III Congreso Internacional sobre Metodos Numericos en Ingenieria y Ciencias Aplicadas
S. Gallegos, I. Herrera, S. Botello, F. Zarate, y G. Ayala (Editores)

SPACE–TIME TECHNIQUES FOR FINITE ELEMENT COMPUTATION OF FLOWS WITH MOVING BOUNDARIES AND INTERFACES

T.E. Tezduyar, S. Sathe, R. Keedy

Mechanical Engineering
Rice University – MS 321
6100 Main St
Houston, TX 77005, USA
Email: tezduyar@rice.edu
http://www.mems.rice.edu/TAFSM/

K. Stein

Department of Physics
Bethel University
St. Paul, MN 55112, USA
Email: k-stein@bethel.edu

Abstract. We describe the space–time finite element techniques we developed for computation of fluid mechanics problems with moving boundaries and interfaces. We specifically focus on flows with fluid–structure interactions. The methods described include the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation and its special version; the mesh update methods, including the Solid-Extension Mesh Moving Technique (SEMMT); and the block-iterative, quasi-direct, and direct coupling techniques for the solution of the fully-discretized, coupled fluid and structural mechanics equations. We present a number of test computations for mesh moving and fluid–structure interactions. Overall, we demonstrate that the techniques we developed have increased the scope and accuracy of the methods used in computation of flows with moving boundaries and interfaces, including fluid–structure interactions.

Key words: Space–time finite elements, Moving boundaries and interfaces, Fluid–structure interactions, Mesh update techniques, Iterative and direct coupling techniques.

1 INTRODUCTION

Fluid mechanics problems with moving boundaries and interfaces is an important but also computationally challenging class of problems. This class of problems includes free-surface and two-fluid flows, fluid–object and fluid–structure interactions, and flows with moving mechanical

components. Some of the computational challenges are common to all of these subclasses of problems. For example, the spatial domain occupied by the fluid changes in time, and the mathematical model to be used will need to be able to handle that. Also, the mesh needs to be updated as the spatial domain occupied by the fluid changes. In addition to the challenges common to all, each of these subclasses of problems offers its own computational challenges, and in this article we focus on fluid–structure interactions.

The Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation[1–4] was developed in early 1990's for computation of flow problems with moving boundaries and interfaces. The DSD/SST method is based on stabilized finite element formulations, which are written over the space–time domains of the fluid mechanics problems considered. The stabilized methods are the streamline-upwind/Petrov-Galerkin (SUPG)[5–8] and pressure-stabilizing/Petrov-Galerkin (PSPG)[1,9] formulations. An earlier version of the pressure-stabilizing formulation for Stokes flows was reported in.[10] These stabilized formulations prevent numerical instabilities that might be encountered in solving problems with high Reynolds or Mach numbers and shocks and strong boundary layers, as well as when using equal-order interpolation functions for velocity and pressure. 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. The stabilized space–time formulations were used earlier by other researchers to solve problems with fixed spatial domains (see for example[11]).

The space–time computations are carried out at one space–time "slab" at a time, where the "slab" is the slice of the space–time domain between the time levels n and $n+1$. This spares a 3D computational problem from becoming a 4D problem including the time dimension. Additionally, in most space–time computations, all the nodes of the space–time slab are at time level n or $n+1$, and the spatial mesh at level $n+1$ is just a deformed version of the spatial mesh at level n . These additional special features are exploited in the Special DSD/SST (S-DSD/SST) formulation for more efficient calculation of the element-level vectors and matrices.

In general, we address the mesh update challenge with an automatic mesh moving method.[4,12] In this method, the motion of the nodes is governed by the equations of elasticity. The mesh deformation is dealt with selectively based on the sizes of the elements. When the mesh becomes too distorted, a full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) is performed. We introduced a number of enhancements to this general mesh update technique, including the the Solid-Extension Mesh Moving Technique (SEMMT).[13–15] The SEMMT was introduced to address the challenge involved in moving the very thin fluid elements typically seen near the solid surfaces.

Solution of the fully-discretized equations also offers some challenges, especially in computation of fluid–structure interactions. The fluid and structural mechanics equations need to be solved in their coupled form, and we propose a number of ways to accomplish that. They are: block-iterative coupling, which we widely use in our computations (see[16–19]); quasi-direct coupling; and direct coupling, which is based on the mixed analytical/numerical element-vector-based (AEVB/NEVB) computation technique introduced in.[4,20,21]

The governing equations are reviewed in Section 2. In Section 3 we describe the finite element formulations, including the DSD/SST formulation. The mesh update concepts are covered in Section 4, and that includes the SEMMT and some test computations. The block-iterative, quasi-direct, and direct coupling techniques are described in Section 5. We give two examples of fluid–structure interaction computations in Section 6, and provide some concluding remarks in Section 7. The S-DSD/SST formulation is described in Appendix A.

2 GOVERNING EQUATIONS

2.1 Fluid dynamics

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

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

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

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

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

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

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

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

2.2 Structural dynamics

Let $\Omega_t^s \subset I\!R^{n_{xd}}$ be the spatial domain bounded by Γ_t^s, where $n_{xd} = 2$ for membranes and $n_{xd} = 1$ for cables. The boundary Γ_t^s is composed of $(\Gamma_t^s)_g$ and $(\Gamma_t^s)_h$. Here, the superscript "s" corresponds to the structure. The equations of motion for the structural system are:

$$\rho^s\left(\frac{d^2\mathbf{y}}{dt^2} + \eta\frac{d\mathbf{y}}{dt} - \mathbf{f}^s\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}^s = \mathbf{0}, \tag{5}$$

where, $\mathbf{y}$ is the displacement, ρ^s is the material density, $\mathbf{f}^s$ are the body forces, $\boldsymbol{\sigma}^s$ is the Cauchy stress tensor, and η is the mass-proportional damping coefficient. The damping provides additional stability and may be used for problems where time-accuracy is not important.

In our numerical method, a Lagrangian formulation of the problem is used. Thus, stresses are expressed in terms of the 2nd Piola–Kirchoff stress tensor $\mathbf{S}$, which is related to the Cauchy stress tensor through a kinematic transformation. Under the assumption of large displacements and rotations, small strains, and no material damping, the membranes and cables are treated as Hookean materials with linear elastic properties. For membranes, under the assumption of plane stress, $\mathbf{S}$ becomes

$$S^{ij} = \left(\bar{\lambda}_m G^{ij}G^{kl} + \mu_m\left[G^{il}G^{jk} + G^{ik}G^{jl}\right]\right)E_{kl}, \tag{6}$$

where for the case of isotropic plane stress

$$\bar{\lambda}_m = \frac{2\lambda_m\mu_m}{(\lambda_m + 2\mu_m)}. \tag{7}$$

Here, E_{kl} are the components of the Cauchy–Green strain tensor, G^{ij} are the components of the contravariant metric tensor in the original configuration, and λ_m and μ_m are Lamé constants. For cables, under the assumption of uniaxial tension, $\mathbf{S}$ becomes

$$S^{11} = E_c G^{11} G^{11} E_{11}, \tag{8}$$

where E_c is the cable Young's modulus. To account for stiffness-proportional material damping, the Hookean stress–strain relationships defined by Eqs. (6) and (8) are modified, and E_{kl} is replaced by $\hat{E}_{kl}$, where

$$\hat{E}_{kl} = E_{kl} + \zeta \dot{E}_{kl}. \tag{9}$$

Here, ζ is the stiffness proportional damping coefficient and $\dot{E}_{kl}$ is the time derivative of E_{kl}.

3 FINITE ELEMENT FORMULATIONS

3.1 DSD/SST formulation of fluid dynamics

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} (see Figure 1). At each time step, the integrations

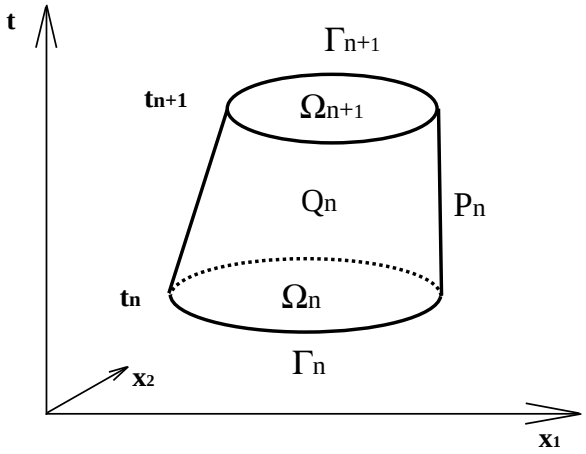


Figure 1: Space–time concept.

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

$p^h \in (\mathcal{S}_p^h)_n$ such that $\forall \mathbf{w}^h \in (\mathcal{V}_{\mathbf{u}}^h)_n$ and $q^h \in (\mathcal{V}_p^h)_n$:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
- &\int_{(P_n)_{\mathrm{h}}} \mathbf{w}^h \cdot \mathbf{h}^h dP + \int_{Q_n} q^h \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho \left((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^- \right) d\Omega \\
+ &\sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) + \tau_{\mathrm{PSPG}} \nabla q^h \right] \cdot \left[\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] dQ \\
+ &\sum_{e=1}^{n_{el}} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot \mathbf{w}^h \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ = 0,
\end{aligned} \tag{10}$$

where

$$\text{Ł}(q^h, \mathbf{w}^h) = \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{w}^h \right) - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h). \tag{11}$$

Here τ_{SUPG}, τ_{PSPG} and ν_{LSIC} are the SUPG, PSPG and LSIC (least-squares on incompressibility constraint) stabilization parameters. For ways of calculating τ_{SUPG}, τ_{PSPG} and ν_{LSIC}, see.[21–23] This formulation is applied to all space–time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, starting with $(\mathbf{u}^h)_0^- = \mathbf{u}_0$. For an earlier, detailed reference on the formulation see.[1]

3.2 Special DSD/SST (S-DSD/SST) formulation

Although it is not a requirement, we assume that the space–time slab Q_n has uniform thickness (temporal dimension). In most space–time computations, all the nodes of the slab Q_n are positioned at time level n or $n+1$, and the spatial mesh at time level $n+1$ is arrived at simply by deformation of the spatial mesh at time level n. These special features, which we expect to see in most DSD/SST computations, are exploited in the Special DSD/SST (S-DSD/SST) formulation for more efficient calculation of the element-level vectors and matrices. The S-DSD/SST formulation is described in more detail in Appendix A. Furthermore, for cases where the spatial mesh consists of linear triangles or tetrahedra, we propose that the spatial part of the integrations over the parent space–time domain be performed analytically. This approach will become more clear in Appendix A.2, where we describe the element-level integrations carried out in the S-DSD/SST formulation.

3.3 Semi-discrete formulation of structural dynamics

With $\mathbf{y}^h$ and $\mathbf{w}^h$ coming from appropriately defined trial and test function spaces, respectively, the semi-discrete finite element formulation of the structural dynamics equations are written as

$$\begin{aligned}
\int_{\Omega_0^s} \mathbf{w}^h \cdot \rho^s \frac{d^2 \mathbf{y}^h}{dt^2} d\Omega^s + \int_{\Omega_0^s} \mathbf{w}^h \cdot \eta \rho^s \frac{d \mathbf{y}^h}{dt} d\Omega^s + \int_{\Omega_0^s} \delta \mathbf{E} : \mathbf{S}^h d\Omega^s = \\
\int_{\Omega_t^s} \mathbf{w}^h \cdot \left(\mathbf{t} + \rho^s \mathbf{f}^s \right) d\Omega^s.
\end{aligned} \tag{12}$$

The fluid dynamics force acting on the structure is represented by vector $\mathbf{t}$. This force term is geometrically nonlinear and thus increases the overall nonlinearity of the formulation. The

left-hand-side terms of Eq. (12) are referred to in the original configuration and the right-hand-side terms for the deformed configuration at time t. From this formulation at each time step we obtain a nonlinear system of equations. In solving that nonlinear system with an iterative method, we use the following incremental form:

$$\left[\frac{\mathbf{M}}{\beta\Delta t^2} + \frac{(1-\alpha)\,\gamma\mathbf{C}}{\beta\Delta t} + (1-\alpha)\,\mathbf{K}\right]\Delta\mathbf{d}^i = \mathbf{R}^i, \tag{13}$$

where

$$\mathbf{C} = \eta\mathbf{M} + \zeta\mathbf{K}. \tag{14}$$

Here $\mathbf{M}$ is the mass matrix, $\mathbf{K}$ is the consistent tangent matrix associated with the internal elastic forces, $\mathbf{C}$ is a damping matrix, $\mathbf{R}^i$ is the residual vector at the i^{th} iteration, and $\Delta\mathbf{d}^i$ is the i^{th} increment in the nodal displacements vector $\mathbf{d}$. The damping matrix $\mathbf{C}$ is used only in stand-alone structural mechanics computations while establishing an inflated canopy shape as the starting shape for the fluid–structure interaction computations. In Eq. (13), all of the terms known from the previous iteration are lumped into the residual vector $\mathbf{R}^i$. The parameters α, β, γ are part of the Hilber–Hughes–Taylor[24] scheme, which is used here for time-integration.

4 MESH UPDATE METHODS

In computation of moving boundaries and interfaces with the DSD/SST formulation, how the mesh is updated depends on several factors. These factors include the complexity of the interface and overall geometry, how unsteady the interface is, and how the starting mesh was generated. In general, the mesh update could have two components: moving the mesh for as long as it is possible, and full or partial remeshing (i.e., generating a new set of elements, and sometimes also a new set of nodes) when the element distortion becomes too high. In mesh moving strategies, the only rule the mesh motion needs to follow is that at the interface the normal velocity of the mesh has to match the normal velocity of the fluid. Beyond that, the mesh can be moved in any way desired, with the main objective being to reduce the frequency of remeshing. In most 3D applications, remeshing requires calling an automatic, unstructured-mesh generator, and therefore reducing the cost of automatic mesh generation becomes a major incentive for reducing the frequency of remeshing. Maintaining the parallel efficiency of the computations is another major incentive for reducing the frequency of remeshing, because parallel efficiency of most automatic mesh generators is substantially lower than that of most flow solvers. Reducing the frequency of remeshing to every ten time steps or less, for example, would sufficiently reduce the influence of remeshing in terms of its added cost and lack of parallel efficiency. In most of the complex flow problems we computed in the past, the frequency of remeshing was far less than every ten time steps. In our current parallel computations on a PC cluster, typically we perform remeshing on one of the nodes, which, with its 2 GigaBytes of memory, is powerful enough to generate very large meshes. If remeshing does not consist of (full or partial) regeneration of just the element connectivities but also involves (full or partial) node regeneration, we need to project the solution from the old mesh to the new one. This involves a search process, which can be carried out in parallel. Still, the computational cost involved in this, and the projection errors introduced by remeshing, add more incentives for reducing the frequency of remeshing.

4.1 Automatic mesh moving technique

In the automatic mesh moving technique introduced in,[12] the motion of the internal nodes is determined by solving the equations of elasticity. As boundary condition, the motion of the nodes at the interfaces is specified to match the normal velocity of the fluid at the interface. Similar mesh moving techniques were used earlier by other researchers (see for example[25]). In[12] the mesh deformation is dealt with selectively based on the sizes of the elements and also the deformation modes in terms of shape and volume changes. Mesh moving techniques with comparable features were later introduced in.[26] In the technique introduced in,[12] selective treatment of the mesh deformation based on shape and volume changes is attained by adjusting the relative values of the Lamé constants of the elasticity equations. The objective would be to stiffen the mesh against shape changes more than we stiffen it against volume changes. Selective treatment based on element sizes, on the other hand, is attained by altering the way we account for the Jacobian of the transformation from the element domain to the physical domain. In this case, the objective is to stiffen the smaller elements, which are typically placed near solid surfaces, more than the larger ones.

4.2 Jacobian-based stiffening

When altering the way the Jacobian is accounted for was first introduced in,[12] it consisted of simply dropping the Jacobian from the finite element formulation of the mesh moving (elasticity) equations. This results in the smaller elements being stiffened more than the larger ones. The method described in[12] was augmented in[27] to a more extensive kind by introducing a stiffening power that determines the degree by which the smaller elements are rendered stiffer than the larger ones. To describe the method, we first write the global integrals involved in the finite element formulation of the mesh moving equations as follows:

$$\int_{\Omega} [\dots] d\Omega = \sum_{e} \int_{\Xi} [\dots]^e J^e d\Xi, \tag{15}$$

where $[\dots]$ symbolically represents what is being integrated, Ξ is the finite element (parent) domain, and the Jacobian for element e is defined as $J^e = \det\left(\partial \boldsymbol{x}/\partial \boldsymbol{\xi}\right)^e$, with $\boldsymbol{x}$ and $\boldsymbol{\xi}$ representing the physical and element (local) coordinates. We alter the way we account for the Jacobian as follows:

$$\int_{\Xi} [\dots]^e J^e d\Xi \quad \longmapsto \quad \int_{\Xi} [\dots]^e J^e \left(\frac{J^0}{J^e}\right)^{\chi} d\Xi, \tag{16}$$

where χ, a non-negative number, is the stiffening power, and J^0, an arbitrary scaling parameter, is inserted into the formulation to make the alteration dimensionally consistent. With $\chi = 0.0$, the method reduces back to an elasticity model with no Jacobian-based stiffening. With $\chi = 1.0$, the method is identical to the one first introduced in.[12] In the general case of $\chi \neq 1.0$, the method stiffens each element by a factor of $(J^e)^{-\chi}$, and χ determines the degree by which the smaller elements are rendered stiffer than the larger ones.

4.3 Solid-Extension Mesh Moving Technique (SEMMT)

In the mesh moving technique introduced in,[12] the structured layers of elements generated around solid objects (to fully control the mesh resolution near solid objects and have more accurate representation of the boundary layers) move "glued" to these solid objects, undergoing

a rigid-body motion. No equations are solved for the motion of the nodes in these layers, because these nodal motions are not governed by the equations of elasticity. This results in some cost reduction. But more importantly, the user has full control of the mesh resolution in these layers. For early examples of automatic mesh moving combined with structured layers of elements undergoing rigid-body motion with solid objects, see.[28] Earlier examples of element layers undergoing rigid-body motion, in combination with deforming structured meshes, can be found in.[1]

In computation of flows with fluid–solid interfaces where the solid is deforming, the motion of the fluid mesh near the interface cannot be represented by a rigid-body motion. Depending on the deformation mode of the solid, we may have to use the automatic mesh moving technique described above. In such cases, the presence of very thin fluid elements near the solid surface becomes a challenge for the automatic mesh moving technique. In the Solid-Extension Mesh Moving Technique (SEMMT),[13–15] we proposed treating those very thin fluid elements almost like an extension of the solid elements. In the SEMMT, in solving the equations of elasticity governing the motion of the fluid nodes, we assign higher rigidity to these thin elements compared to the other fluid elements. Two ways of accomplishing this were proposed in:[13–15] solving the elasticity equations for the nodes connected to the thin elements separate from the elasticity equations for the other nodes, or together. If we solve them separately, for the thin elements, as boundary conditions at the interface with the other elements, we would use traction-free conditions. We refer to the separate solution option as "SEMMT – Multiple Domain (SEMMT–MD)" and the unified solution option as "SEMMT – Single Domain (SEMMT–SD)". In,[29–31] with several test computations, we demonstrated how the SEMMT functions as part of our mesh update method. We employed both of the SEMMT options described above. The test computations included mesh deformation tests[29–31] and a 2D fluid–structure interaction model problem.[31] In Subsections 4.5 and 4.6, we provide brief descriptions of some of those test computations.

4.4 General test conditions and mesh quality measures

The tests reported here are carried out with the standard technique (where $\chi = 1.0$ for all the elements and all the nodes are moved together), SEMMT–SD (with $\chi = 2.0$ for the inner elements and $\chi = 1.0$ for the outer elements), and SEMMT–MD (with $\chi = 1.0$ for all elements in both domains). The mesh over which the elasticity equations are solved is updated at each increment. This update is based on the displacements calculated over the current mesh that has been selectively stiffened. That way, the element Jacobians used in stiffening are updated every time the mesh deforms. As a result, the most current size of an element is used in determining how much it is stiffened. Also as a result, as an element approaches a tangled state, its Jacobian approaches zero, and its stiffening becomes very large. To evaluate the effectiveness of different mesh moving techniques, two measures of mesh quality are defined, similar to those in.[32] They are Element Area Change (f^e_A) and Element Shape Change (f^e_{AR}):

$$\mathsf{f}^e_A = \left|\log\left(\frac{A^e}{A^e_o}\right)\right|, \qquad \mathsf{f}^e_{AR} = \left|\log\left(\frac{AR^e}{AR^e_o}\right)\right|. \tag{17}$$

Here subscript "o" refers to the undeformed mesh (i.e., the mesh obtained after the last remesh), and AR^e is the element aspect ratio, defined as

$$AR^e = \frac{(\ell^e{}_{max})^2}{A^e}, \tag{18}$$

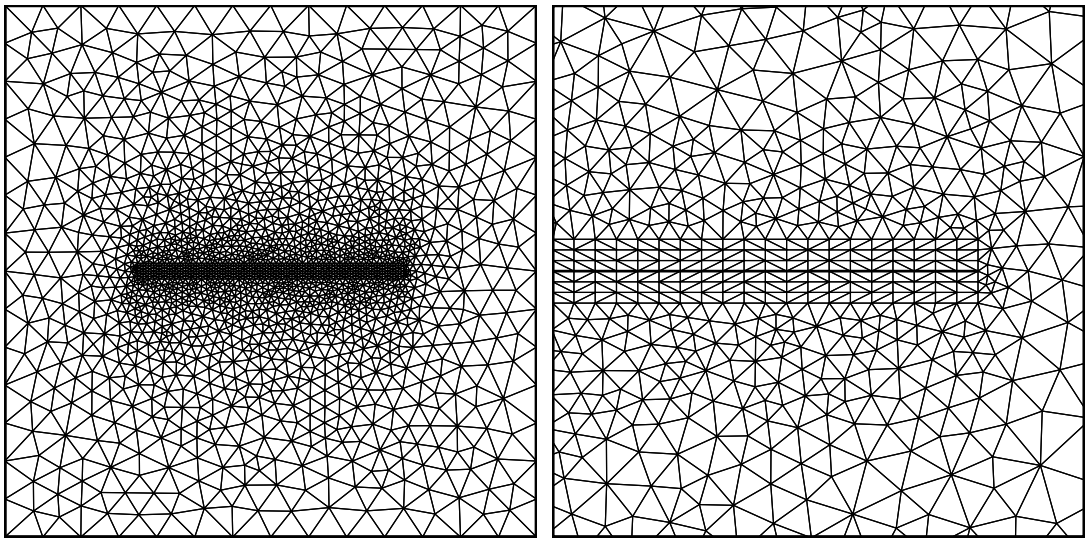


Figure 2: 2D test mesh for SEMMT.

where ${\ell^e}_{max}$ is the maximum edge length for element e. We define array norms for the set of element mesh quality measures as

$$\|\mathsf{f}_A\|_p = \left\{ \sum_e (\mathsf{f}_A^e)^p \right\}^{1/p} , \quad \|\mathsf{f}_{AR}\|_p = \left\{ \sum_e (\mathsf{f}_{AR}^e)^p \right\}^{1/p} , \tag{19}$$

where f_A and f_{AR} are the arrays of mesh quality values f_A and f_{AR} for all elements of interest, and p is the norm of interest. In the following examples, we will consider the norm where $p = \infty$, and

$$\|\mathsf{f}_A\|_\infty = \max_e (\mathsf{f}_A^e) , \quad \|\mathsf{f}_{AR}\|_\infty = \max_e (\mathsf{f}_{AR}^e) . \tag{20}$$

Thus, for a given set of mesh elements, global area and shape changes are defined to be the maximum values of the element area and shape changes, respectively.

4.5 Mesh deformation tests with SEMMT

In this set of tests we use a 2D unstructured mesh consisting of triangular elements and an embedded structure with zero thickness. The mesh spans a region of $|x| \leq 1.0$ and $|y| \leq 1.0$. The structure spans $y = 0.0$ and $|x| \leq 0.5$. Three layers of elements (with $\ell_y = 0.01$) are placed along each side of the structure, with 50 element edges along the structure (i.e. $\ell_x = 0.02$). Figure 2 shows the mesh and its close up view near the structure. The test cases involve three different types of prescribed motion or deformation for the structure identical to the motions prescribed in:[27] rigid-body translation in the y-direction, rigid-body rotation about the origin, and prescribed bending. In the case of prescribed bending, the structure deforms from a line to a circular arc, with no stretch in the structure and no net vertical or horizontal displacement. For each test case the maximum displacement or deformation is reached over 50 increments. Those maximum values are $\Delta y = 0.5$ for the translation test, a rotation of $\Delta\theta = \pi/4$ for the rotation test, and bending to a half circle ($\theta = \pi$) for the bending test. Figure 3 shows, for the SEMMT–MD and the standard mesh moving technique, deformed meshes for the translation, rotation, and bending tests. For more on this set of tests, see.[29–31]

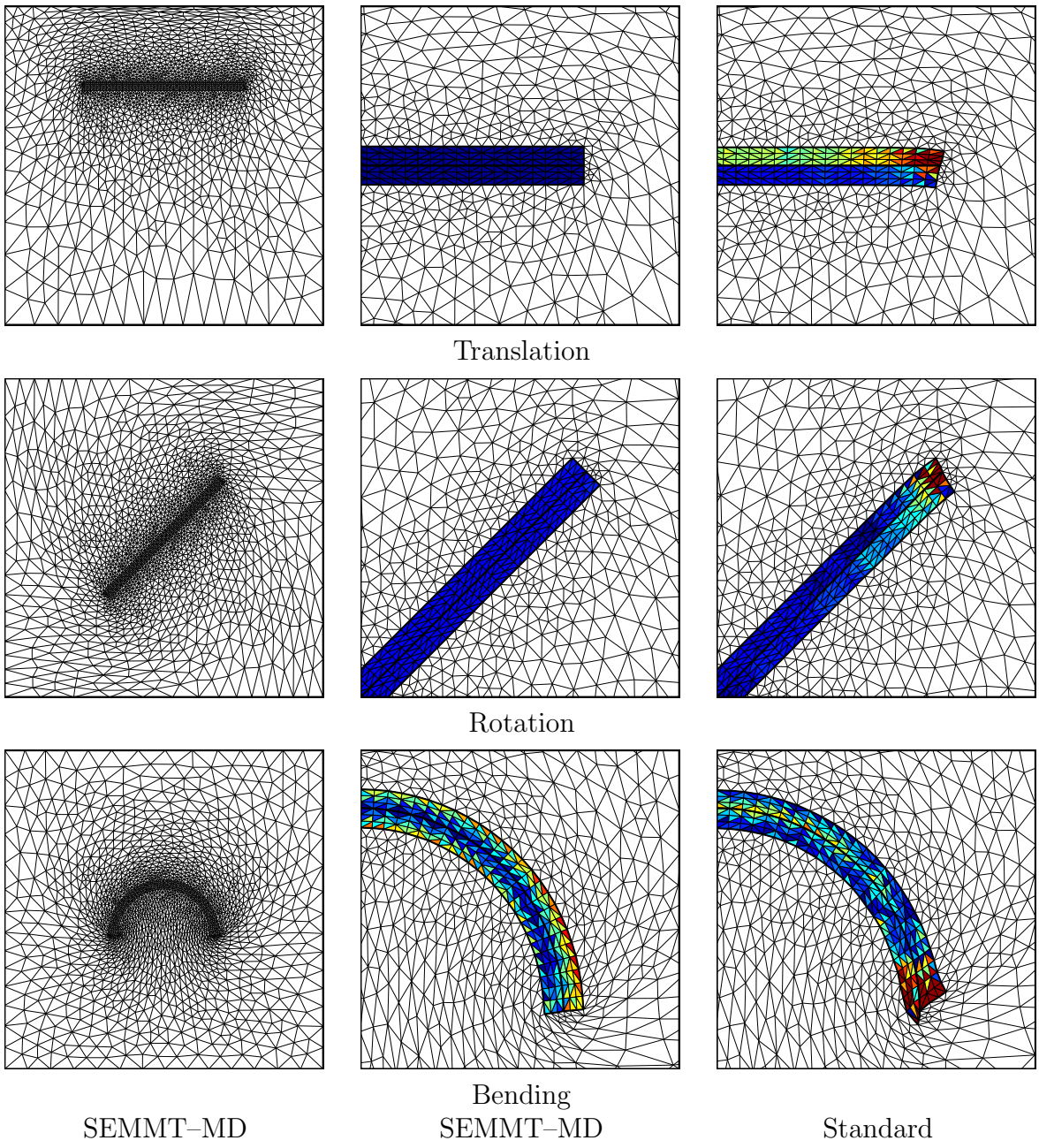


Figure 3: Mesh deformation tests with the SEMMT–MD and the standard mesh moving technique.

4.6 2D fluid–structure interaction model problem with SEMMT

Test computations for this 2D model problem were carried out to show how the SEMMT works in the context of the type of fluid–structure interaction problems encountered in parachute applications. The model represents a parachute-like structure. The "canopy" is modeled with 50 membrane elements and the "suspension lines" with 22 cable elements. Two layers of elements extend outward from the upper and lower surfaces of the canopy. Detailed information on the flow conditions, structural parameters, and the solution steps can be found in.[31] Figure 4 shows the deformed meshes for the standard mesh moving technique and the SEMMT–SD. The orthogonality of the mesh lines at the canopy surface is much better preserved with the SEMMT–SD. For more on this test computation, see.[31]

5 SOLUTION OF FULLY-DISCRETIZED EQUATIONS

Full discretizations of the finite element formulations described in Subsections 3.1 and 3.3 lead to coupled, nonlinear equation systems that need to be solved at every time step of the simulation. In a form that is partitioned with respect to the models represented, these nonlinear equations can be written as follows:

$$\mathbf{N}_1\ (\mathbf{d}_1, \mathbf{d}_2) = \mathbf{F}_1, \tag{21}$$

$$\mathbf{N}_2\ (\mathbf{d}_1, \mathbf{d}_2) = \mathbf{F}_2, \tag{22}$$

where $\mathbf{d}_1$ and $\mathbf{d}_2$ are the vectors of nodal unknowns corresponding to unknown functions $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively. For example, in the context of a coupled fluid–structure interaction problem, $\mathbf{u}_1$ and $\mathbf{u}_2$ might be representing the fluid and structure unknowns, respectively. For the space–time formulation of the fluid dynamics problem, $\mathbf{d}_1$ represents unknowns associated with the finite element formulation written for the space–time slab between the time levels n to $n+1$ (see[1–3]). Solution of these equations with the Newton–Raphson method would necessitate at every Newton–Raphson step solution of the following linear equation system:

$$\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 = \mathbf{b}_1, \tag{23}$$

$$\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 = \mathbf{b}_2, \tag{24}$$

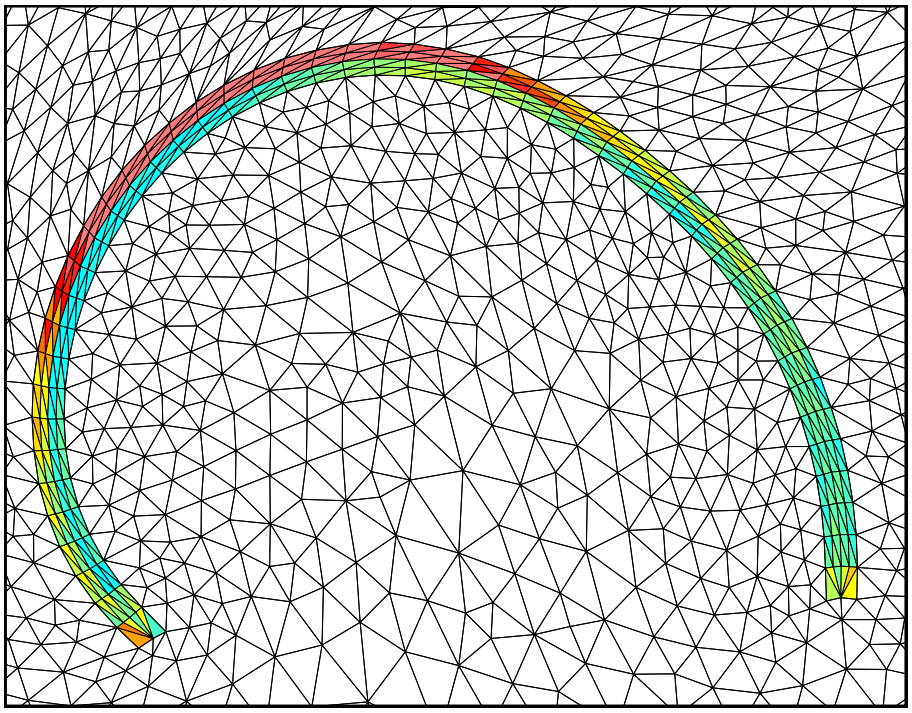
where $\mathbf{b}_1$ and $\mathbf{b}_2$ are the residuals of the nonlinear equation system, $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the correction increments computed for $\mathbf{d}_1$ and $\mathbf{d}_2$, and

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

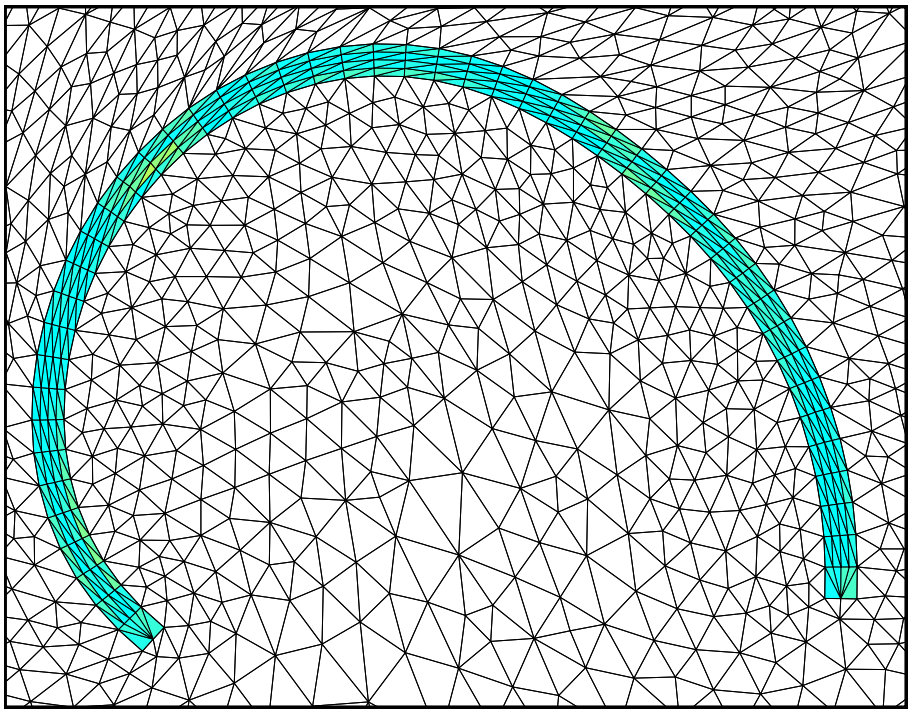
In fluid–structure interactions, keeping the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ in the picture would require taking into account the dependence of Eq. (21) on the mesh motion. In Subsections 5.1–5.3 we describe different ways of handling the coupling between Eqs. (21) and (22) in solving them.

5.1 Block-iterative coupling

In block-iterative coupling, we do not keep the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ in the picture. In an iteration step taking us from iterative solution i to $i+1$, we solve the following set of



Standard Mesh Moving Technique



SEMMT–SD

Figure 4: Mesh deformations of the fluid–structure interaction model problem for the standard mesh moving technique and SEMMT–SD. Colors of the inner elements indicate element distortions as measured by the aspect ratio change. The color range from light blue to light red corresponds to the range $0 \leq \mathrm{f}^{e}_{AR} \leq 0.25$.

equations:

$$\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{\left(\mathbf{d}_1^i,\ \mathbf{d}_2^i\right)} \left(\Delta \mathbf{d}_1^i\right) = \mathbf{F}_1 - \mathbf{N}_1\left(\mathbf{d}_1^i,\ \mathbf{d}_2^i\right), \tag{26}$$

$$\left.\frac{\partial \mathbf{N}_2}{\partial \mathbf{d}_2}\right|_{\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_2^i\right)} \left(\Delta \mathbf{d}_2^i\right) = \mathbf{F}_2 - \mathbf{N}_2\left(\mathbf{d}_1^{i+1},\ \mathbf{d}_2^i\right). \tag{27}$$

The linear equations systems given by Eqs. (26) and (27) are also solved iteratively, using the GMRES search technique.[33]

Because the matrix blocks representing the coupling between the fluid and structural mechanics systems are not in the picture, in fluid–structure interaction computations where the structure is light, structural response becomes very sensitive to small changes in the fluid dynamics forces and convergence becomes difficult to achieve. In Subsections 5.2 and 5.3 we describe ways of keeping the coupling matrix blocks in the picture.

In fluid–structure interaction computations where the structure is light, in the absence of keeping the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ in the picture, we proposed in[21,34] a short cut approach for improving the convergence of the block iterations. In this approach, to reduce "over-correcting" (i.e. "over-incrementing") the structural displacements during the block iterations, we artificially increase the structural mass contribution to the matrix block corresponding to the structural mechanics equations and unknowns. With the understanding that subscript 2 denotes the structure, this would be equivalent to artificially increasing the mass matrix contribution to $\mathbf{A}_{22}$. This is achieved without altering $\mathbf{b}_1$ or $\mathbf{b}_2$ (i.e. $\mathbf{F}_1 - \mathbf{N}_1\left(\mathbf{d}_1, \mathbf{d}_2\right)$ or $\mathbf{F}_2 - \mathbf{N}_2\left(\mathbf{d}_1, \mathbf{d}_2\right)$), and therefore when the block iterations converge, they converge to the solution of the problem with the correct structural mass.

5.2 Quasi-direct coupling

In the quasi-direct coupling approach, we keep the the coupling matrices $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$ in the picture partially, without taking into account the dependence of $\mathbf{A}_{12}$ on the mesh motion. In describing this approach, we re-write the finite element formulations given by Eqs. (10) and (12), with a slight change of notation, and with a clarification of how the fluid–structure interface conditions are handled:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}_{1\mathrm{E}}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h - \mathbf{f}^h\right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}_{1\mathrm{E}}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
- &\int_{(P_n)_\mathrm{h}} \mathbf{w}_{1\mathrm{E}}^h \cdot \mathbf{h}_{1\mathrm{E}}^h dP + \int_{Q_n} q_{1\mathrm{E}}^h \boldsymbol{\nabla}\cdot\mathbf{u}^h dQ + \int_{\Omega_n} (\mathbf{w}_{1\mathrm{E}}^h)_n^+ \cdot \rho\left((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^-\right) d\Omega \\
+ &\sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{1}{\rho}\left[\tau_{\mathrm{SUPG}}\rho\left(\frac{\partial \mathbf{w}_{1\mathrm{E}}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{w}_{1\mathrm{E}}^h\right) + \tau_{\mathrm{PSPG}}\boldsymbol{\nabla} q_{1\mathrm{E}}^h\right] \cdot \left[\mathrm{L}(p^h, \mathbf{u}^h) - \rho\mathbf{f}^h\right] dQ \\
+ &\sum_{e=1}^{n_{el}} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \boldsymbol{\nabla}\cdot\mathbf{w}_{1\mathrm{E}}^h \rho \boldsymbol{\nabla}\cdot\mathbf{u}^h dQ = 0 \,,
\end{aligned} \tag{28}$$

$$\int_{Q_n} q_{1\mathrm{I}}^h \boldsymbol{\nabla}\cdot\mathbf{u}^h dQ + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{1}{\rho}\left[\tau_{\mathrm{PSPG}}\boldsymbol{\nabla} q_{1\mathrm{I}}^h\right] \cdot \left[\mathrm{L}(p^h, \mathbf{u}^h) - \rho\mathbf{f}^h\right] dQ = 0 \,, \tag{29}$$

$$\int_{\Gamma_{1\mathrm{I}}} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot \left((\mathbf{u}_{1\mathrm{I}}^h)_{n+1}^- - \mathbf{u}_{2\mathrm{I}}^h \right) d\Gamma = 0 \ , \tag{30}$$

$$\begin{aligned}
& \int_{Q_n} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} \mathbf{u}^h - \mathbf{f}^h \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}((\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^-) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) dQ \\
- & \int_{(P_n)_{\mathrm{h}}} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot \mathbf{h}_{1\mathrm{I}}^h dP \\
+ & \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \left(\frac{\partial (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^-}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \right) \right] \cdot \left[Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h \right] dQ \\
+ & \sum_{e=1}^{n_{el}} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \boldsymbol{\nabla} \cdot (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \rho \boldsymbol{\nabla} \cdot \mathbf{u}^h dQ = 0 \ ,
\end{aligned} \tag{31}$$

$$\begin{aligned}
& \int_{(\Omega_2)_0} \mathbf{w}_2^h \cdot \rho_2 \frac{d^2 \mathbf{y}^h}{dt^2} d\Omega + \int_{(\Omega_2)_0} \mathbf{w}_2^h \cdot \eta \rho_2 \frac{d \mathbf{y}^h}{dt} d\Omega + \int_{(\Omega_2)_0} \delta \mathbf{E} : \mathbf{S}^h d\Omega \\
= & \int_{\Omega_2} \mathbf{w}_2^h \cdot \rho_2 \mathbf{f}_2^h d\Omega + \int_{\Omega_{2\mathrm{E}}} \mathbf{w}_{2\mathrm{E}}^h \cdot \mathbf{h}_{2\mathrm{E}}^h d\Omega - \int_{\Omega_{2\mathrm{I}}} \mathbf{w}_{2\mathrm{I}}^h \cdot \mathbf{h}_{1\mathrm{I}}^h d\Omega \ .
\end{aligned} \tag{32}$$

Here the subscripts 1 and 2 refer to the fluid and structure, respectively. While the subscript I refers to the fluid–structure interface, the subscript E refers to "elsewhere" in the fluid and structure domains or boundaries. In reconciling the slightly modified notation used here with the notation we used in Eqs. (10) and (12), we note that $\rho_2 = \rho^s$, $\mathbf{f}_2^h = \mathbf{f}^s$, $(\Omega_2)_0 = \Omega_0^s$, $\Omega_2 = \Omega_t^s$, and $\Omega_{2\mathrm{I}}$ and $\Omega_{2\mathrm{E}}$ indicate the partitions of Ω_2 corresponding to the interface and "elsewhere". We also note that $\mathbf{h}_{1\mathrm{I}}^h = -\mathbf{t}$, and $\mathbf{h}_{2\mathrm{E}}^h$ denotes the prescribed external forces acting on the structure in $\Omega_{2\mathrm{E}}$, which is separate from $\mathbf{f}_2^h$. In this formulation, $(\mathbf{u}_{1\mathrm{I}}^h)_{n+1}^-$ and $\mathbf{h}_{1\mathrm{I}}^h$ (the fluid velocity and stress at the interface) are treated as separate unknowns, and Eqs. (30) and (31) can be seen as equations corresponding to these two unknowns, respectively. The structural displacement rate at the interface, $\mathbf{u}_{2\mathrm{I}}^h$, is derived from $\mathbf{y}^h$.

The formulation above is based on allowing for cases when the fluid and structure meshes at the interface are not identical. If they are identical, the same formulation can still be used, but one can also use its reduced version where Eq. (30) is no longer needed and $\mathbf{h}_{1\mathrm{I}}^h$ is no longer treated as a separate unknown.

If the structure is represented by a 3D continuum model instead of a membrane model, the formulation above would still be applicable if the the domain integrations over $\Omega_{2\mathrm{E}}$ and $\Omega_{2\mathrm{I}}$ in the last two terms of Eq. (32) are converted to boundary integrations over $\Gamma_{2\mathrm{E}}$ and $\Gamma_{2\mathrm{I}}$. In such cases, $\mathbf{h}_{2\mathrm{E}}^h$ would represent the prescribed forces acting "elsewhere" on the surface of the structure.

5.3 Direct coupling

The mixed analytical/numerical element-vector-based (AEVB/NEVB) computation technique introduced in[4,20,21] can be employed to keep the coupling matrices in the picture fully by taking into account their dependence on the mesh motion. In describing the mixed AEVB/NEVB

technique, we first write the iterative solution of the equation system given by Eqs. (23) and (24) as follows:

$$\mathbf{P}_{11}\Delta\mathbf{y}_1 + \mathbf{P}_{12}\Delta\mathbf{y}_2 = \mathbf{b}_1 - (\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2)\,, \tag{33}$$

$$\mathbf{P}_{21}\Delta\mathbf{y}_1 + \mathbf{P}_{22}\Delta\mathbf{y}_2 = \mathbf{b}_2 - (\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2)\,, \tag{34}$$

where $\Delta\mathbf{y}_1$ and $\Delta\mathbf{y}_2$ represent the candidate corrections to $\mathbf{x}_1$ and $\mathbf{x}_2$, and $\mathbf{P}_{\beta\gamma}$'s represent the blocks of the preconditioning matrix $\mathbf{P}$. Here we focus our attention on computation of the residual vectors on the right hand side, and explore ways for evaluating the matrix–vector products.

Let us suppose that we are able to compute, without major difficulty, the element-level matrices $\mathbf{A}^e_{11}$ and $\mathbf{A}^e_{22}$ associated with the global matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$, and that we prefer to evaluate $\mathbf{A}_{11}\mathbf{x}_1$ and $\mathbf{A}_{22}\mathbf{x}_2$ by using these element-level matrices. Let us also suppose that calculation of the element-level matrices $\mathbf{A}^e_{12}$ and $\mathbf{A}^e_{21}$ is exceedingly difficult. Reflecting these circumstances, the computations can be carried out by using a mixed element-matrix-based/element-vector-based computation technique:[4,20,21]

$$(\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2) = \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{11}\mathbf{x}_1) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_1\to 0} \left[\frac{\mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2 + \epsilon_1\mathbf{x}_2) - \mathbf{N}^e_1(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_1}\right], \tag{35}$$

$$(\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2) = \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{22}\mathbf{x}_2) + \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_2\to 0} \left[\frac{\mathbf{N}^e_2(\mathbf{d}_1 + \epsilon_2\mathbf{x}_1, \mathbf{d}_2) - \mathbf{N}^e_2(\mathbf{d}_1, \mathbf{d}_2)}{\epsilon_2}\right], \tag{36}$$

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

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

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} (\mathbf{A}^e_{\beta\gamma}\mathbf{x}_\gamma), \tag{37}$$

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_\beta\to 0} \left[\frac{\mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma + \epsilon_\beta\mathbf{x}_\gamma, \ldots) - \mathbf{N}^e_\beta(\ldots, \mathbf{d}_\gamma, \ldots)}{\epsilon_\beta}\right]. \tag{38}$$

Sometimes, computation of the element-level matrices $\mathbf{A}^e_{\beta\gamma}$ might not be exceedingly 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 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 $\mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \Delta\mathbf{u}_\gamma)$ represent 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[4,20,21]

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \frac{\partial \mathbf{N}_\beta}{\partial \mathbf{d}_\gamma}\mathbf{x}_\gamma \quad = \quad \mathop{\mathsf{A}}_{e=1}^{n_{el}} \mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_\gamma), \tag{39}$$

where, $\mathbf{v}_\gamma$ is a function interpolated from $\mathbf{x}_\gamma$ in the same way that $\mathbf{u}_\gamma$ is interpolated from $\mathbf{d}_\gamma$. This element-vector-based computation technique allows us to evaluate matrix–vector products without dealing with numerical evaluation of directional derivatives. To differentiate between the element-vector-based computation techniques defined by Eqs. (38) and (39), we call them, respectively, numerical element-vector-based (NEVB) and analytical element-vector-based (AEVB) computation techniques.

We propose two ways of using the mixed AEVB/NEVB computation technique to take into account the dependence of the coupling matrices on the mesh motion. In the first way, we propose to use the NEVB technique to compute $\mathbf{A}_{12}\mathbf{x}_2$ while including the dependence of $\mathbf{A}_{12}$ on the mesh motion. This would be done as seen in Eq. (35). In the second way we propose, we limit the use of the NEVB technique to evaluation of the matrix–vector products involving coupling matrices representing the dependence on the mesh motion. This would be done by considering a three-block version of the nonlinear equation system given by Eqs. (21) – (22), where $\mathbf{d}_3$ is the vector of nodal unknowns representing the mesh motion, and the third block of equations represents the mesh-moving equations. The three-block version of Eqs. (23) – (24) can be solved iteratively with the three-block of version of Eqs. (33) – (34), which is written as follows:

$$\mathbf{P}_{11}\Delta\mathbf{y}_1 + \mathbf{P}_{12}\Delta\mathbf{y}_2 + \mathbf{P}_{13}\Delta\mathbf{y}_3 \quad = \quad \mathbf{b}_1 - (\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{A}_{13}\mathbf{x}_3)\,, \tag{40}$$

$$\mathbf{P}_{21}\Delta\mathbf{y}_1 + \mathbf{P}_{22}\Delta\mathbf{y}_2 + \mathbf{P}_{23}\Delta\mathbf{y}_3 \quad = \quad \mathbf{b}_2 - (\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{A}_{23}\mathbf{x}_3)\,, \tag{41}$$

$$\mathbf{P}_{31}\Delta\mathbf{y}_1 + \mathbf{P}_{32}\Delta\mathbf{y}_2 + \mathbf{P}_{33}\Delta\mathbf{y}_3 \quad = \quad \mathbf{b}_3 - (\mathbf{A}_{31}\mathbf{x}_1 + \mathbf{A}_{32}\mathbf{x}_2 + \mathbf{A}_{33}\mathbf{x}_3)\,. \tag{42}$$

The NEVB technique can be used for computing $\mathbf{A}_{13}\mathbf{x}_3$ as follows:

$$\mathbf{A}_{13}\mathbf{x}_3 = \mathop{\mathsf{A}}_{e=1}^{n_{el}} \lim_{\epsilon_1 \to 0} \left[\frac{\mathbf{N}_1^e(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 + \epsilon_1 \mathbf{x}_3) - \mathbf{N}_1^e(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)}{\epsilon_1} \right]. \tag{43}$$

6 NUMERICAL EXAMPLES

6.1 Soft landing of a T–10 parachute

In this test problem, we simulate the soft-landing of a T–10 parachute using block-iterative coupling. T–10 is a 35 ft diameter personnel parachute with 30 suspension lines of 29.4 ft length. The suspension lines meet at a single confluence point, which is connected to the payload point mass by a 20 ft pneumatic muscle actuator (PMA). Although T–10 is designed for 222 lb payload and 16 ft/s descent speed, the terminal landing speed is further reduced by retracting the PMA just prior to landing. In the present simulation the PMA is retracted by 5.25 ft in 0.27 s. As seen in Figure 5 the muscle retraction results in a significant decrease in payload descent speed. The minimum payload speed occurs in the post retraction phase of the parachute descent. During the retraction, the mass matrix contribution to $\mathbf{A}_{22}$ is increased by multiplying it by a factor 10. A factor of 5 is used in the post-retraction phase. Figure 6 shows the parachute shapes before and after retraction.

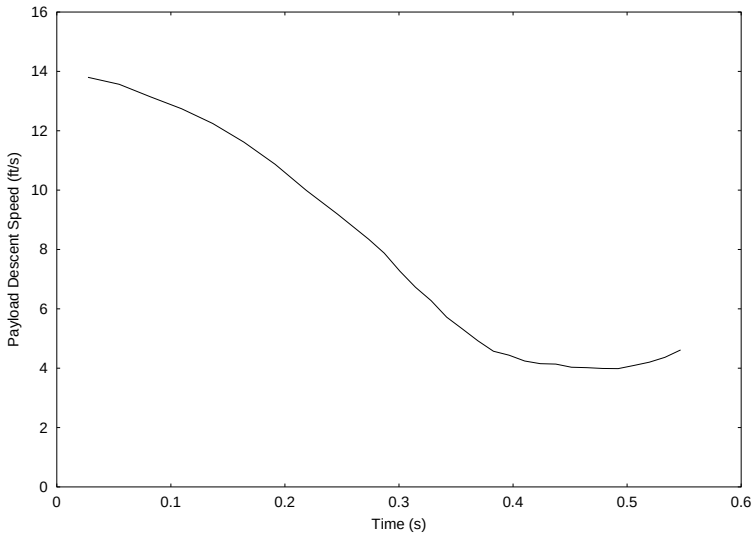


Figure 5: Descent speed of the payload during and after PMA retraction.

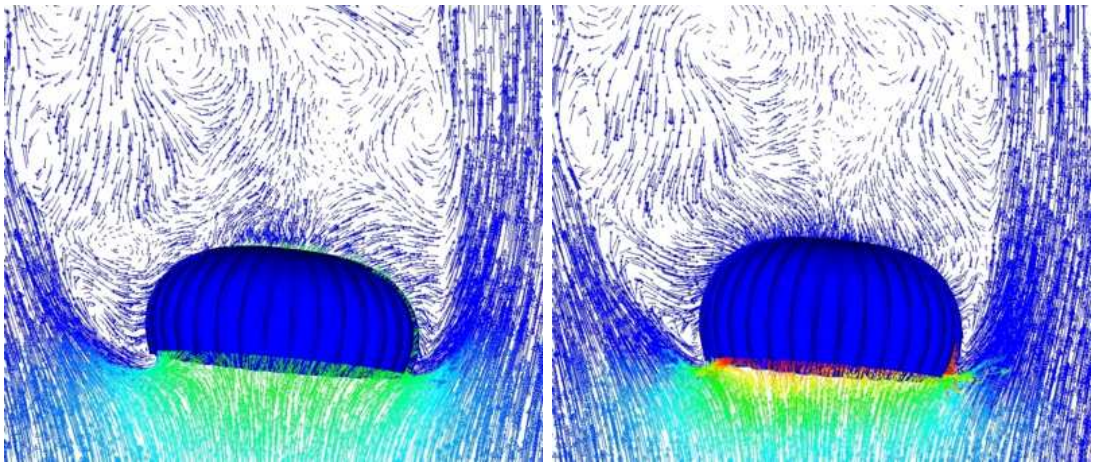


Figure 6: Velocity vectors around T–10, before (left) and after (right) PMA retraction. The velocity vectors are colored with pressure.

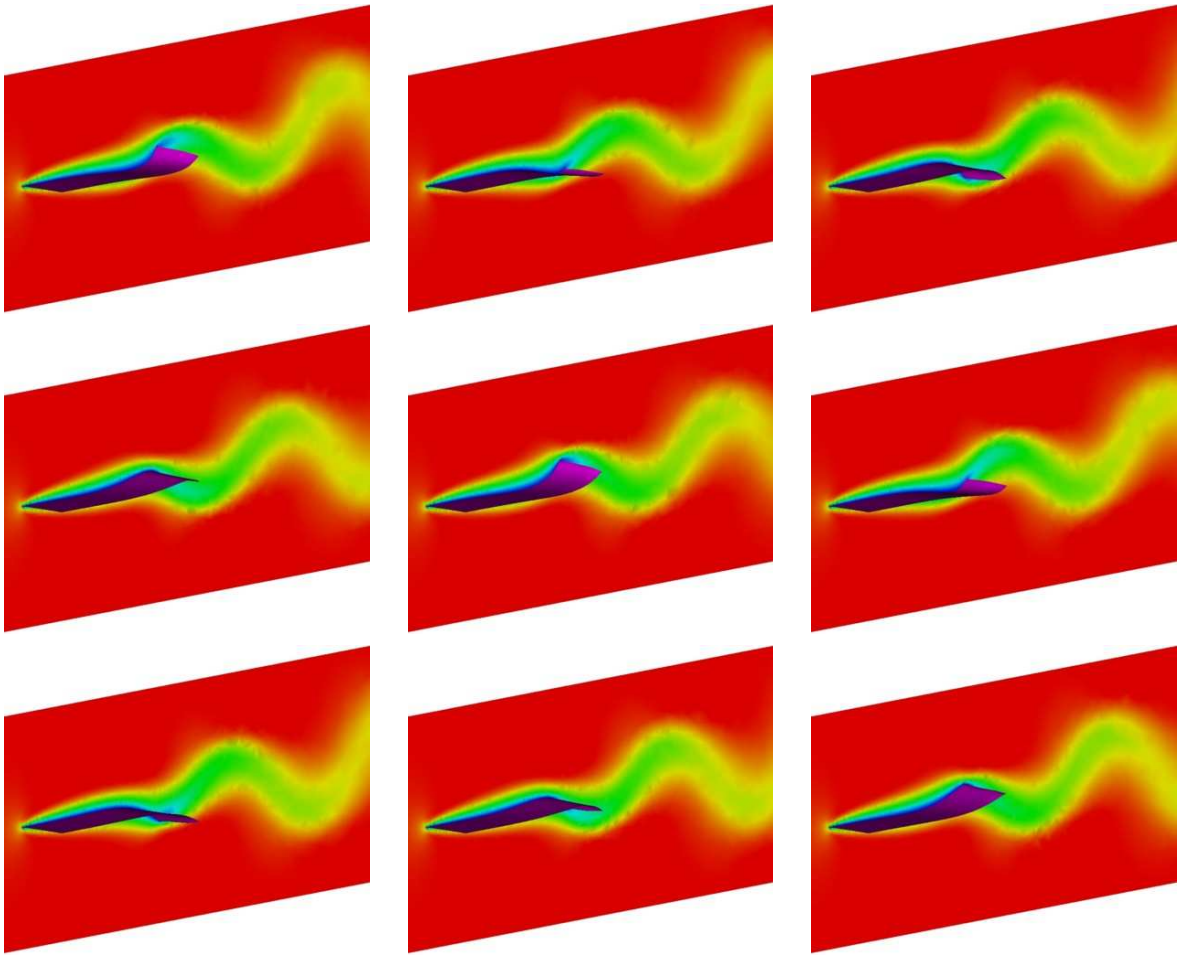


Figure 7: Time history (left to right and top to bottom) of the shape of the flag and the horizontal velocity on a normal plane.

6.2 Flow past a "flag"

In this test problem, we simulate the fluid–structure interactions involved in the flapping of a "flag". We use the quasi-direct coupling approach for this simulation. The length of the flag is 1.5 m in the direction of flow and 1.0 m in the span-wise direction. The fluid velocity, density and kinematic viscosity are 2 m/s, 1 kg/m^3 and 0.008 m^2/s respectively. The flag is modeled as a membrane structure with Young's modulus 40,000 N/m^2 and density 1000 kg/m^3. The thickness of the flag is 0.2 mm. The leading edge of the flag is held fixed and the lateral edges of the flag are constrained to move only in a normal plane. The fluid–structure interaction computations are carried out until a regular pattern of flapping is observed. Figure 7 shows a sequence of snapshots of the shape of the flag and the horizontal velocity on a normal plane. Figure 8 shows the displacement and velocity fluctuations for the midpoint of the free edge of the flag.

7 CONCLUDING REMARKS

We described a set of techniques for computation of fluid mechanics problems with moving boundaries and interfaces, with emphasis on fluid–structure interactions. The core method is the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation. We developed a number of techniques to support and enhance this core method. The Special DSD/SST

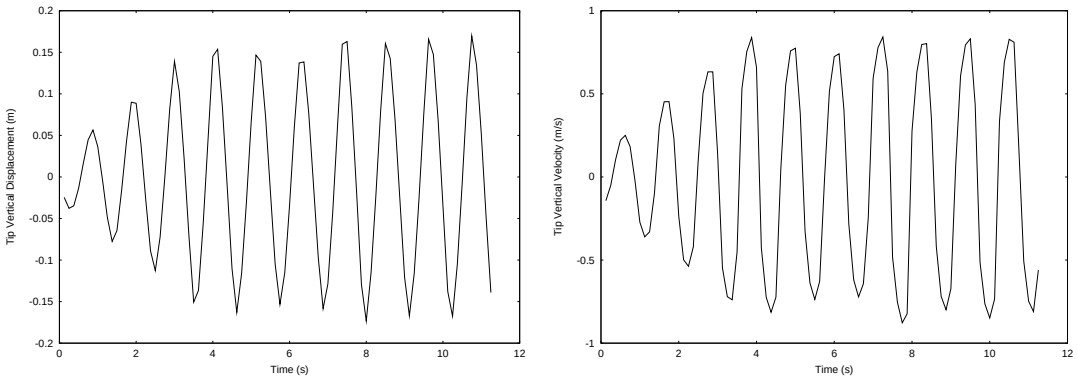


Figure 8: Vertical displacement (left) and velocity (right) of the midpoint of the free edge of the flag.

formulation helps us make the element-level vector and matrix computations more efficient. The mesh update methods, which include the Solid-Extension Mesh Moving Technique (SEMMT), help us update the mesh effectively as the spatial domain occupied by the fluid changes in time. A number of techniques have been developed for the solution of the fully-discretized, coupled fluid and structural mechanics equations. These are the block-iterative, quasi-direct, and direct coupling techniques. The quasi-direct and and direct coupling techniques are particularly suited for fluid–structure interaction computations where the structure is light. We presented some test computations for the mesh moving techniques we developed. We also presented results from some test computations for fluid–structure interactions, including one where the structure is light. We believe we demonstrated in this article that the techniques we are developing are substantially increasing the scope and accuracy of the simulations for moving boundaries and interfaces in general and fluid–structure interactions in particular.

ACKNOWLEDGMENTS

This work was supported by the US Army Natick Soldier Center (Contract No. DAAD16-03-C-0051), NSF (Grant No. EIA-0116289), and NASA Johnson Space Center (Grant No. NAG9-1435).

REFERENCES

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

[2] T.E. Tezduyar, 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.

[3] 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.

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

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

[6] 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.

[7] 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).

[8] T.J.R Hughes and T.E. Tezduyar, "Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations", *Computer Methods in Applied Mechanics and Engineering*, **45** (1984) 217–284.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] T. 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, ASME, New York, New York, (2001).

[14] T.E. Tezduyar, "Stabilized finite element formulations and interface-tracking and interface-capturing techniques for incompressible flows", in M.M. Hafez, editor, *Numerical Simulations of Incompressible Flows*, World Scientific, New Jersey, (2003) 221–239.

[15] 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).

[16] K. Stein, R. Benney, V. Kalro, T.E. Tezduyar, J. Leonard, and M. Accorsi, "Parachute fluid–structure interactions: 3-D Computation", *Computer Methods in Applied Mechanics and Engineering*, **190** (2000) 373–386.

[17] K. Stein, R. Benney, T. Tezduyar, and J. Potvin, "Fluid–structure interactions of a cross parachute: Numerical simulation", *Computer Methods in Applied Mechanics and Engineering*, **191** (2001) 673–687.

[18] K.R. Stein, R.J. Benney, T.E. Tezduyar, J.W. Leonard, and M.L. Accorsi, "Fluid–structure interactions of a round parachute: Modeling and simulation techniques", *Journal of Aircraft*, **38** (2001) 800–808.

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

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

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

[22] 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.

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

[24] H.M. Hilber, T.J.R. Hughes, and R.L. Taylor, "Improved numerical dissipation for time integration algorithms in structural dynamics", *Earthquake Engineering and Structural Dynamics*, **5** (1977) 283–292.

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

[26] A. Masud and T.J.R. Hughes, "A space–time Galerkin/least-squares finite element formulation of the Navier-Stokes equations for moving domain problems", *Computer Methods in Applied Mechanics and Engineering*, **146** (1997) 91–126.

[27] K. Stein, T. Tezduyar, and R. Benney, "Mesh moving techniques for fluid–structure interactions with large displacements", *Journal of Applied Mechanics*, **70** (2003) 58–63.

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

[29] K. Stein, T. Tezduyar, and R. Benney, "Mesh update with solid-extension mesh moving technique", in *Pre-Conference Proceedings of the Sixth Japan-US International Symposium on Flow Simulation and Modeling*, Fukuoka, Japan, (2002).

[30] K. Stein and T. Tezduyar, "Advanced mesh update techniques for problems involving large displacements", in *Proceedings of the Fifth World Congress on Computational Mechanics*, On-line publication: http://wccm.tuwien.ac.at/, Paper-ID: 81489, Vienna, Austria, (2002).

[31] K. Stein, T.E. Tezduyar, and R. Benney, "Automatic mesh update with the solid-extension mesh moving technique", *Computer Methods in Applied Mechanics and Engineering*, **193** (2004) 2019–2032.

[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] 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.

[34] T.E. Tezduyar, "Stabilized finite element methods for computation of flows with moving boundaries and interfaces", in *Lecture Notes on Finite Element Simulation of Flow Problems (Basic - Advanced Course)*, Japan Society of Computational Engineering and Sciences, Tokyo, Japan, (2003).

[35] T.J.R Hughes, *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

A APPENDIX: SPECIAL DSD/SST (S-DSD/SST) FORMULATION

We assume that the space–time slab Q_n has uniform thickness. In most space–time computations, all the nodes of a slab are on its upper or lower surfaces. Also, in most space–time computations, the mesh on the upper surface of the slab is obtained from the deformation of the mesh on the lower surface. These special features are exploited in the S-DSD/SST formulation, so that the element-level vectors and matrices can be calculated more efficiently. The S-DSD/SST formulation can result in two types of savings in computational cost. First, the calculation of shape function derivatives at Gaussian quadrature points can be simplified. Second, the formation of the element-level vectors and matrices can be broken down and simplified.

Because these computations need to be done for each element of the space–time mesh for each iteration of every time step, they constitute a large portion of the overall computational cost for a problem. Therefore, improvements of the way in which they are calculated can have a significant impact on the overall performance of the computer program. The S-DSD/SST formulation can be seamlessly applied to parallel computations as well, because it is implemented at the element level. In this appendix we describe the S-DSD/SST formulation in detail. Furthermore, in computations with spatial meshes made of linear triangles or tetrahedra, we propose that the spatial part of the integrations over the parent space–time domain be performed analytically. This approach is explained more in Appendix A.2.

A.1 Shape functions and their derivatives

The S-DSD/SST formulation can be described by first separating the typical shape function into its spatial and temporal parts:

$$N_a^\alpha \;=\; N_a T^\alpha \;=\; N_a(\boldsymbol{\xi})\; T^\alpha(\theta)\;, \qquad a = 1, 2, \ldots n_{en}\;, \qquad \alpha = 1, 2 \quad , \tag{A1}$$

where n_{en} is the number of spatial nodes, and

$$T^1(\theta) \;=\; \frac{1}{2}(1-\theta) \quad , \quad T^2(\theta) \;=\; \frac{1}{2}(1+\theta) \quad , \tag{A2}$$

$$T^1_{,\theta} \;=\; -\frac{1}{2} \quad , \qquad\qquad T^2_{,\theta} \;=\; +\frac{1}{2} \quad . \tag{A3}$$

Refer to Figure A.1 to better visualize how a and α relate to the geometry of a typical space–time element.

Note that the spatial shape functions are left undefined because they are rather arbitrary and depend on the number of spatial dimensions as well as the element type. On the other hand, the temporal shape functions are defined, based on the special features of the space–time mesh.

No matter how the spatial shape functions are defined for a given number of spatial dimensions and element type, the spatial coordinate vector is interpolated as follows:

$$\boldsymbol{x} \;=\; \sum_{\alpha=1}^{2}\sum_{a=1}^{n_{en}} N_a^\alpha \boldsymbol{x}_a^\alpha \;=\; \sum_{\alpha=1}^{2}\sum_{a=1}^{n_{en}} N_a T^\alpha \boldsymbol{x}_a^\alpha \tag{A4}$$

$$\;=\; \sum_{a=1}^{n_{en}} N_a \left(T^1 \boldsymbol{x}_a^1 + T^2 \boldsymbol{x}_a^2\right) \;=\; \sum_{a=1}^{n_{en}} N_a \boldsymbol{x}_a(\theta) \;=\; \sum_{a=1}^{n_{en}} N_a \boldsymbol{x}_a \quad . \tag{A5}$$

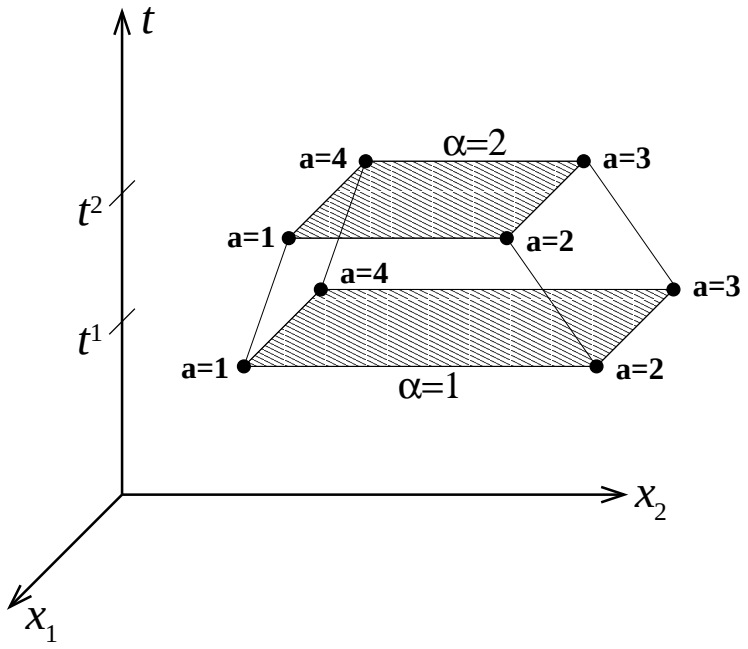


Figure A1: Notation for a space–time element in S-DSD/SST formulation.

We interpolate t in a similar fashion while recognizing that $t_a^\alpha = t^\alpha$ for $a = 1, 2, \ldots n_{en}$:

$$t = \sum_{\alpha=1}^{2}\sum_{a=1}^{n_{en}} N_a^\alpha t_a^\alpha = \sum_{\alpha=1}^{2}\sum_{a=1}^{n_{en}} N_a T^\alpha t^\alpha \tag{A6}$$

$$= \left(\sum_{a=1}^{n_{en}} N_a\right)\sum_{\alpha=1}^{2} T^\alpha t^\alpha = \sum_{\alpha=1}^{2} T^\alpha t^\alpha = T^1 t^1 + T^2 t^2 \ . \tag{A7}$$

Without taking into account the special features of the space–time mesh, one would normally have to calculate the inverse of an $(n_{sd}+1)\times(n_{sd}+1)$ matrix in order to obtain the derivatives of the shape functions with respect to the global coordinates. Implementing the S-DSD/SST formulation would effectively reduce the problem to taking the inverse of an $n_{sd}\times n_{sd}$ matrix. Because the inverse calculation is not a trivial computation, reducing the matrix to be inverted to $n_{sd}\times n_{sd}$ can result in significant computational savings. We will examine the calculations involved for $n_{sd} = 2$, or a 3D mesh in the space–time domain:

$$\begin{pmatrix} N_{a,x}^\alpha \\ N_{a,y}^\alpha \\ N_{a,t}^\alpha \end{pmatrix} = \begin{pmatrix} \xi_{,x} & \eta_{,x} & \theta_{,x} \\ \xi_{,y} & \eta_{,y} & \theta_{,y} \\ \xi_{,t} & \eta_{,t} & \theta_{,t} \end{pmatrix} \begin{pmatrix} N_{a,\xi}^\alpha \\ N_{a,\eta}^\alpha \\ N_{a,\theta}^\alpha \end{pmatrix} = \boldsymbol{R}^{\mathrm{ST}} \begin{pmatrix} N_{a,\xi}^\alpha \\ N_{a,\eta}^\alpha \\ N_{a,\theta}^\alpha \end{pmatrix}, \tag{A8}$$

$$\begin{pmatrix} N_{a,x}^\alpha \\ N_{a,y}^\alpha \\ N_{a,t}^\alpha \end{pmatrix} = \begin{pmatrix} x_{,\xi} & y_{,\xi} & t_{,\xi} \\ x_{,\eta} & y_{,\eta} & t_{,\eta} \\ x_{,\theta} & y_{,\theta} & t_{,\theta} \end{pmatrix}^{-1} \begin{pmatrix} N_{a,\xi}^\alpha \\ N_{a,\eta}^\alpha \\ N_{a,\theta}^\alpha \end{pmatrix} = (\boldsymbol{Q}^{\mathrm{ST}})^{-1} \begin{pmatrix} N_{a,\xi}^\alpha \\ N_{a,\eta}^\alpha \\ N_{a,\theta}^\alpha \end{pmatrix}. \tag{A9}$$

From Eq. (A7) $t_{,\xi} = t_{,\eta} = 0$ and $t_{,\theta} = \frac{\Delta t}{2}$. This simplifies the matrix $\boldsymbol{Q}^{\mathrm{ST}}$ as follows:

$$\boldsymbol{Q}^{\mathrm{ST}} = \begin{pmatrix} x_{,\xi} & y_{,\xi} & 0 \\ x_{,\eta} & y_{,\eta} & 0 \\ x_{,\theta} & y_{,\theta} & \frac{\Delta t}{2} \end{pmatrix}. \tag{A10}$$

Now, taking the inverse becomes as simple as taking the inverse of a 2×2 matrix, as opposed to the more complicated 3×3 matrix inverse. Similarly, a 4×4 matrix can be simplified to a 3×3 matrix. To take the inverse of a 2×2 matrix, we must calculate the determinant of a 2×2 matrix. The determinant and inverse are found to be, respectively:

$$det\boldsymbol{Q}^{\rm ST} \;=\; \frac{\Delta t}{2} det \begin{pmatrix} x_{,\xi} & y_{,\xi} \\ x_{,\eta} & y_{,\eta} \end{pmatrix} \;=\; \frac{\Delta t}{2} J(\xi,\eta,\theta) \;=\; \frac{\Delta t}{2} J \;, \tag{A11}$$

$$(\boldsymbol{Q}^{\rm ST})^{-1} \;=\; \boldsymbol{R}^{\rm ST} \;=\; \frac{1}{\frac{\Delta t}{2}J} \begin{pmatrix} \frac{\Delta t}{2} y_{,\eta} & -\frac{\Delta t}{2} y_{,\xi} & 0 \\ -\frac{\Delta t}{2} x_{,\eta} & \frac{\Delta t}{2} x_{,\xi} & 0 \\ -x_{,\theta} y_{,\eta} + y_{,\theta} x_{,\eta} & x_{,\theta} y_{,\xi} - y_{,\theta} x_{,\xi} & J \end{pmatrix} . \tag{A12}$$

We can rewrite $\boldsymbol{R}^{\rm ST}$ as

$$\boldsymbol{R}^{\rm ST} \;=\; \begin{pmatrix} \frac{1}{J} \begin{pmatrix} y_{,\eta} & -y_{,\xi} \\ -x_{,\eta} & x_{,\xi} \end{pmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} \frac{-x_{,\theta} y_{,\eta} + y_{,\theta} x_{,\eta}}{\frac{\Delta t}{2} J} & \frac{x_{,\theta} y_{,\xi} - y_{,\theta} x_{,\xi}}{\frac{\Delta t}{2} J} \end{matrix} & \frac{2}{\Delta t} \end{pmatrix} , \tag{A13}$$

and define:

$$\boldsymbol{R} \;=\; \frac{1}{J} \begin{pmatrix} y_{,\eta} & -y_{,\xi} \\ -x_{,\eta} & x_{,\xi} \end{pmatrix} , \tag{A14}$$

$$V_\xi \;=\; \frac{-x_{,\theta} y_{,\eta} + y_{,\theta} x_{,\eta}}{\frac{\Delta t}{2} J} \quad , \quad V_\eta \;=\; \frac{x_{,\theta} y_{,\xi} - y_{,\theta} x_{,\xi}}{\frac{\Delta t}{2} J} \quad . \tag{A15}$$

Already for $n_{sd} = 2$, one can see the computational-cost savings afforded by rewriting the derivative calculations for the shape functions. The improvement in efficiency would be even more substantial for $n_{sd} = 3$.

Let us now write the derivatives of the global coordinates with respect to the local coordinates in terms of the shape functions:

$$\boldsymbol{x}_{,\xi} \;=\; \sum_{a=1}^{n_{en}} N_{a,\xi} \boldsymbol{x}_a(\theta) \;, \tag{A16}$$

$$\boldsymbol{x}_{,\eta} \;=\; \sum_{a=1}^{n_{en}} N_{a,\eta} \boldsymbol{x}_a(\theta) \;, \tag{A17}$$

$$\boldsymbol{x}_{,\theta} \;=\; \sum_{a=1}^{n_{en}} N_a \frac{\boldsymbol{x}_a^2 - \boldsymbol{x}_a^1}{2} \;=\; \sum_{a=1}^{n_{en}} N_a \frac{\Delta \boldsymbol{x}_a}{2} \;. \tag{A18}$$

By dividing Eq. (A18) by $\frac{\Delta t}{2}$ and defining the nodal mesh velocity as $\boldsymbol{V}_a = \frac{\Delta \boldsymbol{x}_a}{\Delta t}$ we obtain:

$$\frac{\boldsymbol{x}_{,\theta}}{\frac{\Delta t}{2}} \;=\; \sum_{a=1}^{n_{en}} N_a \frac{\Delta \boldsymbol{x}_a}{\Delta t} \;=\; \boldsymbol{V} \;, \tag{A19}$$

where $\boldsymbol{V}$ is the mesh velocity. We can now define V_ξ and V_η in terms $\boldsymbol{V}$ and $\boldsymbol{R}$:

$$\begin{pmatrix} V_\xi \\ V_\eta \end{pmatrix} \;=\; -\,\boldsymbol{V} \cdot \boldsymbol{R} \;. \tag{A20}$$

Referring back to Eqs. (A8) and (A13)–(A15), we can rewrite our shape function derivatives as

$$\boldsymbol{\nabla} N_a^\alpha = \boldsymbol{R} \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a^\alpha \,, \tag{A21}$$

$$N_{a,t}^\alpha = (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a^\alpha + \frac{2}{\Delta t} N_{a,\theta}^\alpha \,, \tag{A22}$$

where $\boldsymbol{\nabla}_{\boldsymbol{\xi}}$ is the spatial gradient operator with respect to the local coordinates. Using Eq. (A1), we separate the shape functions into their spatial and temporal parts as follows:

$$\boldsymbol{\nabla} N_a^\alpha = \boldsymbol{R} \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) T^\alpha \,, \tag{A23}$$

$$N_{a,t}^\alpha = (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) T^\alpha + \frac{2}{\Delta t} N_a T_{,\theta}^\alpha \,. \tag{A24}$$

By substituting $\alpha = 1, 2$ and using Eq. (A3) to calculate $T_{,\theta}^\alpha$, we can more explicitly write the derivatives as

$$\begin{pmatrix} \boldsymbol{\nabla} N_a^1 \\ \boldsymbol{\nabla} N_a^2 \end{pmatrix} = \boldsymbol{R} \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) \begin{pmatrix} T^1 \\ T^2 \end{pmatrix}, \tag{A25}$$

$$\begin{pmatrix} N_{a,t}^1 \\ N_{a,t}^2 \end{pmatrix} = (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) \begin{pmatrix} T^1 \\ T^2 \end{pmatrix} + \frac{N_a}{\Delta t} \begin{pmatrix} -1 \\ +1 \end{pmatrix}. \tag{A26}$$

It is evident that there are far fewer operations to perform than would be required for the inverse of the full 3×3 matrix.

A.2 Calculation of element-level vectors

Besides savings from reformulating the shape function derivatives, there are also savings to be had by reformulating the way in which the element-level vectors are calculated. To illustrate that let us consider the numerical integration, over a space–time element, of a generic term $N_a^\alpha y$ by using the Gaussian quadrature rule:

$$\int_Q N_a^\alpha y dQ = \sum_{j=1}^{n_{int}^t} \sum_{k=1}^{n_{int}^{sp}} \left(N_a^\alpha y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_{kj}^{\mathrm{ST}} \,. \tag{A27}$$

Here n_{int}^t is the number of Gaussian quadrature points in time, n_{int}^{sp} is the number of Gaussian quadrature points in space, $\tilde{\boldsymbol{\xi}}_k$ is kth Gaussian quadrature point in space, $\tilde{\theta}_j$ is the jth Gaussian quadrature point in time, and W_{kj}^{ST} are the integration weights in the parent space–time domain. We will assume $n_{int}^t = 2$ and use n_{int} in place of n_{int}^{sp}. Using Eq. (A1), we separate the shape functions into their spatial and temporal parts, and rewrite Eq. (A27) as follows:

$$\int_Q N_a^\alpha y dQ = \sum_{j=1}^{2} \sum_{k=1}^{n_{int}} \left(N_a(\boldsymbol{\xi}_k) T^\alpha(\theta_j) y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \,. \tag{A28}$$

Here W_k are the integration weights in the parent spatial domain. The two integration weights in the parent time domain are both unity. Because $T^\alpha(\tilde{\theta}_j)$ is independent of $\tilde{\boldsymbol{\xi}}_k$, we can take it out of the inner summation over k:

$$\int_Q N_a^\alpha y dQ = \sum_{j=1}^{2} \left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \right] T^\alpha(\tilde{\theta}_j) \,. \tag{A29}$$

Expanding Eq. (A29) helps elucidate our potential for savings:

$$\begin{aligned}\int_Q N_a^\alpha y dQ &= \left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_1} W_k\right] T^\alpha(\tilde{\theta}_1) \\ &+ \left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_2} W_k\right] T^\alpha(\tilde{\theta}_2) \ . \end{aligned} \tag{A30}$$

It is clearer that now with the S-DSD/SST formulation, the terms to be integrated will need to be evaluated (at every integration point) only for each $a = 1, 2, \ldots n_{en}$ rather than for each combination of $a = 1, 2, \ldots n_{en}$ and $\alpha = 1, 2$, as was done in Eq. (A27).

Terms with a spatial differential operator acting on N_a^α can also be integrated with the help of Eq. (A30) by simply having the same differential operator also act on N_a on the right-hand-side. For example, in integration of the term $\left(\left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\right) N_a^\alpha\right) y$, we replace N_a with $\left(\left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\right) N_a\right)$ $= \left(\left(\mathbf{u}^h \cdot \boldsymbol{R} \cdot \boldsymbol{\nabla}_{\boldsymbol{\xi}}\right) N_a\right)$ on the right-hand-side of Eq. (A30).

We note that the terms in the brackets in Eq. (A30), apart from $\frac{\Delta t}{2}$, are numerical versions of the integrations over the parent spatial domain at time levels $\tilde{\theta}_2$ and $\tilde{\theta}_2$. If the spatial mesh consists of linear triangles or tetrahedra, we propose that those spatial integrations be performed analytically. Analytical expressions for spatial integrations over straight-edged triangles and flat-surfaced tetrahedra can be found in[35]).

The previous examples were fairly straight-forward, and they represent a majority of the element-level vector calculations. However, the S-DSD/SST formulation for an integral involving the time derivative of a shape function is fundamentally different and slightly more complicated. Therefore, as another example, we consider the integration of a term involving the time-derivative of a shape function:

$$\int_Q N_{a,t}^\alpha y dQ = \sum_{j=1}^{2}\sum_{k=1}^{n_{int}} \left(N_a^\alpha y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k \ . \tag{A31}$$

Separating $N_{a,t}^\alpha$ will not be as simple as it was for $N_a^\alpha = N_a T^\alpha$. Instead, we must refer back to Eq. (A24). Substituting, our numerical integration becomes

$$\int_Q N_{a,t}^\alpha y dQ = \sum_{j=1}^{2}\sum_{k=1}^{n_{int}} \left[\left((-\boldsymbol{V}\cdot\boldsymbol{R})\cdot(\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) T^\alpha + \frac{2}{\Delta t} N_a T_{,\theta}^\alpha\right) y \frac{\Delta t}{2} J\right]\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k \ . \tag{A32}$$

Because $T^\alpha(\tilde{\theta}_j)$ and $T_{,\theta}^\alpha(\tilde{\theta}_j)$ are independent of $\tilde{\boldsymbol{\xi}}_k$, we can take them out of the inner summation over k:

$$\begin{aligned}\int_Q N_{a,t}^\alpha y dQ &= \sum_{j=1}^{2}\left[\sum_{k=1}^{n_{int}} \left((-\boldsymbol{V}\cdot\boldsymbol{R})\cdot(\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k\right] T^\alpha(\tilde{\theta}_j) \\ &+ \sum_{j=1}^{2}\left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k\right] \frac{2}{\Delta t} T_{,\theta}^\alpha(\tilde{\theta}_j) \ . \end{aligned} \tag{A33}$$

We already know the values of $T_{,\theta}^\alpha(\theta_j)$ from Eq. (A3), and once we substitute $j = 1, 2$, we

are left with an expanded equation comparable to Eq. (A30):

$$\begin{aligned}\int_Q N_{a,t}^{\alpha} y dQ &= \left[\sum_{k=1}^{n_{int}} \left((-\boldsymbol{V}\cdot\boldsymbol{R})\cdot(\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_1} W_k\right] T^{\alpha}(\tilde{\theta}_1) \\ &+ \left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_1} W_k\right] \frac{(-1)^{\alpha}}{\Delta t} \\ &+ \left[\sum_{k=1}^{n_{int}} \left((-\boldsymbol{V}\cdot\boldsymbol{R})\cdot(\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_a) y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_2} W_k\right] T^{\alpha}(\tilde{\theta}_2) \\ &+ \left[\sum_{k=1}^{n_{int}} \left(N_a y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_2} W_k\right] \frac{(-1)^{\alpha}}{\Delta t} \quad . \end{aligned} \tag{A34}$$

A.3 Calculation of element-level matrices

The savings to be had by reformulating the way in which the element-level matrices are calculated are even more substantial. To show that let us consider the numerical integration, over a space–time element, of a generic term $N_a^{\alpha} N_b^{\beta} y$:

$$\int_Q N_a^{\alpha} N_b^{\beta} y dQ = \sum_{j=1}^{n_{int}^t} \sum_{k=1}^{n_{int}^{sp}} \left(N_a^{\alpha} N_b^{\beta} y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_{kj}^{\mathrm{ST}} \ , \quad a,b = 1...n_{en} \ , \quad \alpha,\beta = 1,2 \ . \tag{A35}$$

As was done for the element-level vectors, we will assume $n_{int}^t = 2$ and use n_{int} in place of n_{int}^{sp}. Using Eq. (A1), we separate the shape functions into their spatial and temporal parts, and rewrite Eq. (A35) as follows:

$$\int_Q N_a^{\alpha} N_b^{\beta} y dQ = \sum_{j=1}^{2} \sum_{k=1}^{n_{int}} \left(N_a(\boldsymbol{\xi}_k) T^{\alpha}(\theta_j) N_b(\boldsymbol{\xi}_k) T^{\beta}(\theta_j) y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k \ . \tag{A36}$$

Because $T^{\alpha}(\tilde{\theta}_j) T^{\beta}(\tilde{\theta}_j)$ is independent of $\tilde{\boldsymbol{\xi}}_k$, we can take it out of the inner summation over k:

$$\int_Q N_a^{\alpha} N_b^{\beta} y dQ = \sum_{j=1}^{2} \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_j} W_k\right] T^{\alpha}(\tilde{\theta}_j) T^{\beta}(\tilde{\theta}_j) \ . \tag{A37}$$

Expanding Eq. (A37) helps elucidate our potential for savings:

$$\begin{aligned}\int_Q N_a^{\alpha} N_b^{\beta} y dQ &= \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_1} W_k\right] T^{\alpha}(\tilde{\theta}_1) T^{\beta}(\tilde{\theta}_1) \\ &+ \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J\right)\Big|_{\tilde{\boldsymbol{\xi}}_k,\tilde{\theta}_2} W_k\right] T^{\alpha}(\tilde{\theta}_2) T^{\beta}(\tilde{\theta}_2) \quad . \end{aligned} \tag{A38}$$

With the S-DSD/SST formulation, the terms to be integrated will need to be evaluated (at every integration point) only for each combination of $a = 1,2,\ldots n_{en}$ and $b = 1,2,\ldots n_{en}$ rather than for each combination of $a = 1,2,\ldots n_{en}$ and $b = 1,2,\ldots n_{en}$ and $\alpha = 1,2$ and $\beta = 1,2$, as was done in Eq. (A35). For the element-level matrices, we cut the number of evaluations

at integration points by a factor of four, whereas for the element-level vectors, we halve the number of evaluations.

Integration of terms involving the time derivative of a shape function is, again, fundamentally different and slightly more complicated. Therefore, as another example, we consider the integration of a term involving the time-derivative of a shape function:

$$\int_Q N_a^\alpha N_{b,t}^\beta y dQ = \sum_{j=1}^{2} \sum_{k=1}^{n_{int}} \left(N_a^\alpha N_{b,t}^\beta y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \ . \tag{A39}$$

From Eq. (A24), our numerical integration becomes:

$$\int_Q N_a^\alpha N_{b,t}^\beta y dQ = \sum_{j=1}^{2} \sum_{k=1}^{n_{int}} \left(N_a T^\alpha \left((-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_b) T^\beta + \frac{2}{\Delta t} N_b T_{,\theta}^\beta \right) y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \ . \tag{A40}$$

Because $T^\alpha(\tilde{\theta}_j) T^\beta(\tilde{\theta}_j)$ and $T^\alpha(\tilde{\theta}_j) T_{,\theta}^\beta(\tilde{\theta}_j)$ are independent of $\tilde{\boldsymbol{\xi}}_k$, we can take them out of the inner summation over k:

$$\begin{aligned} \int_Q N_a^\alpha N_{b,t}^\beta y dQ \ &= \ \sum_{j=1}^{2} \left[\sum_{k=1}^{n_{int}} \left(N_a (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_b) y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \right] T^\alpha(\tilde{\theta}_j) T^\beta(\tilde{\theta}_j) \\ &+ \ \sum_{j=1}^{2} \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_j} W_k \right] T^\alpha(\tilde{\theta}_j) \frac{2}{\Delta t} T_{,\theta}^\beta(\tilde{\theta}_j) \ . \end{aligned} \tag{A41}$$

We already know the values of $T_{,\theta}^\alpha(\theta_j)$ from Eq. (A3), and once we substitute $j = 1, 2$ we are left with an expanded equation comparable to Eq. (A38):

$$\begin{aligned} \int_Q N_a^\alpha N_{b,t}^\beta y dQ \ &= \ \left[\sum_{k=1}^{n_{int}} \left(N_a (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_b) y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_1} W_k \right] T^\alpha(\tilde{\theta}_1) T^\beta(\tilde{\theta}_1) \\ &+ \ \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_1} W_k \right] T^\alpha(\tilde{\theta}_1) \frac{(-1)^\beta}{\Delta t} \\ &+ \ \left[\sum_{k=1}^{n_{int}} \left(N_a (-\boldsymbol{V} \cdot \boldsymbol{R}) \cdot (\boldsymbol{\nabla}_{\boldsymbol{\xi}} N_b) y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_2} W_k \right] T^\alpha(\tilde{\theta}_2) T^\beta(\tilde{\theta}_2) \\ &+ \ \left[\sum_{k=1}^{n_{int}} \left(N_a N_b y \frac{\Delta t}{2} J \right) \Big|_{\tilde{\boldsymbol{\xi}}_k, \tilde{\theta}_2} W_k \right] T^\alpha(\tilde{\theta}_2) \frac{(-1)^\beta}{\Delta t} \ . \end{aligned} \tag{A42}$$