tag{6}$$

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

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

To generalize Eq. (7), we consider the following time-dependent advection–diffusion equation, written on Ω and $\forall t \in (0, T)$ as

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

where ϕ represents the quantity being transported (e.g., temperature, concentration), and v is the diffusivity, which is separate from (but in mathematical significance very comparable to) the v representing the kinematic viscosity. The essential and natural boundary conditions associated with Eq. (8) are represented as

$$\phi = g \quad \text{on } \Gamma_g, \qquad \mathbf{n} \cdot v\nabla\phi = h \quad \text{on } \Gamma_h. \tag{9}$$

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

3. Stabilized semi-discrete formulations

3.1. Advection–diffusion equation

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

$$\int_\Omega w^h\left(\frac{\partial \phi^h}{\partial t}+\mathbf{u}^h\cdot\nabla\phi^h\right)\mathrm{d}\Omega+\int_\Omega \nabla w^h\cdot v\nabla\phi^h\,\mathrm{d}\Omega-\int_{\Gamma_h} w^h h^h\,\mathrm{d}\Gamma$$
$$+\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\tau_{\mathrm{SUPG}}\mathbf{u}^h\cdot\nabla w^h\left(\frac{\partial \phi^h}{\partial t}+\mathbf{u}^h\cdot\nabla\phi^h-\nabla\cdot(v\nabla\phi^h)\right)\mathrm{d}\Omega=0. \tag{10}$$

Here n_{el} is the number of elements, Ω^e is the domain for element e, and τ_{SUPG} is the SUPG stabilization parameter. For various ways of calculating τ_{SUPG}, see [16,17].

3.2. Navier–Stokes equations of incompressible flows

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

$$\int_\Omega \mathbf{w}^h\cdot\rho\left(\frac{\partial \mathbf{u}^h}{\partial t}+\mathbf{u}^h\cdot\nabla\mathbf{u}^h-\mathbf{f}^h\right)\mathrm{d}\Omega+\int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h):\boldsymbol{\sigma}(p^h,\mathbf{u}^h)\,\mathrm{d}\Omega-\int_{\Gamma_h}\mathbf{w}^h\cdot\mathbf{h}^h\,\mathrm{d}\Gamma$$
$$+\int_\Omega q^h\nabla\cdot\mathbf{u}^h\,\mathrm{d}\Omega+\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\frac{1}{\rho}[\tau_{\mathrm{SUPG}}\rho\mathbf{u}^h\cdot\nabla\mathbf{w}^h+\tau_{\mathrm{PSPG}}\nabla q^h]\cdot[Ł(p^h,\mathbf{u}^h)-\rho\mathbf{f}^h]\,\mathrm{d}\Omega$$
$$+\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\nu_{\mathrm{LSIC}}\nabla\cdot\mathbf{w}^h\rho\nabla\cdot\mathbf{u}^h\,\mathrm{d}\Omega=0, \tag{11}$$

where

$$Ł(q^h,\mathbf{w}^h)=\rho\left(\frac{\partial \mathbf{w}^h}{\partial t}+\mathbf{u}^h\cdot\nabla\mathbf{w}^h\right)-\nabla\cdot\boldsymbol{\sigma}(q^h,\mathbf{w}^h). \tag{12}$$

Here τ_{PSPG} and ν_{LSIC} are the PSPG and LSIC (least-squares on incompressibility constraint) stabilization parameters. For various ways of calculating τ_{PSPG} and ν_{LSIC}, see [16,17].

4. DSD/SST finite element formulation

In the DSD/SST method [1], the finite element formulation of the governing equations is written over a sequence of N space–time slabs Q_n, where Q_n is the slice of the space–time domain between the time levels t_n and t_{n+1}. At each time step, the integrations involved in the finite element formulation are performed over Q_n. The space–time finite element interpolation functions are continuous within a space–time slab, but discontinuous from one space–time slab to another. The notation $(\cdot)^-_n$ and $(\cdot)^+_n$ denotes the function values at t_n as approached from below and above. Each Q_n is decomposed into elements Q^e_n, where $e=1,2,\ldots,(n_{\mathrm{el}})_n$. The subscript n used with n_{el} is for the general case in which the number of space–time elements may change from one space–time slab to another. The Dirichlet- and Neumann-type boundary conditions are enforced over $(P_n)_g$ and $(P_n)_h$, the complementary subsets of the lateral boundary of the space–time slab. The finite element trial function spaces $(\mathscr{S}^h_{\mathbf{u}})_n$ for velocity and $(\mathscr{S}^h_p)_n$ for pressure, and the test function spaces $(\mathscr{V}^h_{\mathbf{u}})_n$ and $(\mathscr{V}^h_p)_n=(\mathscr{S}^h_p)_n$ are defined by using, over Q_n, first-order polynomials in both space and time. The DSD/SST formulation [1,17] is written as follows: given $(\mathbf{u}^h)^-_n$, find $\mathbf{u}^h\in(\mathscr{S}^h_{\mathbf{u}})_n$ and $p^h\in(\mathscr{S}^h_p)_n$ such that $\forall\mathbf{w}^h\in(\mathscr{V}^h_{\mathbf{u}})_n$ and $q^h\in(\mathscr{V}^h_p)_n$:

$$\int_{Q_n} \mathbf{w}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h\right) \mathrm{d}Q + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}Q$$
$$- \int_{(P_n)_h} \mathbf{w}^h \cdot \mathbf{h}^h \, \mathrm{d}P + \int_{Q_n} q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho\big((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^-\big) \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \frac{1}{\rho}\left[\tau_{\mathrm{SUPG}} \rho\left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h\right) + \tau_{\mathrm{PSPG}} \nabla q^h\right] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}Q$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q = 0. \tag{13}$$

This formulation is applied to all space–time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, starting with $(\mathbf{u}^h)_0^- = \mathbf{u}_0$. For an earlier, detailed reference on the DSD/SST formulation see [1]. Similarly, the DSD/SST formulation of Eq. (8) can be written as follows:

$$\int_{Q_n} w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h\right) \mathrm{d}Q + \int_{Q_n} \nabla w^h \cdot \nu \nabla \phi^h \, \mathrm{d}Q - \int_{(P_n)_h} w^h h^h \, \mathrm{d}P + \int_{\Omega_n} (w^h)_n^+ \big((\phi^h)_n^+ - (\phi^h)_n^-\big) \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \tau_{\mathrm{SUPG}} \left(\frac{\partial w^h}{\partial t} + \mathbf{u}^h \cdot \nabla w^h\right) \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h - \nabla \cdot (\nu \nabla \phi^h)\right) \mathrm{d}Q = 0. \tag{14}$$

5. Enhanced-discretization interface-capturing technique (EDICT)

In the enhanced-discretization interface-capturing technique (EDICT) [13], we start with the basic approach of an interface-capturing technique such as the volume of fluid (VOF) method [18]. The Navier–Stokes equations are solved over a non-moving mesh together with the time-dependent advection equation governing the evolution of the interface function ϕ. In writing the stabilized finite element formulation for the EDICT (see [13]), the notation we use here for representing the finite-dimensional function spaces is very similar to the notation we used in the section where we described the DSD/SST formulation. The trial function spaces corresponding to velocity, pressure and interface function are denoted, respectively, by $(\mathcal{S}_{\mathbf{u}}^h)_n$, $(\mathcal{S}_p^h)_n$, and $(\mathcal{S}_\phi^h)_n$. The weighting function spaces corresponding to the momentum equation, incompressibility constraint and time-dependent advection equation are denoted by $(\mathcal{V}_{\mathbf{u}}^h)_n$, $(\mathcal{V}_p^h)_n (= (\mathcal{S}_p^h)_n)$, and $(\mathcal{V}_\phi^h)_n$. The subscript n in this case allows us to use different spatial discretizations corresponding to different time levels.

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

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

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

Here τ_{ϕ} is calculated by applying the definition of τ_{SUPG} to Eq. (16).

A subset of the elements in the base mesh, Mesh-1, are identified as those at or near the interface. A more refined mesh, Mesh-2, is constructed by patching together second-level meshes generated over each element in this subset. The trial functions for velocity and pressure will have two components each, one coming from Mesh-1 and the second one coming from Mesh-2:

$$\mathbf{u}_{n}^{h} = \mathbf{u}_{n}^{1} + \mathbf{u}_{n}^{2}, \tag{17}$$

$$p_{n}^{h} = p_{n}^{1} + p_{n}^{2}. \tag{18}$$

To further increase the accuracy, we construct a third-level mesh, Mesh-3, for the interface function only. The construction of Mesh-3 from Mesh-2 is very similar to the construction of Mesh-2 from Mesh-1. The trial functions for the interface function will have three components, each coming from one of these three meshes:

$$\phi_{n}^{h} = \phi_{n}^{1} + \phi_{n}^{2} + \phi_{n}^{3}. \tag{19}$$

The weighting functions are constructed in a similar fashion:

$$\mathbf{w}_{n}^{h} = \mathbf{w}_{n}^{1} + \mathbf{w}_{n}^{2}, \tag{20}$$

$$q_{n}^{h} = q_{n}^{1} + q_{n}^{2}, \tag{21}$$

$$\psi_{n}^{h} = \psi_{n}^{1} + \psi_{n}^{2} + \psi_{n}^{3}. \tag{22}$$

6. Construction of function spaces for EDICT

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 these elements. Mesh-2 is generated by patching together the second-level meshes generated over each of the elements in $(\epsilon_{n}^{1})_{n}^{2}$. 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. 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 re-used. 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 $(\epsilon_{n}^{2})_{n}^{3}$. The set of elements and nodal points for Mesh-3 are denoted by ϵ_{n}^{3} and η_{n}^{3}.

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^1 and p_n^2 in exactly the same way, except for recognizing the fact that the references to Dirichlet-type boundary conditions do not apply.

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 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.

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. Whenever we re-define 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 reuse the meshes which were generated (and stored) the first time these meshes were needed.

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 mesh zones in such a way that they coincide. One of the advantages in keeping them as non-coinciding is that, by keeping the Mesh-2 zone wider than the Mesh-3 zone, one can chose to limit the existence (as an unknown) of ϕ to the Mesh-2 zone, and therefore solve 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 the Mesh-2 zone. Since our objective will be to keep the interface in the Mesh-3 zone, this would also keep it in the Mesh-2 zone, even if the interface occasionally falls slightly out of the Mesh-3 zone.

7. EDSTT-SM and EDSTT-MM approaches

In the EDSTT-SM approach, a single space–time mesh, unstructured both in space and time, would be used to enhance the time-discretization in regions of the fluid domain where we require smaller time steps. This concept is illustrated in Fig. 1 with a fictitious 2D space–time mesh for the EDSTT-SM approach to a fluid–structure interaction problem. In this example, spatially, the fluid dynamics part is 1D and the structural dynamics part is 0D. The time-step size required in time-integration of the fluid dynamics problem is four times the time-step size required in time-integration of the structural dynamics problem. Application of the EDSTT-SM approach to a spatially 3D problem would in general require a fully unstructured 4D mesh generation.

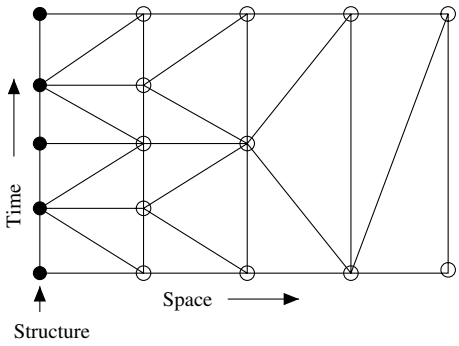

Fig. 1. A fictitious 2D space–time mesh for the EDSTT-SM approach to a fluid–structure interaction problem. Spatially, the fluid dynamics part is 1D and the structural dynamics part is 0D. The time-step size for the fluid dynamics is four times the time-step size for the structural dynamics. The fluid-only nodes are denoted by empty circles, while the nodes shared by the fluid and structure are denoted by solid circles.

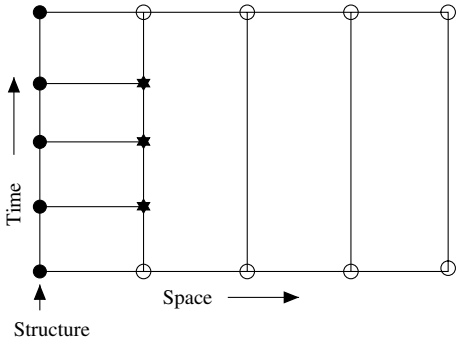

Fig. 2. Fictitious multiple 2D space–time meshes for the EDSTT-MM approach to a fluid–structure interaction problem. Spatially, the fluid dynamics part is 1D and the structural dynamics part is 0D. The time-step size for the fluid dynamics is four times the time-step size for the structural dynamics. The structural dynamics modeling is based on a single space–time mesh, while the fluid dynamics modeling is based on two space–time meshes. The Mesh-1 and Mesh-2 components of the fluid dynamics function space are based on the nodes denoted by the empty and solid circles, respectively. Neither component of the fluid dynamics function space is based on the nodes denoted by the stars. The solid circles also denote the structural dynamics nodes.

The EDSTT-MM approach is based on combining the DSD/SST formulation and the EDICT multi-mesh concept. In this approach, multiple space–time meshes, all structured in time, would be used to enhance the time-discretization in regions of the fluid domain where we require smaller time steps. This concept is illustrated in Fig. 2 with fictitious multiple 2D space–time meshes for the EDSTT-MM approach to a fluid–structure interaction problem. In this example too, spatially, the fluid dynamics part is 1D and the structural dynamics part is 0D. Again, the time-step size required in time-integration of the fluid dynamics problem is four times the time-step size required in time-integration of the structural dynamics problem. The structural dynamics modeling is based on a single space–time mesh, while the fluid dynamics modeling is based on two space–time meshes: Mesh-1 and Mesh-2. All the structural dynamics nodes in the space–time domain also belong to Mesh-2. Fig. 3 shows fictitious multiple 3D space–time meshes for the EDSTT-MM approach to a fluid–structure interaction problem. In this case, spatially, the fluid dynamics part is 2D and the structural dynamics part is 1D. The time-step size required in time-integration of the fluid dynamics problem is three times the time-step size required in time-integration of the structural

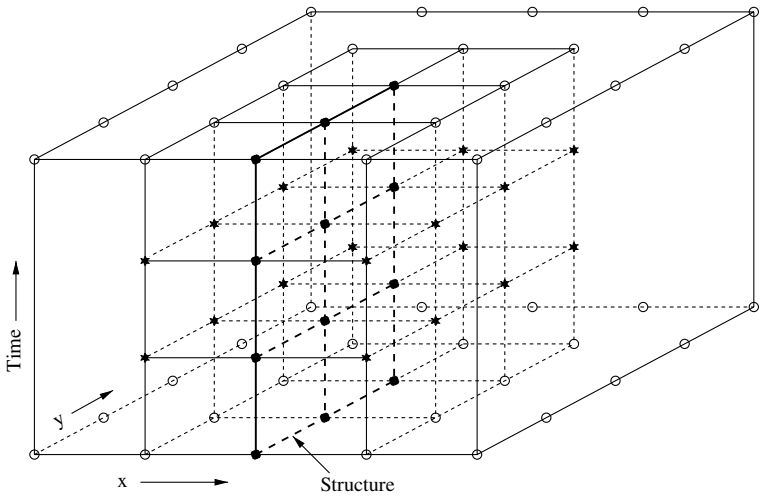

Fig. 3. Fictitious multiple 3D space–time meshes for the EDSTT-MM approach to a fluid–structure interaction problem. Spatially, the fluid dynamics part is 2D and the structural dynamics part is 1D. The time-step size for the fluid dynamics is three times the time-step size for the structural dynamics. The structural dynamics modeling is based on a single space–time mesh, while the fluid dynamics modeling is based on two space–time meshes. The Mesh-1 and Mesh-2 components of the fluid dynamics function space are based on the nodes denoted by the empty and solid circles, respectively. Neither component of the fluid dynamics function space is based on the nodes denoted by the stars. The solid circles also denote the structural dynamics nodes. Not all the element edges of Mesh-1 are shown.

dynamics problem. The EDSTT-MM approach would not require a fully unstructured 4D mesh generation, and therefore would not pose a mesh generation difficulty.

8. Numerical examples

8.1. 1D advection of a cosine wave

In this test problem, we compute with the space–time technique (STT) and EDSTT-SM 1D advection of a cosine wave. The base formulation in both approaches is given by Eq. (14). We use four different space–time meshes, as shown in Fig. 4. Two of the meshes are used with the STT and two with the EDSTT-SM. All four meshes are composed of inner and outer zones. Spatially, the inner zones are four times as refined as the outer zones. For the two STT meshes, the Courant number (Cr_u) based on the inner zones is 2.0 and 4.0. For the inner zones of the EDSTT-SM meshes, the temporal refinement is at such a level that $Cr_u = 1.0$. For the cosine wave being advected, the dimensionless wave number ($q = kh$), defined based on the spatial element length (h) in the inner zones, is 0.3142. Fig. 5 shows the shapes and positions of the cosine wave at the beginning and end of the computations. To a large extent, the solutions for EDSTT-SM at $Cr_u = 2.0$ and $Cr_u = 4.0$ are indistinguishable. The final wave amplitudes are: at $Cr_u = 2.0$, 0.870 for STT and 0.949 for EDSTT-SM, and at $Cr_u = 4.0$, 0.735 for STT and 0.950 for EDSTT-SM. Clearly, the EDSTT-SM performs better.

8.2. 2D Translation of a cosine puff

In this test problem, we compute with the STT and EDSTT-SM 2D translation of a "puff" with cross-sectional profile of a cosine wave. The spatial footprint of the space–time meshes used in the computations

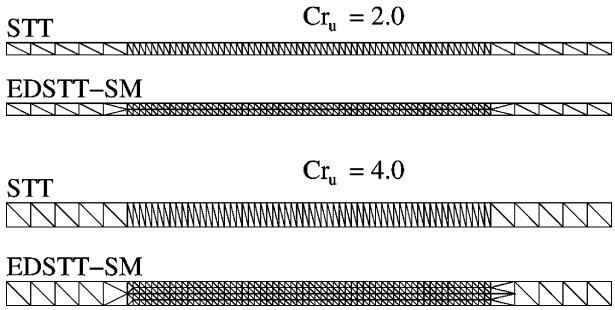

Fig. 4. 1D advection of a cosine wave. Space–time meshes for computations with the space–time technique (STT) and EDSTT-SM. Spatially, the inner mesh zones are four times as refined as the outer zones. Courant number $(Cr_u) = 2.0$ and 4.0, based on the inner zones of the upper and lower STT meshes, respectively. For the inner zones of the EDSTT-SM meshes, $Cr_u = 1.0$.

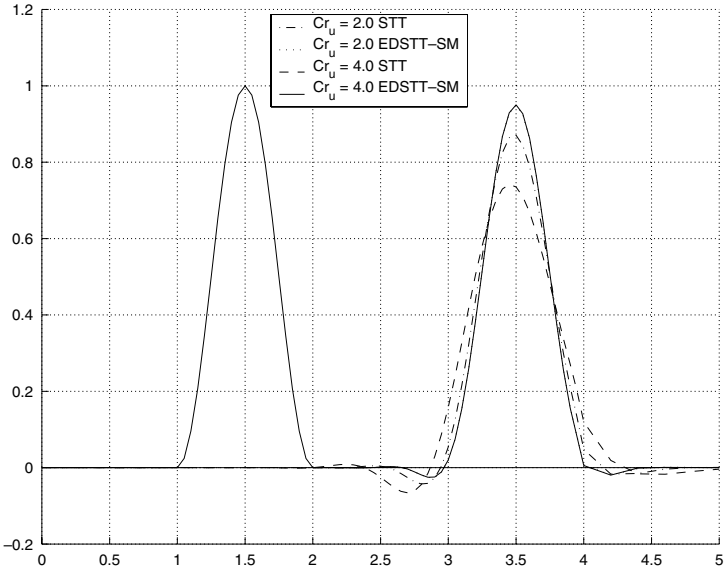

Fig. 5. 1D advection of a cosine wave. Solutions from computations with the STT and EDSTT-SM for $Cr_u = 2.0$ and 4.0. Dimensionless wave number $(q = kh) = 0.3142$, defined based on the spatial element length (h) in the inner zones. The shapes and positions of the cosine wave at the beginning and end of the computations. The curves for EDSTT-SM at $Cr_u = 2.0$ and $Cr_u = 4.0$ essentially coincide. The final wave amplitudes: at $Cr_u = 2.0$, 0.870 for STT and 0.949 for EDSTT-SM, and at $Cr_u = 4.0$, 0.735 for STT and 0.950 for EDSTT-SM.

is shown in Fig. 6. The size of the spatial domain is 1.25×0.75, and the mesh has a spatially refined zone. That refined zone is four times as refined in the horizontal direction as the unrefined zones. Based on the puff translation velocity, the element length in the horizontal direction in the refined zone, and the temporal refinement of the STT mesh, $Cr_u = 8.0$. For the refined zone of the EDSTT-SM mesh, $Cr_u = 2.0$. For the puff being advected, the wave length of the cross-sectional cosine is 0.5. The puff translates a distance of 0.5. Fig. 7 shows the shapes and positions of the puff at the end of the translation. The final puff amplitudes are: 0.927 for STT and 0.997 for EDSTT-SM. The EDSTT-SM performs better.

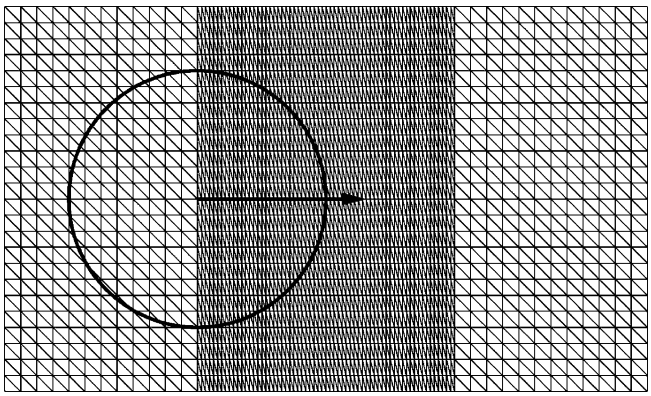

Fig. 6. 2D translation of a cosine puff. Spatial footprint of the space–time meshes used in the computations with the STT and EDSTT-SM. The spatial domain size is 1.25×0.75. The spatially refined zone is four times as refined in the horizontal direction as the unrefined zones. $Cr_u = 8.0$, based on the puff translation velocity, the element length in the horizontal direction in the refined zone, and the temporal refinement of the STT mesh. For the refined zone of the EDSTT-SM mesh, $Cr_u = 2.0$. The initial position of the puff and the translation direction are also shown.

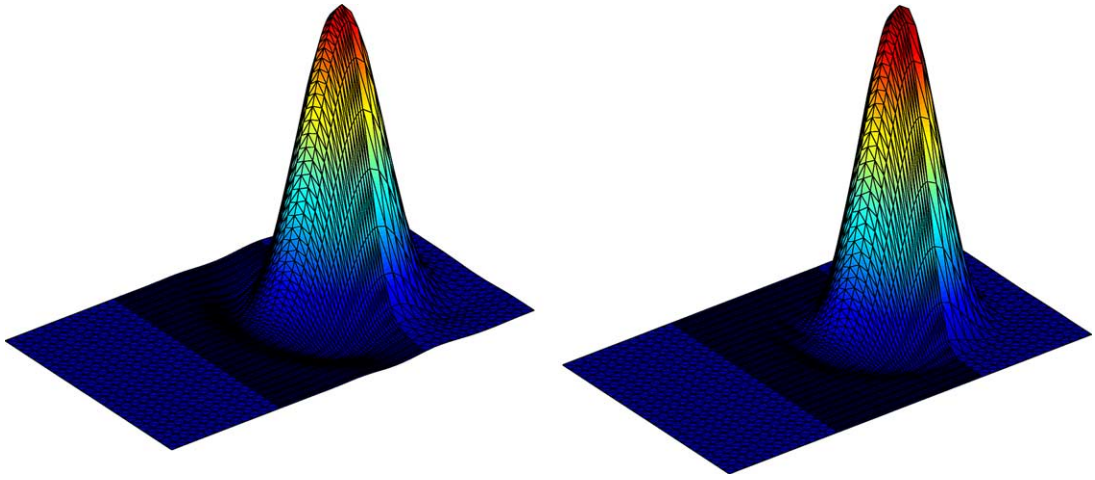

Fig. 7. 2D translation of a cosine puff. Solutions from computations with the STT (left) and EDSTT-SM (right). The wave length of the cross-sectional cosine is 0.5. The puff translates a distance of 0.5. The shapes and positions of the puff at the end of the translation. The final puff amplitudes: 0.927 for STT and 0.997 for EDSTT-SM.

8.3. 2D rotation of a cosine puff

Here we compute with the STT and EDSTT-SM 2D rotation of a puff with cross-sectional profile of a cosine wave. The spatial footprint of the space–time meshes used in the computations is shown in Fig. 8. The size of the spatial domain is 1.00×1.25, and the mesh has a spatially refined zone. That refined zone is four times as refined in the horizontal direction as the unrefined zones. Based on the maximum flow velocity, the element length in the horizontal direction in the refined zone, and the temporal refinement of the STT mesh, $Cr_u = 16.0$. For the refined zone of the EDSTT-SM mesh, $Cr_u = 4.0$. For the puff being

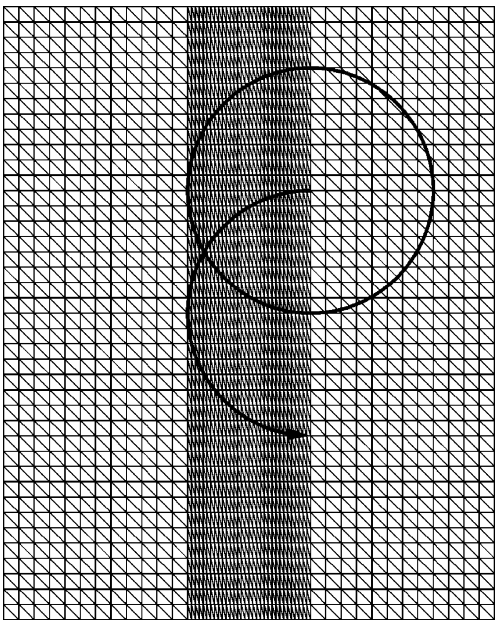

Fig. 8. 2D rotation of a cosine puff. Spatial footprint of the space–time meshes used in the computations with the STT and EDSTT-SM. The spatial domain size is 1.00×1.25. The spatially refined zone is four times as refined in the horizontal direction as the unrefined zones. $Cr_u = 16.0$, based on the maximum flow velocity, the element length in the horizontal direction in the refined zone, and the temporal refinement of the STT mesh. For the refined zone of the EDSTT-SM mesh, $Cr_u = 4.0$. The initial position of the puff and the rotation direction are also shown.

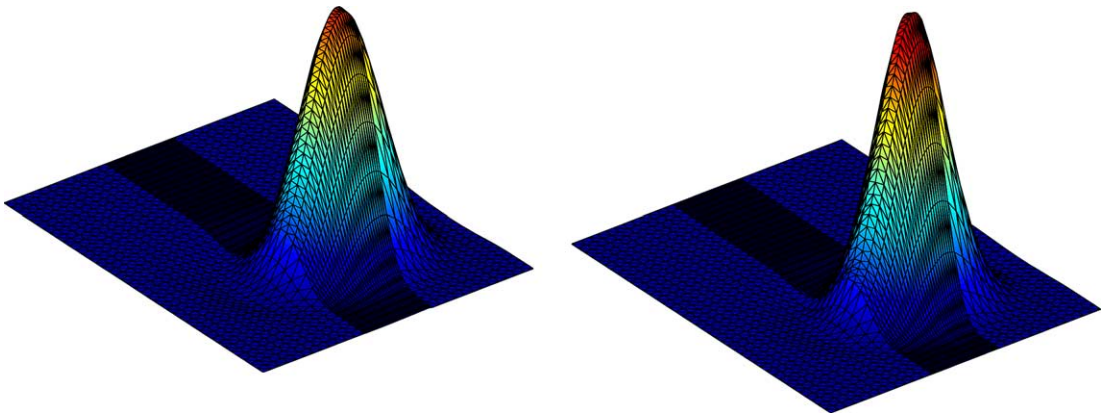

Fig. 9. 2D rotation of a cosine puff. Solutions from computations with the STT (left) and EDSTT-SM (right). The wave length of the cross-sectional cosine is 0.5. The puff undergoes 1/2 revolutions. The shapes and positions of the puff at the end of the rotation. The final puff amplitudes: 0.845 for STT and 0.973 for EDSTT-SM.

rotated, the wave length of the cross-sectional cosine is 0.5. The puff undergoes 1/2 revolutions. Fig. 9 shows the shapes and positions of the puff at the end of the rotation. The final puff amplitudes are: 0.845 for STT and 0.973 for EDSTT-SM. The EDSTT-SM performs better.

8.4. 2D flow past a thin flexible beam

We compute with the EDSTT-MM approach fluid–structure interactions in 2D flow past a thin flexible beam. The problem set up is shown in Fig. 10. The fluid density and viscosity are 1000 kg/m^3 and 1.792×10^{-3} N s/m^2. The flow boundary conditions are set to a uniform inflow velocity of 1.0 m/s at the upstream boundary, traction-free conditions at the downstream, slip conditions at the lateral boundaries, and no slip conditions at the beam. The length and thickness of the beam are 2.0 m and 5.848 mm. Its density and modulus of elasticity are 2750 kg/m^3 and 7.5×10^{10} N/m^2. The beam is assumed to be governed by the Bernoulli–Euler beam equation, which is solved with a finite element formulation using piecewise cubic Hermite shape functions (see [19]). At the midpoint of the beam the displacement and slope are set to zero. First we calculate analytically the period for the first mode of the natural vibrations of the beam, and determine that period to be 0.2 s. Based on that, we set the time-step size for the structural dynamics to

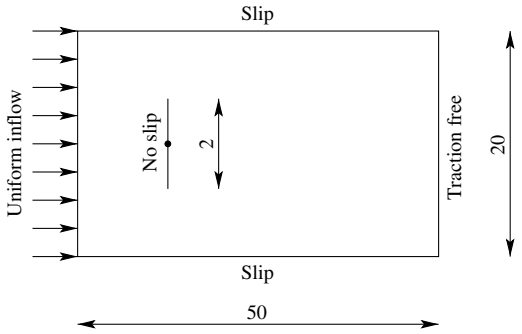

Fig. 10. 2D flow past a thin flexible beam. Problem set up. The dimensions indicated are in meters. The fluid density and viscosity are 1000 kg/m^3 and 1.792×10^{-3} N s/m^2. Flow boundary conditions are indicated next to each boundary. The inflow velocity is 1.0 m/s. The length and thickness of the beam are 2.0 m and 5.848 mm. Its density and modulus of elasticity are 2750 kg/m^3 and 7.5×10^{10} N/m^2. The beam is assumed to be governed by the Bernoulli–Euler beam equation. At the midpoint of the beam the displacement and slope are set to zero. The period for the first mode of the natural vibrations of the beam is 0.2 s. In the computations, the time-step size is set to 0.20 s for the fluid dynamics and 0.01 s for the structural dynamics.

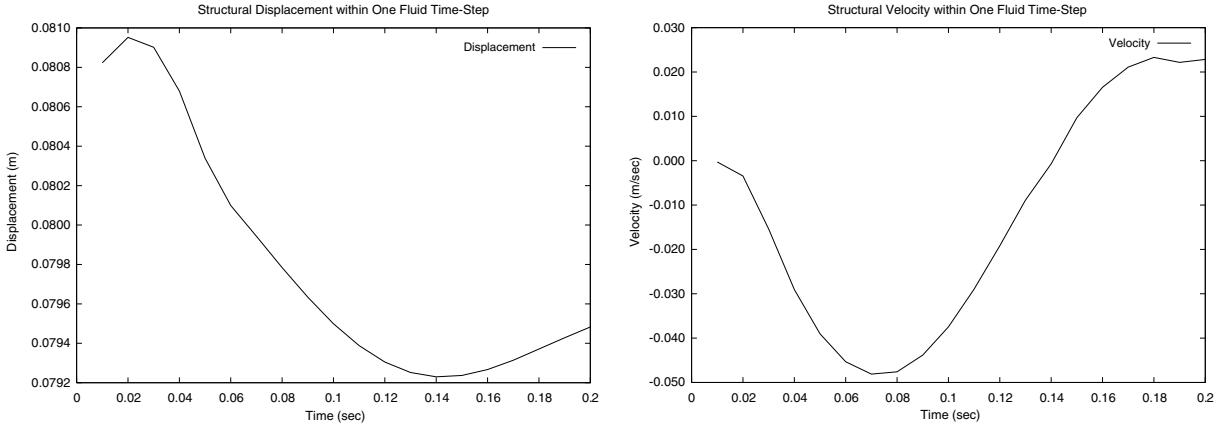

Fig. 11. 2D flow past a thin flexible beam. Tip displacement (left) and velocity (right) of the beam during one fluid time step.

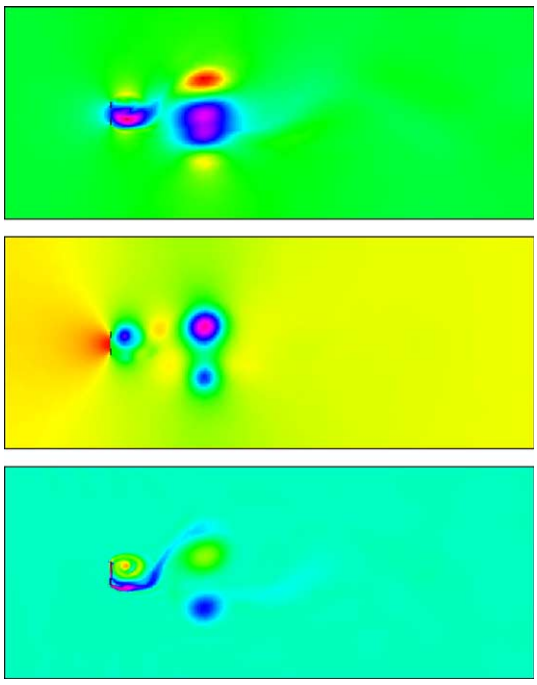

Fig. 12. 2D flow past a thin flexible beam. Flow horizontal-velocity (top), pressure (middle), and vorticity (bottom). The color range from pink (through blue, green and yellow) to red corresponds to the ranges of velocity, pressure and vorticity values from low to high.

0.01 s. The time-step size for the fluid dynamics is set, based on Courant number considerations, to 0.20 s. For each space–time slab we perform three nonlinear iterations. The fluid–structure coupling, which is handled iteratively, is embedded in these iterations. Fig. 11 shows the tip displacement and velocity of the beam during one fluid time step. Fig. 12 shows the flow horizontal-velocity, pressure, and vorticity.

9. Concluding remarks

We described the EDSTT, which was developed to use, in the context of a space–time formulation, enhanced time-discretization in regions of the fluid domain requiring smaller time steps. Such requirements are often encountered in computation of fluid–structure interactions, because the time-step size required by the structural dynamics part for time-accurate computations is smaller than time-step size required by the fluid dynamics part. Carrying out the entire computation with that time-step size would be too inefficient for the fluid dynamics, which typically has one higher spatial dimension than the structural dynamics. Increasing the time-step size to a level required by the fluid dynamics, on the other hand, would often not give us an acceptable temporal accuracy for the structural dynamics.

We described two ways of formulating the EDSTT. In the EDSTT-SM approach, a single space–time mesh, unstructured both in space and time, is used to enhance the time-discretization in regions of the fluid domain requiring smaller time steps. Application of the EDSTT-SM approach to spatially 3D problems would in general require fully unstructured 4D mesh generation, and this would not be very desirable. In the EDSTT-MM approach, we combine the space–time concept of the DSD/SST formulation with the multi-mesh concept of the EDICT. When we apply the EDSTT-MM approach to fluid–structure

interaction problems, the structural dynamics modeling is based on a single space–time mesh and the fluid dynamics modeling is based on two space–time meshes. Because the structural dynamics interface nodes in the space–time domain also belong to the second fluid mesh, time-step requirement of the structural dynamics is accommodated. In spatially 3D problems, the two fluid meshes would not require fully unstructured 4D mesh generation, and therefore the EDSTT-MM approach would not involve a daunting mesh generation challenge.

With the test problems we presented here, we showed how the EDSTT works in various contexts. We also showed that the EDSTT would be a good strategy in computation of fluid–structure interactions, especially when it is not desirable to accept a trade off between the time-accuracy requirements of the structural dynamics and the efficiency requirements of the fluid dynamics.

Acknowledgements

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

References

(1992)

[1] T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Adv. Appl. Mech. 28 (1991) 1–44.
[2] T.E. Tezduyar, M. Behr, S. Mittal, 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, pp. 7–24.
[3] T.J.R. Hughes, G.M. Hulbert, Space–time finite element methods for elastodynamics: formulations and error estimates, Comput. Methods Appl. Mech. Engrg. 66 (1988) 339–363.
[4] A. Masud, T.J.R. Hughes, A space–time Galerkin/least-squares finite element formulation of the Navier–Stokes equations for moving domain problems, Comput. Methods Appl. Mech. Engrg. 146 (1997) 91–126.
[5] K.R. Stein, R.J. Benney, T.E. Tezduyar, J.W. Leonard, M.L. Accorsi, Fluid–structure interactions of a round parachute: modeling and simulation techniques, J. Aircraft 38 (2001) 800–808.
[6] M. Lesoinne, C. Farhat, Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations, Comput. Methods Appl. Mech. Engrg. 134 (1996) 71–90.
[7] T.E. Tezduyar, Finite element methods for flow problems with moving boundaries and interfaces, Arch. Comput. Methods Engrg. 8 (2001) 83–130.
[8] T.E. Tezduyar, Finite element interface-tracking and interface-capturing techniques for flows with moving boundaries and interfaces, in: Proceedings of the ASME Symposium on Fluid-Physics and Heat Transfer for Macro- and Micro-Scale Gas–Liquid and Phase-Change Flows (CD-ROM), ASME Paper IMECE2001/HTD-24206, New York, ASME, New York, 2001.
[9] T. Tezduyar, Interface-tracking and interface-capturing techniques for computation of moving boundaries and interfaces, in: Proceedings of the Fifth World Congress on Computational Mechanics, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81513, Vienna, Austria, 2002.
[10] T.J.R. Hughes, A.N. Brooks, A multi-dimensional upwind scheme with no crosswind diffusion, in: T.J.R. Hughes (Ed.), Finite Element Methods for Convection Dominated Flows, AMD-Vol. 34, ASME, New York, 1979, pp. 19–35.
[11] T. Tezduyar, 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.
[12] T.J.R. Hughes, L.P. Franca, 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, Comput. Methods Appl. Mech. Engrg. 59 (1986) 85–99.
[13] T. Tezduyar, S. Aliabadi, M. Behr, Enhanced-discretization interface-capturing technique (EDICT) for computation of unsteady flows with interfaces, Comput. Methods Appl. Mech. Engrg. 155 (1998) 235–248.
[14] S. Mittal, S. Aliabadi, T. Tezduyar, Parallel computation of unsteady compressible flows with the EDICT, Comput. Mech. 23 (1999) 151–157.
[15] T. Tezduyar, Y. Osawa, K. Stein, R. Benney, V. Kumar, J. McCune, Computational methods for parachute aerodynamics, in: M. Hafez, K. Morinishi, J. Periaux (Eds.), Computational Fluid Dynamics for the 21st Century, Springer, 2001.

[16] T.E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, Comput. Methods Appl. Mech. Engrg. 190 (2000) 411–430.
[17] T. Tezduyar, Stabilization parameters and local length scales in SUPG and PSPG formulations, in: Proceedings of the Fifth World Congress on Computational Mechanics, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81508, Vienna, Austria, 2002.
[18] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[19] T.J.R Hughes, The Finite Element Method. Linear Static and Dynamic Finite Element Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.