Comput Mech (2014) 53:1–15
DOI 10.1007/s00466-013-0888-x

ORIGINAL PAPER

Space–time VMS computation of wind-turbine rotor and tower aerodynamics

Kenji Takizawa · Tayfun E. Tezduyar · Spenser McIntyre · Nikolay Kostov · Ryan Kolesar · Casey Habluetzel

Received: 24 April 2013 / Accepted: 1 June 2013 / Published online: 13 July 2013

Abstract We present the space–time variational multiscale (ST-VMS) computation of wind-turbine rotor and tower aerodynamics. The rotor geometry is that of the NREL 5MW offshore baseline wind turbine. We compute with a given wind speed and a specified rotor speed. The computation is challenging because of the large Reynolds numbers and rotating turbulent flows, and computing the correct torque requires an accurate and meticulous numerical approach. The presence of the tower increases the computational challenge because of the fast, rotational relative motion between the rotor and tower. The ST-VMS method is the residual-based VMS version of the Deforming-Spatial-Domain/Stabilized ST (DSD/SST) method, and is also called "DSD/SST-VMST" method (i.e., the version with the VMS turbulence model). In calculating the stabilization parameters embedded in the method, we are using a new element length definition for the diffusion-dominated limit. The DSD/SST method, which was introduced as a general-purpose moving-mesh method for computation of flows with moving interfaces, requires a mesh update method. Mesh update typically consists of moving the mesh for as long as possible and remeshing as needed. In the computations reported here, NURBS basis functions are used for the temporal representation of the rotor motion, enabling us to represent the circular paths associated with that motion exactly and specify a constant angular velocity corresponding to the invariant speeds along those paths. In addition, temporal NURBS basis functions are used in representation of the motion and deformation of the volume meshes computed and also in remeshing. We name this "ST/NURBS Mesh Update Method (STNMUM)." The STNMUM increases computational efficiency in terms of computer time and storage, and computational flexibility in terms of being able to change the time-step size of the computation. We use layers of thin elements near the blade surfaces, which undergo rigid-body motion with the rotor. We compare the results from computations with and without tower, and we also compare using NURBS and linear finite element basis functions in temporal representation of the mesh motion.

Keywords Space–time VMS method · DSD/SST-VMST · Wind-turbine rotor and tower aerodynamics · Mesh motion · Remeshing · Temporal NURBS functions · ST/NURBS Mesh Update Method · STNMUM

K. Takizawa
Department of Modern Mechanical Engineering and Waseda Institute for Advanced Study, Waseda University, 1-6-1 Nishi-Waseda, Shinjuku-ku, Tokyo 169-8050, Japan

T. E. Tezduyar (✉) · S. McIntyre · N. Kostov · R. Kolesar · C. Habluetzel
Mechanical Engineering, Rice University, MS 321, 6100 Main Street, Houston, TX 77005, USA
e-mail: tezduyar@rice.edu

1 Introduction

The Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) formulation [1–8] is a general-purpose moving-mesh method for computation of flows with moving interfaces. Its stabilization components are the Streamline-Upwind/Petrov-Galerkin (SUPG) [9] and Pressure-Stabilizing/ Petrov-Galerkin (PSPG) [1,10] methods. The ST variational multiscale (ST-VMS) method [7] is a shorter name for the VMS version of the DSD/SST method, which was originally called "DSD/SST-VMST" (i.e. the version with the VMS turbulence model) in [6]. The VMS components are from the residual-based VMS method given in [11–14]. The original DSD/SST formulation was named "DSD/SST-SUPS" in [6] (i.e. the version with the SUPG/PSPG stabilization), which was also called "ST-SUPS" in [8].

The Arbitrary Lagrangian–Eulerian (ALE) finite element formulation [15] is the most commonly used moving-mesh approach in computation of flow problems with moving interfaces, including fluid–structure interaction (FSI) problems (see, for example, [16–34]). However, the DSD/SST formulation has also been applied to some of the most challenging moving-interface problems, including FSI (see, for example, [5,7,8,35–49] and references therein).

An ST method will naturally involve more computational cost per time step than an ALE method. However, as the stability and accuracy analysis reported in [6,7,50] for the DSD/SST formulation of the advection equation shows, versions of the DSD/SST method with higher-order basis functions in time have also higher-order accuracy in time. Consequently, the DSD/SST versions with higher-order basis functions in time can attain the desired accuracy with larger time steps. This, in some cases, might make those higher-order versions more computationally efficient than the lower-order ones. Considerations for parallel-computing efficiency also make the higher-order versions more favorable, because increasing the computational cost per time step is better for parallel efficiency than increasing the number of time steps. Furthermore, when higher-order spatial basis functions (such as NURBS [18,21,51,52]) are used, it is much more effective to use also higher-order temporal basis functions. The DSD/SST method motivated a number of ideas and studies [53–57] on increasing the computational efficiency in iterative solution of the large, coupled linear equation systems encountered in every nonlinear iteration of every time step of a DSD/SST computation.

Moving-mesh methods require mesh update methods. Mesh update typically consists of moving the mesh for as long as possible and remeshing as needed. With the key objectives being to maintain the element quality near solid surfaces and to minimize frequency of remeshing, a number of advanced mesh update methods [5,35,40,58,59] were developed in conjunction with the DSD/SST method, including those that minimize the deformation of the layers of small elements placed near solid surfaces. The ST context, with higher-order functions in time, gives us more effective ways of mesh moving and remeshing (see [42,45–47,60]). Temporal versions of the NURBS basis functions are very good choices for higher-order functions in time (see [42,45–47,60]).

The DSD/SST method was earlier applied to flows involving two objects in fast, rotational relative motion. This was accomplished with the Shear–Slip Mesh Update Method (SSMUM) [37,61,62]. The SSMUM was first introduced for computation of flow around two high-speed trains passing each other in a tunnel (see [37]). The challenge was to accurately and efficiently update the meshes used in computations based on the DSD/SST formulation and involving two objects in fast, linear relative motion. The idea behind the SSMUM was to restrict the mesh moving and remeshing to a thin layer of elements between the objects in relative motion. The mesh update at each time step can be accomplished by a "shear" deformation of the elements in this layer, followed by a "slip" in node connectivities. The slip in the node connectivities, to an extent, un-does the deformation of the elements and results in elements with better shapes than those that were shear-deformed. Because the remeshing consists of simply re-defining the node connectivities, both the projection errors and the mesh generation cost are minimized. A few years after the high-speed train computations, the SSMUM was implemented for objects in fast, rotational relative motion and applied to computation of flow past a rotating propeller [61] and flow around a helicopter with its rotor in motion [62].

The ST-SUPS computation of wind-turbine rotor aerodynamics was first reported in [28], followed by the first ST-VMS computation in [63], and numerical performance studies for ST-SUPS and ST-VMS computation of rotor aerodynamics in [64]. The rotor geometry was that of the NREL 5MW offshore baseline wind turbine. All this work, which was reviewed in [8,33], was done as part of a collaboration with Bazilevs et al. , who, starting with what they reported also in [28], have done the most extensive work in modeling of wind-turbines, including the FSI and tower effects (see [8,29,33,65–70]). To handle the interaction between the rotor and tower, in [68–70] the authors used a sliding-interface approach, which was first proposed in [52]. This technique requires no remeshing, however, the presence of sliding interfaces, where the kinematics and tractions are imposed weakly, requires additional computer implementation.

The DSD/SST formulation, like most stabilized formulations, involves stabilization parameters that play an important role in determining the accuracy of the formulation. For the ST-SUPS method, these stabilization parameters are called τ_{SUPG}, τ_{PSPG} and ν_{LSIC}, the last one being the stabilization parameter embedded in the "least squares on incompressibility constraint (LSIC)" stabilization. For the ST-VMS method, they are called τ_{M} (τ_{SUPS} in [8,33,44]) and ν_{C} (ν_{LSIC} in [8,33,44]). There are various ways of defining the stabilization parameters (see, for example, [1,4,5,71–85]). The ones used with the DSD/SST formulation in recent years have mostly been those given in [4,5]. They involve two different element length definitions. For the advection-dominated limit it is h_{UGN} (originating from [71]), and for the diffusion-dominated limit h_{RGN} (originating from [4]).

The ST-VMS computations of wind-turbine rotor aerodynamics in [63] included tests with two definitions of ν_{LSIC}: the one given by Eq. (17) in [6] (originating from Eq. (17) in [5]), which is called "TC2," and the one given by Eq. (18) in [6] (originating from [14]), which is called "TGI." The computations in [64] involved a third definition, given by Eq. (1) in [64], which is called "LHC." The computations in [63] showed that both better subgrid scale modeling and better mesh refinement (which is what [63] had compared to [28]) were important in obtaining correct torque values in this class of problems. For that reason, the rotor aerodynamics computations in [64] included extensive mesh refinement studies. In this paper, we introduce a new element length definition for the diffusion-dominated limit, which posseses better stabilization features and which we will call "h_{RGNT}."

The DSD/SST computations in [28,63,64] did not include a wind-turbine tower, and therefore a mesh update method was not required. The presence of a tower in our computations here requires a mesh update method that can handle the fast, rotational relative motion between the rotor and tower. The SSMUM would have been one option, but we decided to use a mesh update method that is more general. We use NURBS basis functions for the temporal representation of the rotor motion, mesh motion and also in remeshing. This is essentially the same computational technology used in the ST-VMS computations of flapping-wing aerodynamics reported in [42,45–47]. We name it "ST/NURBS Mesh Update Method (STNMUM)" in this paper.

The rotor surface geometry is a NURBS surface, generated by starting from the quadratic NURBS patches that were created by Bazilevs et al. [28] and generating a quadratic NURBS surface with G^2 and G^1 continuity between the patches around and along the blade, respectively. The motion of the rotor surface mesh created from the NURBS geometry is represented by quadratic temporal NURBS basis functions, with sufficient number of temporal patches for one rotation. This enables us to represent the circular paths associated with the rotor motion exactly and, with a "secondary mapping" [6–8,42], specify a constant angular velocity corresponding to the invariant speeds along those paths.

Given the motion of the surface mesh, we compute meshes that serve as temporal-control points. This is done by creating with an automatic mesh generator a new mesh at the central control point of the temporal patch, and computing the meshes at the other two control points by using the mesh moving technique [5,35,40,58,59] mentioned earlier. The STNMUM allows us to do mesh computations with longer time in between, but get the mesh-related information for each ST slab, such as the coordinates and their time derivatives, from the temporal representation whenever we need. This approach where the mesh-related information is computed "directly" will be called in this paper "Direct Temporal Representation (DTR)." As an alternative approach, we can get the mesh-related data after first computing the finite element meshes associated with each ST slab by interpolation from the temporal NURBS representation of the mesh. We will call this approach "Interpolated-Mesh Temporal Representation (IMTR)." For better mesh resolution, we use layers of thin elements near the blade surfaces. These layers of elements are created with a special mesh generation process and are not part of what we create with the automatic mesh generation process. They undergo rigid-body motion with the rotor.

We perform the ST-VMS computations of the wind-turbine rotor and tower aerodynamics with a given wind speed and a specified rotor speed. We test both the DTR and IMTR approaches. For comparison purposes, we compute the rotor aerodynamics also without the tower. The rotation representation with constant angular velocity and the new element length definition for the diffusion-dominated limit are given in Sect. 2. The rotor and tower geometries are described in Sect. 3. The problem setup, mesh generation and computations are presented in Sect. 4. The concluding remarks are given in Sect. 5.

2 Computational techniques

2.1 Rotation representation with constant angular velocity

We use quadratic NURBS functions, as described in [6–8,42], to represent a circular arc. We discretize time and position as follows:

$$t = \sum_{\alpha=1}^{n_{\mathrm{ent}}} T^{\alpha}(\Theta_t(\theta))t^{\alpha}, \tag{1}$$

$$\mathbf{x} = \sum_{\alpha=1}^{n_{\mathrm{ent}}} T^{\alpha}(\Theta_x(\theta))\mathbf{x}^{\alpha}. \tag{2}$$

Here n_{ent} is the number of temporal element nodes, T^{α} is the basis function, $\Theta_t(\theta)$ and $\Theta_x(\theta)$ are the secondary mappings for time and position, and t^{α} and $\mathbf{x}^{\alpha}$ are the time and position values corresponding to the basis function T^{α}. The basis functions could be finite element or NURBS basis functions. For the circular arc, $n_{\mathrm{ent}} = 3$ and they are quadratic NURBS. The secondary mapping concept above was introduced in [6], and the velocity can be expressed as follows:

$$\frac{d\mathbf{x}}{dt} = \left(\sum_{\alpha=1}^{n_{\mathrm{ent}}} \frac{dT^{\alpha}}{d\Theta_x}\frac{d\Theta_x}{d\theta}\mathbf{x}^{\alpha}\right)\left(\sum_{\alpha=1}^{n_{\mathrm{ent}}} \frac{dT^{\alpha}}{d\Theta_t}\frac{d\Theta_t}{d\theta}t^{\alpha}\right)^{-1}, \tag{3}$$

leading to

$$\frac{d\mathbf{x}}{dt} = \left(\sum_{\alpha=1}^{n_{\mathrm{ent}}} \frac{dT^{\alpha}}{d\Theta_x}\mathbf{x}^{\alpha}\right)\left(\sum_{\alpha=1}^{n_{\mathrm{ent}}} \frac{dT^{\alpha}}{d\Theta_t}t^{\alpha}\right)^{-1}\left(\frac{d\Theta_x}{d\theta}\frac{d\theta}{d\Theta_t}\right). \tag{4}$$

Thus, the speed along the path can be specified only by modifying the secondary mapping. For a circular arc, two methods were introduced in [7,42] and also described in [8]; one is modifying the secondary mapping for position and the other one is modifying both such that $\frac{dt}{d\theta}$ is constant. We note that, in theory, the secondary mapping selections do not make any difference as long as the relationship $\frac{d\Theta_x}{d\Theta_t}$ is the same.

In our implementation, to keep the process general, we search for the parametric coordinate θ by using an iterative solution method [7,8,42]. We use the latter set of the secondary mappings, having constant $\frac{dt}{d\theta}$.

Remark 1 When we use a secondary mapping for discretization of unknowns, the selection of the mappings affects the numerical integration accuracy in the physical domain.

For the IMTR, we find the parametric coordinate corresponding to each time level and interpolate the position to obtain the corresponding mesh. For the DTR, we first calculate time corresponding to each integration point, including the time step size because of the jump term, and then calculate Θ_x and Θ_t to interpolate the position and velocity from Eqs. (2) and (4).

2.2 Element length definition for the diffusion-dominated limit

The element length definition for the diffusion-dominated limit is used in calculating the diffusion-dominated limit of the stabilization parameters τ_{SUPG}, τ_{PSPG} and τ_{M} $(= \tau_{\mathrm{SUPS}})$, and, directly or indirectly (through τ_{SUPS}), in calculating all options of ν_{LSIC} and ν_{C} $(= \nu_{\mathrm{LSIC}})$ except for TGI. That includes the option that was defined in [64] as a component of the LHC version, which was named in [8] "HRGN":

$$\nu_{\mathrm{LSIC-HRGN}} = \frac{h_{\mathrm{RGN}}^2}{\tau_{\mathrm{SUPS}}}. \tag{5}$$

The element length definition introduced in [4] for the diffusion-dominated limit is expressed as follows:

$$h_{\mathrm{RGN}} = 2\left(\sum_{a=1}^{n_{\mathrm{en}}} |\mathbf{r}\cdot\nabla N_a|\right)^{-1}, \tag{6}$$

where n_{en} is the number of ST element nodes, N_a is the ST basis functions associated with the ST node a, and

$$\mathbf{r} = \frac{\nabla\|\mathbf{u}^h\|}{\left\|\nabla\|\mathbf{u}^h\|\right\|} \tag{7}$$

represents the solution gradient. Here $\nabla\|\mathbf{u}^h\|$ is calculated as

$$\nabla\|\mathbf{u}^h\| = \nabla\left(\left(\left\|\mathbf{u}^h\right\|^2\right)^{\frac{1}{2}}\right) = \frac{1}{2\|\mathbf{u}^h\|}\nabla\left(\left\|\mathbf{u}^h\right\|^2\right), \tag{8}$$

$$\nabla\|\mathbf{u}^h\| = \nabla\mathbf{u}^h\cdot\frac{\mathbf{u}^h}{\|\mathbf{u}^h\|}, \tag{9}$$

resulting in

$$\mathbf{r} = \frac{\nabla\mathbf{u}^h\cdot\mathbf{u}^h}{\|\nabla\mathbf{u}^h\cdot\mathbf{u}^h\|}. \tag{10}$$

This expression becomes ill defined when $\nabla\mathbf{u}^h\cdot\mathbf{u}^h = \mathbf{0}$.

We introduce a new element length definition for the diffusion-dominated limit:

$$h_{\mathrm{RGNT}} = \left(\sum_{i=1}^{n} w_i^2\frac{1}{h_i^2}\right)^{-\frac{1}{2}}, \tag{11}$$

where $h_i > 0$ is element length for the i^{th} direction, $w_i \geq 0$ is the weight for that direction, n is the number of directions, and

$$\sum_{i=1}^{n} w_i = 1. \tag{12}$$

Equation (11) is well defined if the element length for at least one of the directions is nonzero.

We define the element length and weight for each of the n directions based on an $n_{\mathrm{sd}}\times n$ tensor $\mathbf{R}$:

$$h_i = 2\,\|\mathbf{R}_i\|\left(\sum_{a=1}^{n_{\mathrm{en}}}|\mathbf{R}_i\cdot\nabla N_a|\right)^{-1}, \tag{13}$$

$$w_i = \frac{\|\mathbf{R}_i\|}{\|\mathbf{R}\|}, \tag{14}$$

where n_{sd} is the number of space dimensions, $\mathbf{R}_i$ is the i^{th} column vector of $\mathbf{R}$, the vector norm is L^2, and the matrix norm is Frobenius. From Eqs. (11), (13) and (14) we obtain

$$h_{\mathrm{RGNT}} = 2\,\|\mathbf{R}\|\left(\sum_{i=1}^{n}\left(\sum_{a=1}^{n_{\mathrm{en}}}|\mathbf{R}_i\cdot\nabla N_a|\right)^2\right)^{-\frac{1}{2}}, \tag{15}$$

which is well defined if at least one of the column vectors is nonzero. We propose to define the tensor $\mathbf{R}$ as

$$\mathbf{R}_i = \nabla u_i, \tag{16}$$

with $n = n_{\mathrm{sd}}$. If all column vectors are zero, which implies uniformness in the flow field, to make the expression for h_{RGNT} well defined, we define $\mathbf{R}$ as

$$\mathbf{R}_i = \nabla N_i, \tag{17}$$

with $n = n_{\mathrm{en}}$. The following is a brief justification for Eq. (17). Suppose where there is uniformness in the flow field we change only one coefficient, $(u_b)_i$, which means that the solution gradient is in the ∇N_b direction. Therefore, it is reasonable to assume that there is an equal chance of having a solution gradient in all ∇N_b directions. Therefore we use all those directions as the column vectors of $\mathbf{R}$. As an alternative to the definition given by Eq. (17), we propose

$$\mathbf{R}_i = \nabla N_i - \left(\frac{\mathbf{u}^h}{\|\mathbf{u}^h\|} \cdot \nabla N_i \right) \frac{\mathbf{u}^h}{\|\mathbf{u}^h\|}, \tag{18}$$

and this is based on the assumption that there is an equal chance of having a solution gradient in all ∇N_b directions perpendicular to $\mathbf{u}^h$. In this alternative option, $\mathbf{u}^h = \mathbf{0}$ would revert the definition back to the one given by Eq. (17).

With the new element length h_{RGNT} for the diffusion-dominated limit, the HRGN option of ν_{LSIC} becomes

$$\nu_{\text{LSIC-HRGN}} = \frac{h_{\text{RGNT}}^2}{\tau_{\text{SUPS}}}. \tag{19}$$

3 Geometry construction for the wind-turbine rotor blade, hub, and tower

The geometry construction for the wind-turbine rotor blade and hub we are using in the computations was described in [28,63], and also partially in [64]. For completeness we repeat some of that information here. The geometry of the rotor blade is based on the NREL 5MW offshore baseline wind turbine reported in [86]. A 61 m blade is attached to a hub with radius of 2 m, making the total rotor radius, R, 63 m. The blade is composed of several airfoil types. The first portion of the blade is a perfect cylinder. Farther away from the root the cylinder is smoothly blended into a series of DU (Delft University) airfoils. Starting at 44.55 m from the root and all the way to the tip, the NACA64 profile is used. For each cross-section, we use quadratic NURBS to represent the 2D airfoil shape. The weights of the NURBS functions are set to unity. The weights are adjusted near the root to represent the circular cross-sections exactly. The cross-sections are lofted along the blade axis direction, also using quadratic NURBS and unit weights. This geometry-construction process yields a smooth blade surface with a relatively small number of input parameters, which is an advantage of the isogeometric representation. Images of the airfoil types used in the wind-turbine rotor blade and the final blade including the twisting cross-sections can be found in [28,63,64].

The tower geometry was created based on the tower design specified for the NREL 5MW offshore baseline wind turbine, which describes a circular tower with a height of 87.6 m, a base diameter of 6 m, and a top diameter of 3.87 m. This geometry was generated by lofting between NURBS curves for the top and base of the tower. The rotor axis is 90° to the tower, and there is no tilt or precone. The distance between the tower axis and the point where the three blade axes intersect is 5 m. For most of the blade, the clearance from the tower is in the range 2.3 m to 2.8 m.

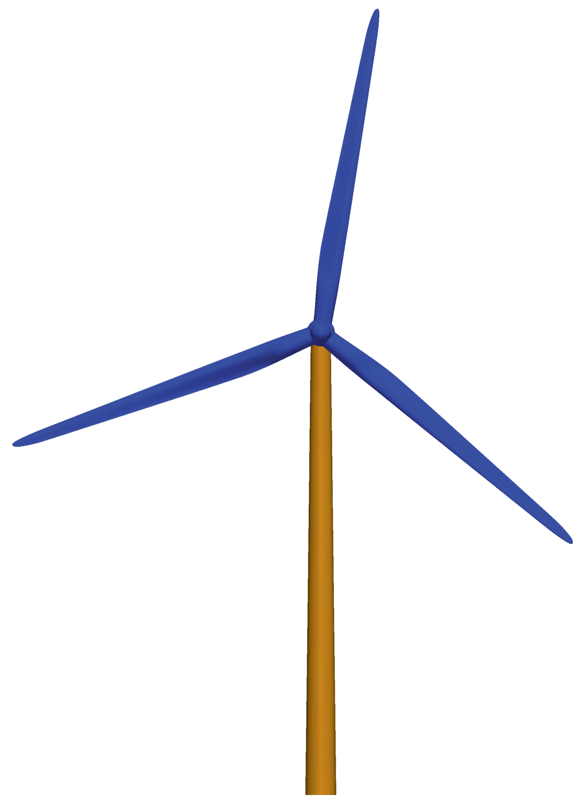

Fig. 1 Wind-turbine rotor and tower geometries

4 Computations

4.1 Problem setup

We compute the aerodynamics of the rotor with and without its tower for a given rotor shape and wind speed and a specified rotor speed. The rotor and tower geometries are shown in Fig. 1.

The wind speed is uniform at 9 m/s and the rotor speed is 1.08 rad/s, giving a tip speed ratio of 7.55 (see [87] for wind-turbine terminology). We use air properties at standard sea-level conditions. The Reynolds number (based on the chord length at $\frac{3}{4}R$ and the relative velocity there) is approximately 12 million. At the inflow boundary the velocity is set to the wind velocity, at the outflow boundary the stress vector is set to zero, and at the top, side, and bottom boundaries slip conditions are imposed.

4.2 Rotor motion

The circular turbine rotation is represented with temporal NURBS basis functions and secondary mapping, described in Sect. 2.1. Because the 3 blades of the turbine are 120° apart, rotational geometric periodicity is used such that a full 360° rotation is defined by 3 identical 120° segments. Each 120° segment is divided into 6 patches to keep the

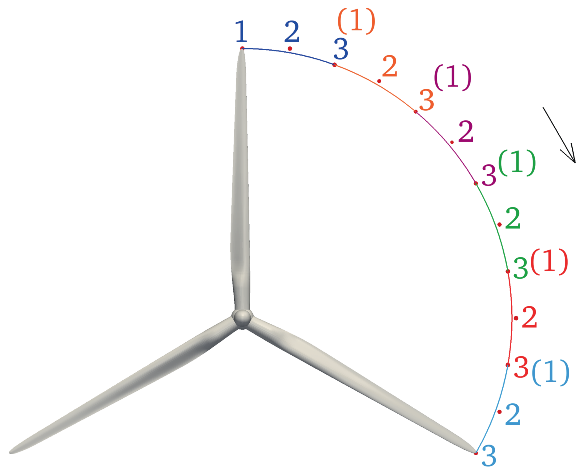

Fig. 2 Path of a blade tip with temporal patches and control point numbering local to each patch. A control point at the start of a patch and colocated with a control point at the end of the previous patch is in parentheses. For the color code, see Table 1

Table 1 Figure 2 color code for the temporal patches

Temporal patch	Fig. 2 color
1	Blue
2	Orange
3	Purple
4	Green
5	Red
6	Teal

mesh distortion under control. Each patch is a 20° arc, with 3 temporal-control points. The 6 temporal patches and their control points are illustrated in Fig. 2 and Table 1.

4.3 Surface mesh

The rotor surface mesh is generated by discretizing the NURBS surface geometry at each knot intersection, subdividing the knot spans into quadrilateral finite elements in a structured way, and subdividing the quadrilateral elements into two triangles. Small adjustments are made to improve the mesh near the hub. The surface mesh position is calculated at each temporal-control point shown in Fig. 2. Figure 3 shows the rotor surface at the three temporal-control points of the first patch. We note that control points 1 and 3 lie on the path traveled by the points on the blades and a portion of the hub at the start and end of the 20° rotation, but control point 2 lies outside the circular arc. This means that the temporal-control mesh 2 is deformed compared to the temporal-control meshes 1 and 3. A temporal-control mesh 2 has to be generated for the part of the surface between the hub cross-sections rotating with the blades and fixed to the tower. The tower surface mesh is generated

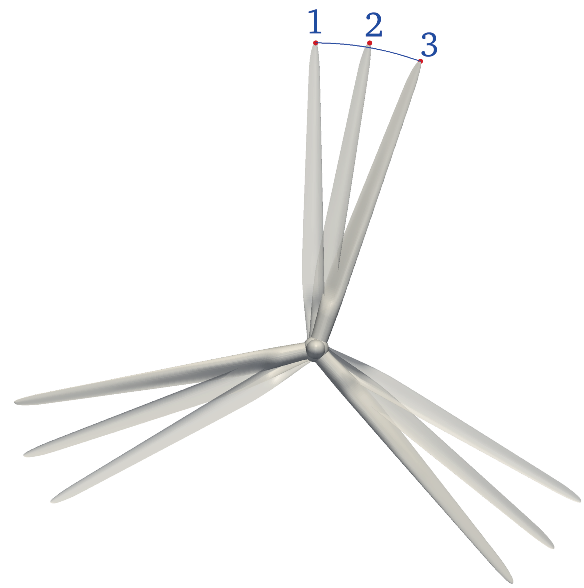

Fig. 3 Rotor surface at the three temporal-control points of the first patch

from the NURBS representation of the surface by using an unstructured triangular mesh generator and matched with the previously generated hub mesh at the intersection. The rotor surface mesh consists of 34,087 nodes and 68,112 triangles. The tower surface mesh consists of 6,952 nodes and 13,806 triangles.

4.4 Volume mesh

4.4.1 Boundary-layer mesh

The layers of thin elements near the blades are generated by extruding the NURBS surface geometry into NURBS volume representation, subdividing the knot spans into hexahedral finite elements in a structured way, and subdividing the hexahedral elements into six tetrahedral elements. The resulting boundary-layer mesh for each blade consists of 4 layers with a first-layer thickness of about 2.85×10^{-2} m and a total thickness of about 2.85×10^{-1} m, 52 nodes in the circumferential direction around the blade, and approximately 145 nodes in the longitudinal direction. The tower boundary-layer mesh is generated by extruding the tower surface mesh to layers of prismatic elements, which are then subdivided into 3 tetrahedral elements each. It consists of 4 layers, with a first-layer thickness of 2.85×10^{-2} m and a total thickness of 3.0×10^{-1} m. The blade and tower boundary-layer meshes do not undergo any mesh deformation. This maintains the mesh quality in the boundary-layer regions. Figure 4 illustrates the outer surface of the blade boundary-layer mesh and cutplanes showing the tower and blade boundary-layer meshes.

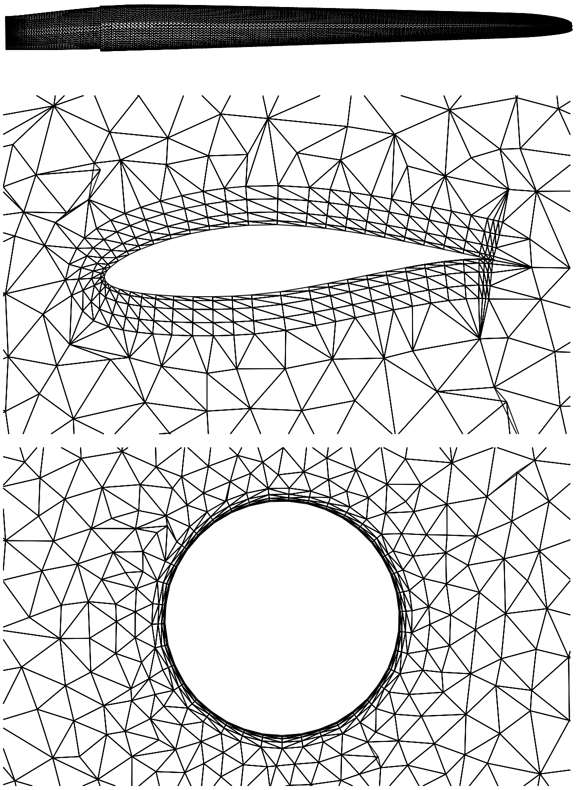

Fig. 4 *Top* Outer surface of blade and boundary-layer mesh. *Middle* Boundary-layer mesh at $\frac{3}{4}R$. *Bottom* Tower boundary-layer mesh

4.4.2 Overall mesh

Three different meshes are used in the computations: Mesh 1, Mesh 2, and Mesh 3. Mesh 2 has both the rotor and the tower, with boundary-layer mesh only for the blades. Mesh 1 has only the rotor, and is identical to Mesh 2 except the tower is filled with volume elements. Mesh 3 has both the rotor and the tower, with boundary-layer mesh for both the blades and the tower, and a mesh refinement region downstream of the tower. All three meshes have an outer, coarser region, with an inner cylindrical refinement region surrounding the rotor. This inner refinement region includes most of the tower for Mesh 2 and Mesh 3, and the mesh refinement region downstream of the tower for Mesh 3. Figure 5 illustrates, as an example, cut planes of Mesh 3, and Fig. 6 shows zoomed longitudinal cut planes of all three meshes. The inflow and outflow boundaries are at $3.79R$ and $10.35R$ from the hub center, respectively. The side, top, and bottom boundaries are at $2.29R$, $3.17R$, and $1.43R$, respectively (see Fig. 5). The volume mesh is generated once per patch using an automatic mesh generator (a total of 6 times). The mesh is generated at control point 2 of each patch to minimize mesh distortion between control points. We note that only the mesh in the inner cylindrical refinement region surrounding the rotor is generated for each patch. The outer, coarser mesh is generated only once, and is kept the same when the inner meshes are generated for each patch. The mesh moving

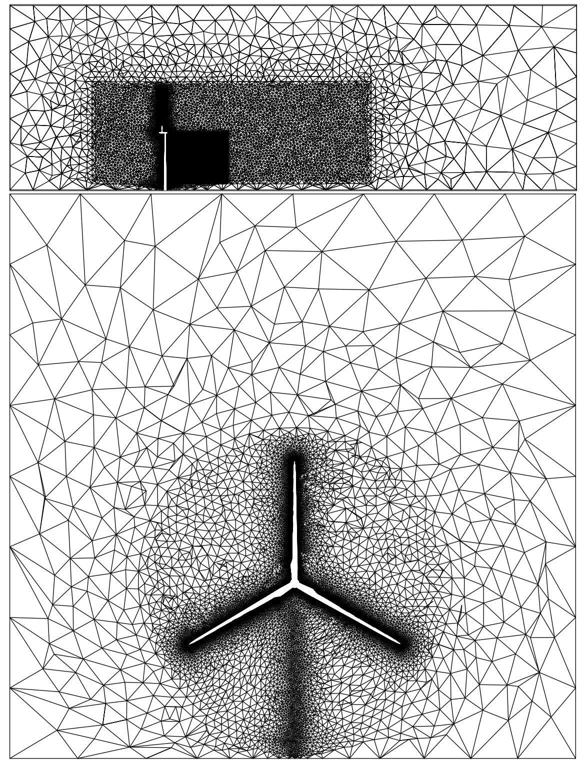

Fig. 5 Cut planes of temporal-control mesh 1 of patch 1 for Mesh 3

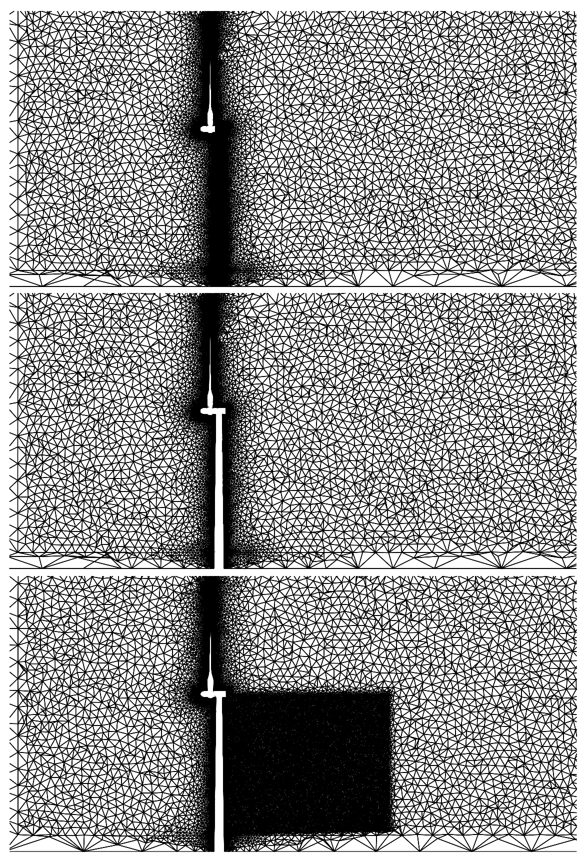

Fig. 6 Zoomed cut planes of temporal-control mesh 1 of patch 1 for Mesh 1 (*top*), Mesh 2 (*middle*), and Mesh 3 (*bottom*)

Table 2 Number of nodes (nn) and elements (ne) for the fluid mechanics meshes used in each temporal patch

Mesh	Temporal patch	nn	ne
1	1	470,880	2,725,614
1	2	466,983	2,701,657
1	3	460,932	2,665,562
1	4	462,733	2,676,747
1	5	464,712	2,687,745
1	6	468,529	2,711,069
2	1	446,709	2,553,100
2	2	442,876	2,529,556
2	3	436,825	2,493,524
2	4	438,802	2,505,789
2	5	440,870	2,517,233
2	6	444,517	2,539,512
3	1	598,125	3,454,865
3	2	596,111	3,442,699
3	3	592,345	3,420,273
3	4	590,628	3,410,226
3	5	595,719	3,440,031
3	6	596,522	3,445,407

technique [5,35,40,58,59] mentioned earlier is used to compute the mesh position for control points 1 and 3. The outer surfaces of the boundary-layer meshes serve as the boundaries where we specify the inner boundary conditions for the mesh motion. The external boundaries of the computational domain serve as the boundaries where we specify the outer boundary conditions, with zero displacement. In the elasticity equations of the mesh moving technique, a Young's modulus of 1.0, a Poisson's ratio of −0.20, and a Jacobian-based stiffening exponent of 1.5 are used. We use 1,500 GMRES [88] iterations for each step of the mesh motion, with diagonal preconditioner. Each 10° range of motion is computed over 40 steps. Number of nodes and elements for all 6 temporal patches of the 3 volume meshes are given in Table 2.

4.5 Computational conditions

In the ST-VMS computations, $\tau_{\mathrm{M}} (= \tau_{\mathrm{SUPS}})$ comes from the τ_{SUPG} definition in [4], specifically the definition given by Eqs. (107)–(109) in [4], which can also be found as the definition given by Eqs. (7)–(9) in [5], with $h_{\mathrm{RGN}} (= h_{\mathrm{RGNT}})$ given by Eq. (15). For $\nu_{\mathrm{C}} (= \nu_{\mathrm{LSIC}})$, we use the $\nu_{\mathrm{LSIC-HRGN}}$ definition given by Eq. (19). The DTR and IMTR approaches are used on all three meshes. Least-squares projection is used to interpolate the velocity and pressure between temporal patches. Because the boundary-layer meshes and the tower and rotor surface meshes remain identical between temporal patches, the velocity values are transferred exactly for those nodes. The computations performed are summarized in Table 3.

Table 3 Summary of the computations

Mesh	Tower	Temporal representation
1	No	DTR
2	Yes	DTR
3	Yes	DTR
1	No	IMTR
2	Yes	IMTR
3	Yes	IMTR

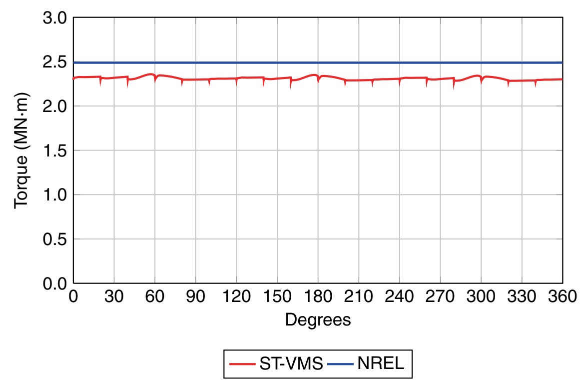

Fig. 7 Torque for Mesh 1 with the DTR approach, compared with the NREL data

The time-step size is 2.23×10^{-3} s (145 time steps per patch), with 4 nonlinear iterations per time-step. First we develop the flow field for 500 time steps while the rotor is static, ramping up the inflow velocity during the first 300 steps from zero to the wind speed using a cosine ramp. During this flow-development stage of the computation, we use 150, 150, 200, and 400 GMRES iterations for the 4 nonlinear iterations. In computations with the rotor in motion, we use 150, 150, 200, and 400 GMRES iterations for Mesh 1, and 150, 250, 350, and 500 GMRES iterations for Mesh 2 and Mesh 3. With the GMRES iterations in flow computations, we use nodal-block-diagonal preconditioner. The mesh is partitioned based on the METIS algorithm [89] to improve parallel efficiency in the computations.

4.6 Results

Figure 7 shows the torque for Mesh 1 with the DTR approach, for the last 360° rotation of a blade, with the rotation amount measured from the orientation seen in Fig. 2. For reference purposes, Fig. 7 includes the NREL data for the 5MW offshore wind turbine reported in [86]. The torque is within 8 % of the NREL data.

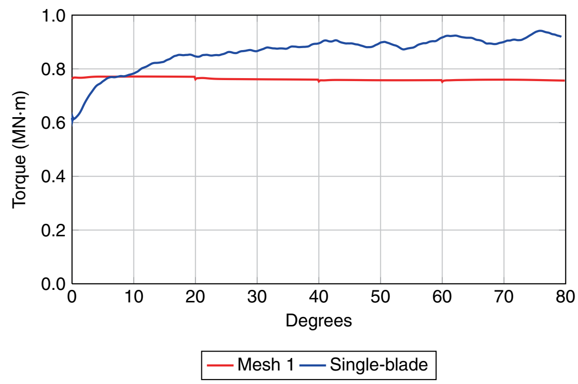

Fig. 8 Torque for a single blade of Mesh 1 with the DTR approach, compared with the torque from an earlier single-blade computation using the TGI option of $\nu_{\mathrm{C}} (= \nu_{\mathrm{LSIC}})$

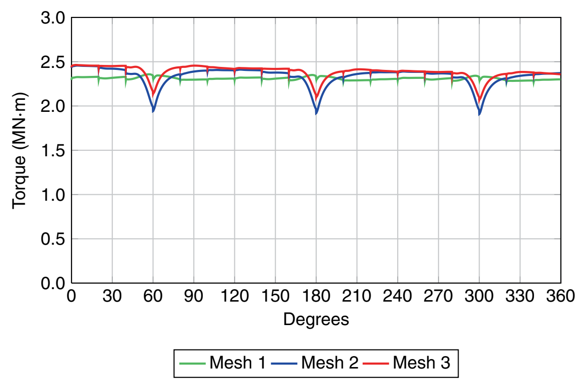

Fig. 9 Torque for Mesh 1, Mesh 2 and Mesh 3 with the DTR approach

Figure 8 shows the torque for the last 80° rotation of a single blade of Mesh 1 with the DTR approach, compared with the torque from an earlier single-blade computation using the TGI option of ν_{C} ($= \nu_{\mathrm{LSIC}}$). The single-blade computation has the same blade geometry, wind speed, and rotor speed, but has a single-blade mesh in a rotationally-periodic domain. It has a more refined boundary-layer mesh and a time-step size that is approximately 5 times smaller. The higher torque seen for the single-blade computation may be due to the fact that the computation was carried out for a much shorter duration, only 80° of rotation versus 1,080° for the Mesh 1 computation. Therefore the current computation likely represents a more settled torque value.

Figures 9 and 10 show the torque for all three meshes with the DTR and IMTR approaches. As can be seen from these figures, Mesh 1 (no tower) has a very stable torque, while Mesh 2 and Mesh 3 (with tower) exhibit a significant but expected drop in torque each time a blade passes the tower.

Figure 11 shows, for each of the three meshes, the torque obtained with the DTR and IMTR approaches. The figure illustrates that the DTR and IMTR approaches result in a nearly identical torque magnitude for all 3 meshes.

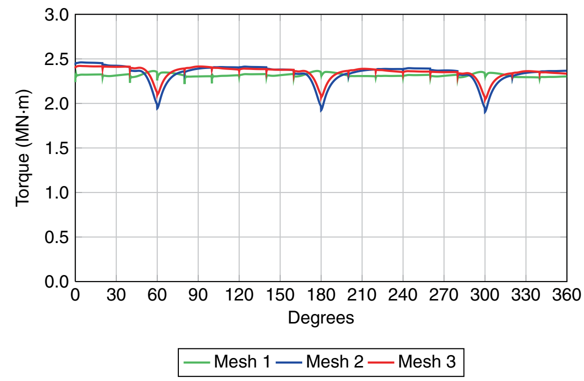

Fig. 10 Torque for Mesh 1, Mesh 2 and Mesh 3 with the IMTR approach

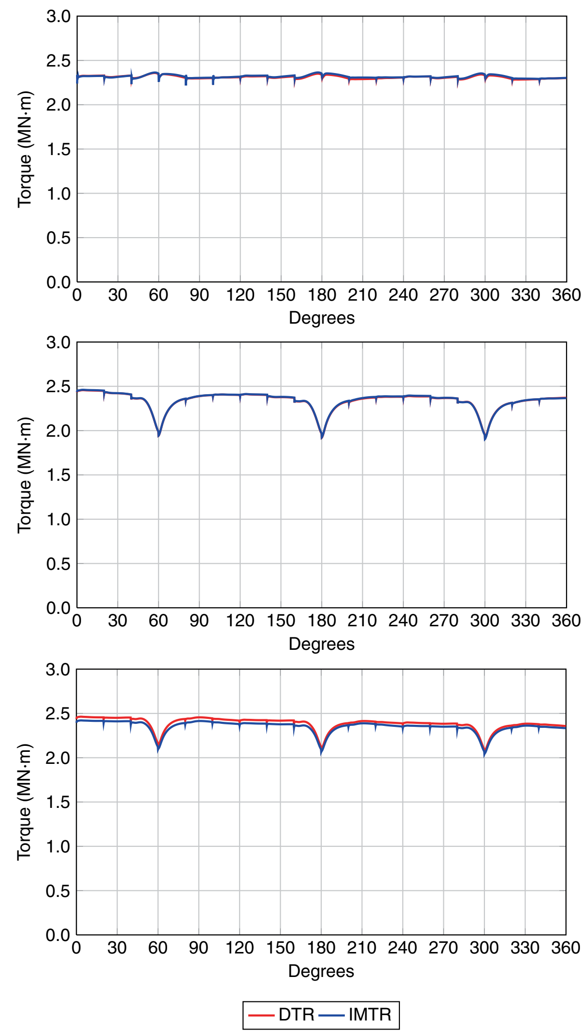

Fig. 11 Torque with the DTR and IMTR approaches for Mesh 1 (*top*), Mesh 2 (*middle*), and Mesh 3 (*bottom*)

Figure 12 shows the torque for Mesh 1 with the DTR approach, using two different time-step sizes: 2.23×10^{-3} s (145 time steps per patch) and 4.49×10^{-3} s (72 time steps per patch). Doubling the time-step size still yields a compa-

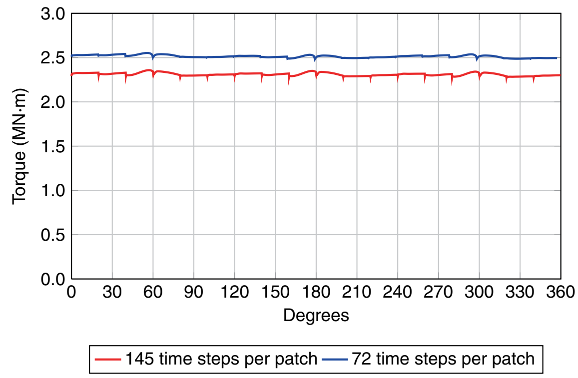

Fig. 12 Torque for Mesh 1 with the DTR approach, using two different time-step sizes: 2.23×10^{-3} s (145 time steps per patch) and 4.49×10^{-3} s (72 time steps per patch)

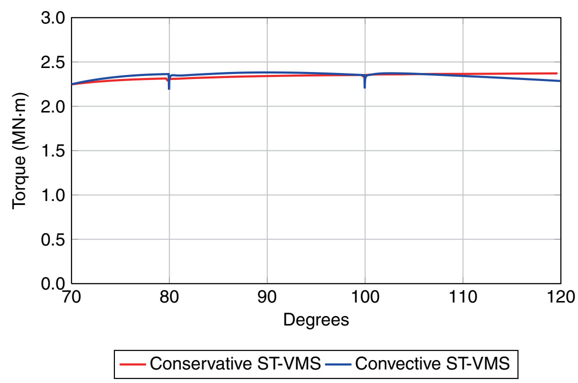

Fig. 13 Torque for Mesh 2 with the DTR approach and the conservative and convective forms of the ST-VMS formulation. The time-step sizes: 4.46×10^{-4} s (725 time steps per patch) for the convective form and 2.23×10^{-3} s (145 time steps per patch) for the conservative form. The torques shown are from the same period in a rotation cycle, but the conservative-form torque is from the last 360° of the computation, and the convective-form torque is from a recently-started, ongoing computation

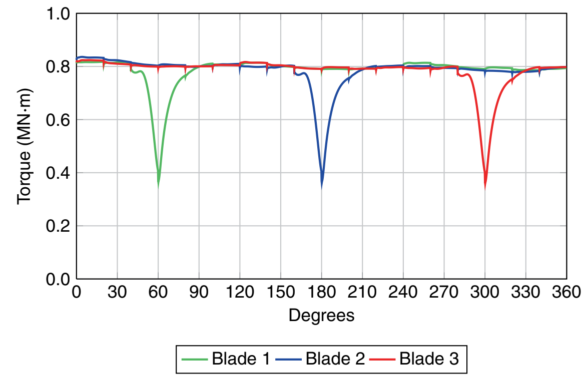

Fig. 14 Torque for the individual blades of Mesh 2 with the DTR approach

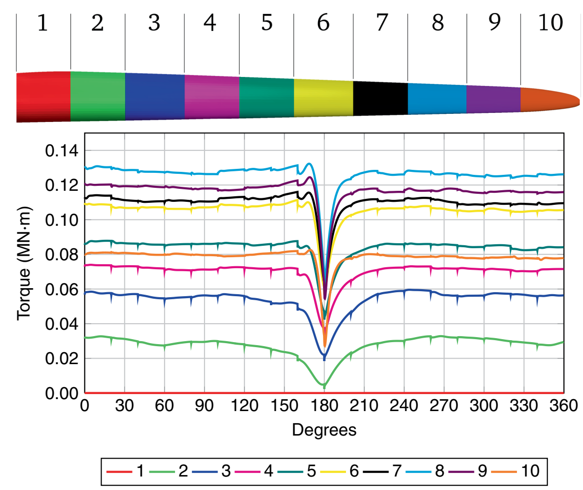

Fig. 15 Torque for 10 equal-length spanwise sections of a blade of Mesh 2 with the DTR approach

rable torque value, within 10 % of the value for the smaller time-step size.

We also carried out a computation with the convective form of the ST-VMS formulation (see Eq. (8.17) in [7]), but with a smaller time-step size: 4.46×10^{-4} s (725 time steps per patch). Figure 13 shows the torque for Mesh 2 with the DTR approach and the conservative and convective forms of the ST-VMS formulation. The conservative-form computation is with the standard time-step size: 2.23×10^{-3} s (145 time steps per patch).

Figure 14 shows the torque for the individual blades of Mesh 2 with the DTR approach. The figure clearly shows the expected torque drop for each blade as it passes the tower, while the other two blades maintain relatively constant torque.

Figure 15 shows the torque for 10 equal-length spanwise sections of a blade of Mesh 2 with the DTR approach. Greatest amount of torque is generated in sections 6–9 of the blade, while section 10 at the tip and the other lower sections generate less torque.

Figure 16 shows a volume rendering of the vorticity for Mesh 2 with the DTR approach. The flow patterns vary considerably along each blade length, illustrating the necessity to carry out the computations in 3D.

Figure 17 shows the pressure coefficient at $0.90R$ for the last 0° orientation of a blade of Mesh 2, with the DTR and IMTR approaches, with the last 0° orientation being common between the two computations. There is very little difference in the pressure coefficient around the blades between the DTR and IMTR approaches.

Figure 18 shows the pressure coefficient at $0.90R$ for the last 180° orientation of a blade of Mesh 1, Mesh 2 and Mesh 3, with the DTR approach, with the last 180° orientation being common between Mesh 2 and Mesh 3 computations.

Table 4 provides the averaged torque for the last 360° rotation in all 6 computations. The values show that the difference in torque between the DTR and IMTR approaches,

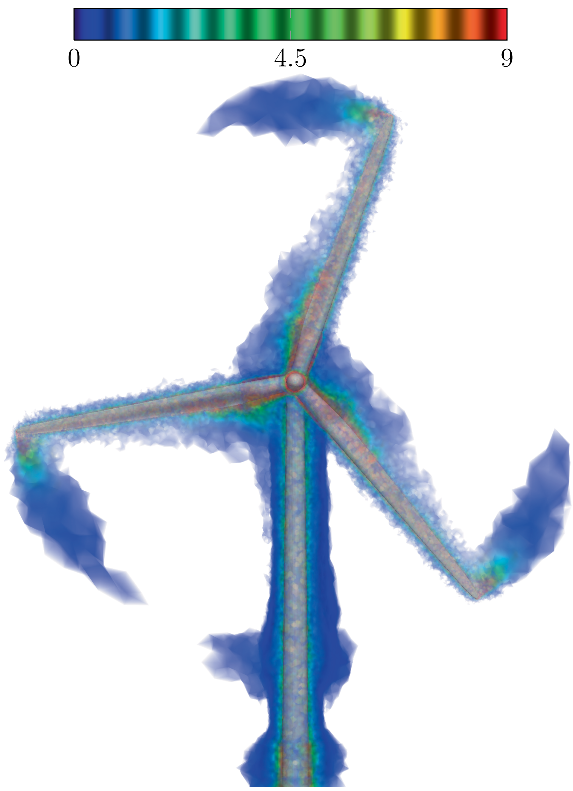

Fig. 16 Volume rendering of the vorticity (in s^{-1}) from the last 360° of the computation for Mesh 2 with the DTR approach

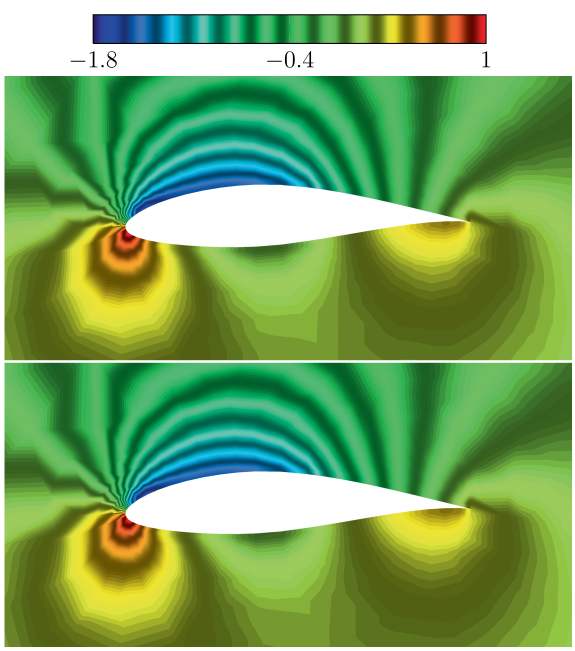

Fig. 17 Pressure coefficient at $0.90R$ for the last 0° orientation of a blade of Mesh 2, with the DTR (*top*) and IMTR (*bottom*) approaches

and between Mesh 2 and Mesh 3, is rather small. The difference in torque between Mesh 1 and Mesh 2 and 3 illustrates effect of the tower.

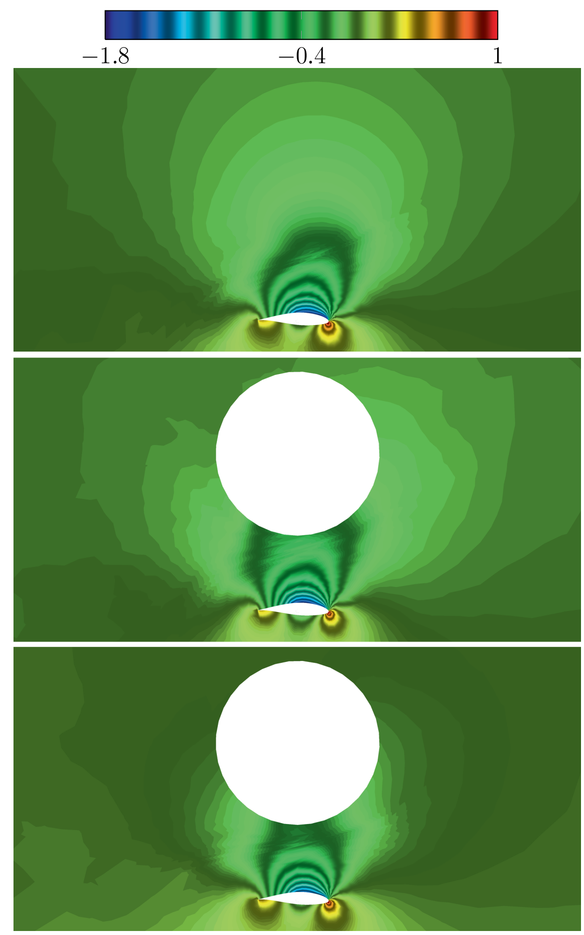

Fig. 18 Pressure coefficient at $0.90R$ for the last 180° orientation of a blade of Mesh 1 (*top*), Mesh 2 (*middle*), and Mesh 3 (*bottom*), with the DTR approach

Table 4 Averaged torque (MN·m) for the last 360° rotation in all 6 computations

Mesh	DTR	IMTR
1	2.31	2.32
2	2.34	2.34
3	2.39	2.35

Figure 19 shows the vorticity around the tower for Mesh 2 and Mesh 3 with the DTR approach. Mesh 3 is able to represent the wake behind the tower far more effectively, although no vortex shedding is observed at this stage of the computation, possibly due to insufficient computing duration or mesh refinement.

5 Concluding remarks

We presented the ST techniques we have developed for computation of wind-turbine rotor and tower aerodynamics. The

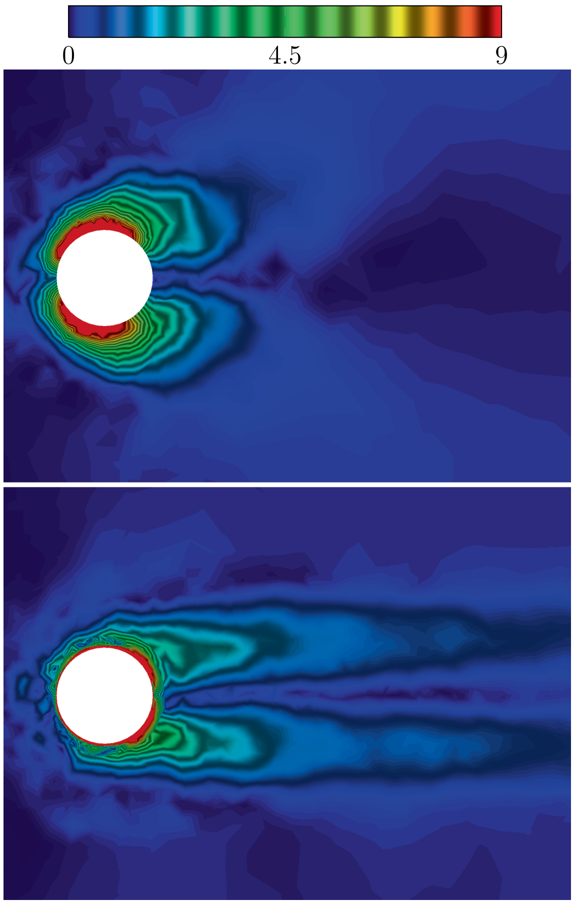

Fig. 19 Vorticity (in s^{-1}) around the tower at a cross-section $1.1R$ from the hub center for Mesh 2 (*top*) and Mesh 3 (*bottom*) with the DTR approach. The pictures are from the last time step of the computation

main computational challenges are the large Reynolds numbers and rotating turbulent flows, the care and high-resolution meshes needed in computing the correct torque values, and the presence of the tower, which requires a method that can deal with the fast, rotational relative motion between the rotor and tower. The core numerical technology is the ST-VMS method, which is the residual-based VMS version of the DSD/SST method, and which is also called DSD/SST-VMST. In calculating the stabilization parameters embedded in the ST-VMS method used in the computations reported here, we are using a new element length definition for the diffusion-dominated limit, and this has better stabilization features. The rotor geometry is that of the NREL 5MW offshore baseline wind turbine. We compute with a given wind speed and a specified rotor speed. The mesh update technique is based on using NURBS basis functions for the temporal representation of the rotor and mesh motion and in remeshing. We named it "ST/NURBS Mesh Update Method (STNMUM)" in this paper. We used 6 quadratic temporal NURBS patches for a $1/3$ rotation of the turbine. Using NURBS basis functions for the temporal representation of the rotor motion enables us to represent the circular paths associated with that motion exactly and, with a secondary mapping, specify a constant angular velocity corresponding to the invariant speeds along those paths. Given the rotor surface mesh and its motion, the mesh at the central control point of each patch is created with an automatic mesh generator, and the meshes at the other two control points are computed by using the mesh moving method that is normally used with the DSD/SST method. The STNMUM allows us to do mesh computations with longer time in between, but get the mesh-related information for each ST slab from the temporal representation whenever we need. The automatic mesh generator is used only 6 times. For better mesh resolution, we have layers of thin elements near the blade surfaces, created with a special process outside the automatic mesh generation. These layers of elements undergo rigid-body motion with the rotor. We used two different ways of getting the mesh-related information for each ST slab from the temporal representation: (a) directly as described above, which we call Direct Temporal Representation (DTR), and (b) after first computing the finite element meshes associated with each ST slab by interpolation from the temporal NURBS representation of the mesh, which we call Interpolated-Mesh Temporal Representation (IMTR). We presented results from the computations with and without the tower, with both the DTR and IMTR approaches. We compared the torque values obtained to the NREL data. The results demonstrate that the ST-VMS method and using NURBS basis functions in temporal representation of the mesh provide an accurate and flexible computational technology for computation of wind-turbine rotor and tower aerodynamics.

Acknowledgments This work was supported in part by the Rice–Waseda research agreement (first author). Method analysis and evaluation components of this work were also supported in part by ARO Grant W911NF-12-1-0162 (second through sixth authors). The starting NURBS geometry for the turbine blade was provided by Yuri Bazilevs (UCSD). NURBS layers near the blade and the mesh on the blade surface were generated by Anthony Puntel.

References

1. Tezduyar TE (1992) Stabilized finite element formulations for incompressible flow computations. Adv Appl Mech 28:1–44. doi:10.1016/S0065-2156(08)70153-4
2. Tezduyar TE, Behr M, Liou J (1992) 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. Comput Methods Appl Mech Eng 94:339–351. doi:10.1016/0045-7825(92)90059-S
3. Tezduyar TE, Behr M, Mittal S, Liou J (1992) 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. Comput Methods Appl Mech Eng 94:353–371. doi:10.1016/0045-7825(92)90060-W
4. Tezduyar TE (2003) Computation of moving boundaries and interfaces and stabilization parameters. Int J Numer Methods Fluids 43:555–575. doi:10.1002/fld.505
5. Tezduyar TE, Sathe S (2007) Modeling of fluid-structure interactions with the space-time finite elements: solution techniques. Int J Numer Methods Fluids 54:855–900. doi:10.1002/fld.1430
6. Takizawa K, Tezduyar TE (2011) Multiscale space-time fluid-structure interaction techniques. Comput Mech 48:247–267. doi:10.1007/s00466-011-0571-z
7. Takizawa K, Tezduyar TE (2012) Space–time fluid-structure interaction methods. Math Models Methods Appl Sci 22:1230001. doi:10.1142/S0218202512300013
8. Bazilevs Y, Takizawa K, Tezduyar TE (2013) Computational fluid–structure interaction: methods and applications. Wiley, New York
9. Brooks AN, Hughes TJR (1982) Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. Comput Methods Appl Mech Eng 32:199–259
10. Tezduyar TE, Mittal S, Ray SE, Shih R (1992) Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. Comput Methods Appl Mech Eng 95:221–242. doi:10.1016/0045-7825(92)90141-6
11. Hughes TJR (1995) Multiscale phenomena: Green’s functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles, and the origins of stabilized methods. Comput Methods Appl Mech Eng 127:387–401
12. Hughes TJR, Oberai AA, Mazzei L (2001) Large eddy simulation of turbulent channel flows by the variational multiscale method. Phys Fluids 13:1784–1799
13. Bazilevs Y, Calo VM, Cottrell JA, Hughes TJR, Reali A, Scovazzi G (2007) Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Comput Methods Appl Mech Eng 197:173–201
14. Bazilevs Y, Akkerman I (2010) Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method. J Comput Phys 229:3402–3414
15. Hughes TJR, Liu WK, Zimmermann TK (1981) Lagrangian–Eulerian finite element formulation for incompressible viscous flows. Comput Methods Appl Mech Eng 29:329–349
16. Ohayon R (2001) Reduced symmetric models for modal analysis of internal structural-acoustic and hydroelastic-sloshing systems. Comput Methods Appl Mech Eng 190:3009–3019
17. van Brummelen EH, de Borst R (2005) On the nonnormality of subiteration for a fluid–structure interaction problem. SIAM J Sci Comput 27:599–621
18. Bazilevs Y, Calo VM, Zhang Y, Hughes TJR (2006) Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. Comput Mech 38:310–322
19. Lohner R, Cebral JR, Yang C, Baum JD, Mestreau EL, Soto O (2006) Extending the range of applicability of the loose coupling approach for FSI simulations. In: Bungartz H-J, Schafer M (eds) Fluid–structure interaction, volume 53 of lecture notes in Computational Science and Engineering. Springer, Berlin, p 82–100
20. Bletzinger K-U, Wuchner R, Kupzok A (2006) Algorithmic treatment of shells and free form-membranes in FSI. In: Bungartz H-J, Schafer M (eds) Fluid–structure interaction, volume 53 of lecture notes in Computational Science and Engineering. Springer, Berlin, p 336–355
21. Bazilevs Y, Calo VM, Hughes TJR, Zhang Y (2008) Isogeometric fluid–structure interaction: theory, algorithms, and computations. Comput Mech 43:3–37
22. Dettmer WG, Peric D (2008) On the coupling between fluid flow and mesh motion in the modelling of fluid–structure interaction. Comput Mech 43:81–90
23. Bazilevs Y, Gohean JR, Hughes TJR, Moser RD, Zhang Y (2009) Patient-specific isogeometric fluid-structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik (2000) left ventricular assist device. Comput Methods Appl Mech Eng 198:3534–3550
24. Bazilevs Y, Hsu M-C, Benson D, Sankaran S, Marsden A (2009) Computational fluid–structure interaction: methods and application to a total cavopulmonary connection. Comput Mech 45:77–89
25. Calderer R, Masud A (2010) A multiscale stabilized ALE formulation for incompressible flows with moving boundaries. Comput Mech 46:185–197
26. Bazilevs Y, Hsu M-C, Zhang Y, Wang W, Liang X, Kvamsdal T, Brekken R, Isaksen J (2010) A fully-coupled fluid–structure interaction simulation of cerebral aneurysms. Comput Mech 46:3–16
27. Bazilevs Y, Hsu M-C, Zhang Y, Wang W, Kvamsdal T, Hentschel S, Isaksen J (2010) Computational fluid–structure interaction: methods and application to cerebral aneurysms. Biomech Model Mechanobiol 9:481–498
28. Bazilevs Y, Hsu M-C, Akkerman I, Wright S, Takizawa K, Henicke B, Spielman T, Tezduyar TE (2011) 3D simulation of wind turbine rotors at full scale. Part I: Geometry modeling and aerodynamics. Int J Numer Methods Fluids 65:207–235. doi:10.1002/fld.2400
29. Bazilevs Y, Hsu M-C, Kiendl J, Wüchner R, Bletzinger K-U (2011) 3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades. Int J Numer Methods Fluids 65:236–253
30. Akkerman I, Bazilevs Y, Kees CE, Farthing MW (2011) Isogeometric analysis of free-surface flow. J Comput Phys 230:4137–4152
31. Hsu M-C, Bazilevs Y (2011) Blood vessel tissue prestress modeling for vascular fluid-structure interaction simulations. Finite Elem Anal Des 47:593–599
32. Nagaoka S, Nakabayashi Y, Yagawa G, Kim YJ (2011) Accurate fluid–structure interaction computations using elements without mid-side nodes. Comput Mech 48:269–276. doi:10.1007/s00466-011-0620-7
33. Bazilevs Y, Hsu M-C, Takizawa K, Tezduyar TE (2012) ALE-VMS and ST-VMS methods for computer modeling of wind-turbine rotor aerodynamics and fluid–structure interaction. Math Models Methods Appl Sci 22:1230002. doi:10.1142/S0218202512300025
34. Akkerman I, Bazilevs Y, Benson DJ, Farthing MW, Kees CE (2012) Free-surface flow and fluid–object interaction modeling with emphasis on ship hydrodynamics. J Appl Mech 79:010905
35. Tezduyar T, Aliabadi S, Behr M, Johnson A, Mittal S (1993) Parallel finite-element computation of 3D flows. Computer 26:27–36. doi:10.1109/2.237441
36. Tezduyar TE, Aliabadi SK, Behr M, Mittal S (1994) Massively parallel finite element simulation of compressible and incompressible flows. Comput Methods Appl Mech Eng 119:157–177. doi:10.1016/0045-7825(94)00082-4
37. Tezduyar T, Aliabadi S, Behr M, Johnson A, Kalro V, Litke M (1996) Flow simulation and high performance computing. Comput Mech 18:397–412. doi:10.1007/BF00350249
38. Tezduyar TE (1999) CFD methods for three-dimensional computation of complex flow problems. J Wind Eng Ind Aerodyn 81:97–116. doi:10.1016/S0167-6105(99)00011-2
39. Tezduyar T, Osawa Y (1999) Methods for parallel computation of complex flow problems. Parallel Comput 25:2039–2066. doi:10.1016/S0167-8191(99)00080-0
40. Tezduyar TE (2001) Finite element methods for flow problems with moving boundaries and interfaces. Arch Comput Methods Eng 8:83–130. doi:10.1007/BF02897870

41. Tezduyar TE, Takizawa K, Brummer T, Chen PR (2011) Space-time fluid–structure interaction modeling of patient-specific cerebral aneurysms. Int J Numer Methods Biomed Eng 27:1665–1710. doi:10.1002/cnm.1433
42. Takizawa K, Henicke B, Puntel A, Spielman T, Tezduyar TE (2012) Space–time computational techniques for the aerodynamics of flapping wings. J Appl Mech 79:010903. doi:10.1115/1.4005073
43. Takizawa K, Tezduyar TE (2012) Computational methods for parachute fluid–structure interactions. Arch Comput Methods Eng 19:125–169. doi:10.1007/s11831-012-9070-4
44. Takizawa K, Bazilevs Y, Tezduyar TE (2012) Space–time and ALE-VMS techniques for patient-specific cardiovascular fluid–structure interaction modeling. Arch Comput Methods Eng 19:171–225. doi:10.1007/s11831-012-9071-3
45. Takizawa K, Henicke B, Puntel A, Kostov N, Tezduyar TE (2012) Space–time techniques for computational aerodynamics modeling of flapping wings of an actual locust. Comput Mech 50:743–760. doi:10.1007/s00466-012-0759-x
46. Takizawa K, Kostov N, Puntel A, Henicke B, Tezduyar TE (2012) Space–time computational analysis of bio-inspired flapping-wing aerodynamics of a micro aerial vehicle. Comput Mech 50:761–778. doi:10.1007/s00466-012-0758-y
47. Takizawa K, Henicke B, Puntel A, Kostov N, Tezduyar TE (2012) Computer modeling techniques for flapping-wing aerodynamics of a locust. Comput Fluids. doi:10.1016/j.compfluid.2012.11.008
48. Takizawa K, Tezduyar TE (2012) Bringing them down safely. Mech Eng 134:34–37
49. Bazilevs Y, Takizawa K, Tezduyar TE (2013) Challenges and directions in computational fluid–structure interaction. Math Models Methods Appl Sci 23:215–221. doi:10.1142/S0218202513400010
50. Takizawa K, Wright S, Moorman C, Tezduyar TE (2011) Fluid–structure interaction modeling of parachute clusters. Int J Numer Methods Fluids 65:286–307. doi:10.1002/fld.2359
51. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. Comput Methods Appl Mech Eng 194:4135–4195
52. Bazilevs Y, Hughes TJR (2008) NURBS-based isogeometric analysis for the computation of flows about rotating components. Comput Mech 43:143–150
53. Manguoglu M, Sameh AH, Tezduyar TE, Sathe S (2008) A nested iterative scheme for computation of incompressible flows in long domains. Comput Mech 43:73–80. doi:10.1007/s00466-008-0276-0
54. Manguoglu M, Sameh AH, Saied F, Tezduyar TE, Sathe S (2009) Preconditioning techniques for nonsymmetric linear systems in the computation of incompressible flows. J Appl Mech 76:021204. doi:10.1115/1.3059576
55. Manguoglu M, Takizawa K, Sameh AH, Tezduyar TE (2010) Solution of linear systems in arterial fluid mechanics computations with boundary layer mesh refinement. Comput Mech 46:83–89. doi:10.1007/s00466-009-0426-z
56. Manguoglu M, Takizawa K, Sameh AH, Tezduyar TE (2011) Nested and parallel sparse algorithms for arterial fluid mechanics computations with boundary layer mesh refinement. Int J Numer Methods Fluids 65:135–149. doi:10.1002/fld.2415
57. Manguoglu M, Takizawa K, Sameh AH, Tezduyar TE (2011) A parallel sparse algorithm targeting arterial fluid mechanics computations. Comput Mech 48:377–384. doi:10.1007/s00466-011-0619-0
58. Tezduyar TE, Behr M, Mittal S, Johnson AA (1992) 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,p 7–24
59. Johnson AA, Tezduyar TE (1994) Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. Comput Methods Appl Mech Eng 119:73–94. doi:10.1016/0045-7825(94)00077-8
60. Takizawa K, Montes D, Fritze M, McIntyre S, Boben J, Tezduyar TE (2013) Methods for FSI modeling of spacecraft parachute dynamics and cover separation. Math Models Methods Appl Sci 23:307–338. doi:10.1142/S0218202513400058
61. Behr M, Tezduyar T (1999) The shear–slip mesh update method. Comput Methods Appl Mech Eng 174:261–274. doi:10.1016/S0045-7825(98)00299-0
62. Behr M, Tezduyar T (2001) Shear–slip mesh update in 3D computation of complex flow problems with rotating mechanical components. Comput Methods Appl Mech Eng 190:3189–3200. doi:10.1016/S0045-7825(00)00388-1
63. Takizawa K, Henicke B, Tezduyar TE, Hsu M-C, Bazilevs Y (2011) Stabilized space–time computation of wind-turbine rotor aerodynamics. Comput Mech 48:333–344. doi:10.1007/s00466-011-0589-2
64. Takizawa K, Henicke B, Montes D, Tezduyar TE, Hsu M-C, Bazilevs Y (2011) Numerical-performance studies for the stabilized space-time computation of wind-turbine rotor aerodynamics. Comput Mech 48:647–657. doi:10.1007/s00466-011-0614-5
65. Bazilevs Y, Hsu M-C, Kiendl J, Benson DJ (2012) A computational procedure for pre-bending of wind turbine blades. Int J Numer Methods Eng 89:323–336
66. Hsu M-C, Akkerman I, Bazilevs Y (2011) High-performance computing of wind turbine aerodynamics using isogeometric analysis. Comput Fluids 49:93–100
67. Hsu M-C, Akkerman I, Bazilevs Y (2012) Wind turbine aerodynamics using ALE-VMS: validation and role of weakly enforced boundary conditions. Comput Mech 50:499–511
68. Hsu M-C, Bazilevs Y (2012) Fluid–structure interaction modeling of wind turbines: simulating the full machine. Comput Mech 50:821–833
69. Korobenko A, Hsu M-C, Akkerman I, Tippmann J, Bazilevs Y (2013) Structural mechanics modeling and FSI simulation of wind turbines. Math Models Methods Appl Sci 23:249–272
70. Hsu M-C, Akkerman I, Bazilevs Y (2013) Finite element simulation of wind turbine aerodynamics: validation study using NREL Phase VI experiment. Wind Energy. doi:10.1002/we.1599
71. Tezduyar TE, Park YJ (1986) Discontinuity capturing finite element formulations for nonlinear convection-diffusion-reaction equations. Comput Methods Appl Mech Eng 59:307–325. doi:10.1016/0045-7825(86)90003-4
72. Tezduyar TE, Osawa Y (2000) Finite element stabilization parameters computed from element matrices and vectors. Comput Methods Appl Mech Eng 190:411–430. doi:10.1016/S0045-7825(00)00211-5
73. Akin JE, Tezduyar T, Ungor M, Mittal S (2003) Stabilization parameters and Smagorinsky turbulence model. J Appl Mech 70:2–9. doi:10.1115/1.1526569
74. Akin JE, Tezduyar TE (2004) Calculation of the advective limit of the SUPG stabilization parameter for linear and higher-order elements. Comput Methods Appl Mech Eng 193:1909–1922. doi:10.1016/j.cma.2003.12.050
75. Catabriga L, Coutinho ALGA, Tezduyar TE (2005) Compressible flow SUPG parameters computed from element matrices. Commun Numer Methods Eng 21:465–476. doi:10.1002/cnm.759
76. Corsini A, Rispoli F, Santoriello A, Tezduyar TE (2006) Improved discontinuity-capturing finite element techniques for reaction effects in turbulence computation. Comput Mech 38:356–364. doi:10.1007/s00466-006-0045-x
77. Catabriga L, Coutinho ALGA, Tezduyar TE (2006) Compressible flow SUPG parameters computed from degree-of-freedom submatrices. Comput Mech 38:334–343. doi:10.1007/s00466-006-0033-1

78. Tezduyar TE (2007) Finite elements in fluids: stabilized formulations and moving boundaries and interfaces. Comput Fluids 36:191–206. doi:10.1016/j.compfluid.2005.02.011
79. Rispoli F, Corsini A, Tezduyar TE (2007) Finite element computation of turbulent flows with the discontinuity-capturing directional dissipation (DCDD). Comput Fluids 36:121–126. doi:10.1016/j.compfluid.2005.07.004
80. Catabriga L, de Souza DAF, Coutinho ALGA, Tezduyar TE (2009) Three-dimensional edge-based SUPG computation of inviscid compressible flows with YZβ shock-capturing. J Appl Mech 76:021208. doi:10.1115/1.3062968
81. Corsini A, Iossa C, Rispoli F, Tezduyar TE (2010) A DRD finite element formulation for computing turbulent reacting flows in gas turbine combustors. Comput Mech 46:159–167. doi:10.1007/s00466-009-0441-0
82. Hsu M-C, Bazilevs Y, Calo VM, Tezduyar TE, Hughes TJR (2010) Improving stability of stabilized and multiscale formulations in flow simulations at small time steps. Comput Methods Appl Mech Eng 199:828–840. doi:10.1016/j.cma.2009.06.019
83. Corsini A, Rispoli F, Tezduyar TE (2011) Stabilized finite element computation of NOx emission in aero-engine combustors. Int J Numer Methods Fluids 65:254–270. doi:10.1002/fld.2451
84. Corsini A, Rispoli F, Tezduyar TE (2012) Computer modeling of wave-energy air turbines with the SUPG/PSPG formulation and discontinuity-capturing technique. J Appl Mech 79:010910. doi:10.1115/1.4005060
85. Corsini A, Rispoli F, Sheard AG, Tezduyar TE (2012) Computational analysis of noise reduction devices in axial fans with stabilized finite element formulations. Comput Mech 50:695–705. doi:10.1007/s00466-012-0789-4
86. Jonkman J, Butterfield S, Musial W, Scott G (2009) Definition of a 5-MW reference wind turbine for offshore system development. Technical Report NREL/TP-500-38060, National Renewable Energy Laboratory
87. Spera DA (1994) Introduction to modern wind turbines. In: Spera DA (ed) Wind turbine technology: fundamental concepts of wind turbine engineering. ASME, New York, pp 47–72
88. Saad Y, Schultz M (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 7:856–869
89. Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20:359–392