INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **54**:855–900
Published online 25 January 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1430

Modelling of fluid–structure interactions with the space–time finite elements: Solution techniques

Tayfun E. Tezduyar*, † and Sunil Sathe

Team for Advanced Flow Simulation and Modeling (T★AFSM), Mechanical Engineering, Rice University — MS 321, 6100 Main Street, Houston, TX 77005, U.S.A.

SUMMARY

The space–time fluid–structure interaction (FSI) techniques developed by the Team for Advanced Flow Simulation and Modeling (T★AFSM) have been applied to a wide range of 3D computation of FSI problems, some as early as in 1994 and many with challenging complexities. In this paper, we review these space–time FSI techniques and describe the enhancements introduced recently by the T★AFSM to increase the scope, accuracy, robustness and efficiency of these techniques. The aspects of the FSI solution process enhanced include the deforming-spatial-domain/stabilized space–time (DSD/SST) formulation, the fluid–structure interface conditions, the preconditioning techniques used in iterative solution of the linear equation systems, and a contact algorithm protecting the quality of the fluid mechanics mesh between the structural surfaces coming into contact. We present a number of 3D numerical examples computed with these new stabilized space–time FSI (SSTFSI) techniques.

Received 22 November 2006; Accepted 6 December 2006

KEY WORDS: fluid–structure interactions; finite elements; space–time methods; segregated solution techniques

1. INTRODUCTION

Modelling of fluid–structure interaction (FSI) problems has been receiving much attention in recent years, and we are seeing many good papers and PhD theses on this subject (see, for example, [1–28]). The main reason is that most engineering applications involve some sort of FSI problem, and the FSI researchers have been quite successful in modelling such applications, from pipes [29] to parachutes [30–38] and from blood flow [14, 16, 39–42] to bridges [23] and tents [28]. Another reason is that FSI modelling offers a number of numerical challenges for computational researchers.

*Correspondence to: Tayfun E. Tezduyar, Team for Advanced Flow Simulation and Modeling (T★AFSM), Mechanical Engineering, Rice University — MS 321, 6100 Main Street, Houston, TX 77005, U.S.A.
†E-mail: tezduyar@rice.edu

The spatial domain occupied by the fluid changes in time as the fluid–structure interface moves, and the numerical model will need to handle that. Accurate representation of the flow field near the interface requires that the mesh be updated to track the interface, and this requires special attention in 3D problems with complex geometries. Solution of the coupled fluid and structural mechanics equations offers additional challenges. What technique should be used to solve those coupled equations depends on a number of factors, including (a) how sensitive the structure is to the variations in the fluid dynamics forces; (b) how much programme modularity is desired for the fluid and structural mechanics solvers, and (c) how much previous experience one has with the kind of FSI problem that needs to be computed. Structural surfaces coming into contact create additional challenges for the generation and update of the fluid mechanics mesh, and this needs to be addressed with a contact algorithm that can guard the quality of the mesh. In this paper, we describe the computational methods developed by the Team for Advanced Flow Simulation and Modeling (T⋆AFSM)[‡] to address these FSI challenges. We also describe the new versions of these methods, developed recently by the T⋆AFSM to increase the scope, accuracy, robustness and efficiency of the their FSI computations.

In FSI modelling, the preference of the T⋆AFSM has always been using an interface-tracking technique. In this category of techniques, as the structure moves and the spatial domain occupied by the fluid changes its shape, the mesh moves to accommodate this shape change and to follow (i.e. 'track') the fluid–structure interface. Moving the fluid mesh to track the interface enables us to control the mesh resolution near that interface and obtain more accurate solutions in such critical flow regions. One of the most well-known examples of the interface-tracking techniques is the arbitrary Lagrangian–Eulerian (ALE) finite element formulation [43], which by itself has been receiving much attention (see, for example, [17, 44, 45]). In fact, a good majority of the interface-tracking FSI algorithms are based on the ALE formulation (see, for example, [1, 2, 8, 11, 15, 16, 18–23, 26, 28]). Most of these ALE-based FSI algorithms also include the streamline-upwind/Petrov–Galerkin (SUPG) [46, 47] and pressure-stabilizing/Petrov–Galerkin (PSPG) [48, 49] formulations. The SUPG formulation prevents numerical instabilities that might be encountered when we have high Reynolds number and strong boundary layers. With the PSPG formulation, we can use, without numerical instabilities, equal-order interpolation functions for velocity and pressure. An earlier version of the pressure stabilization, for Stokes flows, was introduced in [50].

As the interface-tracking technique the T⋆AFSM has always been using the deforming-spatial-domain/stabilized space–time (DSD/SST) formulation [48, 51, 52], which was introduced by the T⋆AFSM in 1991. In the DSD/SST formulation, the stabilization is again based on the SUPG and PSPG formulations. The DSD/SST formulation was originally intended to be a general-purpose interface-tracking technique for simulation of problems involving moving boundaries or interfaces, whether fluid–solid or fluid–fluid. The stabilized space–time formulations were introduced and tested earlier by other researchers in the context of problems with fixed spatial domains (see, for example, [53]), mainly because of the superior stability and accuracy characteristics of these formulations. In the DSD/SST method, the space–time computations are carried out for 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

[‡]This team name was intended to imply the research team led by Tezduyar also prior to when the team assumed this specific name.

DOI: 10.1002/fld

the time dimension. Some additional special features are exploited in the special DSD/SST (S-DSD/SST) formulation [35, 36] to make the calculation of the element-level vectors and matrices more efficient. It is also worth noting here that the version of the DSD/SST formulation given in [54] is clearer in representing the implementation used in the computations the T★AFSM has been practicing from the beginning (see Remark 1 in Section 3.1).

In the mesh update strategy envisioned originally with the DSD/SST formulation, the updating is based on moving it for as many time steps as we can and remeshing (generating fully or partially new set of nodes or elements) only as frequently as we need to. Moving the mesh is accomplished with the method introduced in [55] and described in more detail in [56]. The motion of the nodes is governed by the equations of elasticity and the mesh deformation is dealt with selectively based on the sizes of the elements (see also [57]). The Jacobian of the mapping from the element domain to the physical domain is dropped in the finite element formulation of the elasticity equations. This 'Jacobian-based stiffening' is equivalent to dividing the elastic modulus by the element Jacobian and results in an increase in the stiffness of the smaller elements, which are typically placed near the fluid–structure interfaces. Mesh-moving techniques with functionally comparable features were later introduced in [58]. The T★AFSM introduced a number of enhancements to the general mesh update technique originally introduced in [55]. A mesh-moving optimization study based on using different ratios of the Lamé parameters of the elasticity equations was reported in [56]. A 'stiffening exponent' was introduced in [59] for the Jacobian-based stiffening, together with a mesh-moving optimization study based on using different values of this exponent. The solid-extension mesh-moving technique (SEMMT) [60, 61] addresses the challenges involved in moving a mesh with very thin fluid elements near the solid surfaces. In the move-reconnect-renode mesh update method (MRRMUM) [38, 62], two remeshing options are defined, with each one proposed to be used when it is most effective to do so. The mesh update and mesh-moving topics, including the various enhancements introduced by the T★AFSM, will be discussed more in Section 4.

The space–time FSI techniques introduced by the T★AFSM use as their core technologies the DSD/SST formulation and mesh update methods described in earlier paragraphs of this section. The structural mechanics equations are solved using a semi-discrete finite element formulation. We see no compelling reason to use a space–time formulation for those equations. The first applications of these techniques were reported in [63] for 2D flow computation of vortex-induced vibrations of a cylinder (0D structure), and in [29] for 3D computation of flow in a flexible, cantilevered pipe (1D structure). The earliest application to axisymmetric FSI computations was reported in [30] for modelling the inflation of a parachute. Applications to 3D FSI computations with incompressible flows and membranes and cables were first reported, in the context of parachute simulations, in [31–33, 64, 65]. More applications to 3D parachute FSI computations were reported in [34, 66].

In the space–time FSI techniques (or in other interface-tracking FSI techniques), at each time step, one needs to solve the fully discretized, coupled fluid and structural mechanics and mesh-moving equations. As it was mentioned in the first paragraph, the solution approach would be determined by factors such as the sensitivity of the structure to the variations in the fluid dynamics forces, the desired level of programme modularity for the fluid and structural solvers, and the level of previous experience with the FSI problem to be computed. The different solution approaches developed by the T★AFSM are suitable for emphasizing different selections of these factors. The quasi-direct [35–37] and direct coupling techniques [35–37] give us more robust algorithms for FSI computations where the structure is light and therefore more sensitive to the variations in the fluid dynamics forces. They would be preferred especially when one does not have much previous experience with the kind of FSI problem that needs to be computed. We do not get that level of

DOI: 10.1002/fld

robustness from the block-iterative coupling technique [35–37, 67–71] or its earlier and even less robust versions [29–31]. But the block-iterative technique gives us more flexibility in terms of algorithmic modularity and independence of the fluid and structural mechanics solvers and also better parallel efficiency.

As we discussed in the earlier paragraphs of this section, the space–time FSI techniques developed by the T⋆AFSM have been applied to a wide range of 3D computation of FSI problems, some as early as in 1994 (see [29]) and many with challenging complexities (see, for example, [31, 33–38, 62, 72]). The enhancements we describe in this paper increase the scope, accuracy, robustness and efficiency of the FSI techniques developed by the T⋆AFSM. The aspects of the FSI solution process enhanced include the DSD/SST formulation, the fluid–structure interface conditions, the preconditioning techniques used in iterative solution of the linear equation systems, and a contact algorithm protecting the quality of the fluid mechanics mesh between the structural surfaces coming into contact. The contact algorithm, which is called the surface-edge-node contact-tracking (SENCT) technique and is in early stages of its development and testing, was introduced in [38]. The 3D computation of a rather complex test problem was also reported in [38]. The SENCT will be described in Section 7 briefly and in another T⋆AFSM publication in more detail and with more test computations.

The governing equations are reviewed in Section 2. The finite element formulations, including the DSD/SST formulation, are described in Section 3. In Section 4 we review the mesh update techniques, including the SEMMT and MRRMUM. In Sections 5 and 6 we describe the iterative solution techniques, block-iterative, quasi-direct, and direct coupling techniques, and the preconditioning techniques. The SENCT contact algorithm is described briefly in Section 7. In Sections 8 and 9, for the purpose of comparison and generalization, we briefly describe how the SUPG and PSPG stabilizations can be extended to the ALE formulation and problems with thermal coupling. In Section 10, we provide a brief history of the parallel implementations of the space–time FSI solvers developed by the T⋆AFSM. A brief history of parachute modelling with those FSI solvers is provided in Section 11. Numerical examples are presented in Section 12, and the concluding remarks are given in Section 13.

2. GOVERNING EQUATIONS

2.1. Fluid mechanics

Let $\Omega_t \subset \mathbb{R}^{n_{sd}}$ be the spatial 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

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

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

where ρ, $\mathbf{u}$ and $\mathbf{f}$ are the density, velocity and the external force, respectively. The stress tensor $\boldsymbol{\sigma}$ is defined as $\boldsymbol{\sigma}(p, \mathbf{u}) = -p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\mathbf{u})$, with $\boldsymbol{\varepsilon}(\mathbf{u}) = ((\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}})/2$. 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 Equation (1) are represented as $\mathbf{u} = \mathbf{g}$ on $(\Gamma_t)_g$ and $\mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h}$ on $(\Gamma_t)_h$, where $(\Gamma_t)_g$ and $(\Gamma_t)_h$ are complementary subsets of the boundary

DOI: 10.1002/fld

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 mechanics

Let $\Omega_t^s \subset \mathbb{R}^{n_{xd}}$ be the spatial domain with boundary Γ_t^s, where $n_{xd}=2$ for membranes and $n_{xd}=1$ for cables. The superscript 's' indicates the structure. The parts of Γ_t^s corresponding to the essential and natural boundary conditions are represented by $(\Gamma_t^s)_g$ and $(\Gamma_t^s)_h$. The equations of motion are written as

$$\rho^s \left(\frac{\mathrm{d}^2\mathbf{y}}{\mathrm{d}t^2} + \eta \frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} - \mathbf{f}^s \right) - \nabla \cdot \boldsymbol{\sigma}^s = \mathbf{0} \tag{3}$$

where ρ^s, $\mathbf{y}$, $\mathbf{f}^s$ and $\boldsymbol{\sigma}^s$ are the material density, structural displacement, external force and the Cauchy stress tensor, respectively. Here, η is an artificial-damping coefficient, which is non-zero only in computations where time accuracy is not required, such as in determining the deformed shape of the structure for specified fluid mechanics forces acting on it. Such computations typically precede any fluid mechanics or FSI computations, and the artificial damping facilitates reaching that initial shape in a robust way. The stresses are expressed in terms of the second 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 characterized with linearly elastic material properties. For membranes, under the assumption of plane stress, $\mathbf{S}$ becomes:

$$S^{ij} = (\bar{\lambda}^s G^{ij} G^{kl} + \mu^s (G^{il} G^{jk} + G^{ik} G^{jl})) E_{kl} \tag{4}$$

where for the case of isotropic plane stress $\bar{\lambda}^s = 2\lambda^s \mu^s / (\lambda^s + 2\mu^s)$. Here, E_{kl} are the components of the Cauchy–Green strain tensor, G^{ij} are the contravariant components of the metric tensor in the original configuration, and λ^s and μ^s are the Lamé constants. For cables, under the assumption of uniaxial tension, $\mathbf{S}$ becomes $S^{11} = E_c G^{11} G^{11} E_{11}$, where E_c is Young's modulus for the cable.

3. FINITE ELEMENT FORMULATIONS

3.1. DSD/SST formulation of fluid mechanics

In the DSD/SST method [48, 51, 52, 54], the finite element formulation 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 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 $(\cdot)_n^+$ will denote the function values at t_n as approached from below and above. Each Q_n is decomposed into elements Q_n^e, where $e = 1, 2, \ldots, (n_{\mathrm{el}})_n$. The subscript n used with n_{el} is for the general case where the number of space–time elements may change from one space–time slab to another. The essential and natural boundary conditions are enforced over $(P_n)_g$ and $(P_n)_h$, the complementary subsets of the lateral boundary of the space–time slab. The finite element trial function spaces $(\mathscr{S}_{\mathbf{u}}^h)_n$ for velocity and $(\mathscr{S}_p^h)_n$ for pressure, and the test function spaces $(\mathscr{V}_{\mathbf{u}}^h)_n$ and $(\mathscr{V}_p^h)_n = (\mathscr{S}_p^h)_n$ are defined by using, over Q_n, first-order polynomials in space and time.

DOI: 10.1002/fld

The DSD/SST formulation (from [54]) is written as follows: given $(\mathbf{u}^h)_n^-$, find $\mathbf{u}^h \in (\mathscr{S}_{\mathbf{u}}^h)_n$ and $p^h \in (\mathscr{S}_p^h)_n$ such that $\forall \mathbf{w}^h \in (\mathscr{V}_{\mathbf{u}}^h)_n$ and $\forall q^h \in (\mathscr{V}_p^h)_n$:

$$\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) \mathrm{d}Q + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \, \mathrm{d}Q$$
$$- \int_{(P_n)_h} \mathbf{w}^h \cdot \mathbf{h}^h \, \mathrm{d}P + \int_{Q_n} q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^-) \, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) + \tau_{\mathrm{PSPG}} \nabla q^h \right] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}Q$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q = 0 \tag{5}$$

where

$$Ł(q^h, \mathbf{w}^h) = \rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}^h \right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h) \tag{6}$$

This formulation is applied to all space–time slabs $Q_0, Q_1, Q_2, \ldots, Q_{N-1}$, starting with $(\mathbf{u}^h)_0^- = \mathbf{u}_0$. Here, τ_{SUPG}, τ_{PSPG} and ν_{LSIC} are the SUPG, PSPG and LSIC (least squares on incompressibility constraint) stabilization parameters. There are various ways of defining these stabilization parameters. Here, we provide the definitions given in [54]:

$$\tau_{\mathrm{SUPG}} = \left(\frac{1}{\tau_{\mathrm{SUGN12}}^2} + \frac{1}{\tau_{\mathrm{SUGN3}}^2} \right)^{-1/2} \tag{7}$$

$$\tau_{\mathrm{SUGN12}} = \left(\sum_{a=1}^{n_{\mathrm{en}}} \left| \frac{\partial N_a}{\partial t} + \mathbf{u}^h \cdot \nabla N_a \right| \right)^{-1} \tag{8}$$

$$\tau_{\mathrm{SUGN3}} = \frac{h_{\mathrm{RGN}}^2}{4\nu} \tag{9}$$

$$h_{\mathrm{RGN}} = 2 \left(\sum_{a=1}^{n_{\mathrm{en}}} |\mathbf{r} \cdot \nabla N_a| \right)^{-1} \tag{10}$$

$$\mathbf{r} = \frac{\nabla \|\mathbf{u}^h\|}{\|\nabla \|\mathbf{u}^h\|\|} \tag{11}$$

$$\tau_{\mathrm{PSPG}} = \tau_{\mathrm{SUPG}} \tag{12}$$

$$\nu_{\mathrm{LSIC}} = \tau_{\mathrm{SUPG}} \|\mathbf{u}^h\|^2 \tag{13}$$

where n_{en} is the number of (space–time) element nodes and N_a is the space–time shape function associated with the space–time node a. As an alternative to the construction of τ_{SUPG} as given by

DOI: 10.1002/fld

Equations (7)–(8), we propose the option of constructing τ_{SUPG} based on separate definitions for the advection-dominated and transient-dominated limits:

$$\tau_{\text{SUPG}} = \left(\frac{1}{\tau_{\text{SUGN1}}^2} + \frac{1}{\tau_{\text{SUGN2}}^2} + \frac{1}{\tau_{\text{SUGN3}}^2} \right)^{-1/2} \tag{14}$$

$$\tau_{\text{SUGN1}} = \left(\sum_{a=1}^{n_{\text{en}}} |(\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla N_a| \right)^{-1} \tag{15}$$

$$\tau_{\text{SUGN2}} = \frac{\Delta t}{2} \tag{16}$$

where $\mathbf{v}^h$ is the mesh velocity. We note that separating τ_{SUGN12} into its advection-dominated and transient-dominated components as given by Equations (15)–(16) is equivalent to excluding the $(\partial N_a/\partial t|_{\boldsymbol{\xi}})$ part of $(\partial N_a/\partial t)$ in Equation (8), making that the definition for τ_{SUGN1}, and accounting for the $(\partial N_a/\partial t|_{\boldsymbol{\xi}})$ part in the definition for τ_{SUGN2} given by Equation (16). Here, $\boldsymbol{\xi}$ is the vector of element (parent-domain) coordinates. We also propose the option of modifying the ν_{LSIC} definition given by Equation (13) to take the mesh velocity into account:

$$\nu_{\text{LSIC}} = \tau_{\text{SUPG}} \|\mathbf{u}^h - \mathbf{v}^h\|^2 \tag{17}$$

For more ways of calculating τ_{SUPG}, τ_{PSPG} and ν_{LSIC}, see [54, 69, 73, 74]. References [54, 69, 74] also include the discontinuity-capturing directional dissipation (DCDD) stabilization, which was introduced as an alternative to the LSIC stabilization.

It was remarked in [54, 69, 73–76] that in marching from time level n to $n+1$, there are advantages in calculating the τ's from the flow field at time level n. That is

$$\tau \leftarrow \tau_n \tag{18}$$

where τ is the stabilization parameter to be used in marching from time level n to $n+1$, and τ_n is the stabilization parameter calculated from the flow field at time level n. One of the main advantages in doing that, as it was pointed out in [54, 69, 74–76], is avoiding another level of nonlinearity coming from the way τ's are defined. In general, it is desirable to make τ's less dependent on short-term variations in the flow field. For this purpose, a recursive time-averaging approach was proposed in [69, 74] for determining the τ's to be used in marching from time level n to $n+1$:

$$\tau \leftarrow z_1\tau_n + z_2\tau_{n-1} + (1 - z_1 - z_2)\tau \tag{19}$$

where τ_n and τ_{n-1} are the stabilization parameters calculated from the flow field at time levels n and $n-1$, and the τ on the right-hand side is the stabilization parameter that was used in marching from time level $n-1$ to n. The magnitudes and the number of the 'averaging parameters' $z_1, z_2, \ldots$ can be adjusted to create the desired outcome in terms of giving more weight to recently calculated τs or making the averaging closer to being a trailing average.

In addition, for high-aspect-ratio elements near solid surfaces, we propose the option of setting $\mathbf{r} = \mathbf{n}$. This would spare h_{RGN} from undesirable fluctuations as $\|\mathbf{u}^h\|$ gets smaller and smaller for elements that become thinner and thinner. By setting $\mathbf{r} = \mathbf{n}$ we still get the desired outcome from the computation of h_{RGN}, but without hard wiring the computation for any particular element type or shape.

DOI: 10.1002/fld

Remark 1
Strictly speaking, the DSD/SST formulation given by Equation (5) was introduced in [54] and is slightly different from the formulation given in [48, 51]. The two formulations are equivalent if the stabilization parameters τ_{SUPG} and τ_{PSPG} are defined to be identical, $\nu_{\text{LSIC}} = 0$, and $\nabla \cdot (\mu\boldsymbol{\varepsilon}(\mathbf{w}^h))$ is zero (which will be the case for linear elements) or neglected.

Remark 2
As an alternative to the way the SUPG test function is defined in Equation (5), we propose the SUPG test function option of replacing $(\partial\mathbf{w}^h/\partial t + \mathbf{u}^h \cdot \nabla\mathbf{w}^h)$ with $((\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla\mathbf{w}^h)$. This replacement is equivalent to excluding the $(\partial\mathbf{w}^h/\partial t|_\xi)$ part of $(\partial\mathbf{w}^h/\partial t)$. We call this option 'WTSE', and the option where the $(\partial\mathbf{w}^h/\partial t|_\xi)$ term is active 'WTSA'.

Remark 3
With the function spaces defined in the paragraph preceding Equation (5), for each space–time slab velocity and pressure assume double unknown values at each spatial node. One value corresponds to the lower end of the slab, and the other one upper end. The option of using double unknown values at a spatial node will be called 'DV' for velocity and 'DP' for pressure. In this case, we use two integration points over the time interval of the space–time slab, and this time-integration option will be called 'TIP2'. This version of the DSD/SST formulation, with the options set DV, DP and TIP2, will be called 'DSD/SST-DP'.

Remark 4
We propose here the option of using, for each space–time slab, a single unknown pressure value at each spatial node, and we will call this option 'SP'. With this, we propose another version of the DSD/SST formulation, where the options set is DV, SP and TIP2, and we will call this version 'DSD/SST-SP'. Because the number of unknown pressure values is halved, the computational cost is reduced substantially.

Remark 5
To reduce the computational cost further, we propose the option of using only one integration point over the time interval of the space–time slab, and we call this time-integration option 'TIP1'. With this, we propose a third version of the DSD/SST formulation, where the options set is DV, SP and TIP1, and we will call this version 'DSD/SST-TIP1'.

Remark 6
As a third way of reducing the computational cost, we propose the option of using, for each space–time slab, a single unknown velocity value at each spatial node, and we will call this option 'SV'. In the SV option, of the two parts of Equation (5), the one generated by $(\mathbf{w}^h)_n^+$ is removed, and we explicitly set $(\mathbf{u}^h)_n^+ = (\mathbf{u}^h)_n^-$, which makes the velocity field continuous in time. Based on the SV option, we propose a fourth version of the DSD/SST formulation, where the options set is SV, SP and TIP1, and we will call this version 'DSD/SST-SV'. With this version of the DSD/SST formulation, we propose to use the SUPG test function option WTSE.

Remark 7
In terms of computational cost the DSD/SST-SV formulation would be quite comparable to the ALE formulations. This makes the DSD/SST-SV formulation very competitive in computational efficiency.

DOI: 10.1002/fld

Remark 8
Versions DSD/SST-TIP1 and DSD/SST-SV render the DSD/SST versions used earlier obsolete. For example, the version DSD/SST-DP has twice the number of unknown pressure values and twice the number of time-integration points compared to DSD/SST-TIP1 and DSD/SST-SV, and is no longer attractive in terms of computational efficiency. Therefore, we now consider DSD/SST-DP obsolete. As another example, the version with the option set DV, DP and TIP1 would reduce the number of time-integration points by half and bring some computational efficiency that way. But it still would have twice the number of unknown pressure values and generate twice the number of pressure equations, where half of those equations would be linearly dependent on the other half. As it can be seen from the numerous computations carried out in the past by the T★AFSM, because of the iterative nature of the solution process and the special circumstances, this linear dependency, by itself, does not create a convergence problem. Still, having twice the number of pressure equations is not computationally efficient and needlessly taking chances with linearly dependent equation systems is not prudent. Therefore, the version of the DSD/SST formulation with the option set DV, DP and TIP1 is now obsolete.

3.2. Semi-discrete formulation of structural mechanics

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 mechanics equations (see [33, 77, 78]) is written as

$$\int_{\Omega_0^s} \mathbf{w}^h \cdot \rho^s \frac{\mathrm{d}^2 \mathbf{y}^h}{\mathrm{d}t^2}\,\mathrm{d}\Omega^s + \int_{\Omega_0^s} \mathbf{w}^h \cdot \eta\rho^s \frac{\mathrm{d}\mathbf{y}^h}{\mathrm{d}t}\,\mathrm{d}\Omega^s + \int_{\Omega_0^s} \delta\mathbf{E}^h : \mathbf{S}^h\,\mathrm{d}\Omega^s$$
$$= \int_{\Omega_t^s} \mathbf{w}^h \cdot (\mathbf{t}^h + \rho^s \mathbf{f}^s)\,\mathrm{d}\Omega^s \tag{20}$$

The fluid mechanics forces acting on the structure are represented by vector $\mathbf{t}^h$. This force term is geometrically nonlinear and thus increases the overall nonlinearity of the formulation. The left-hand side terms of Equation (20) are referred to in the original configuration and the right-hand side terms in 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 an incremental form (see [31, 33, 77, 78]), which is expressed as

$$\left[\frac{\mathbf{M}}{\beta\Delta t^2} + \frac{(1-\alpha)\gamma\mathbf{C}}{\beta\Delta t} + (1-\alpha)\mathbf{K}\right]\Delta\mathbf{d}^i = \mathbf{R}^i \tag{21}$$

Here, $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the artificial-damping matrix, $\mathbf{K}$ is the consistent tangent matrix associated with the internal elastic forces, $\mathbf{R}^i$ is the residual vector at the ith iteration and $\Delta\mathbf{d}^i$ is the ith increment in the nodal displacements vector $\mathbf{d}$. The artificial-damping matrix $\mathbf{C}$ is used, as mentioned in Section 2.2, only in computations where time accuracy is not required, and for spatially constant η it can be written as $\mathbf{C} = \eta\mathbf{M}$. 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 [79] scheme, which is the time-integration technique used here.

DOI: 10.1002/fld

3.3. Stabilized space–time fluid–structure interaction (SSTFSI) method

We will describe the SSTFSI method based on the finite element formulations given by Equations (5) and (20), with a slight change of notation and with a clarification of how the fluid–structure interface conditions are handled. In this notation subscripts 1 and 2 will refer to fluid and structure, respectively. Furthermore, while subscript I will refer to the fluid–structure interface, subscript E will refer to 'elsewhere' in the fluid and structure domains or boundaries. Then the equations representing the SSTFSI method are written as follows:

$$\int_{Q_n} \mathbf{w}_{1\mathrm{E}}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) \mathrm{d}Q + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}_{1\mathrm{E}}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}Q$$
$$- \int_{(P_n)_h} \mathbf{w}_{1\mathrm{E}}^h \cdot \mathbf{h}_{1\mathrm{E}}^h \, \mathrm{d}P + \int_{Q_n} q_{1\mathrm{E}}^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q + \int_{\Omega_n} (\mathbf{w}_{1\mathrm{E}}^h)_n^+ \cdot \rho((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^-) \, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho \left(\frac{\partial \mathbf{w}_{1\mathrm{E}}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{w}_{1\mathrm{E}}^h \right) + \tau_{\mathrm{PSPG}} \nabla q_{1\mathrm{E}}^h \right] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}Q$$
$$+ \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}_{1\mathrm{E}}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q = 0 \tag{22}$$

$$\int_{Q_n} q_{1\mathrm{I}}^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}Q + \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \frac{1}{\rho} [\tau_{\mathrm{PSPG}} \nabla q_{1\mathrm{I}}^h] \cdot [Ł(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}Q = 0 \tag{23}$$

$$\int_{(\Gamma_{1\mathrm{I}})_{\mathrm{REF}}} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot ((\mathbf{u}_{1\mathrm{I}}^h)_{n+1}^- - \mathbf{u}_{2\mathrm{I}}^h) \, \mathrm{d}\Gamma = 0 \tag{24}$$

$$\int_{(P_n)_h} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot \mathbf{h}_{1\mathrm{I}}^h \, \mathrm{d}P = - \int_{(P_n)_h} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot p\mathbf{n} \, \mathrm{d}P + \int_{Q_n} 2\mu \boldsymbol{\varepsilon}((\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^-) : \boldsymbol{\varepsilon}(\mathbf{u}) \, \mathrm{d}Q$$
$$+ \int_{Q_n} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot \nabla \cdot (2\mu \boldsymbol{\varepsilon}(\mathbf{u})) \, \mathrm{d}Q \tag{25}$$

$$\int_{(\Omega_{2\mathrm{I}})_{\mathrm{REF}}} \mathbf{w}_{2\mathrm{I}}^h \cdot (\mathbf{h}_{2\mathrm{I}}^h + (\mathbf{h}_{1\mathrm{I}}^h)_{\mathrm{A}} + (\mathbf{h}_{1\mathrm{I}}^h)_{\mathrm{B}}) \, \mathrm{d}\Omega = 0 \tag{26}$$

$$\int_{(\Omega_2)_0} \mathbf{w}_2^h \cdot \rho_2 \frac{\mathrm{d}^2 \mathbf{y}^h}{\mathrm{d}t^2} \, \mathrm{d}\Omega + \int_{(\Omega_2)_0} \mathbf{w}_2^h \cdot \eta \rho_2 \frac{\mathrm{d}\mathbf{y}^h}{\mathrm{d}t} \, \mathrm{d}\Omega + \int_{(\Omega_2)_0} \delta \mathbf{E}^h : \mathbf{S}^h \, \mathrm{d}\Omega$$
$$= \int_{\Omega_2} \mathbf{w}_2^h \cdot \rho_2 \mathbf{f}_2^h \, \mathrm{d}\Omega + \int_{\Omega_{2\mathrm{E}}} \mathbf{w}_{2\mathrm{E}}^h \cdot \mathbf{h}_{2\mathrm{E}}^h \, \mathrm{d}\Omega + \int_{\Omega_{2\mathrm{I}}} \mathbf{w}_{2\mathrm{I}}^h \cdot \mathbf{h}_{2\mathrm{I}}^h \, \mathrm{d}\Omega \tag{27}$$

DOI: 10.1002/fld

Here, $(\Gamma_{2\mathrm{I}})_{\mathrm{REF}}$ and $(\Omega_{2\mathrm{I}})_{\mathrm{REF}}$ represent some reference configurations of $\Gamma_{2\mathrm{I}}$ and $\Omega_{2\mathrm{I}}$, respectively. In reconciling the slightly modified notation used here with the notation we used in Equations (5) and (20), 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 $\mathbf{h}_{2\mathrm{I}}^h = \mathbf{t}^h$, and $(\mathbf{h}_{1\mathrm{I}}^h)_{\mathrm{A}}$ and $(\mathbf{h}_{1\mathrm{I}}^h)_{\mathrm{B}}$ represent the values of $\mathbf{h}_{1\mathrm{I}}^h$ associated with the fluid surfaces above and below the membrane structure. The symbol $\mathbf{h}_{2\mathrm{E}}^h$ 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, $(\mathbf{u}_{1\mathrm{I}}^h)_{n+1}^-$, $\mathbf{h}_{1\mathrm{I}}^h$ and $\mathbf{h}_{2\mathrm{I}}^h$ (the fluid velocity, fluid stress and structural stress at the interface) are treated as separate unknowns, and Equations (24)–(26) can be seen as equations corresponding to these three 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. If the structure is represented by a 3D continuum model instead of a membrane model, the formulation above would still be applicable if the domain integrations over $\Omega_{2\mathrm{E}}$ and $\Omega_{2\mathrm{I}}$ in the last two terms of Equation (27) 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.

We note that, for constant viscosity, the term $\nabla\cdot(2\mu\boldsymbol{\varepsilon}(\mathbf{u}))$ in Equation (25) vanishes for tetrahedral elements and in most cases can be neglected for hexahedral elements. The same statement can be made also in the context of that term being a part of the expression $Ł(p^h, \mathbf{u}^h)$ appearing in Equations (22) and (23).

In computations where we account for the porosity of the membrane fabric, we replace Equation (24) with the following one:

$$\int_{\Gamma_{1\mathrm{I}}} (\mathbf{w}_{1\mathrm{I}}^h)_{n+1}^- \cdot ((\mathbf{u}_{1\mathrm{I}}^h)_{n+1}^- - \mathbf{u}_{2\mathrm{I}}^h + k_{\mathrm{PORO}}(\mathbf{n}\cdot\mathbf{h}_{1\mathrm{I}}^h)\mathbf{n})\,\mathrm{d}\Gamma = 0 \tag{28}$$

where k_{PORO} is the porosity coefficient. This coefficient is typically given in units of 'CFM'. When a fabric with a porosity coefficient of 1 CFM is subjected to a pressure differential of $\frac{1}{2}$ in of water, the amount of flow crossing is 1 $\mathrm{ft}^3/\mathrm{min}$ across a sample size of 1 ft^2, which translates to a normal velocity of 1 ft/min. In our current implementation, in Equation (28) we take into account only the pressure component of $\mathbf{h}_{1\mathrm{I}}^h$.

Remark 9

In FSI computations with membranes and shells, the pressure at the interface has split nodal values corresponding to the fluid surfaces above and below the membrane or shell structure. We propose to use such split nodal values for pressure also at the boundaries (i.e. edges) of a membrane structure submerged in the fluid. Our computations show that this provides additional numerical stability for the edges of the membrane.

Remark 10

The versions of the SSTFSI method corresponding to the DSD/SST-DP, DSD/SST-SP, DSD/SST-TIP1 and DSD/SST-SV formulations (see Remarks 3–6) will be called 'SSTFSI-DP', 'SSTFSI-SP', 'SSTFSI-TIP1' and 'SSTFSI-SV', respectively.

DOI: 10.1002/fld

Remark 11
In terms of computational cost the SSTFSI-SV formulation would be quite comparable to the ALE FSI formulations. This makes the SSTFSI-SV formulation very competitive in computational efficiency.

Remark 12
Versions SSTFSI-TIP1 and SSTFSI-SV render the SSTFSI versions used earlier obsolete. For example, the version SSTFSI-DP has twice the number of unknown pressure values and twice the number of time-integration points compared to SSTFSI-TIP1 and SSTFSI-SV, and is no longer attractive in terms of computational efficiency. Therefore, we now consider SSTFSI-DP obsolete. As another example, the SSTFSI version with the option set DV, DP and TIP1 would reduce the number of time-integration points by half and bring some computational efficiency that way. But it still would have twice the number of unknown pressure values and generate twice the number of pressure equations, where half of those equations would be linearly dependent on the other half. As it can be seen from the numerous computations carried out in the past by the T$\star$AFSM, because of the iterative nature of the solution process and the special circumstances, this linear dependency, by itself, does not create a convergence problem. Still, having twice the number of pressure equations is not computationally efficient and needlessly taking chances with linearly dependent equation systems is not prudent. Therefore, the version of the SSTFSI formulation with the option set DV, DP and TIP1 is now obsolete.

4. MESH UPDATE METHODS

The mesh update has 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. Provided that at the fluid–solid interfaces the normal velocities of the mesh and the fluid interface are matched, we can move the mesh anyway we find most suitable for the purpose of reducing the frequency of remeshing. In general remeshing requires calling an automatic, unstructured-mesh generator. Reducing the cost associated with that in 3D applications 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. For example, reducing the frequency of remeshing to every 10 time steps or less 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 the T$\star$AFSM computed in the past, the frequency of remeshing was far less than every 10 time steps. In our current parallel computations on PC clusters, we typically perform the remeshing on one of the computing nodes, which, with the memory sizes such nodes come with these days, is powerful enough to generate 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.

DOI: 10.1002/fld

4.1. Automatic mesh-moving technique

In the automatic mesh-moving technique introduced in [55, 56], the motion of the internal nodes is determined by solving the equations of elasticity. As the boundary condition, the motion of the nodes at the fluid–solid interfaces is specified to match the velocity of the fluid interface. Similar mesh-moving techniques were used earlier by other researchers (see, for example, [80]). In [55, 56] the mesh deformation is dealt with selectively based on the sizes of the elements. Mesh-moving techniques with comparable features were later introduced in [58]. In the technique introduced in [55, 56], selective treatment based on element sizes is attained by altering the way we account for the Jacobian of the transformation from the element domain to the physical domain. The objective is to stiffen the smaller elements, which are typically placed near solid surfaces, more than the larger ones. When this technique was first introduced in [55, 56], 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 [55, 56] was augmented in [59] 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. This approach, when the stiffening power is set to 1.0, would be identical to the one first introduced in [55]. Here, we propose also an option of the automatic mesh-moving technique where the elements are stiffened proportional to an invariant measure of the shear strain associated with the mesh deformation. For that invariant measure of the shear strain, we propose to use the second invariant of the strain deviator tensor.

4.2. Solid-extension mesh-moving technique (SEMMT)

In dealing with fluid–solid interfaces, sometimes we need to generate structured layers of elements around the solid objects to fully control the mesh resolution there and have more accurate representation of the boundary layers. In the mesh-moving technique introduced in [55, 56], such structured layers of elements move ‘glued’ to the 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 [56, 81]. Earlier examples of element layers undergoing rigid-body motion, in combination with deforming structured meshes, can be found in [48].

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, the automatic mesh-moving technique described above may need to be used. In such cases, the thin fluid elements near the solid surface becomes a challenge for the automatic mesh-moving technique. In the SEMMT [60, 61], it was proposed to treat those 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, higher rigidity is assigned to these thin elements compared to the other fluid elements. Two ways of accomplishing this were proposed in [60, 61]: solving the elasticity equations for the nodes connected to the thin elements separate from the elasticity equations for the other nodes, or together. If they are solved separately, for the thin elements, as boundary conditions at the interface with the other elements, traction-free conditions would be used. The separate solution option is referred to as ‘SEMMT-multiple domain (SEMMT-MD)’ and the unified solution option as ‘SEMMT-single domain (SEMMT-SD)’.

DOI: 10.1002/fld

In [82, 83], test computations were presented to demonstrate how the SEMMT functions as part of the T⋆AFSM mesh update method. Both SEMMT options described above were employed. The test computations included mesh deformation tests [82, 83] and a 2D FSI model problem [83].

4.3. Move-reconnect-renode mesh update method (MRRMUM)

The MRRMUM was proposed in [62]. In the MRRMUM (see [38, 62]), two remeshing options were defined, with each one proposed to be used when it is most effective to do so. In the 'reconnect' option, only the way the nodes are connected is changed and thus only the elements are replaced (fully or partially) with a new set of elements. The mesh generator developed in [84] provides the reconnect option. In the 'renode' option, the existing nodes are replaced (fully or partially) with a new set of nodes. This, of course, results in also replacing the existing elements with a new set of elements. Because the reconnect option is simpler and involves less projection errors, it is preferable to the renode option. In the MRRMUM, we move the mesh for as many time steps as we can, reconnect only as frequently as we need to, and renode only when doing so is the only remaining option.

In [62], for the prescribed rigid-body rotation of a parachute, the performances of the two remeshing options described above were compared. By examining the aerodynamical forces acting on the parachute in all three directions, performances of remeshing with the 'reconnect' and 'renode' options were evaluated. The evaluations showed that the force oscillations seen immediately after the remeshing are reduced substantially with the 'reconnect' option.

4.4. 'Pressure clipping'

'Pressure clipping' was introduced in [85] for the purpose reducing the pressure 'spikes' typically encountered after the mesh-to-mesh projection following a remeshing. After such a projection, the incompressibility constraint is slightly violated, but it is recovered at the next nonlinear iteration. However, at that nonlinear iteration, the pressure, as it performs its duty of 'enforcing' the constraint, changes more than it should. Therefore, in the time step following a remeshing, in calculating the fluid mechanics forces at the fluid–solid interface, we propose to use 'clipped' pressure values. The 'clipped' values are obtained by the least-squares projection from the pressure values prior to remeshing. We also propose to use those 'clipped' values as the initial guess for the nonlinear iterations of the subsequent time step. Using the 'pressure clipping' in conjunction with the MRRMUM should further improve the quality of the solution improved by using, whenever we can, the 'reconnect' option of remeshing.

4.5. FSI geometric smoothing technique (FSI-GST)

For computations where the geometric complexity of the structure at the interface would require a fluid mechanics mesh that is not affordable or not desirable or just not manageable in mesh moving, we propose the FSI geometric smoothing technique (FSI-GST). In this technique, the structural mesh and displacement rates at the interface are projected to the fluid mesh after a geometric smoothing. In the geometric smoothing, a value (mesh coordinate or displacement rate) at a given node is replaced by a weighted average of the values at that node and a limited set of nearby nodes. When projecting the stress values from the smoothened interface to the structure, currently we propose to just transfer those values to the corresponding nodes of the structure. In some computations, we may need not an isotropic geometric smoothing but a directional smoothing

DOI: 10.1002/fld

along some preferred direction. For such computations, we propose, as a version of the FSI-GST, the FSI directional geometric smoothing technique (FSI-DGST). We propose to, whenever we can, generate the interface mesh in such a fashion that the preferred smoothing directions can approximately be represented by the gridlines of the interface mesh. Then the weighted averaging for a node on such a gridline would involve a limited set of nearby nodes only along that gridline. The directional smoothing concept is similar to the directional 'upwind' concept of the SUPG formulation, where the residual-based numerical dissipation is active only along the streamline direction.

5. SOLUTION OF THE FULLY DISCRETIZED COUPLED EQUATIONS

Full discretization of the FSI formulation described in Section 3.3 leads to coupled, nonlinear equation systems that need to be solved at every time step. In a conceptual form that is partitioned with respect to the models represented, such nonlinear equation systems can be written as follows:

$$\mathbf{N}_1(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) = \mathbf{F}_1 \tag{29}$$

$$\mathbf{N}_2(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) = \mathbf{F}_2 \tag{30}$$

$$\mathbf{N}_3(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) = \mathbf{F}_3 \tag{31}$$

where $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$ are the vectors of nodal unknowns corresponding to generic unknown functions $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$, respectively. In the context of an FSI problem, the generic functions $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ represent the fluid, structure and mesh unknowns, respectively. For the space–time formulation of the fluid mechanics 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 [48, 51, 52, 54]). Solution of these equations with the Newton–Raphson method would necessitate at every Newton–Raphson step solution of the following linear equation system:

$$\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{A}_{13}\mathbf{x}_3 = \mathbf{b}_1 \tag{32}$$

$$\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{A}_{23}\mathbf{x}_3 = \mathbf{b}_2 \tag{33}$$

$$\mathbf{A}_{31}\mathbf{x}_1 + \mathbf{A}_{32}\mathbf{x}_2 + \mathbf{A}_{33}\mathbf{x}_3 = \mathbf{b}_3 \tag{34}$$

where $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ are the residuals of the nonlinear equations, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are the correction increments for $\mathbf{d}_1$, $\mathbf{d}_2$ and $\mathbf{d}_3$, and $\mathbf{A}_{\beta\gamma} = \partial \mathbf{N}_\beta / \partial \mathbf{d}_\gamma$.

Remark 13
In FSI computations with a fluctuating traction boundary condition at the outflow, to improve the convergence of the nonlinear iterations, we propose to calculate the initial guess for p_{n+1} with the expression $(p_{n+1})^0 = p_n + (\Delta p_{\mathrm{OUTF}})_n$. In this expression, $(\Delta p_{\mathrm{OUTF}})_n$ is a measure of the change in the outflow traction from time level n to $n+1$.

5.1. Block-iterative coupling

In the block-iterative coupling [35–37, 67–71], the fluid, structure and mesh systems are treated as separate blocks, and the nonlinear iterations are carried out one block at a time. In solving a block

DOI: 10.1002/fld

of equations for the block of unknowns it is associated with, we use the most current values of the other blocks of unknowns. Assuming a cyclic order of $1 \to 2 \to 3$, in an iteration step taking us from iterative solution i to $i+1$, the following three blocks of equations are solved:

$$\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_1^i) = \mathbf{F}_1 - \mathbf{N}_1(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i) \tag{35}$$

$$\left.\frac{\partial \mathbf{N}_2}{\partial \mathbf{d}_2}\right|_{(\mathbf{d}_1^{i+1}, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_2^i) = \mathbf{F}_2 - \mathbf{N}_2(\mathbf{d}_1^{i+1}, \mathbf{d}_2^i, \mathbf{d}_3^i) \tag{36}$$

$$\left.\frac{\partial \mathbf{N}_3}{\partial \mathbf{d}_3}\right|_{(\mathbf{d}_1^{i+1}, \mathbf{d}_2^{i+1}, \mathbf{d}_3^i)} (\Delta \mathbf{d}_3^i) = \mathbf{F}_3 - \mathbf{N}_3(\mathbf{d}_1^{i+1}, \mathbf{d}_2^{i+1}, \mathbf{d}_3^i) \tag{37}$$

Each of the three blocks of linear equations systems given by Equations (35)–(37) is also solved iteratively, using the GMRES search technique [86]. In the block-iterative implementation of the T⋆AFSM, at each time step the cycle starts with block number 2 (i.e. $2 \to 3 \to 1 \to 2 \to 3 \to 1 \ldots$).

In FSI computations where the structure is light, structural response becomes very sensitive to small changes in the fluid mechanics forces. In such cases, when the coupling between the three blocks of equations given by Equations (32)–(34) is handled with a block-iterative coupling technique rather than a direct coupling technique, convergence becomes difficult to achieve. In Sections 5.2 and 5.3 we describe 'more direct' techniques for handling the coupling. A shortcut approach was proposed in [67–69] (and was also described in [35–37, 70, 71]) for improving the convergence of the block-iterative coupling technique. In this approach, to reduce 'over-correcting' (i.e. 'over-incrementing') the structural displacements during the block iterations, the mass matrix contribution to $\mathbf{A}_{22}$ is increased. This is achieved without altering $\mathbf{b}_1$, $\mathbf{b}_2$ or $\mathbf{b}_3$ (i.e. without altering $\mathbf{F}_1 - \mathbf{N}_1(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$, $\mathbf{F}_2 - \mathbf{N}_2(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$ or $\mathbf{F}_3 - \mathbf{N}_3(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$), 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 [35–37], the fluid + structure and mesh systems are treated as two separate blocks, and the nonlinear iterations are carried out one block at a time. In solving a block of equations for the block of unknowns it is associated with, we use the most current values of the other block of unknowns. In an iteration step taking us from iterative solution i to $i+1$, the following two blocks of equations are solved:

$$\left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_1}\right|_{(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_1^i) + \left.\frac{\partial \mathbf{N}_1}{\partial \mathbf{d}_2}\right|_{(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_2^i) = \mathbf{F}_1 - \mathbf{N}_1(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i) \tag{38}$$

$$\left.\frac{\partial \mathbf{N}_2}{\partial \mathbf{d}_1}\right|_{(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_1^i) + \left.\frac{\partial \mathbf{N}_2}{\partial \mathbf{d}_2}\right|_{(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i)} (\Delta \mathbf{d}_2^i) = \mathbf{F}_2 - \mathbf{N}_2(\mathbf{d}_1^i, \mathbf{d}_2^i, \mathbf{d}_3^i) \tag{39}$$

$$\left.\frac{\partial \mathbf{N}_3}{\partial \mathbf{d}_3}\right|_{(\mathbf{d}_1^{i+1}, \mathbf{d}_2^{i+1}, \mathbf{d}_3^i)} (\Delta \mathbf{d}_3^i) = \mathbf{F}_3 - \mathbf{N}_3(\mathbf{d}_1^{i+1}, \mathbf{d}_2^{i+1}, \mathbf{d}_3^i) \tag{40}$$

Each of the two blocks of linear equations systems given by Equations (38)–(39) and Equation (40) is also solved iteratively, using the GMRES search technique.

DOI: 10.1002/fld

Remark 14
In the iterative solution of the combined fluid+structure (i.e. $1+2$) block with the GMRES search technique and a diagonal preconditioner, depending on the nature of the problem, one of these two parts might pose a greater convergence challenge than the other one. The convergence challenges might be created by an incompressibility constraint, having thin or shallow computational domains, or some other factors. The scaling provided by diagonal preconditioning is unlikely to remedy such disparities in the convergence challenges offered by the two parts, which are typically exhibited as disparities in the residual-decay rates for the two parts rather than disparities in the residual magnitudes. In some cases, the scaling provided by diagonal preconditioning might not even be able to properly account for the disparities in the residual magnitudes corresponding to the fluid and structure parts. Here, we propose 'selective scaling' to place, in GMRES iterations, greater emphasis on the part posing greater convergence challenge. With this additional scaling (beyond diagonal preconditioning), in constructing the Krylov vectors of the GMRES search technique, the relative weights given to the residual vectors associated with the fluid and structure parts are determined based on the relative convergence challenges posed by those two parts. We propose to determine those relative weights on a case-by-case basis as well as on a more automated basis, where the weights increase with decreasing residual-decay rates.

5.3. Direct coupling

In the direct coupling [35–37], the fluid + structure + mesh system is treated as a single block, and the linear equation system given by Equations (32)–(34) is solved iteratively:

$$\mathbf{P}_{11}\mathbf{z}_1 + \mathbf{P}_{12}\mathbf{z}_2 + \mathbf{P}_{13}\mathbf{z}_3 = \mathbf{b}_1 - (\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{A}_{13}\mathbf{x}_3) \tag{41}$$

$$\mathbf{P}_{21}\mathbf{z}_1 + \mathbf{P}_{22}\mathbf{z}_2 + \mathbf{P}_{23}\mathbf{z}_3 = \mathbf{b}_2 - (\mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{A}_{23}\mathbf{x}_3) \tag{42}$$

$$\mathbf{P}_{31}\mathbf{z}_1 + \mathbf{P}_{32}\mathbf{z}_2 + \mathbf{P}_{33}\mathbf{z}_3 = \mathbf{b}_3 - (\mathbf{A}_{31}\mathbf{x}_1 + \mathbf{A}_{32}\mathbf{x}_2 + \mathbf{A}_{33}\mathbf{x}_3) \tag{43}$$

where $\mathbf{P}_{\beta\gamma}$'s represent the blocks of the preconditioning matrix $\mathbf{P}$. The most computing-intensive part here is the evaluation of the matrix–vector products of the form $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ (for $\beta, \gamma = 1, 2, \ldots, N$ and no sum). In the FSI computations carried out by the T⋆AFSM, those evaluations are performed with the element-vector-based (EVB) computation techniques (see [57, 69, 71, 87, 88]), which do not require computation of any matrices, not even at the element level. The EVB computations can be carried out in two ways: numerical EVB (NEVB) and analytical EVB (AEVB) computations.

5.3.1. NEVB computations. In the NEVB computation technique, which is also called the matrix-free computation technique (see [87, 88]), a matrix–vector product of the form $\mathbf{Ax}$, which is the directional derivative of $\mathbf{N}$ in $\mathbf{x}$ direction, is evaluated by the expression:

$$\mathbf{Ax} = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \left[\frac{\mathbf{N}^e(\mathbf{d} + \epsilon\mathbf{x}) - \mathbf{N}^e(\mathbf{d})}{\epsilon} \right] \tag{44}$$

where $\mathbf{N}^e$ is the element-level vector representing the contribution of element e to $\mathbf{N}$, and ϵ is a small parameter used in the numerical calculation of the limit representing the directional derivative. This concept was extended in [57, 69, 71] to FSI computations, where we need to evaluate

DOI: 10.1002/fld

the matrix–vector products of the form $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$:

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \left[\frac{\mathbf{N}_\beta^e(\ldots, \mathbf{d}_\gamma + \epsilon_{\beta\gamma}\mathbf{x}_\gamma, \ldots) - \mathbf{N}_\beta^e(\ldots, \mathbf{d}_\gamma, \ldots)}{\epsilon_{\beta\gamma}} \right] \tag{45}$$

where $\mathbf{N}_\beta^e$ is the element-level vector representing the contribution of element e to $\mathbf{N}_\beta$, and $\epsilon_{\beta\gamma}$ is the limit-evaluation parameter selected for the unknown set γ in the equation set β. If we decide to use a single limit-evaluation parameter ϵ_β for all the unknown sets in the equation set β, then the computations can be carried out as

$$\sum_{\gamma=1}^{N} \mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \left[\frac{\mathbf{N}_\beta^e(\mathbf{d} + \epsilon_\beta\mathbf{x}) - \mathbf{N}_\beta^e(\mathbf{d})}{\epsilon_\beta} \right] \tag{46}$$

Remark 15
Using a single limit-evaluation parameter for all the unknown sets in the equation set β would be computationally more economical. On the other hand, using a different limit-evaluation parameter for each unknown set would give us the option of taking separately into account the dependence of $\mathbf{N}_\beta$ on each unknown $\mathbf{d}_\gamma$, including how $\partial\mathbf{N}_\beta/\partial\mathbf{d}_\gamma$ varies with $\mathbf{d}_\gamma$. This is an important consideration because of the multi-physics and multi-scale nature of FSI computations.

5.3.2. AEVB computations. The AEVB computation technique (see [57, 69, 71]) can be used for evaluating the matrix–vector products of the form $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ if deriving expressions for such matrix–vector products is not painful and we prefer not to deal with limit-evaluation parameters and numerical evaluation of directional derivatives.

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 expressions for the first-order terms in the expansion of $\mathbf{B}_\beta(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N)$ in $\mathbf{u}_\gamma$. Those first-order terms in $\Delta\mathbf{u}_\gamma$ will be represented by the finite element integral form $\mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \Delta\mathbf{u}_\gamma)$. For example, $\mathbf{G}_{11}(\mathbf{W}_1, \mathbf{u}_1, \ldots, \mathbf{u}_N, \Delta\mathbf{u}_1)$ will represent the first-order terms obtained by expanding the finite element formulation of the fluid mechanics equations (i.e. momentum equation and incompressibility constraint) in fluid mechanics unknowns (i.e. fluid velocity and pressure). We note that the integral form $\mathbf{G}_{\beta\gamma}$ will generate $\partial\mathbf{N}_\beta/\partial\mathbf{d}_\gamma$. Consequently, as it was pointed out in [57, 69, 71], the product $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ can be evaluated as follows:

$$\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma = \frac{\partial\mathbf{N}_\beta}{\partial\mathbf{d}_\gamma}\mathbf{x}_\gamma = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \mathbf{G}_{\beta\gamma}(\mathbf{W}_\beta, \mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{v}_\gamma) \tag{47}$$

where $\mathbf{v}_\gamma$ is a function interpolated from $\mathbf{x}_\gamma$ in the same way $\mathbf{u}_\gamma$ is interpolated from $\mathbf{d}_\gamma$.

In the mixed AEVB/NEVB computation technique [57, 69, 71], in evaluation of $\mathbf{A}_{\beta\gamma}\mathbf{x}_\gamma$ for each combination of β and γ, depending on the nature of what is involved in that particular evaluation, one can select between the AEVB and NEVB computation techniques. In the direct coupling [35–37, 71], the matrix–vector product $\mathbf{A}_{13}\mathbf{x}_3$ is computed with the NEVB technique:

$$\mathbf{A}_{13}\mathbf{x}_3 = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \left[\frac{\mathbf{N}_1^e(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 + \epsilon_{13}\mathbf{x}_3) - \mathbf{N}_1^e(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)}{\epsilon_{13}} \right] \tag{48}$$

DOI: 10.1002/fld

Remark 16
Remark 14, with the wording expanded to the combined fluid + structure + mesh (i.e. 1 + 2 + 3) system, becomes applicable to the direct coupling.

6. SEGREGATED EQUATION SOLVERS

To simplify the context for the fundamental concepts of this section, let us first consider only the Navier–Stokes equations of incompressible flows, without any structure. The stabilized formulation given by Equation (5) leads to a nonlinear equation system that needs to be solved at every time step. In a form that is partitioned (segregated) with respect to velocity and pressure, that nonlinear equation system can be written as follows:

$$\mathbf{N}_{\mathrm{U}}(\mathbf{d}_{\mathrm{U}}, \mathbf{d}_{\mathrm{P}}) = \mathbf{F}_{\mathrm{U}} \tag{49}$$

$$\mathbf{N}_{\mathrm{P}}(\mathbf{d}_{\mathrm{U}}, \mathbf{d}_{\mathrm{P}}) = \mathbf{F}_{\mathrm{P}} \tag{50}$$

where $\mathbf{d}_{\mathrm{U}}$ and $\mathbf{d}_{\mathrm{P}}$ are the vectors of nodal unknowns corresponding to velocity and pressure, respectively. Solution of this nonlinear equation system with the Newton–Raphson method would necessitate at every Newton–Raphson step solution of the following linear equation system:

$$\mathbf{A}_{\mathrm{UU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{UP}}\mathbf{x}_{\mathrm{P}} = \mathbf{b}_{\mathrm{U}} \tag{51}$$

$$\mathbf{A}_{\mathrm{PU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{PP}}\mathbf{x}_{\mathrm{P}} = \mathbf{b}_{\mathrm{P}} \tag{52}$$

where $\mathbf{b}_{\mathrm{U}}$ and $\mathbf{b}_{\mathrm{P}}$ are the residuals of the nonlinear equations, $\mathbf{x}_{\mathrm{U}}$ and $\mathbf{x}_{\mathrm{P}}$ are the correction increments for $\mathbf{d}_{\mathrm{U}}$ and $\mathbf{d}_{\mathrm{P}}$, and $\mathbf{A}_{\beta\gamma} = \partial\mathbf{N}_{\beta}/\partial\mathbf{d}_{\gamma}$, with β, $\gamma = \mathrm{U}$, P.

6.1. Segregated equation solver for nonlinear systems (SESNS)

The SESNS technique originates from the segregated solvers reported in [47–49, 89, 90]. In [47], a segregated solver was first used with the SUPG formulation based on elements with bilinear velocity and constant pressure. The overly dissipative nature of the one-step SUPG formulation with constant pressure motivated the introduction of the multi-step SUPG formulations reported in [89], where the segregated solution approach was extended to multi-step methods. In [90], segregated solvers were provided for the SUPG formulation based on elements with higher-order interpolations for velocity and pressure, in the context of both one-step and multi-step formulations. In [48, 49], a segregated solver was provided for the SUPG/PSPG formulation based on elements with equal-order interpolations for velocity and pressure, where, because of the PSPG stabilization, the submatrix $\mathbf{A}_{\mathrm{PP}}$ was no longer zero.

In SESNS, instead of solving the equation system given by Equations (51)–(52) in its given form, we solve its approximate version where $\mathbf{A}_{\mathrm{UU}}$ is approximated by a diagonal matrix $\mathbf{D}_{\mathrm{UU}}$:

$$\mathbf{D}_{\mathrm{UU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{UP}}\mathbf{x}_{\mathrm{P}} = \mathbf{b}_{\mathrm{U}} \tag{53}$$

$$\mathbf{A}_{\mathrm{PU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{PP}}\mathbf{x}_{\mathrm{P}} = \mathbf{b}_{\mathrm{P}} \tag{54}$$

where $\mathbf{D}_{\mathrm{UU}} = \mathrm{DIAG}(\mathbf{A}_{\mathrm{UU}})$. From Equations (53)–(54), we obtain the following sets of equations:

$$\mathbf{x}_{\mathrm{U}} + \mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{A}_{\mathrm{UP}}\mathbf{x}_{\mathrm{P}} = \mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{b}_{\mathrm{U}} \tag{55}$$

DOI: 10.1002/fld

$$(\mathbf{A}_{\mathrm{PU}}\mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{A}_{\mathrm{UP}} - \mathbf{A}_{\mathrm{PP}})\mathbf{x}_{\mathrm{P}} = \mathbf{A}_{\mathrm{PU}}\mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{b}_{\mathrm{U}} - \mathbf{b}_{\mathrm{P}} \tag{56}$$

Equation (56) can be solved iteratively with the GMRES method. After solving Equation (56) for $\mathbf{x}_{\mathrm{P}}$, we substitute that solution into Equation (55) and compute $\mathbf{x}_{\mathrm{U}}$. Doing this twice would be similar to using a predictor/multi-corrector algorithm with two passes. With two passes, we can get second-order accuracy in time, but, because of stability considerations, we still would have a limit on the time-step size.

6.2. Segregated equation solver for linear systems (SESLS)

In SESLS, we do not replace the equation system given by Equations (51)–(52) with its approximate version. We solve it with preconditioned iterations. By using the concepts we used in Equations (41)–(43) and a comparable notation, we write the iterative solution of Equations (51)–(52) as follows:

$$\mathbf{P}_{\mathrm{UU}}\mathbf{z}_{\mathrm{U}} + \mathbf{P}_{\mathrm{UP}}\mathbf{z}_{\mathrm{P}} = \mathbf{b}_{\mathrm{U}} - (\mathbf{A}_{\mathrm{UU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{UP}}\mathbf{x}_{\mathrm{P}}) \tag{57}$$

$$\mathbf{P}_{\mathrm{PU}}\mathbf{z}_{\mathrm{U}} + \mathbf{P}_{\mathrm{PP}}\mathbf{z}_{\mathrm{P}} = \mathbf{b}_{\mathrm{P}} - (\mathbf{A}_{\mathrm{PU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{PP}}\mathbf{x}_{\mathrm{P}}) \tag{58}$$

We define:

$$\mathbf{r}_{\mathrm{U}} = \mathbf{b}_{\mathrm{U}} - (\mathbf{A}_{\mathrm{UU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{UP}}\mathbf{x}_{\mathrm{P}}) \tag{59}$$

$$\mathbf{r}_{\mathrm{P}} = \mathbf{b}_{\mathrm{P}} - (\mathbf{A}_{\mathrm{PU}}\mathbf{x}_{\mathrm{U}} + \mathbf{A}_{\mathrm{PP}}\mathbf{x}_{\mathrm{P}}) \tag{60}$$

select the preconditioning matrix blocks as

$$\mathbf{P}_{\mathrm{UU}} = \mathbf{D}_{\mathrm{UU}}, \quad \mathbf{P}_{\mathrm{UP}} = \mathbf{A}_{\mathrm{UP}} \tag{61}$$

$$\mathbf{P}_{\mathrm{PU}} = \mathbf{A}_{\mathrm{PU}}, \quad \mathbf{P}_{\mathrm{PP}} = \mathbf{A}_{\mathrm{PP}} \tag{62}$$

and rewrite Equations (57)–(58) as follows:

$$\mathbf{D}_{\mathrm{UU}}\mathbf{z}_{\mathrm{U}} + \mathbf{A}_{\mathrm{UP}}\mathbf{z}_{\mathrm{P}} = \mathbf{r}_{\mathrm{U}} \tag{63}$$

$$\mathbf{A}_{\mathrm{PU}}\mathbf{z}_{\mathrm{U}} + \mathbf{A}_{\mathrm{PP}}\mathbf{z}_{\mathrm{P}} = \mathbf{r}_{\mathrm{P}} \tag{64}$$

Now, we do to Equations (63)–(64) exactly what we did to Equations (53)–(54) and obtain:

$$\mathbf{z}_{\mathrm{U}} + \mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{A}_{\mathrm{UP}}\mathbf{z}_{\mathrm{P}} = \mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{r}_{\mathrm{U}} \tag{65}$$

$$(\mathbf{A}_{\mathrm{PU}}\mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{A}_{\mathrm{UP}} - \mathbf{A}_{\mathrm{PP}})\mathbf{z}_{\mathrm{P}} = \mathbf{A}_{\mathrm{PU}}\mathbf{D}_{\mathrm{UU}}^{-1}\mathbf{r}_{\mathrm{U}} - \mathbf{r}_{\mathrm{P}} \tag{66}$$

Equation (66) can be solved iteratively with the GMRES method. We will call these iterations 'sub-level' (or 'lower') iterations. After solving Equation (66) for $\mathbf{z}_{\mathrm{P}}$, we substitute that solution into Equation (65) and compute $\mathbf{z}_{\mathrm{U}}$. This completes the solution of Equations (63)–(64), which is equivalent to applying the preconditioner defined by Equations (61)–(62) in the iterative solution of Equations (51)–(52).

The SESLS is simply an extension of the SESNS concept to the linear equation systems that need to be solved at every Newton–Raphson step. For more sophisticated iterations techniques with sub-levels, see [91, 92].

DOI: 10.1002/fld

6.3. Segregated equation solver for fluid–structure interactions (SESFSI)

The SESFSI is based on extending the SESLS concept to FSI computations. In describing this solver, for the identification of the equation and unknown blocks, we will use an expanded notation that will remain local to this subsection. The vectors $\mathbf{x}$, $\mathbf{b}$, $\mathbf{z}$ and $\mathbf{r}$ are defined as

$$\mathbf{x}=\begin{pmatrix}\mathbf{x}_{\mathrm{E}}\\ \mathbf{x}_{\mathrm{I}}\\ \mathbf{x}_{\mathrm{Y}}\\ \mathbf{x}_{\mathrm{S}}\\ \mathbf{x}_{\mathrm{F}}\\ \mathbf{x}_{\mathrm{H}}\\ \mathbf{x}_{\mathrm{P}}\end{pmatrix},\quad \mathbf{b}=\begin{pmatrix}\mathbf{b}_{\mathrm{E}}\\ \mathbf{b}_{\mathrm{I}}\\ \mathbf{b}_{\mathrm{Y}}\\ \mathbf{b}_{\mathrm{S}}\\ \mathbf{b}_{\mathrm{F}}\\ \mathbf{b}_{\mathrm{H}}\\ \mathbf{b}_{\mathrm{P}}\end{pmatrix},\quad \mathbf{z}=\begin{pmatrix}\mathbf{z}_{\mathrm{E}}\\ \mathbf{z}_{\mathrm{I}}\\ \mathbf{z}_{\mathrm{Y}}\\ \mathbf{z}_{\mathrm{S}}\\ \mathbf{z}_{\mathrm{F}}\\ \mathbf{z}_{\mathrm{H}}\\ \mathbf{z}_{\mathrm{P}}\end{pmatrix},\quad \mathbf{r}=\begin{pmatrix}\mathbf{r}_{\mathrm{E}}\\ \mathbf{r}_{\mathrm{I}}\\ \mathbf{r}_{\mathrm{Y}}\\ \mathbf{r}_{\mathrm{S}}\\ \mathbf{r}_{\mathrm{F}}\\ \mathbf{r}_{\mathrm{H}}\\ \mathbf{r}_{\mathrm{P}}\end{pmatrix} \tag{67}$$

where the index translation is given in Table I.

The matrix $\mathbf{A}$ is expressed as follows:

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{\mathrm{EE}} & \mathbf{A}_{\mathrm{EI}} & & & & & \mathbf{A}_{\mathrm{EP}}\\ & \mathbf{A}_{\mathrm{II}} & \mathbf{A}_{\mathrm{IY}} & & & & \mathbf{A}_{\mathrm{IP}}\\ & & \mathbf{A}_{\mathrm{YY}} & \mathbf{A}_{\mathrm{YS}} & \mathbf{A}_{\mathrm{YF}} & & \\ & & \mathbf{A}_{\mathrm{SY}} & \mathbf{A}_{\mathrm{SS}} & & & \\ & & & & \mathbf{A}_{\mathrm{FF}} & \mathbf{A}_{\mathrm{FH}} & \\ \mathbf{A}_{\mathrm{HE}} & \mathbf{A}_{\mathrm{HI}} & & & & \mathbf{A}_{\mathrm{HH}} & \mathbf{A}_{\mathrm{HP}}\\ \mathbf{A}_{\mathrm{PE}} & \mathbf{A}_{\mathrm{PI}} & & & & & \mathbf{A}_{\mathrm{PP}}\end{pmatrix} \tag{68}$$

We note that the matrix $\mathbf{A}_{\mathrm{IP}}$ is generated by the porosity term, when in Equation (28) we take into account only the pressure component of $\mathbf{h}_{\mathrm{II}}^{h}$. If we take into account $\mathbf{h}_{\mathrm{II}}^{h}$ fully, then the coupling matrix generated would be $\mathbf{A}_{\mathrm{IH}}$ instead of $\mathbf{A}_{\mathrm{IP}}$.

Table I. SESFSI index translation.

E	All other fluid velocities
I	Fluid velocity $(\mathbf{u}_{\mathrm{II}}^{h})_{n+1}^{-}$ at the interface
Y	Structural displacements at the interface
S	All other structural displacements
F	Interface stresses acting on the structure
H	Interface stresses acting on the fluid
P	Fluid pressure

DOI: 10.1002/fld

Now, similar to what we did by means of Equations (61)–(62), we define various preconditioner options for the SESFSI. The first one is a very simple preconditioner, and we will call that $\mathbf{P}_{\mathrm{SIMP}}$:

$$\mathbf{P}_{\mathrm{SIMP}} = \begin{pmatrix} \mathbf{D}_{\mathrm{EE}} & \mathbf{0} & & & & & \mathbf{A}_{\mathrm{EP}} \\ & \mathbf{L}_{\mathrm{II}} & \mathbf{0} & & & & \mathbf{0} \\ & & \mathbf{D}_{\mathrm{YY}} & \mathbf{0} & \mathbf{0} & & \\ & & \mathbf{0} & \mathbf{D}_{\mathrm{SS}} & & & \\ & & & & \mathbf{L}_{\mathrm{FF}} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & & & & \mathbf{L}_{\mathrm{HH}} & \mathbf{0} \\ \mathbf{A}_{\mathrm{PE}} & \mathbf{A}_{\mathrm{PI}} & & & & & \mathbf{A}_{\mathrm{PP}} \end{pmatrix} \tag{69}$$

where $\mathbf{D}_{\mathrm{EE}} = \mathrm{DIAG}(\mathbf{A}_{\mathrm{EE}})$, $\mathbf{L}_{\mathrm{II}} = \mathrm{LUMP}(\mathbf{A}_{\mathrm{II}})$, $\mathbf{D}_{\mathrm{YY}} = \mathrm{DIAG}(\mathbf{A}_{\mathrm{YY}})$, $\mathbf{D}_{\mathrm{SS}} = \mathrm{DIAG}(\mathbf{A}_{\mathrm{SS}})$, $\mathbf{L}_{\mathrm{FF}} = \mathrm{LUMP}(\mathbf{A}_{\mathrm{FF}})$, $\mathbf{L}_{\mathrm{HH}} = \mathrm{LUMP}(\mathbf{A}_{\mathrm{HH}})$, and the operator 'LUMP' represents the matrix-lumping operation. We note that the replacement of $\mathbf{A}_{\mathrm{IP}}$ by $\mathbf{0}$ constitutes an approximation only if the structure has porosity.

The second preconditioner is an augmented version, in the sense that, compared to $\mathbf{P}_{\mathrm{SIMP}}$, it includes more of the coupling matrices, and we will call that $\mathbf{P}_{\mathrm{AUGM}}$:

$$\mathbf{P}_{\mathrm{AUGM}} = \begin{pmatrix} \mathbf{D}_{\mathrm{EE}} & \mathbf{0} & & & & & \mathbf{A}_{\mathrm{EP}} \\ & \mathbf{L}_{\mathrm{II}} & \mathbf{0} & & & & \mathbf{A}_{\mathrm{IP}} \\ & & \mathbf{D}_{\mathrm{YY}} & \mathbf{0} & \mathbf{A}_{\mathrm{YF}} & & \\ & & \mathbf{0} & \mathbf{D}_{\mathrm{SS}} & & & \\ & & & & \mathbf{L}_{\mathrm{FF}} & \mathbf{0} & \\ \mathbf{A}_{\mathrm{HE}} & \mathbf{A}_{\mathrm{HI}} & & & & \mathbf{L}_{\mathrm{HH}} & \mathbf{0} \\ \mathbf{A}_{\mathrm{PE}} & \mathbf{A}_{\mathrm{PI}} & & & & & \mathbf{A}_{\mathrm{PP}} \end{pmatrix} \tag{70}$$

The third preconditioner, compared to $\mathbf{P}_{\mathrm{SIMP}}$, includes the coupling matrices that lead to a more direct treatment of the projections, and we will call that $\mathbf{P}_{\mathrm{DPRO}}$:

$$\mathbf{P}_{\mathrm{DPRO}} = \begin{pmatrix} \mathbf{D}_{\mathrm{EE}} & \mathbf{0} & & & & & \mathbf{A}_{\mathrm{EP}} \\ & \mathbf{L}_{\mathrm{II}} & \mathbf{B}_{\mathrm{IY}} & & & & \mathbf{0} \\ & & \mathbf{D}_{\mathrm{YY}} & \mathbf{0} & \mathbf{0} & & \\ & & \mathbf{0} & \mathbf{D}_{\mathrm{SS}} & & & \\ & & & & \mathbf{L}_{\mathrm{FF}} & \mathbf{B}_{\mathrm{FH}} & \\ \mathbf{0} & \mathbf{0} & & & & \mathbf{L}_{\mathrm{HH}} & \mathbf{B}_{\mathrm{HP}} \\ \mathbf{A}_{\mathrm{PE}} & \mathbf{A}_{\mathrm{PI}} & & & & & \mathbf{A}_{\mathrm{PP}} \end{pmatrix} \tag{71}$$

DOI: 10.1002/fld

To define $\mathbf{B}_{\mathrm{IY}}$ and $\mathbf{B}_{\mathrm{FH}}$, we first consider the number of nodal points in blocks E, I, Y, S, F, H and P, and denote them by $(n_n)_{\mathrm{E}}$, $(n_n)_{\mathrm{I}}$, $(n_n)_{\mathrm{Y}}$, $(n_n)_{\mathrm{S}}$, $(n_n)_{\mathrm{F}}$, $(n_n)_{\mathrm{H}}$ and $(n_n)_{\mathrm{P}}$, respectively.

If $(n_n)_{\mathrm{I}}<(n_n)_{\mathrm{Y}}$, for each node in Block I, we select the nearest node in Block Y, with the condition that a node in Block Y can be selected only once as the nearest node. We use the index YI to denote the part of Block Y associated with that set of nodes. We use the index YO to denote the part of Block Y associated with the other (i.e. remaining) nodes in Block Y. Based on this, we partition $\mathbf{D}_{\mathrm{YY}}$ into $\mathbf{D}_{\mathrm{YIYI}}$ and $\mathbf{D}_{\mathrm{YOYO}}$. With that, we define $\mathbf{B}_{\mathrm{IY}}$ as follows:

$$\begin{pmatrix} \mathbf{L}_{\mathrm{II}} & \mathbf{B}_{\mathrm{IY}} \\ & \mathbf{D}_{\mathrm{YY}} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{\mathrm{II}} & -\mathbf{L}_{\mathrm{II}} & \mathbf{0} \\ & \mathbf{D}_{\mathrm{YIYI}} & \\ & & \mathbf{D}_{\mathrm{YOYO}} \end{pmatrix} \quad \text{for } (n_n)_{\mathrm{I}}<(n_n)_{\mathrm{Y}} \tag{72}$$

If $(n_n)_{\mathrm{I}}>(n_n)_{\mathrm{Y}}$, for each node in Block Y, we select the nearest node in Block I. Index IY denotes the part of Block I associated with that set of nodes, and index IO denotes the part of Block I associated with the other nodes in Block I. Based on this, we partition $\mathbf{L}_{\mathrm{II}}$ into $\mathbf{L}_{\mathrm{IYIY}}$ and $\mathbf{L}_{\mathrm{IOIO}}$. With that, we define $\mathbf{B}_{\mathrm{IY}}$ as follows:

$$\begin{pmatrix} \mathbf{L}_{\mathrm{II}} & \mathbf{B}_{\mathrm{IY}} \\ & \mathbf{D}_{\mathrm{YY}} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{\mathrm{IOIO}} & & \mathbf{0} \\ & \mathbf{L}_{\mathrm{IYIY}} & -\mathbf{L}_{\mathrm{IYIY}} \\ & & \mathbf{D}_{\mathrm{YY}} \end{pmatrix} \quad \text{for } (n_n)_{\mathrm{I}} > (n_n)_{\mathrm{Y}} \tag{73}$$

Using the nearest-node concept, if $(n_n)_{\mathrm{F}}<(n_n)_{\mathrm{H}}$, we partition Block H into blocks HF and HO, partition $\mathbf{L}_{\mathrm{HH}}$ into $\mathbf{L}_{\mathrm{HFHF}}$ and $\mathbf{L}_{\mathrm{HOHO}}$, and define $\mathbf{B}_{\mathrm{FH}}$ as follows:

$$\begin{pmatrix} \mathbf{L}_{\mathrm{FF}} & \mathbf{B}_{\mathrm{FH}} \\ & \mathbf{L}_{\mathrm{HH}} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{\mathrm{FF}} & -\mathbf{L}_{\mathrm{FF}} & \mathbf{0} \\ & \mathbf{L}_{\mathrm{HFHF}} & \\ & & \mathbf{L}_{\mathrm{HOHO}} \end{pmatrix} \quad \text{for } (n_n)_{\mathrm{F}}<(n_n)_{\mathrm{H}} \tag{74}$$

If $(n_n)_{\mathrm{F}}>(n_n)_{\mathrm{H}}$, we partition Block F into blocks FH and FO, partition $\mathbf{L}_{\mathrm{FF}}$ into $\mathbf{L}_{\mathrm{FHFH}}$ and $\mathbf{L}_{\mathrm{FOFO}}$, and define $\mathbf{B}_{\mathrm{FH}}$ as follows:

$$\begin{pmatrix} \mathbf{L}_{\mathrm{FF}} & \mathbf{B}_{\mathrm{FH}} \\ & \mathbf{L}_{\mathrm{HH}} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{\mathrm{FOFO}} & & \mathbf{0} \\ & \mathbf{L}_{\mathrm{FHFH}} & -\mathbf{L}_{\mathrm{FHFH}} \\ & & \mathbf{L}_{\mathrm{HH}} \end{pmatrix} \quad \text{for } (n_n)_{\mathrm{F}}>(n_n)_{\mathrm{H}} \tag{75}$$

To define $\mathbf{B}_{\mathrm{HP}}$, we first partition Block H into blocks H1–H3 (corresponding to the three spatial directions) and partition $\mathbf{L}_{\mathrm{HH}}$ into $\mathbf{L}_{\mathrm{H1H1}}$, $\mathbf{L}_{\mathrm{H2H2}}$ and $\mathbf{L}_{\mathrm{H3H3}}$. We note that $\mathbf{L}_{\mathrm{H2H2}}=\mathbf{L}_{\mathrm{H1H1}}$ and $\mathbf{L}_{\mathrm{H3H3}}=\mathbf{L}_{\mathrm{H1H1}}$. We also partition Block P into blocks PH and PD (corresponding to the nodal values of the pressure at the interface and elsewhere in the fluid domain) and partition $\mathbf{A}_{\mathrm{PP}}$ into $\mathbf{A}_{\mathrm{PHPH}}$, $\mathbf{A}_{\mathrm{PHPD}}$, $\mathbf{A}_{\mathrm{PDPH}}$ and $\mathbf{A}_{\mathrm{PDPD}}$. In addition, we define three diagonal matrices $\mathbf{D}^1$, $\mathbf{D}^2$ and $\mathbf{D}^3$, representing the three spatial components of the unit normal vectors at the interface nodes. These diagonal matrices are defined as

$$\mathbf{D}^j = [(\mathbf{n}_A)^j \delta_{AB}] \quad \text{(no sum)}, \quad j = 1, 2, 3 \tag{76}$$

DOI: 10.1002/fld

where δ_{AB} are the components of the identity tensor, and $(\mathbf{n}_A)^j$ is the j component of the unit normal vector at the interface node A. With that, we define $\mathbf{B}_{HP}$ as follows:

$$\begin{pmatrix} \mathbf{L}_{HH} & \mathbf{B}_{HP} \\ & \mathbf{A}_{PP} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{H1H1} & & & -\mathbf{L}_{H1H1}\mathbf{D}^1 & \\ & \mathbf{L}_{H2H2} & & -\mathbf{L}_{H2H2}\mathbf{D}^2 & \\ & & \mathbf{L}_{H3H3} & -\mathbf{L}_{H3H3}\mathbf{D}^3 & \\ & & & \mathbf{A}_{PHPH} & \mathbf{A}_{PHPD} \\ & & & \mathbf{A}_{PDPH} & \mathbf{A}_{PDPD} \end{pmatrix} \tag{77}$$

7. SURFACE-EDGE-NODE CONTACT TRACKING (SENCT)

The SENCT technique was introduced in [38] as a contact algorithm. In this technique, which is in early stages of its development and testing, the objective is to prevent the structural surfaces from coming closer than a predetermined minimum distance we would like to maintain to protect the quality of the fluid mechanics mesh between the structural surfaces.

The contact detection is based on searching and calculating for each node the projection distance from that node to all the structural surface elements ('surfaces'), edges and nodes. During this search, we check to see if that projection distance is smaller than the minimum distance we would like to maintain between the structural surfaces. If it is, then we declare that surface or edge or node to be a contact surface or edge or node for the node we are conducting the search for. We note that for each node we are conducting a search for, the predetermined minimum distance between the structural surfaces is calculated locally (i.e. based on the local length scales related to that node).

The search algorithm involves some exclusion criteria. For example, edges belonging to contact surfaces are excluded, and the nodes belonging to contact surfaces or edges are excluded. The search algorithm also involves some more obvious exclusion criteria. For example, we exclude the surfaces containing the node we are conducting the search for, and exclude the edges and nodes that belong to the surface that also contains the node we are conducting the search for. The surfaces, edges and nodes beyond a certain distance from the node we are conducting the search for are excluded from the projection distance calculations, because they are not expected to be candidates for a contact surface or edge or node. The contact detection search is conducted at every n_{tsbcd} time steps, a parameter specified by the user. Once all the contact surfaces, edges and nodes are identified for the 'contacted node' (i.e. the node we were conducting the search for), the nodes belonging to those surfaces, edges and nodes are declared to be the 'contact-node set' for the contacted node.

We proposed two variations of the SENCT technique in [38]. In the SENCT-force (SENCT-F) technique, the contacted node is subjected to penalty forces that are inversely proportional to the projection distances to the contacting surfaces, edges and nodes. In the SENCT-displacement (SENCT-D) technique, the displacement of the contacted node is adjusted to correlate with the motion of the contacting surfaces, edges and nodes. There are various ways of accomplishing that. For example, at every time step, the contacted node can be allowed to move for a certain number of nonlinear iterations without any contact restrictions, followed by some more nonlinear iterations

DOI: 10.1002/fld

where the motion of the contacted node is set to the mean displacement of the contact-node set. If the displacement of the contacted node at the end of the first set of nonlinear iterations shows that it no longer qualifies as a contacted node, then during the next set of nonlinear iterations the node is allowed to move without any contact restrictions.

8. ALE FORMULATION WITH SUPG AND PSPG STABILIZATIONS

As it was pointed out in Remark 11, in terms of computational cost the SSTFSI-SV formulation would be quite comparable to the ALE FSI formulations. For completeness, we provide here a stabilized ALE formulation of the Navier–Stokes equations of incompressible flows, where the stabilization is based on the SUPG and PSPG methods. In ALE FSI computations, the formulation we provide here would replace the fluid mechanics parts of the SSTFSI formulation given in Section 3.3. Many of the equations in that section would, in substance, remain the same, with proper modifications reflecting the change from a space–time context to a semi-discrete context.

The ALE formulation with the SUPG and PSPG stabilizations can be written as follows:

$$\begin{aligned}
&\int_{\Omega_t} \mathbf{w}^h \cdot \rho \left(\left.\frac{\partial \mathbf{u}^h}{\partial t}\right|_{\boldsymbol{\xi}} + (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla \mathbf{u}^h - \mathbf{f}^h \right) \mathrm{d}\Omega + \int_{\Omega_t} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}\Omega \\
&\quad - \int_{(\Gamma_t)_h} \mathbf{w}^h \cdot \mathbf{h}^h \, \mathrm{d}\Gamma + \int_{\Omega_t} q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \frac{1}{\rho} \left[\tau_{\mathrm{SUPG}} \rho (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla \mathbf{w}^h + \tau_{\mathrm{PSPG}} \nabla q^h \right] \cdot [\text{Ł}(p^h, \mathbf{u}^h) - \rho \mathbf{f}^h] \, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \, \mathrm{d}\Omega = 0
\end{aligned} \tag{78}$$

where

$$\text{Ł}(q^h, \mathbf{w}^h) = \rho \left(\left.\frac{\partial \mathbf{w}^h}{\partial t}\right|_{\boldsymbol{\xi}} + (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla \mathbf{w}^h \right) - \nabla \cdot \boldsymbol{\sigma}(q^h, \mathbf{w}^h) \tag{79}$$

and the stabilization parameters are given by Equations (14)–(16), (9), (12) and (17).

9. THERMAL COUPLING

In computations with thermal coupling, the momentum equation given by Equation (1) would be replaced with the equation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - (1 - \beta_{\mathrm{T}}(T - T_{\mathrm{ref}}))\mathbf{a}_{\mathrm{GRAV}} \right) - \nabla \cdot \boldsymbol{\sigma} = 0 \tag{80}$$

DOI: 10.1002/fld

where the temperature T is governed by the following equation:

$$\rho C_p \left(\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) - \nabla \cdot (\kappa \nabla T) = 0 \tag{81}$$

Here, β_{T} is the coefficient of thermal expansion, T_{ref} is a reference temperature, $\mathbf{a}_{\mathrm{GRAV}}$ is the gravitational acceleration, C_p is the constant-pressure specific heat and κ is the thermal conductivity. For ideal gases, $\beta_{\mathrm{T}} = 1/T$, with T in the expression representing the absolute temperature. This expression is not valid for water, and the β_{T} values need to be extracted from tabulated data for water. In computations, we propose to use a polynomial representation of that data, expressed as a function of temperature, for the expected temperature range $T_1 \leqslant T \leqslant T_2$. A simple way would be to use a quadratic polynomial

$$\beta_{\mathrm{T}}(T) = (\beta_{\mathrm{T}})_1 + b_1(T - T_1) + b_2(T - T_1)^2 \tag{82}$$

where $(\beta_{\mathrm{T}})_1 = \beta_{\mathrm{T}}(T_1)$, and the coefficients b_1 and b_2 are determined by a least-squares fit to the data tabulated for the range $T_1 \leqslant T \leqslant T_2$.

We write the stabilized formulations in the context of the ALE method. The stabilized formulation corresponding to Equation (80) can be written as

$$\begin{aligned}
&\int_{\Omega_t} \mathbf{w}^h \cdot \rho \left(\left. \frac{\partial \mathbf{u}^h}{\partial t} \right|_{\boldsymbol{\xi}} + (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla \mathbf{u}^h - (1 - \beta_{\mathrm{T}}(T^h - T_{\mathrm{ref}}))\mathbf{a}_{\mathrm{GRAV}} \right) \mathrm{d}\Omega \\
&+ \int_{\Omega_t} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h)\, \mathrm{d}\Omega - \int_{(\Gamma_t)_h} \mathbf{w}^h \cdot \mathbf{h}^h\, \mathrm{d}\Gamma + \int_{\Omega_t} q^h \nabla \cdot \mathbf{u}^h\, \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \frac{1}{\rho} [\tau_{\mathrm{SUPG}} \rho (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla \mathbf{w}^h + \tau_{\mathrm{PSPG}} \nabla q^h] \\
&\cdot [Ł(p^h, \mathbf{u}^h) - \rho(1 - \beta_{\mathrm{T}}(T^h - T_{\mathrm{ref}}))\mathbf{a}_{\mathrm{GRAV}}]\, \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \nu_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h\, \mathrm{d}\Omega + S_{\mathrm{DC}} = 0
\end{aligned} \tag{83}$$

where the stabilization parameters are again given by Equations (14)–(16), (9), (12) and (17). The symbol S_{DC} represents the discontinuity-capturing term, which is helpful in computations with thermal coupling, especially for the equation governing the temperature. For the Navier–Stokes equations the discontinuity-capturing term is given as

$$S_{\mathrm{DC}} = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \rho \nabla \mathbf{w}^h : (\boldsymbol{\nu}_{\mathrm{DC}} \cdot \nabla \mathbf{u}^h)\, \mathrm{d}\Omega \tag{84}$$

where $\boldsymbol{\nu}_{\mathrm{DC}}$ is the discontinuity-capturing parameter. For examples of ways of calculating this parameter, including the DCDD parameter, see [54, 69, 74, 75]. We note that the DCDD stabilization was originally introduced as an alternative to the LSIC stabilization, and therefore normally only one of these stabilizations should be retained.

DOI: 10.1002/fld

The stabilized formulation corresponding to Equation (81) can be written as

$$\begin{aligned}
&\int_{\Omega_t} w^h \rho C_p \left(\left.\frac{\partial T^h}{\partial t}\right|_{\xi} + \left(\mathbf{u}^h - \mathbf{v}^h\right) \cdot \nabla T^h \right) \mathrm{d}\Omega \\
&+ \int_{\Omega_t} \nabla w^h \cdot \kappa \nabla T^h \, \mathrm{d}\Omega - \int_{(\Gamma_t)_h} w^h \mathrm{h}^h \, \mathrm{d}\Gamma \\
&+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} (\tau_{\mathrm{SUPG}})_{\mathrm{T}} (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla w^h \\
&\times \left(\rho C_p \left(\left.\frac{\partial T^h}{\partial t}\right|_{\xi} + \left(\mathbf{u}^h - \mathbf{v}^h\right) \cdot \nabla T^h \right) - \nabla \cdot \left(\kappa \nabla T^h \right) \right) \mathrm{d}\Omega \\
&+ (S_{\mathrm{DC}})_{\mathrm{T}} = 0
\end{aligned} \tag{85}$$

where

$$(S_{\mathrm{DC}})_{\mathrm{T}} = \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega_t^e} \nabla w^h \cdot \boldsymbol{\kappa}_{\mathrm{DC}} \nabla T^h \, \mathrm{d}\Omega \tag{86}$$

The stabilization parameters are defined as follows:

$$(\tau_{\mathrm{SUPG}})_{\mathrm{T}} = \left(\frac{1}{((\tau_{\mathrm{SUGN1}})_{\mathrm{T}})^2} + \frac{1}{((\tau_{\mathrm{SUGN2}})_{\mathrm{T}})^2} + \frac{1}{((\tau_{\mathrm{SUGN3}})_{\mathrm{T}})^2} \right)^{-1/2} \tag{87}$$

$$(\tau_{\mathrm{SUGN1}})_{\mathrm{T}} = \tau_{\mathrm{SUGN1}} \tag{88}$$

$$(\tau_{\mathrm{SUGN2}})_{\mathrm{T}} = \tau_{\mathrm{SUGN2}} \tag{89}$$

$$(\tau_{\mathrm{SUGN3}})_{\mathrm{T}} = \frac{((h_{\mathrm{RGN}})_{\mathrm{T}})^2}{4(\kappa/(\rho C_p))} \tag{90}$$

$$(h_{\mathrm{RGN}})_{\mathrm{T}} = 2 \left(\sum_{a=1}^{n_{\mathrm{en}}} |\mathbf{r}_{\mathrm{T}} \cdot \nabla N_a| \right)^{-1} \tag{91}$$

$$\mathbf{r}_{\mathrm{T}} = \frac{\nabla T^h}{\|\nabla T^h\|} \tag{92}$$

For early examples of ways of calculating $\boldsymbol{\kappa}_{\mathrm{DC}}$, see [93, 94]. Newer examples of ways of calculating $\boldsymbol{\kappa}_{\mathrm{DC}}$ are those based on the DCDD stabilization [54, 69, 74, 75] and '$YZ\beta$ shock-capturing' [68, 69, 74, 95] techniques (the 'β' in $YZ\beta$ shock-capturing is not related to the β_{T} representing the coefficient of thermal expansion).

For completeness, we describe here how the $YZ\beta$ shock capturing and DCDD stabilization techniques can be applied to the calculation of $\boldsymbol{\kappa}_{\mathrm{DC}}$. We can define $\boldsymbol{\kappa}_{\mathrm{DC}}$ as $\boldsymbol{\kappa}_{\mathrm{DC}} = \rho C_p (v_{\mathrm{DC}})_{\mathrm{T}} \mathbf{I}$ or as $\boldsymbol{\kappa}_{\mathrm{DC}} = \rho C_p (v_{\mathrm{DC}})_{\mathrm{T}} \mathbf{r}_{\mathrm{T}} \mathbf{r}_{\mathrm{T}}$ or in some more complex way, such as by using '*switch*' functions, as described in [68, 69, 74].

DOI: 10.1002/fld

Based on $YZ\beta$ shock capturing (with $\beta = 2$), $(\nu_{\mathrm{DC}})_{\mathrm{T}}$ can be calculated by using the expression

$$(\nu_{\mathrm{DC}})_{\mathrm{T}} = (\nu_{\mathrm{YZ}\beta})_{\mathrm{T}} = \left| Y^{-1} \left(\left. \frac{\partial T^h}{\partial t} \right|_{\xi} + (\mathbf{u}^h - \mathbf{v}^h) \cdot \nabla T^h \right) \right| \left(\frac{(h_{\mathrm{RGN}})_{\mathrm{T}}}{2} \right)^2 \tag{93}$$

where Y is a scaling value for T, which can be selected as $Y = T_{\max} - T_{\min}$.

Based on DCDD stabilization, $(\nu_{\mathrm{DC}})_{\mathrm{T}}$ can be calculated by using the expression

$$(\nu_{\mathrm{DC}})_{\mathrm{T}} = (\nu_{\mathrm{DCDD}})_{\mathrm{T}} = \|\mathbf{u}^h - \mathbf{v}^h\| \left(\frac{(h_{\mathrm{RGN}})_{\mathrm{T}}}{2} \right) \frac{\|\nabla T^h\| (h_{\mathrm{RGN}})_{\mathrm{T}}}{(\Delta T)_{\mathrm{ref}}} \tag{94}$$

where $(\Delta T)_{\mathrm{ref}}$ is a reference value for ΔT, which can be selected as $(\Delta T)_{\mathrm{ref}} = Y$.

10. PARALLEL IMPLEMENTATIONS

The T⋆AFSM's parallel implementations of their FSI algorithms, including the fluid mechanics, structural mechanics, mesh-moving and mesh-to-mesh projection components, are based on their earlier parallel implementations (see [55, 81, 96–104]) of T⋆AFSM's semi-discrete flow solvers, space–time flow solvers for moving boundaries and interfaces, mesh-moving methods and mesh-to-mesh projection techniques. Various aspects of the implementations of the flow solvers and mesh-to-mesh projection techniques were extensively described in several of the references listed above, and the implementation of the mesh-moving methods is essentially a subset of the implementation of the semi-discrete flow solvers. The implementation of the T⋆AFSM space–time FSI algorithm was described in [31] in the context of parallel computation of parachute FSI. The description was less extensive than the earlier, published descriptions of the T⋆AFSM's semi-discrete and space–time flow solvers, mesh-moving methods, and mesh-to-mesh projection techniques. This is because the parallel implementation of the additional component (structural mechanics solver) was, as it was stated in [31], 'essentially identical' to the implementation of the flow solver. In fact, the T⋆AFSM accomplished that parallel implementation by simply starting with a parallel, semi-discrete flow solver developed earlier by the T⋆AFSM, and importing into it the ingredients of the structural mechanics algorithm [77, 78] from an existing, non-parallel structural mechanics solver [78]. The fully parallel T⋆AFSM space–time FSI solver, in its various algorithmic forms, was used extensively for parachute FSI computations, and those computations were reported in a large number of journal papers (see, for example, [33, 34, 36, 37, 65, 66, 105, 106]), conference papers and PhD theses. In all those papers and theses, the parallel computations reported were based on parallel implementation of *all* main components of the FSI solver, including the structural mechanics component.

11. PARACHUTE MODELLING

The earliest applications of the DSD/SST formulation to parachute FSI modelling were reported by the T⋆AFSM in a 1997 AIAA paper [30]. These were axisymmetric computations focusing on the inflation of the parachute and were carried out on parallel computers. The 3D parachute FSI computations reported by the T⋆AFSM following that were also all carried out on parallel computers, using fully parallel FSI solvers (see Section 10), based on the DSD/SST formulation

DOI: 10.1002/fld

and the mesh update methods (see Section 4) developed by the T★AFSM. We will mention here some of those 3D computations reported earlier.

Computations for a ram-air parachute were reported in [31], together with a description of the parallel implementation of the T★AFSM space–time FSI algorithm. Modelling of a round parachute with emphasis on performance and control was reported in [32, 33]. Computations for a cross parachute and comparison of numerical predictions with wind-tunnel data were reported in [64, 65]. Computation of a round parachute crossing the far wake of an aircraft was reported in [34]. Modelling of a round parachute with emphasis on soft landing was reported in [66]. Computations of the aerodynamic interactions between multiple parachute canopies were reported in [70, 105, 106].

The parachute computations mentioned in this section so far were based on the block-iterative coupling technique (see Section 5.1). The quasi-direct and direct coupling techniques (see Sections 5.2 and 5.3) introduced in a January 2004 conference paper [35] brought the T★AFSM FSI modelling techniques, including parachute modelling techniques, to a new level. The journal version of the conference paper was published in 2006 (see [36]). These newer techniques had more computational robustness and were better suited for a new FSI application (such as a new and complex parachute design or a new parachute manoeuvre) that one might encounter and have less computing experience with. Soft-landing of a T–10 parachute with pneumatic muscle actuator (PMA) was reported in [35, 36, 107]. The new techniques put the T★AFSM in a better position in modelling of large cargo parachutes such, as G–12 (see [37, 38, 62]) and G–11 (see [5]), and modelling of complex manoeuvres with those large cargo parachutes, such as soft landing (see [37, 38, 62]) and disreefing (see [38, 62]).

As it was stated in Remarks 8 and 12 in Section 3, the new versions of the DSD/SST formulation and space–time FSI techniques introduced in this paper render the earlier versions obsolete. Consequently, the new parachute FSI modelling techniques based on these new space–time FSI techniques render the earlier versions of the parachute FSI modelling techniques obsolete. The new parachute FSI modelling techniques offer better computational efficiency (see Remarks 7 and 11). They also offer a more realistic representation of the parachutes, because now the fabric porosity can be taken into account in the computations.

12. TEST COMPUTATIONS

All computations were carried out in a parallel computing environment, using PC clusters. In all cases, the fully discretized, coupled fluid and structural mechanics and mesh-moving equations were solved with the quasi-direct coupling technique (see Section 5.2).

12.1. A cloth piece falling over a rigid rod

A $1.0\,\mathrm{m} \times 1.0\,\mathrm{m}$ piece of cloth is dropped in the air over a rigid rod that is 0.1 m thick and 1.2 m long, from a height of 0.02 m above the surface of the rod. The thickness, density and stiffness of the cloth are 0.002 m, $100\,\mathrm{kg/m^3}$ and $1.0 \times 10^4\,\mathrm{N/m^2}$, respectively. The mesh for the cloth consists of 2909 nodes and 5616 three-node triangular membrane elements. The initial configuration is shown by the top picture in Figure 1. The fluid mechanics mesh contains approximately 58 000 nodes and 335 000 four-node tetrahedral elements. The computation is carried out with the SSTFSI-TIP1 technique (see Remarks 5 and 10) and the SUPG test function option WTSA (see Remark 2). The

DOI: 10.1002/fld

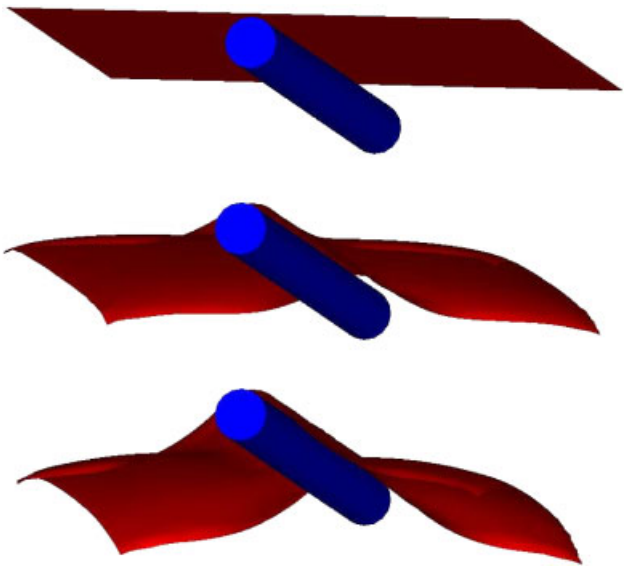

Figure 1. A cloth piece falling over a rigid rod, at $t = 0.00, 0.20$ and 0.30 s.

stabilization parameters used are those given by Equations (7)–(13). Split nodal values for pressure are used at the edges of the membrane structure (see Remark 9). The SENCT-D technique (see Section 7) is used as the contact algorithm. The GMRES search technique is used with a diagonal preconditioner. The time-step size is 0.01 s. The number of nonlinear iterations per time step is 8, and the number of GMRES iterations per nonlinear iteration is 30. The entire computation was completed with only one remeshing. Figures 1–3 show the cloth at various instants during the simulation.

12.2. Flow in a tube constricted with a flexible diaphragm

A tube with diameter 3 mm and length 6 mm is constricted in the middle with a flexible diaphragm that has a hole with diameter 1.5 mm. The fluid has properties similar to that of human blood, with density and viscosity $1000\,\mathrm{kg/m^3}$ and $4.0 \times 10^{-6}\,\mathrm{m^2/s}$. Figure 4 shows the problem setup. A pulsating inflow is specified in the form of a Cosine wave with period 0.3 s. The minimum and maximum values of the magnitude of the inflow velocity are 0.075 and 0.675 m/s. A pulsating traction boundary condition, also in the form of that Cosine wave, is specified at the outflow, with minimum and maximum values 80 and 120 mmHg. The tube walls are rigid with no-slip boundary surfaces. The diaphragm surfaces are also no-slip boundaries. The thickness, density and stiffness of the diaphragm are 0.3 mm, $1000\,\mathrm{kg/m^3}$ and $5.0 \times 10^5\,\mathrm{N/m^2}$, respectively. The mesh for the diaphragm consists of 337 nodes and 578 three-node triangular membrane elements. The fluid mechanics mesh contains 9187 nodes and 50 212 four-node tetrahedral elements. The computation is carried out with the SSTFSI-TIP1 technique (see Remarks 5 and 10) and the SUPG test function option WTSA (see Remark 2). The stabilization parameters used are those given by Equations (7)–(13). Split nodal values for pressure are used at the edges of the membrane structure (see Remark 9). The GMRES search technique is used with a preconditioner based on

DOI: 10.1002/fld

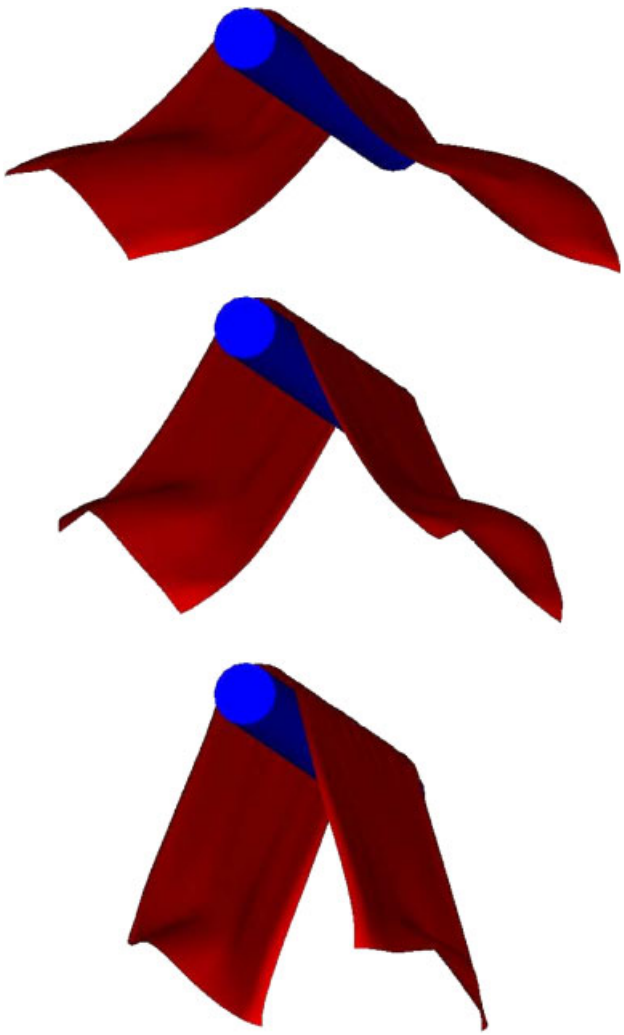

Figure 2. A cloth piece falling over a rigid rod, at $t = 0.40, 0.50$ and 0.60 s.

the segregated solver SESFSI (see Section 6.3). Specifically, we use the preconditioner $\mathbf{P}_{\text{SIMP}}$, given by Equation (69). We use 'Selective Scaling' (see Remark 14) and scale up the structural mechanics equations by a factor of 2.0 to enhance the convergence of that part. The time-step size is 5.333×10^{-4} s. The number of nonlinear iterations per time step is 5, the number of (upper) GMRES iterations per nonlinear iteration is 30, and the number of lower GMRES iterations per upper iteration is 20. The entire computation was completed without any remeshing. As seen in Figures 5–7, the diaphragm bulges and flattens in synchronization with the inflow velocity. We also note from Figure 7 that there is a good match between the volumetric flow rates for the inflow and outflow.

DOI: 10.1002/fld

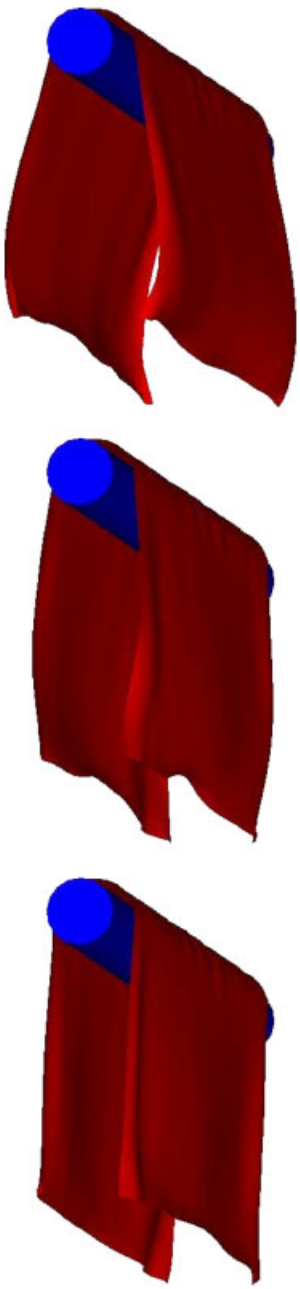

Figure 3. A cloth piece falling over a rigid rod, at $t = 0.70, 0.85$ and 0.95 s.

DOI: 10.1002/fld

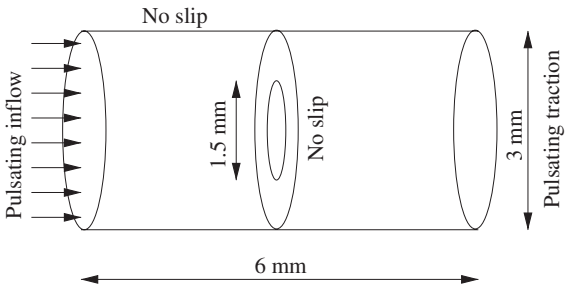

Figure 4. Flow in a tube constricted with a flexible diaphragm. Problem setup.

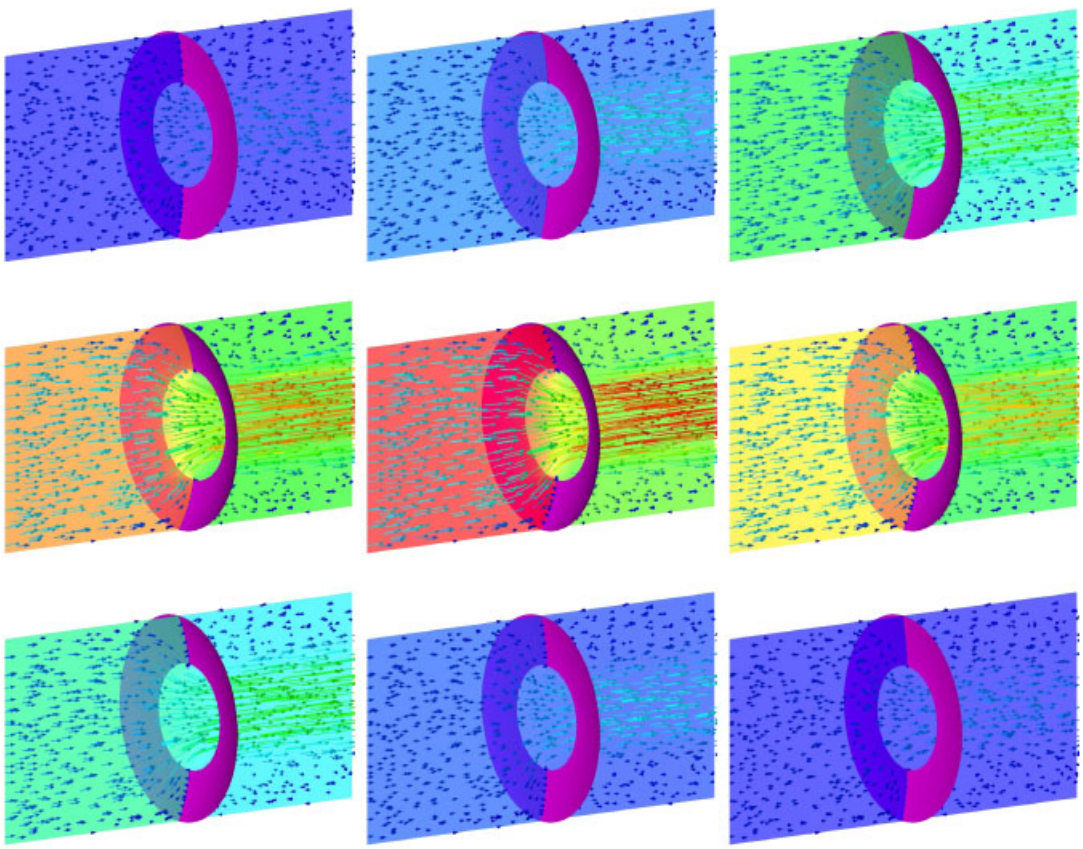

Figure 5. Flow in a tube constricted with a flexible diaphragm. Time history (left to right and top to bottom) of the velocity field and pressure on a vertical plane. Velocity vectors are coloured by their magnitudes.

DOI: 10.1002/fld

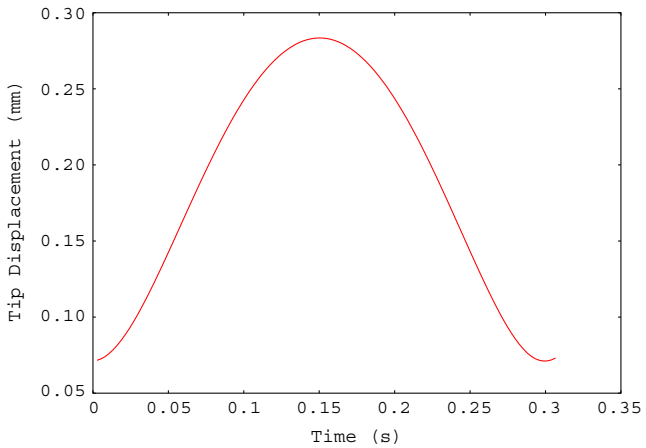

Figure 6. Flow in a tube constricted with a flexible diaphragm. Displacement of the diaphragm at a point along its inner edge.

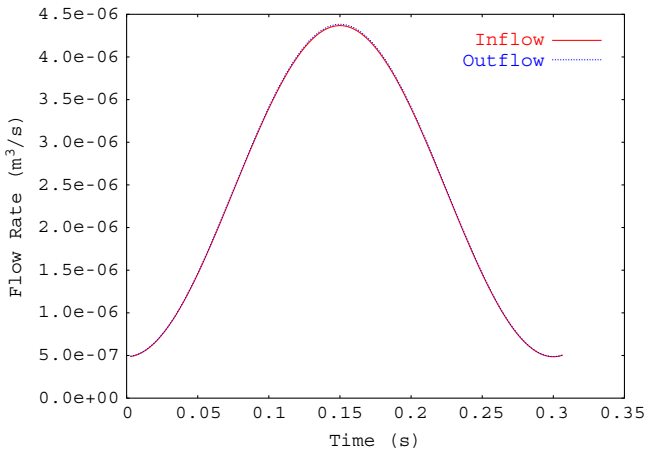

Figure 7. Flow in a tube constricted with a flexible diaphragm. Volumetric flow rate for inflow and outflow.

12.3. Inflation of a balloon

A balloon, initially spherical, is inflated by pumping air through a circular hole as shown in Figure 8. The inflow is pulsating in the form of a Cosine wave with period 2 s. The minimum and maximum values of the magnitude of the inflow velocity are 0.0 and 2.0 m/s. Initially, the diameter of the balloon is 2 m and the diameter of the circular hole is 0.6245 m. The thickness, density and stiffness of the balloon are 2.0 mm, 100 kg/m^3 and 1.0×10^3 N/m^2, respectively. The mesh for the balloon consists of 1479 nodes and 2936 three-node triangular membrane elements. The fluid mechanics mesh for the air inside balloon contains 6204 nodes and 32 455 four-node tetrahedral elements. The computation is carried out with the SSTFSI-TIP1 technique (see Remarks 5 and 10) and the SUPG test function option WTSA (see Remark 2). The stabilization parameters used are those given by Equations (7)–(13). The GMRES search technique is used with a diagonal

DOI: 10.1002/fld

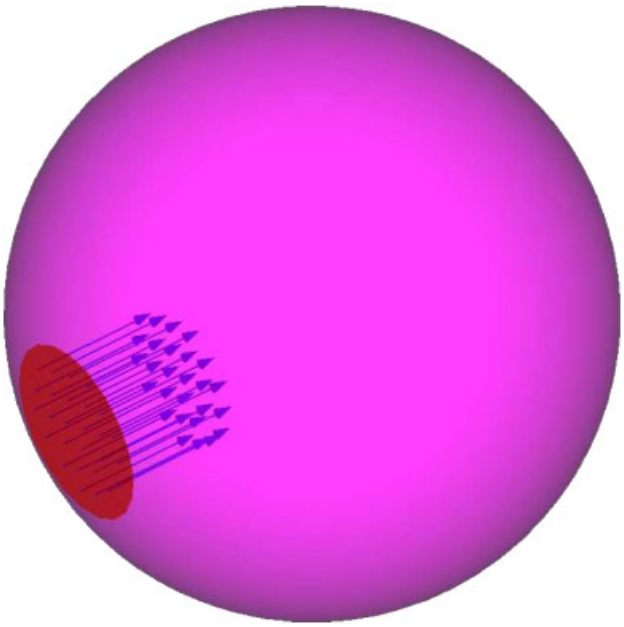

Figure 8. Inflation of a balloon. Problem setup.

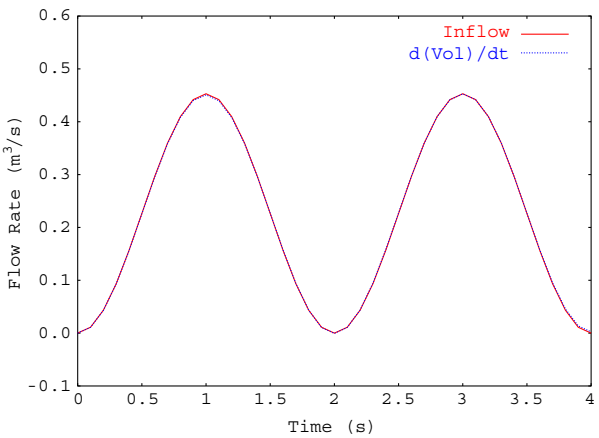

Figure 9. Inflation of a balloon. Volumetric flow rate for the inflow and the rate of change for the balloon volume.

preconditioner. The time-step size is 0.1 s, and the computation duration is 4 s. The number of nonlinear iterations per time step is 5, and the number of GMRES iterations per nonlinear iteration is 30. The entire computation was completed without any remeshing. Figure 9 shows that volumetric flow rate for the inflow matches the rate of change for the balloon volume. Figure 10 shows that the instantaneous balloon volume matches the initial balloon volume plus the volume of air added. Figures 11 and 12 show the flow field during the two periods of inflation.

DOI: 10.1002/fld

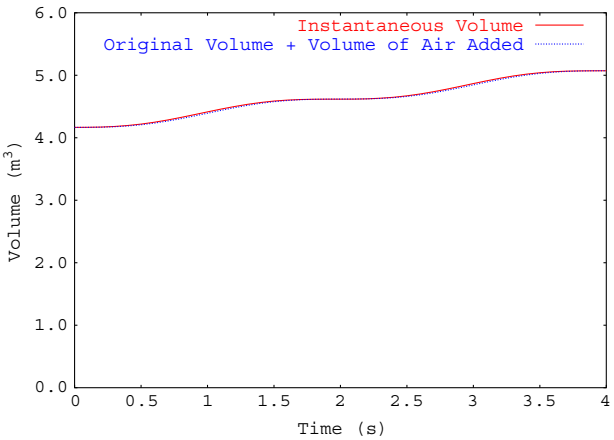

Figure 10. Inflation of a balloon. Instantaneous balloon volume compared with the initial balloon volume plus the volume of air added.

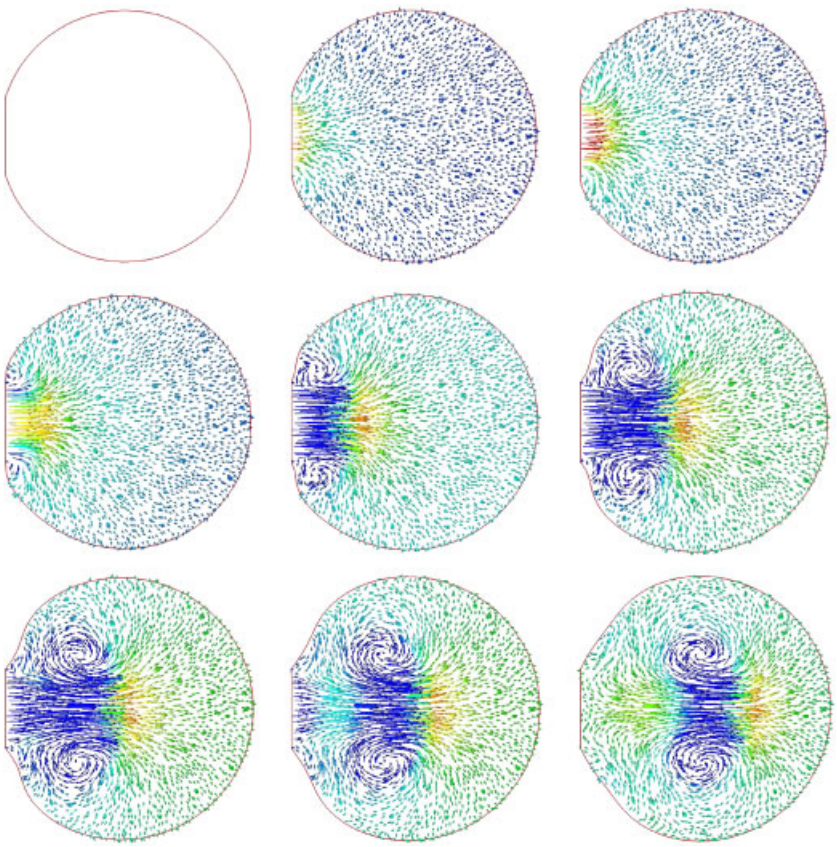

Figure 11. Inflation of a balloon. Velocity vectors coloured by air pressure, from 0 to 2 s.

DOI: 10.1002/fld

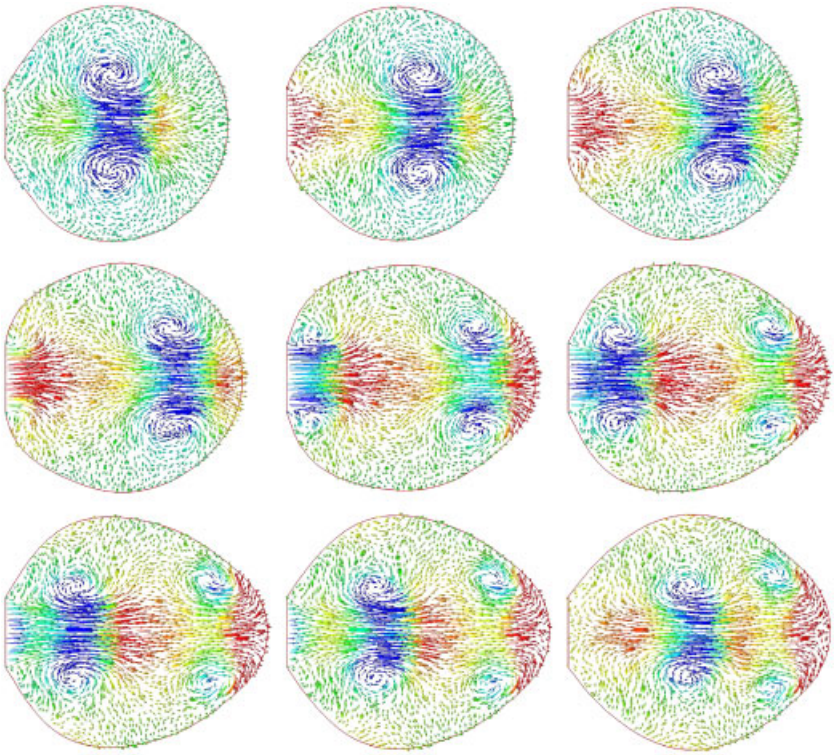

Figure 12. Inflation of a balloon. Velocity vectors coloured by air pressure, from 2 to 4 s.

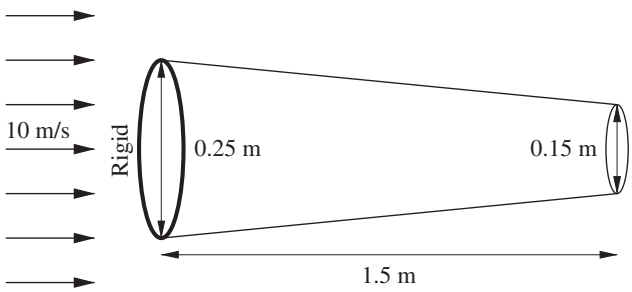

Figure 13. Flow through and around a windsock. Problem setup.

12.4. Flow through and around a windsock

The windsock has a length of 1.5 m and a diameter ranging from 0.25 m upstream to 0.15 m downstream (see Figure 13). Initially the windsock is in a horizontal position, and the starting condition for the flow field is the developed flow field corresponding to a rigid windsock held in that horizontal position. Then the gravity is turned on for the windsock, the FSI starts, and the windsock starts falling down. The wind velocity is constant at 10 m/s. The thickness, density and

DOI: 10.1002/fld

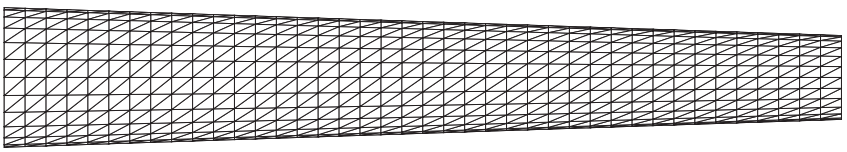

Figure 14. Flow through and around a windsock. The initial windsock mesh.

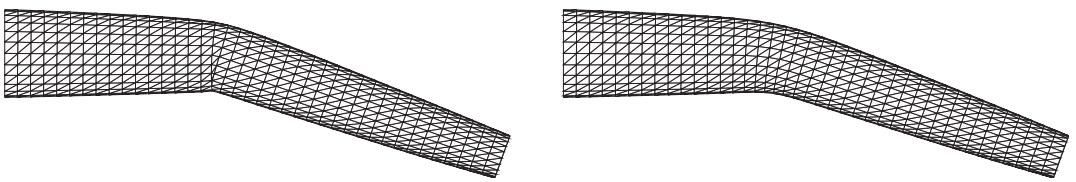

Figure 15. Flow through and around a windsock. Meshes at the interface: structure (left) and fluid (right).

stiffness of the windsock are 2.0 mm, $100\,\mathrm{kg/m^3}$ and $1.0 \times 10^6\,\mathrm{N/m^2}$, respectively. The upstream edge of the structure is held fixed while the remaining structure is free and flaps in cycles. The mesh for the windsock is semi-structured and consists of 984 nodes and 1920 three-node triangular membrane elements (see Figure 14). The fluid mechanics mesh contains 19 579 nodes and 113 245 four-node tetrahedral elements. Initially, the fluid mesh at the interface is identical to the windsock mesh. The computation is carried out with the SSTFSI-SV technique (see Remarks 6 and 10) and the SUPG test function option WTSE (see Remark 2). The stabilization parameters used are those given by Equations (9)–(12) and Equations (14)–(17). The GMRES search technique is used with a diagonal preconditioner. The time-step size is 0.0125 s, and the computation duration is two cycles of flapping. The number of nonlinear iterations per time step is 5, and the number of GMRES iterations per nonlinear iteration is 30.

We expected the windsock to develop kinks as it flaps in the wind. Therefore, we used the FSI-GST (see Section 4.5) for smoothing the fluid mesh at the interface. The nodes of the windsock mesh were generated on straight longitudinal gridlines, and with that we were able to use the directional version of the FSI-GST, i.e. FSI-DGST (see Section 4.5). For a node A on such a gridline, we use a weighted averaging involving four nearby nodes on each side: $A \pm 1$, $A \pm 2$, $A \pm 3$ and $A \pm 4$. The weighted averaging formula is given as follows:

$$\mathbf{X}_A^{\mathrm{SMOOTH}} = 0.2\mathbf{X}_A + 0.16(\mathbf{X}_{A-1} + \mathbf{X}_{A+1}) + 0.12(\mathbf{X}_{A-2} + \mathbf{X}_{A+2}) + 0.08(\mathbf{X}_{A-3} + \mathbf{X}_{A+3}) + 0.04(\mathbf{X}_{A-4} + \mathbf{X}_{A+4})$$

We note that this directional smoothing does not introduce any smoothing in the circumferential direction. During the FSI computations the structure develops kinks, which would make mesh updating more difficult and increase the frequency of remeshing. With the FSI-DGST, two cycles of flapping were computed without any remeshing. Figure 15 shows the structural and fluid mechanics meshes at the interface, one with a kink and the other one smooth. Figure 16 shows the zoomed (around the kink) versions of the pictures in Figure 15. Figure 17 shows the windsock and the flow field at various instants.

DOI: 10.1002/fld

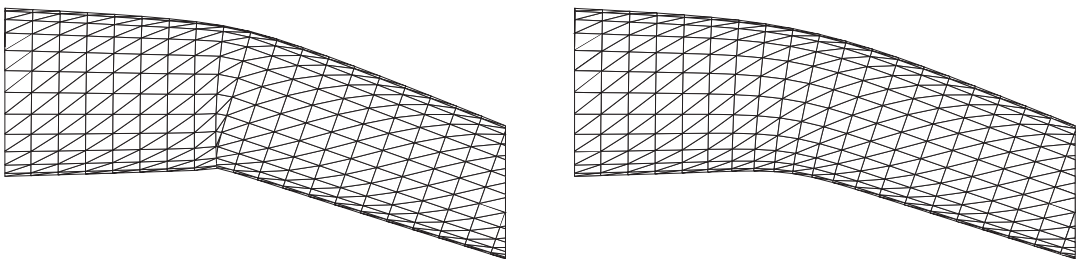

Figure 16. Flow through and around a windsock. Meshes at the interface: structure (left) and fluid (right).

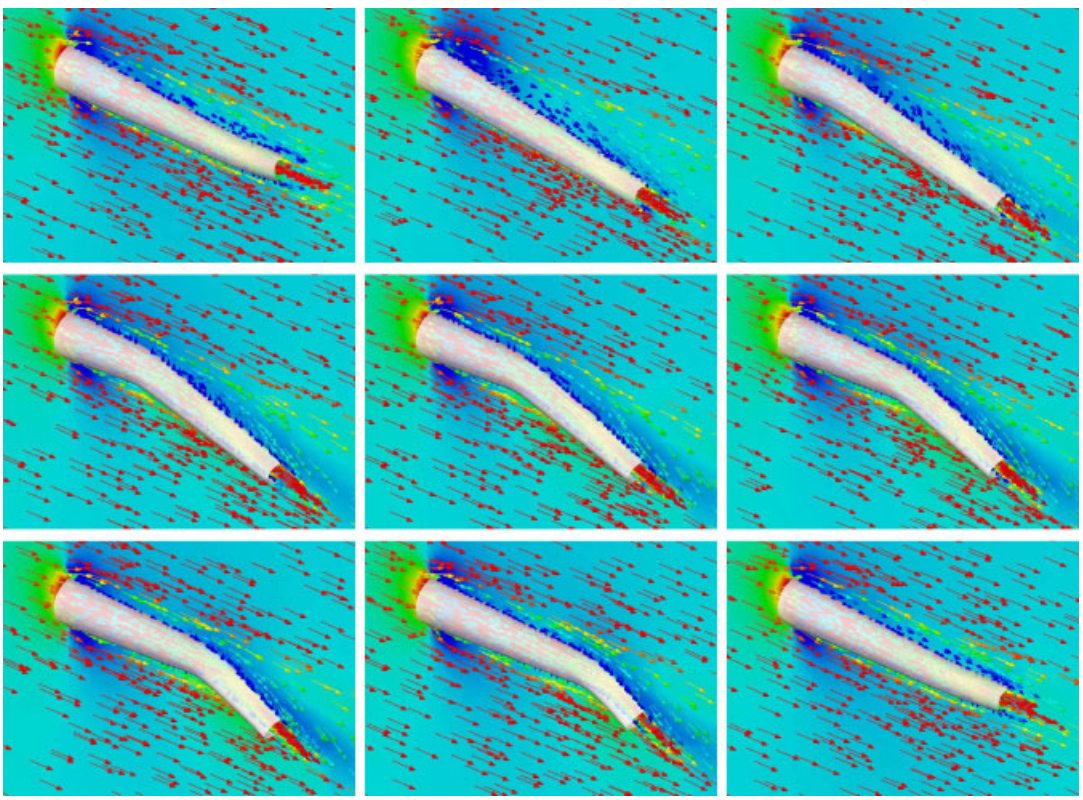

Figure 17. Flow through and around a windsock. The windsock and the flow field (velocity and pressure) at various instants. Velocity vectors are coloured by their magnitudes.

12.5. Descent of a T–10 parachute with fabric porosity

T–10 is a 35-ft diameter personnel parachute with 30 suspension lines each 29.4 ft long. The thickness, density and stiffness of the canopy membrane are 1.0×10^{-4} ft, 2.374 slugs/ft^3 and 2.0×10^5 lb/ft^2, respectively. The fabric porosity for the canopy membrane is 100 CFM. The cross-sectional area, density and stiffness of the cables are 3.358×10^{-5} ft^2, 2.374 slugs/ft^3 and

DOI: 10.1002/fld

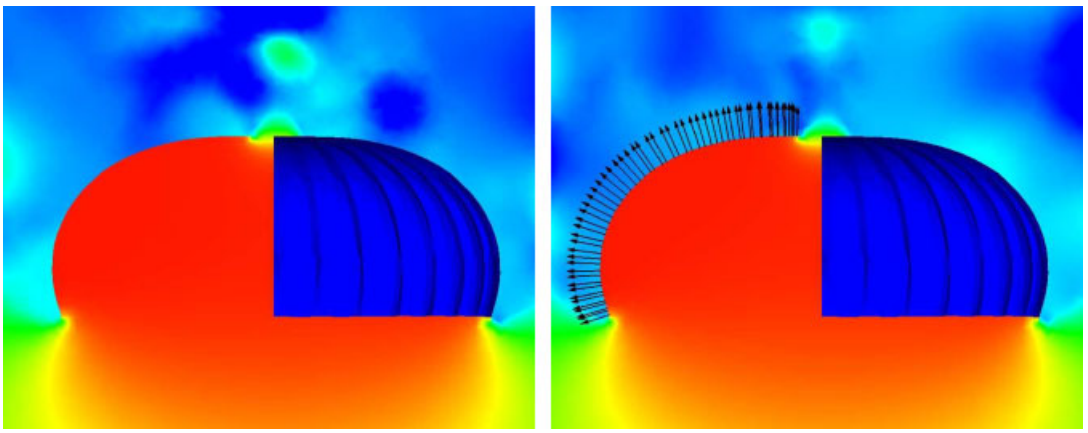

Figure 18. Descent of a T–10 parachute with porosity. Pressure field and velocity vectors without (left) and with (right) fabric porosity.

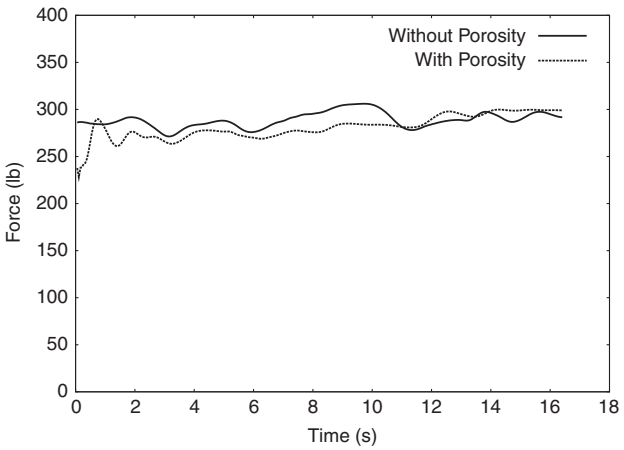

Figure 19. Descent of a T–10 parachute with fabric porosity. Drag force with and without porosity.

1.117×10^7 lb/ft^2, respectively. The payload is 250 lb. The mesh for the parachute consists of 3385 nodes and 5880 three-node triangular membrane elements, 1496 two-node cable elements, and 4 one-node payload elements. The fluid mechanics mesh contains 133 740 nodes and 810 213 four-node tetrahedral elements. The computation is carried out with the SSTFSI-TIP1 technique (see Remarks 5 and 10) and the SUPG test function option WTSA (see Remark 2). The stabilization parameters used are those given by Equations (7)–(12) and Equation (17). In Equation (8), the $\partial N_a/\partial t$ term is dropped. The GMRES search technique is used with a diagonal preconditioner. The time-step size is 0.0547 s. The number of nonlinear iterations per time step is 6, and the number of GMRES iterations per nonlinear iteration is 30. First, we carried out a fluid mechanics simulation with no FSI. Figure 18 shows the pressure field and velocity vectors with and without fabric porosity. Figure 19 shows the drag force with and without porosity. After that we carried out the

DOI: 10.1002/fld

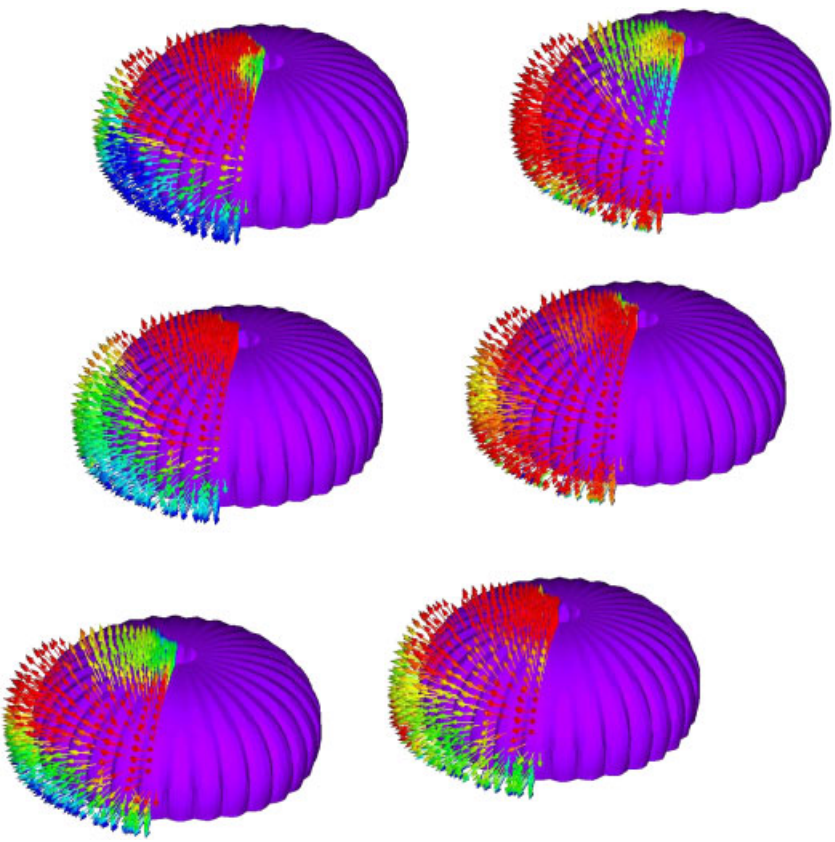

Figure 20. Descent of a T–10 parachute with fabric porosity. Canopy shape and velocity vectors at various instants.

FSI modelling of the descent. The entire FSI computation was completed without any remeshing. Figure 20 shows the canopy shape and velocity vectors at various instants. The magnitude of the velocity vectors crossing the canopy fabric is 1.6–1.7% of the descent speed.

13. CONCLUDING REMARKS

We provided a review of the space–time fluid–structure interaction (FSI) techniques developed by the Team for Advanced Flow Simulation and Modelling (T★AFSM) and described the enhancements introduced recently by the T★AFSM to increase the scope, accuracy, robustness and efficiency of these space–time FSI techniques. We described how the deforming-spatial-domain/stabilized space–time (DSD/SST) formulation is enhanced, how the fluid–structure interface conditions are handled in a different way, how preconditioning techniques more sophisticated than diagonal preconditioning can be used in iterative solution of the linear equation systems, and how a contact algorithm can protect the quality of the fluid mechanics mesh between the structural surfaces

DOI: 10.1002/fld

coming into contact. We also briefly described, for the purpose of comparison and generalization, how the SUPG and PSPG stabilizations can be extended to the ALE formulation and problems with thermal coupling. We presented a number of 3D test problems computed with the new stabilized space–time FSI (SSTFSI) techniques and demonstrated how these new SSTFSI techniques work.

REFERENCES

1. Ohayon R. Reduced symmetric models for modal analysis of internal structural-acoustic and hydroelastic-sloshing systems. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**:3009–3019.
2. Wall W. Fluid–structure interaction with stabilized finite elements. *Ph.D. Thesis*, University of Stuttgart, 1999.
3. Michler C, van Brummelen EH, Hulshoff SJ, de Borst R. The relevance of conservation for stability and accuracy of numerical methods for fluid–structure interaction. *Computer Methods in Applied Mechanics and Engineering* 2003; **192**:4195–4215.
4. Gerbeau J-F, Vidrascu M. A quasi-Newton algorithm based on a reduced model for fluid–structure interaction problems in blood flows. *Mathematical Modelling and Numerical Analysis* 2003; **37**:663–680.
5. Sathe S. Enhanced-discretization and solution techniques in flow simulations and parachute fluid–structure interactions. *Ph.D. Thesis*, Rice University, 2004.
6. Heil M. An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:1–23.
7. Hubner B, Walhorn E, Dinkler D. A monolithic approach to fluid–structure interaction using space–time finite elements. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:2087–2104.
8. Dettmer W. Finite element modeling of fluid flow with moving free surfaces and interfaces including fluid–solid interaction. *Ph.D. Thesis*, University of Wales Swansea, 2004.
9. Michler C, van Brummelen EH, Hulshoff SJ, de Borst R. A monolithic approach to fluid–structure interaction. *Computers and Fluids* 2004; **33**:839–848.
10. van Brummelen EH, de Borst R. On the nonnormality of subiteration for a fluid–structure interaction problem. *SIAM Journal on Scientific Computing* 2005; **27**:599–621.
11. Dettmer W, Peric D. A computational framework for fluid–rigid body interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:1633–1666.
12. Michler C, van Brummelen EH, de Borst R. An interface Newton–Krylov solver for fluid–structure interaction. *International Journal for Numerical Methods in Fluids* 2005; **47**:1189–1195.
13. Fernandez MA, Moubachir M. A Newton method using exact Jacobians for solving fluid–structure coupling. *Computers and Structures* 2005; **83**:127–142.
14. Gerbeau J-F, Vidrascu M, Frey P. Fluid–structure interaction in blood flow on geometries based on medical images. *Computers and Structures* 2005; **83**:155–165.
15. Dettmer W, Peric D. A computational framework for fluid–structure interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering* 2006 (Published Online).
16. Bazilevs Y, Calo VM, Zhang Y, Hughes TJR. Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. *Computational Mechanics* 2006; **38**:310–322.
17. Masud A, Khurram RA. A multiscale finite element method for the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:1750–1777.
18. Khurram RA, Masud A. A multiscale/stabilized formulation of the incompressible Navier–Stokes equations for moving boundary flows and fluid–structure interaction. *Computational Mechanics* 2006; **38**:403–416.
19. Kuttler U, Forster C, Wall WA. A solution for the incompressibility dilemma in partitioned fluid–structure interaction with pure Dirichlet fluid domains. *Computational Mechanics* 2006; **38**:417–429.
20. Masud A, Bhanabhagvanwala M, Khurram RA. An adaptive mesh rezoning scheme for moving boundary flows and fluid–structure interaction. *Computers and Fluids* 2007; **36**:77–91.
21. Sawada T, Hisada T. Fluid–structure interaction analysis of the two dimensional flag-in-wind problem by an interface tracking ALE finite element method. *Computers and Fluids* 2007; **36**:136–146.
22. Wall WA, Genkinger S, Ramm E. A strong coupling partitioned approach for fluid–structure interaction with free surfaces. *Computers and Fluids* 2007; **36**:169–183.
23. Lohner R, Cebral JR, Yang C, Baum JD, Mestreau EL, Soto O. Extending the range of applicability of the loose coupling approach for FSI simulations. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 82–100.

DOI: 10.1002/fld

24. Dunne T, Rannacher R. Adaptive finite element approximation of fluid–structure interaction based on Eulerian variational formulation. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 110–145.
25. Schafer M, Heck M, Yigit S. An implicit partitioned method for the numerical simulation of fluid–structure interaction. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 171–194.
26. Wall WA, Gerstenberger A, Gamnitzer P, Forster C, Ramm E. Large deformation fluid–structure interaction–advances in ALE methods and new fixed grid approaches. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 195–232.
27. Brenk M, Bungartz H-J, Mehl M, Neckel T. Fluid–structure interaction on Cartesian grids: flow simulation and coupling environment. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 233–269.
28. Bletzinger K-U, Wuchner R, Kupzok A. Algorithmic treatment of shells and free form-membranes in FSI. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 336–355.
29. Mittal S, Tezduyar TE. Parallel finite element simulation of 3D incompressible flows—fluid–structure interactions. *International Journal for Numerical Methods in Fluids* 1995; **21**:933–953.
30. Stein KR, Benney RJ, Kalro V, Johnson AA, Tezduyar TE. Parallel computation of parachute fluid–structure interactions. *Proceedings of AIAA 14th Aerodynamic Decelerator Systems Technology Conference*, *AIAA Paper 97-1505*, San Francisco, California, 1997.
31. Kalro V, Tezduyar TE. A parallel 3D computational method for fluid–structure interactions in parachute systems. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:321–332.
32. Stein K, Benney R, Tezduyar T, Kalro V, Leonard J, Accorsi M. 3-D computation of parachute fluid–structure interactions: performance and control. *Proceedings of CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, *AIAA Paper 99-1714*, Toulouse, France, 1999.
33. Stein K, Benney R, Kalro V, Tezduyar TE, Leonard J, Accorsi M. Parachute fluid–structure interactions: 3-D computation. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:373–386.
34. Tezduyar T, Osawa Y. Fluid–structure interactions of a parachute crossing the far wake of an aircraft. *Computer Methods in Applied Mechanics and Engineering* 2001; **191**:717–726.
35. Tezduyar TE, Sathe S, Keedy R, Stein K. Space–time techniques for finite element computation of flows with moving boundaries and interfaces. *Proceedings of the III International Congress on Numerical Methods in Engineering and Applied Science* (CD-ROM), Gallegos S, Herrera I, Botello S, Zarate F, Ayala G (eds), 2004.
36. Tezduyar TE, Sathe S, Keedy R, Stein K. Space–time finite element techniques for computation of fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:2002–2027.
37. Tezduyar TE, Sathe S, Stein K. Solution techniques for the fully-discretized equations in computation of fluid–structure interactions with the space–time formulations. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:5743–5753.
38. Tezduyar TE, Sathe S, Stein K, Aureli L. Modeling of fluid–structure interactions with the space–time techniques. In *Fluid–Structure Interaction*, Bungartz H-J, Schafer M (eds). Lecture Notes in Computational Science and Engineering. Springer: Berlin, 2006; 50–81.
39. Torii R, Oshima M, Kobayashi T, Takagi K, Tezduyar TE. Influence of wall elasticity on image-based blood flow simulation. *Japan Society of Mechanical Engineers Journal, Series A* 2004; **70**:1224–1231 (in Japanese).
40. Torii R, Oshima M, Kobayashi T, Takagi K, Tezduyar TE. Computer modeling of cardiovascular fluid–structure interactions with the deforming-spatial-domain/stabilized space–time formulation. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:1885–1895.
41. Torii R, Oshima M, Kobayashi T, Takagi K, Tezduyar TE. Fluid–structure interaction modeling of aneurysmal conditions with high and normal blood pressures. *Computational Mechanics* 2006; **38**:482–490.
42. Torii R, Oshima M, Kobayashi T, Takagi K, Tezduyar TE. Influence of wall elasticity in patient-specific hemodynamic simulations. *Computers and Fluids* 2007; **36**:160–168.
43. Hughes TJR, Liu WK, Zimmermann TK. Lagrangian–Eulerian finite element formulation for incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering* 1981; **29**:329–349.
44. Farhat C, Geuzaine P. Design and analysis of robust ALE time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:4073–4095.
45. Masud A. Effects of mesh motion on the stability and convergence of ALE based formulations for moving boundary flows. *Computational Mechanics* 2006; **38**:430–439.

DOI: 10.1002/fld

46. Hughes TJR, Brooks AN. A multi-dimensional upwind scheme with no crosswind diffusion. In *Finite Element Methods for Convection Dominated Flows*, Hughes TJR (ed.), AMD-vol. 34. ASME: New York, 1979; 19–35.
47. Brooks AN, Hughes TJR. 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* 1982; **32**:199–259.
48. Tezduyar TE. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics* 1992; **28**:1–44.
49. Tezduyar TE, Mittal S, Ray SE, Shih R. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity–pressure elements. *Computer Methods in Applied Mechanics and Engineering* 1992; **95**:221–242.
50. Hughes TJR, Franca LP, Balestra M. 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* 1986; **59**:85–99.
51. Tezduyar TE, Behr M, Liou J. 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* 1992; **94**:339–351.
52. Tezduyar TE, Behr M, Mittal S, Liou J. 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* 1992; **94**:353–371.
53. Hughes TJR, Hulbert GM. Space–time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering* 1988; **66**:339–363.
54. Tezduyar TE. Computation of moving boundaries and interfaces and stabilization parameters. *International Journal for Numerical Methods in Fluids* 2003; **43**:555–575.
55. Tezduyar TE, Behr M, Mittal S, Johnson AA. 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.
56. Johnson AA, Tezduyar TE. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering* 1994; **119**:73–94.
57. Tezduyar TE. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering* 2001; **8**:83–130.
58. Masud A, Hughes TJR. 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* 1997; **146**:91–126.
59. Stein K, Tezduyar T, Benney R. Mesh moving techniques for fluid–structure interactions with large displacements. *Journal of Applied Mechanics* 2003; **70**:58–63.
60. Tezduyar T. Finite element interface-tracking and interface-capturing techniques for flows with moving boundaries and interfaces. *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, 2001.
61. Tezduyar TE. Stabilized finite element formulations and interface-tracking and interface-capturing techniques for incompressible flows. In *Numerical Simulations of Incompressible Flows*, Hafez MM (ed.). World Scientific: New Jersey, 2003; 221–239.
62. Tezduyar TE, Sathe S, Senga M, Aureli L, Stein K, Griffin B. Finite element modeling of fluid–structure interactions with space–time and advanced mesh update techniques. *Proceedings of the 10th International Conference on Numerical Methods in Continuum Mechanics* (CD-ROM), Zilina, Slovakia, 2005.
63. Mittal S, Tezduyar TE. A finite element study of incompressible flows past oscillating cylinders and aerofoils. *International Journal for Numerical Methods in Fluids* 1992; **15**:1073–1118.
64. Stein K, Benney R, Tezduyar T, Kalro V, Potvin J, Bretl T. Fluid–structure interaction simulation of a cross parachute: comparison of numerical predictions with wind tunnel data. *Proceedings of CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, *AIAA Paper 99-1725*, Toulouse, France, 1999.
65. Stein K, Benney R, Tezduyar T, Potvin J. Fluid–structure interactions of a cross parachute: numerical simulation. *Computer Methods in Applied Mechanics and Engineering* 2001; **191**:673–687.
66. Stein KR, Benney RJ, Tezduyar TE, Leonard JW, Accorsi ML. Fluid–structure interactions of a round parachute: modeling and simulation techniques. *Journal of Aircraft* 2001; **38**:800–808.

DOI: 10.1002/fld

67. Tezduyar TE. Stabilized finite element methods for computation of flows with moving boundaries and interfaces. Lecture Notes on Finite Element Simulation of Flow Problems (Basic, Advanced Course). Japan Society of Computational Engineering and Sciences: Tokyo, Japan, 2003.
68. Tezduyar TE. Stabilized finite element methods for flows with moving boundaries and interfaces. *HERMIS*: *The International Journal of Computer Mathematics and its Applications* 2003; **4**:63–88.
69. Tezduyar TE. Finite element methods for fluid dynamics with moving boundaries and interfaces. In *Encyclopedia of Computational Mechanics*, vol. 3: *Fluids*, Stein E, De Borst R, Hughes TJR (eds). Wiley: New York, 2004 (Chapter 17).
70. Tezduyar TE. Moving boundaries and interfaces. In *Finite Element Methods*: *1970's and Beyond*, Franca LP, Tezduyar TE, Masud A (eds). CIMNE: Barcelona, Spain, 2004; 205–220.
71. Tezduyar TE. Finite elements in fluids: special methods and enhanced solution techniques. *Computers and Fluids* 2007; **36**:207–223.
72. Sathe S, Benney R, Charles R, Doucette E, Miletti J, Senga M, Stein K, Tezduyar TE. Fluid–structure interaction modeling of complex parachute designs with the space–time finite element techniques. *Computers and Fluids* 2007; **36**:127–135.
73. Tezduyar TE, Osawa Y. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:411–430.
74. Tezduyar TE. Finite elements in fluids: stabilized formulations and moving boundaries and interfaces. *Computers and Fluids* 2007; **36**:191–206.
75. Tezduyar TE. Adaptive determination of the finite element stabilization parameters. *Proceedings of the ECCOMAS Computational Fluid Dynamics Conference 2001* (CD-ROM), Swansea, Wales, U.K., 2001.
76. Tezduyar T. Stabilization parameters and local length scales in SUPG and PSPG formulations. *Proceedings of the Fifth World Congress on Computational Mechanics*, http://wccm.tuwien.ac.at/ (On-line publication); *Paper-ID*: *81508*, Vienna, Austria, 2002.
77. Lo A. Nonlinear dynamic analysis of cable and membrane structure. *Ph.D. Thesis*, Department of Civil Engineering, Oregon State University, 1982.
78. Leonard JW, Benney RJ, Stein KR, Accorsi ML. Current 3-D structural dynamic finite element modeling capabilities. *Proceedings of AIAA 14th Aerodynamic Decelerator Systems Technology Conference*, *AIAA Paper 97-1506*, San Francisco, California, 1997.
79. Hilber HM, Hughes TJR, Taylor RL. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics* 1997; **5**:283–292.
80. Lynch DR. Wakes in liquid–liquid systems. *Journal of Computational Physics* 1982; **47**:387–411.
81. Tezduyar T, Aliabadi S, Behr M, Johnson A, Mittal S. Parallel finite-element computation of 3D flows. *Computer* 1993; **26**:27–36.
82. Stein K, Tezduyar T. Advanced mesh update techniques for problems involving large displacements. *Proceedings of the Fifth World Congress on Computational Mechanics*, http://wccm.tuwien.ac.at/ (On-line publication); *Paper-ID*: *81489*, Vienna, Austria, 2002.
83. Stein K, Tezduyar TE, Benney R. Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:2019–2032.
84. Fujisawa T, Inaba M, Yagawa G. Parallel computing of high-speed compressible flows using a node-based finite element method. *International Journal for Numerical Methods in Fluids* 2003; **58**:481–511.
85. Johnson AA, Tezduyar TE. Simulation of multiple spheres falling in a liquid-filled tube. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:351–373.
86. Saad Y, Schultz M. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**:856–869.
87. Johan Z, Hughes TJR, Shakib F. A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids. *Computer Methods in Applied Mechanics and Engineering* 1991; **87**:281–304.
88. Johan Z, Mathur KK, Johnsson SL, Hughes TJR. A case study in parallel computation: viscous flow around an Onera M6 wing. *International Journal for Numerical Methods in Fluids* 1995; **21**:877–884.
89. Tezduyar TE, Liou J, Ganjoo DK. Incompressible flow computations based on the vorticity–stream function and velocity–pressure formulations. *Computers and Structures* 1990; **35**:445–472.
90. Tezduyar TE, Mittal S, Shih R. Time-accurate incompressible flow computations with quadrilateral velocity–pressure elements. *Computer Methods in Applied Mechanics and Engineering* 1991; **87**:363–384.
91. Sameh A, Sarin V. Hybrid parallel linear solvers. *International Journal of Computational Fluid Dynamics* 1999; **12**:213–223.

DOI: 10.1002/fld

92. Sameh A, Sarin V. Parallel algorithms for indefinite linear systems. *Parallel Computing* 2002; **28**:285–299.
93. Hughes TJR, Mallet M, Mizukami A. A new finite element formulation for computational fluid dynamics: II. Beyond SUPG. *Computer Methods in Applied Mechanics and Engineering* 1986; **54**:341–355.
94. Tezduyar TE, Park YJ. Discontinuity capturing finite element formulations for nonlinear convection–diffusion-reaction equations. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**:307–325.
95. Tezduyar TE. Determination of the stabilization and shock-capturing parameters in SUPG formulation of compressible flows. *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering*, *ECCOMAS 2004* (CD-ROM), Jyvaskyla, Finland, 2004.
96. Behr M, Johnson A, Kennedy J, Mittal S, Tezduyar T. Computation of incompressible flows with implicit finite element implementations on the connection machine. *Computer Methods in Applied Mechanics and Engineering* 1993; **108**:99–118.
97. Behr M, Tezduyar TE. Finite element solution strategies for large-scale flow simulations. *Computer Methods in Applied Mechanics and Engineering* 1994; **112**:3–24.
98. Tezduyar TE, Aliabadi SK, Behr M, Mittal S. Massively parallel finite element simulation of compressible and incompressible flows. *Computer Methods in Applied Mechanics and Engineering* 1994; **119**:157–177.
99. Kennedy JG, Behr M, Kalro V, Tezduyar TE. Implementation of implicit finite element methods for incompressible flows on the CM-5. *Computer Methods in Applied Mechanics and Engineering* 1994; **119**:95–111.
100. Tezduyar T, Aliabadi S, Behr M, Johnson A, Kalro V, Litke M. Flow simulation and high performance computing. *Computational Mechanics* 1996; **18**:397–412.
101. Tezduyar T, Aliabadi S, Behr M, Johnson A, Kalro V, Litke M. High performance computing techniques for flow simulations. In *Solving Large-Scale Problems in Mechanics*: *Parallel Solution Methods in Computational Mechanics*, Papadrakakis M (ed.). Wiley: New York, 1997; 363–398 (Chapter 10).
102. Johnson AA, Tezduyar TE. 3D simulation of fluid-particle interactions with the number of particles reaching 100. *Computer Methods in Applied Mechanics and Engineering* 1997; **145**:301–321.
103. Tezduyar TE. CFD methods for three-dimensional computation of complex flow problems. *Journal of Wind Engineering and Industrial Aerodynamics* 1999; **81**:97–116.
104. Tezduyar T, Osawa Y. Methods for parallel computation of complex flow problems. *Parallel Computing* 1999; **25**:2039–2066.
105. Stein K, Tezduyar T, Kumar V, Sathe S, Benney R, Thornburg E, Kyle C, Nonoshita T. Aerodynamic interactions between parachute canopies. *Journal of Applied Mechanics* 2003; **70**:50–57.
106. Stein K, Tezduyar T, Benney R. Computational methods for modeling parachute systems. *Computing in Science and Engineering* 2003; **5**:39–46.
107. Tezduyar TE. Interface-tracking and interface-capturing techniques for finite element computation of moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:2983–3000.

DOI: 10.1002/fld