FAST Modular Framework for Wind Turbine Simulation: New Algorithms and Numerical Examples

Michael A. Sprague, Jason M. Jonkman, and Bonnie J. Jonkman
National Renewable Energy Laboratory

Presented at AIAA SciTech 2015: 33rd Wind Energy Symposium
Kissimmee, Florida
January 5-9, 2015

NREL is a national laboratory of the U.S. Department of Energy
Office of Energy Efficiency & Renewable Energy
Operated by the Alliance for Sustainable Energy, LLC

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Conference Paper
NREL/CP-2C00-63203
December 2015

Contract No. DE-AC36-08GO28308

NOTICE

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Available electronically at SciTech Connect http:/www.osti.gov/scitech

Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from:

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
OSTI http://www.osti.gov
Phone: 865.576.8401
Fax: 865.576.5728
Email: reports@osti.gov

Available for sale to the public, in paper, from:

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandra, VA 22312
NTIS http://www.ntis.gov
Phone: 800.553.6847 or 703.605.6000
Fax: 703.605.6900
Email: orders@ntis.gov

Cover Photos by Dennis Schroeder: (left to right) NREL 26173, NREL 18302, NREL 19758, NREL 29642, NREL 19795.

NREL prints on paper that contains recycled content.

FAST Modular Framework for Wind Turbine Simulation: New Algorithms and Numerical Examples*

Michael A. Sprague†, Jason M. Jonkman‡,
and Bonnie J. Jonkman§
National Renewable Energy Laboratory, Golden, Colorado, 80401, USA

Over the past few years, the FAST wind turbine simulation tool has undergone a major restructuring. FAST is now, at its core, an algorithm and software framework for coupling time-dependent multi-physics modules relevant to computer-aided engineering (CAE) of wind turbines. Each module, which represents one or more turbine components or physics control volumes, is constituted by a mathematical model composed of time-dependent constraint and/or differential equations that are typically nonlinear. Under this new modular form, modules can interact through matching or non-matching spatial meshes and can be time advanced with different time steps and different time integrators. Sharing of data between modules is accomplished with a predictor-corrector approach, which allows for either implicit or explicit time integration within each module. This new modularity positions FAST as a backbone for coupling both high-fidelity and engineering-level wind turbine physics models. In this paper, we describe new features of the FAST modular framework. In particular, we describe a new mixed-time-step algorithm, sparse-matrix storage, a direct solver for sparse linear systems, and interpolation of rotation fields in space for mesh mapping and in time for time advancement. We also show several numerical examples that demonstrate the performance and flexibility of the FAST framework, and we use those results to provide modeling guidance to users.

I. Introduction

The wind turbine industry relies heavily on CAE tools for analyzing wind turbine performance, loading, and stability. Over the past two decades, the U.S. Department of Energy has sponsored the National Renewable Energy Laboratory's (NREL's) development of CAE tools for wind turbine analysis. NREL's premier tool is FAST,[1] which is a modular assembly of advanced CAE codes. FAST is an open-source, professional-grade software package. FAST encompasses modules for aerodynamics (AeroDyn[2,3]), substructure hydrodynamics (HydroDyn[4,5]) for offshore systems, control and electrical systems (ServoDyn), and structural dynamics (ElastoDyn). Blade modeling capabilities are currently being augmented with addition of the BeamDyn[6] module, which is a high-fidelity code appropriate for highly flexible composite blades. The modules are coupled to allow for nonlinear analysis of aero-hydro-servo-elastic interactions in the time domain. The FAST tool enables the analysis of a range of wind turbine configurations, including two- or three-blade horizontal-axis rotors, pitch or stall regulation, rigid or teetering hub, upwind or downwind rotor, and lattice or tubular towers. A wind turbine (or wind plant) can be modeled on land or offshore on fixed-bottom or floating substructures.

†Senior Scientist, Computational Science Center, 15013 Denver West Parkway, Golden, CO 80401, AIAA Professional Member.

‡Senior Engineer, National Wind Technology Center, 15013 Denver West Parkway, Golden, CO 80401, AIAA Professional Member.

§Senior Scientist, National Wind Technology Center, 15013 Denver West Parkway, Golden, CO 80401, AIAA Professional Member.

This work is motivated by a major restructuring of the FAST tool suite, which is described in Jonkman.[7] FAST-restructuring goals include (1) improving the ability to read, implement, and maintain source code; (2) increasing module sharing and shared-code development across the wind community; (3) improving numerical performance and robustness; and (4) greatly enhancing flexibility and expandability to enable further developments of functionality without the need to recode established modules. It is envisioned that the new modularization framework will transform FAST into a powerful, robust, and flexible wind turbine modeling tool with a large number of developers and a range of modeling fidelities across the aerodynamic, hydrodynamic, servo-dynamic, and structural-dynamic components.

In general, when modeling wind turbine multi-physics, each physics is represented as a system of nonlinear, time-dependent equations. These equations may be some combination of partial-differential equations (PDEs), ordinary-differential equations (ODEs), or, more generally, differential and algebraic equations (DAEs). Here, we restrict our scope to situations where spatial differential operators have been discretized and only temporal differential operators and/or algebraic constraints remain. This is known as semi-discretization or the method-of-lines approach in numerical PDE analysis.[8]

In Gasmi et al.,[9] we outlined our chosen taxonomy for the various approaches to multi-physics modeling and numerical simulation; Figure 1 summarizes that taxonomy. In that paper we focused on loose temporal coupling of partitioned models, where each module was time integrated separately, but in lock step, and where information was passed between modules at each time step. For explicit coupling, all modules were time advanced from known information. For implicit coupling, the time advancement of one or more modules depends on the data from other modules at the end of a time step. We introduced a predictor-corrector (PC) approach for implicit coupling. We found that the PC approach was more stable and was, in general, significantly more accurate than explicit coupling (when one or more modules had an implicit dependence on other-module solutions). Loose coupling is appealing because it allows modules to use spatial and temporal grids and ODE/DAE time integrators that are chosen to accurately represent a module's physics, and not to accommodate the grid of another module. Further, it allows for the use of existing software for a particular module. Alternatively, in a tightly coupled system, solutions to all equations must be time advanced with the same time step and same time integrator; use of existing software is problematic. Readers are referred to Felippa et al.[10] for a discussion of the benefits of loose coupling of partitions over tight coupling.

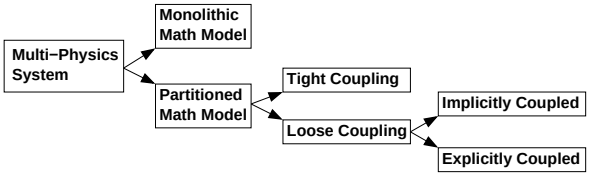


Figure 1. Schematic illustrating our "taxonomy" for models that describe a multi-physics systems.

In Sprague et al.,[11] we greatly expanded and improved upon the coupling strategies described in Gasmi et al.[9] Namely, we enhanced the numerical algorithm to handle non-matching temporal and spatial meshes, although noted limitations included mixed time steps based only on time step subcycling and the absence of interpolation of rotation fields, relying instead on a nearest-neighbor approach. In this paper, we describe new features of the FAST modular framework. In particular, we describe a new mixed-time-step algorithm supporting module steps both smaller than and larger than the coupling step, sparse-matrix storage, a direct solver for sparse linear systems, and interpolation of rotation fields in space (for mesh mapping) and in time (for time advancement). We also show several numerical examples that demonstrate the performance and flexibility of the FAST framework, and we use those results to provide modeling guidance to users.

II. Formulation

In this section we present a general framework for describing time-dependent partitioned systems that is adopted by the new modular framework in FAST.[7] The formulation described here is an extension of that described in Sprague et al.[11] The interface matching for non-matching spatial meshes is largely unchanged (Section II.B), except for the addition of a rigorous algorithm for the interpolation and extrapolation of rotation, which is described in Section II.C. In Sprague et al.[11] we described an algorithm where each module was independently time advanced with a time increment that was less than, or equal to, the global

interaction time increment. Interaction between modules was handled with a predictor-corrector approach. In Section II.D we extend that algorithm to allow modules to have a larger time increment than the global interaction increment. The purpose of this "large step" option is to accelerate simulations involving modules that evolve on much slower timescales than the rest of the system. Finally, in Section II.E, we describe our implementation of an open-source sparse-system direct solver.

A. Partitioned-System Representation

We assume that the entire multi-physics system (i.e., the wind turbine) is subdivided into N partitions as described in Jonkman[7] and Sprague et al.[11] Further, we assume that a method-of-lines approach (or semi-discretization) has been followed, where PDE partition models have been reduced to time-dependent ODEs, or, more generally, DAEs, through numerical discretization of spatial operators. For example, spatial differential operators can be approximated through finite-element or finite-difference discretization. Under this approach, each partition model can be represented in a general DAE form:

$$\dot{\mathbf{x}}^{(i)} = \mathbf{X}^{(i)}\left(t, \mathbf{x}^{(i)}, \mathbf{x}^{d,(i)}, \mathbf{z}^{(i)}, \mathbf{u}^{(i)}\right), \tag{1}$$

$$\mathbf{x}^{d,(i)}_{m^{(i)}+1} = \mathbf{X}^{d,(i)}\left(m^{(i)}, \mathbf{x}^{(i)}, \mathbf{x}^{d,(i)}_{m^{(i)}}, \mathbf{z}^{(i)}, \mathbf{u}^{(i)}\right), \tag{2}$$

$$\mathbf{0} = \mathbf{Z}^{(i)}\left(t, \mathbf{x}^{(i)}, \mathbf{x}^{d,(i)}, \mathbf{z}^{(i)}, \mathbf{u}^{(i)}\right), \tag{3}$$

$$\mathbf{y}^{(i)} = \mathbf{Y}^{(i)}\left(t, \mathbf{x}^{(i)}, \mathbf{x}^{d,(i)}, \mathbf{z}^{(i)}, \mathbf{u}^{(i)}\right), \tag{4}$$

where the superscript in parentheses $i \in \{1, 2, \ldots, N\}$ corresponds to the i^{th} partition; $\mathbf{X}^{(i)}$, $\mathbf{X}^{d,(i)}$, $\mathbf{Z}^{(\mathbf{i})}$, and $\mathbf{Y}^{(\mathbf{i})}$ are multi-variable vector functions, corresponding to the continuous-state, discrete-state, constraint-state, and output equations, respectively; $\mathbf{x}^{(i)}$, $\mathbf{x}^{d,(i)}$, and $\mathbf{z}^{(i)}$, are the continuous-state, discrete-state, and constraint-state dependent variables, respectively; $\mathbf{y}^{(i)}$ is the output-vector variable; $\mathbf{u}^{(i)}$ is a vector of inputs derived from outputs (and, in general, inputs) of all coupled partitions. For the discrete states, $m^{(i)}$ denotes the position in time ($t = m^{(i)}\Delta t^{(i)}$, for $m^{(i)} \in \{0, 1, \ldots\}$); values are calculated at fixed intervals $\Delta t^{(i)}$ and are constant over $m^{(i)}\Delta t^{(i)} \leq t < (m^{(i)} + 1)\Delta t^{(i)}$ (see Jonkman[7] for more details). The input to partition i is determined through the additional implicit input-output relationship

$$\mathbf{0} = \mathbf{U}^{(i)}\left(t, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(N)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\right), \tag{5}$$

where it is assumed that "mapping" of non-matching inputs and outputs is embodied in the input-output relationship. Equations (1)–(5) for $i \in \{1, \ldots, N\}$ constitute what we consider a *partitioned-system representation*. We describe some simple examples of systems in this partitioned representation below.

As described in the Introduction, a goal of the new modularized FAST framework is to allow for non-matching spatial and temporal meshes. This is aimed at modeling flexibility and simulation efficiency; each partition can be spatially and temporally refined independently of other partitions. However, partition coupling can introduce new physics that require additional refinement in time and/or space of the coupled partitions for stability and/or accuracy requirements. We describe below our methods for non-matching spatial and temporal meshes.

B. Non-Matching Spatial Meshes: Interface Matching

In this section we describe the algorithms used to couple non-matching spatial meshes. The information described here is largely taken from our earlier work,[11] and is included for completeness.

We focus on partition communication where data transfer occurs through domain interfaces (often corresponding to the domain boundary), which are zero-, one-, two-, or three-dimensional (0D, 1D, 2D, or 3D) entities (located in three-dimensional space). For example, a wind turbine blade domain might be represented as an assembly of 2D-shell and 3D-volume finite elements. Its domain boundary, and its interface with other modules, would be surface elements. Alternatively, a blade could be modeled as an assembly of beam elements, and its interface would be composed of line elements that may interact with an aerodynamic module, and a point element that may interact with the wind turbine hub module. Aerodynamic response may be modeled at the blade-interface alone, whereby the model "domain" and interface are both represented as surface or line elements. For higher fidelity simulations, aerodynamic response may be modeled with

computational fluid dynamics, whereby the fluid domain and its interface would be represented as volume and surface elements, respectively.

In the new FAST modularization framework, we are confronted potentially with the task of "matching" extremely disparate meshes. For example, consider the fluid-structure interaction of a simplified submerged truss structure shown schematically in Figure 2. Here, the hydrodynamic model well describes the true "wet" boundary of the truss structure. It is composed of line elements for segments between joints and point elements at joints. These elements take displacement, velocity, and acceleration as their input; lines and points output distributed and point loads, respectively. The structural model could be simply an assembly of beam elements with sectional properties that represent the full truss structure. The structural model takes distributed and point forces as input and outputs displacement, velocity, and acceleration. As shown in the figure, these domains are highly non-matching.

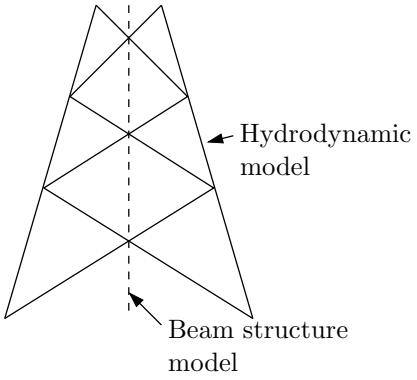


Figure 2. A multi-member hydrodynamics partition overlayed on a single-beam structural partition.

To facilitate module coupling, we have equipped the FAST code with the interface elements shown in Figure 3. A spatial mesh consists of a set of nodes, their connectivity (elements), nodal reference locations (position and orientation), and one or more nodal fields, which include motion, load, and/or scalar quantities. Point elements are physically assumed to represent rigid bodies or concentrated (lumped) loads applied on rigid bodies, Line2 elements are physically assumed to represent beams or distributed loads (per unit length) applied along beams, surface elements (Tri3 and Quad4) are physically assumed to represent plates/shells or surface traction loads (per unit area) applied across plates/shells, and volume elements (Tet4, Wedge6, and Hex8) are physically assumed to represent solids or body loads (per unit volume) applied within solids. Rotational displacement (orientation) is stored as a direction cosine matrix. Scalar quantities can include, e.g., color, temperature, or other attributes, independent of motion and load quantities. Details regarding the computational aspects of creation and manipulation of these entities are described in the FAST programmer's handbook.[12]

Module coupling is embodied in the creation of the "mapped" version of the input-output equation (5)

$$\mathbf{0} = \mathbf{U}^{M,(i)}\left(t\,, \mathbf{M}_{\mathbf{u}}^{1i}\left(\mathbf{u}^{(1)}\right), \ldots, \mathbf{u}^{(i)}, \ldots, \mathbf{M}_{\mathbf{u}}^{Ni}\left(\mathbf{u}^{(N)}\right), \right.$$
$$\left. \mathbf{M}_{\mathbf{y}}^{1i}\left(\mathbf{y}^{(1)}\right), \ldots, \mathbf{M}_{\mathbf{y}}^{ii}\left(\mathbf{y}^{(i)}\right), \ldots, \mathbf{M}_{\mathbf{y}}^{Ni}\left(\mathbf{y}^{(N)}\right)\right), \tag{6}$$

where $\mathbf{M}_{\mathbf{y}}^{ji}$ and $\mathbf{M}_{\mathbf{u}}^{ji}$ are vector functions that map the interface input or output mesh, respectively, of module j onto the interface *input* mesh of module i. Equation (6) is written in its most general form. In most cases, however, it is expected that a partition's input-output equation will depend only on its input, and the output from a few, or even one, other partitions. Inputs and outputs are expected to be composed of load, motion, and reference-position data, as well as scalar quantities.

Our mapping procedures are based on a straight-forward spatial projection between interfaces, and our guiding principles for transferring loads and motion quantities are

1. Loads (distributed/point forces and moments) are balanced between source and destination meshes.

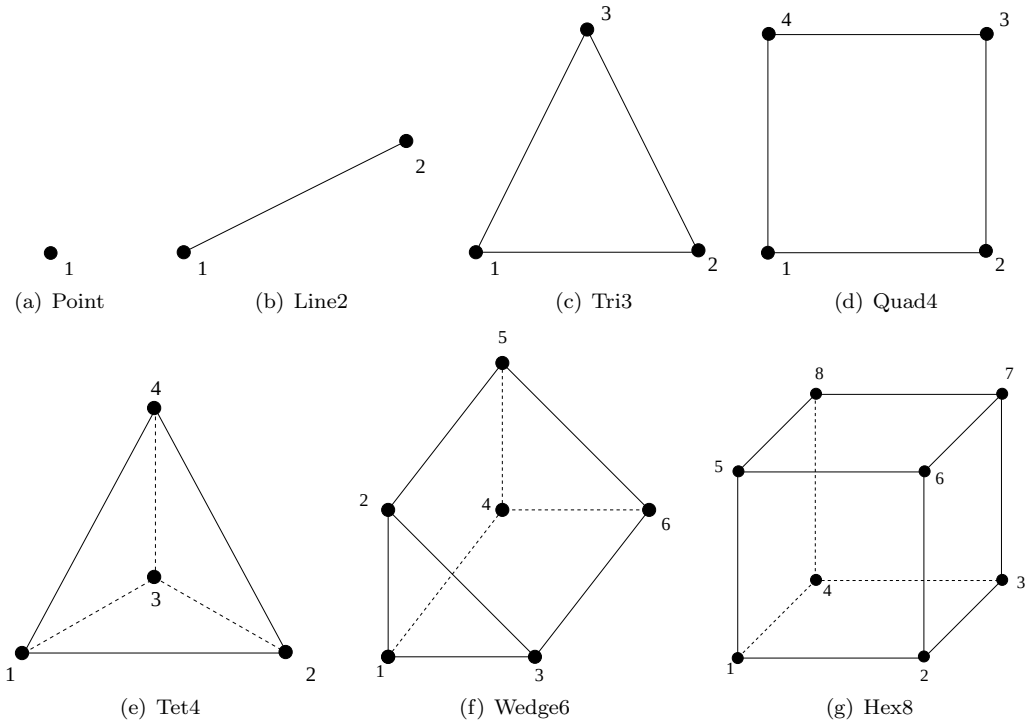


Figure 3. Interface-element types in the modularized FAST framework for coupling partitions.

2. Motion quantities are mapped in a physically relevant manner such that motions are properly preserved; e.g., rigid-body motions are transferred.

3. Load and motion mappings should be conjugate.

4. When source and destination meshes are identical (same element types and element locations), there is a one-to-one mapping of load and motion quantities.

It is our goal to create mapping algorithms for all of the entities shown in Figure 3. However, we have currently created the following algorithms: *Point_to_Point*, *Line2_to_Point*, *Point_to_Line2*, and *Line2_to_Line2*. Our mapping approach is related to what is known as *consistent interpolation*.[13,14] Detailed qualitative descriptions of motion and load mapping are given in Tables 1 and 2, respectively, for Point and Line2 elements; quantitative theoretical formulations of the algorithms are given in the Appendix. Mapping between independent spatial discretizations involves two steps: (1) a mapping search where nearest-neighbor nodes/elements are found between source and destination meshes and (2) a mapping transfer where nodal field quantities are transferred to the destination mesh from the mapped nodes of the source mesh. Augmentation of the source mesh for load quantities of Line2 elements is needed so that loads are properly transferred in the case of a coarsely discretized source mesh mapped to a finely discretized destination mesh. The mapping transfers for load quantities of Line2 elements involve lumping distributed source loads to point loads, splitting the point loads while transferring to destination loads, and transforming the destination point loads to distributed loads. This multi-step procedure is used to ensure that the first guiding principle is maintained even when the source and destination meshes are extremely disparate. (A simple interpolation of distributed loads would not result in the same total load transfer if the source and destination meshes are not of the same total length.) All mapping searches are based on the reference configuration of the meshes; as such, mapping-search procedure between source and destination meshes need only be performed at initialization and when the reference configuration is changed, or if the intent is for one mesh to move relative to another. These mappings may violate our fourth guiding principle in the case where a module's mesh contains multiple nodes at the same location in space. An example where such a mesh would be appropriate is the case where a distributed load using Line2 elements is defined with a jump discontinuity. This pitfall can be mitigated by introducing a small offset between the nodes of the mesh.

Table 1. Summary description of mapping algorithms for motion and scalar quantities between Point and Line2 elements (see Figure 3).

	Source: Point	Source: Line2
Destination: Point	Algorithm: Point_to_Point (motion mapping) Mapping search: For each Point-element node of the destination mesh, a nearest-neighbor Point-element node of the source mesh is found in the reference configuration. A source-mesh Point-element node may be associated with multiple destination-mesh Point-element nodes. Mapping transfer: For each destination-mesh Point-element node, motion and scalar quantities are transferred from its mapped source Point-element node. In the case that the source and destination Point-element nodes are not coincident in the current configuration, rotations and moment arms (including displacements) are used to augment transferred translations such that overall motion is maintained.	Algorithm: Line2_to_Point (motion mapping) Mapping search: For each Point-element node of the destination mesh, a nearest-neighbor Line2 element of the source mesh is found[¶] in the reference configuration in a manner identical to the Line2_to_Line2 motion-mapping search. Mapping transfer: For each destination-mesh Point-element node, motion and scalar quantities are interpolated (based on projection) and are transferred from its mapped source Line2 element in a manner identical to the Line2_to_Line2 motion-mapping transfer.
Destination: Line2	Algorithm: Point_to_Line2 (motion mapping) Mapping search: For each node of the Line2-element destination mesh, a nearest-neighbor Point-element node of the source mesh is found in the reference configuration in a manner identical to the Point_to_Point motion-mapping search. Mapping transfer: For each destination-mesh Line2-element node, motion and scalar quantities are transferred from its mapped source Point-element node in a manner identical to Point_to_Point motion-mapping transfer.	Algorithm: Line2_to_Line2 (motion mapping) Mapping search: For each Line2-element node of the destination mesh, a nearest-neighbor Line2 element of the source mesh is found[¶] in the reference configuration, for which the destination Line2-element node projects orthogonally onto the source Line2-element domain. A source-mesh Line2 element may be associated with multiple destination-mesh Line2-element nodes. Mapping transfer: For each destination-mesh Line2-element node, motion and scalar quantities are interpolated (based on projection) and are transferred from its mapped source Line2 element. In the case that the destination Line2-element node does not lie in its source Line2-element domain in the current configuration, interpolated rotations and moment arms (including displacements) are used to augment transferred translations such that overall motion is maintained.
[¶]If a Line2 element is not found, the mapping is aborted with an error.		

Table 2. Summary description of mapping algorithms for load quantities between Point and Line2 elements (see Figure 3).

	Source: Point	Source: Line2
Destination: Point	Algorithm: Point_to_Point (load mapping) Mapping search: For each Point-element node of the source mesh, a nearest-neighbor Point-element node of the destination mesh is found in the reference configuration. A destination-mesh Point-element node may be associated with multiple source-mesh Point-element nodes. Mapping transfer: For each source-mesh Point-element node, forces and moments are transferred to its mapped destination Point-element node; forces and moments are superposed when a destination element has more than one source element. In the case that the source and destination Point-element nodes are not coincident in the current configuration, forces and moment arms (including displacements) are used to augment transferred moments such that the overall load balance is maintained.	Algorithm: Line2_to_Point (load mapping) Mapping search: An augmented Line2-element source mesh is first formed by splitting the original Line2-element source mesh at each location where a destination-mesh Point-element node projects orthogonally onto the Line2-element source mesh. For each node of the augmented Line2-element source mesh, a nearest-neighbor Point-element node of the destination mesh is found in the reference configuration in a manner identical to the Point_to_Point load-mapping search. Mapping transfer: For each Line2 element of the augmented source mesh, distributed loads are lumped as point loads at the two nodes (of the source Line2 element) such that the lumped loads maintain the overall load balance with the Line2-element distributed loads; lumped loads are superposed at nodes shared by multiple elements in a manner identical to lumping in the Line2_to_Line2 load mapping. The lumped nodal loads from each Line2-element node of the augmented source mesh are transferred to its mapped destination Point-element node in a manner identical to Point_to_Point load mapping.
Destination: Line2	Algorithm: Point_to_Line2 (load mapping) Mapping search: For each Point-element node of the source mesh, a nearest-neighbor Line2 element of the destination mesh is found‖ in the reference configuration in a manner identical to the Line2_to_Line2 load-mapping search (but without augmentation of the source mesh). Mapping transfer: For each source-mesh Point-element node, the point load is split based on its projected location in the mapped destination Line2 element, and is transferred as two point loads at the destination Line2-element nodes and transformed to distributed loads in a manner identical to the Line2_to_Line2 load-mapping transfer (but without augmentation and lumping of the source mesh).	Algorithm: Line2_to_Line2 (load mapping) Mapping search: An augmented Line2-element source mesh is first formed by splitting the original Line2-element source mesh at each location where a destination-mesh Line2-element node projects orthogonally from the destination mesh. For each Line2-element node of the augmented source mesh, a nearest-neighbor Line2 element of the destination mesh is found‖ in the reference configuration, for which the source Line2-element node projects orthogonally onto the destination Line2-element domain. A destination-mesh Line2 element may be associated with multiple source-mesh Line2-element nodes. Mapping transfer: For each Line2 element of the augmented source mesh, distributed loads are lumped as point loads at the two nodes (of the source Line2 element) such that the lumped loads maintain the overall load balance with the Line2-element distributed loads; lumped loads are superposed at nodes shared by multiple elements. For each Line2-element node of the augmented source mesh, the lumped load is split based on its projected location in the mapped destination Line2 element, and is transferred as two point loads at the destination Line2-element nodes. Forces and moments are superposed when a destination Line2-element node has more than one source element. In the case that the source Line2-element node does not lie in its destination Line2-element domain in the current configuration, forces and moment arms (including displacements) are used to augment transferred moments such that the overall load balance is maintained. The transferred point loads are transformed to distributed loads that maintain the overall load balance.

‖If a Line2 element is not found, the mapping is aborted with an error.

C. Rotation Interpolation

Within the FAST modular framework, orientation at a given point in space and time is stored as a direction cosine matrix (DCM). In order to accomodate non-matching spatial and temporal meshes, we require a means of inerpolating and extrapolating orientation fields in space or time. This is complicated by the fact that DCMs do not reside in a linear space, but are in the Lie group $SO(3)$. For example, if one were to linearly interpolate each entry of two DCMs at different locations in space, the resulting matrix would not, in general, be a DCM.

We describe here our algorithm for interpolating and extrapolating DCMs, which follows closely that described in Mota et al.[15] The algorithm described here is appropriate for interpolating and extrapolating DCMS defined at different locations in time or along a curve defined in three-dimensional space. Given $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3 \in SO(3)$, which are DCMs at time stations (or space locations) t_1, t_2, and t_3, respectively, we wish to determine an interpolated/extrapolated DCM $\mathbf{\Lambda}_t \in SO(3)$ at time t. The steps are as follows:

1. Calculate the logarithmic maps for each of the DCMs:

$$\widetilde{\boldsymbol{\lambda}}_j = \log\left(\mathbf{\Lambda}_j\right), \tag{7}$$

for each $j \in \{1,2,3\}$, where $\widetilde{\boldsymbol{\lambda}}_j \in so(3)$ is a skew-symmetric tensor and $so(3)$ is the Lie algebra associated with the Lie group $SO(3)$. The logrithm of rotation is calculated as follows: For each $j \in \{1,2,3\}$ for quadratic interpolation/extrapolation, or for each $j \in \{1,2\}$ for linear interpolation/extrapolation:

   (a) Calculate the angle of rotation as

$$\theta_j = \arccos\left\{\frac{1}{2}\left[\operatorname{trace}\left(\mathbf{\Lambda}_j\right) - 1\right]\right\}, \tag{8}$$

   where $\theta_j \in [0, \pi]$.

   (b) The logarithm of the rotation, $\log(\mathbf{\Lambda}_j) \in so(3)$, is calculated as

$$\widetilde{\boldsymbol{\lambda}}_j = \log\left(\mathbf{\Lambda}_j\right) = \begin{cases} \mathbf{0}, & \theta_j = 0, \\ \frac{\theta_j}{2\sin(\theta_j)}\left(\mathbf{\Lambda}_j - \left(\mathbf{\Lambda}_j\right)^T\right), & \theta_j \in (0,\pi), \\ \pm\pi\widetilde{\mathbf{v}}_j, & \theta_j = \pi, \end{cases} \tag{9}$$

   where $\widetilde{\mathbf{v}}_j$ is the skew-symmetric matrix associated with $\mathbf{v}_j$, which is the unit-length eigenvector of $\mathbf{\Lambda}_j$ associated with the eigenvalue 1 and, given $\mathbf{v} \in \mathbb{R}^3$,

$$\widetilde{\mathbf{v}} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}. \tag{10}$$

   For the case $\theta_j = \pi$ in (9), the sign is chosen according to continuity conditions from the field in the neighborhood.

2. Calculate $\boldsymbol{\lambda}_t$ at time t by interpolating or extrapolating with a polynomial fit to data $\boldsymbol{\lambda}_j$ (and taking into consideration 2π periodicity), where $\boldsymbol{\lambda}_j$ is the axial vector associated with the skew-symmetric matrix $\widetilde{\boldsymbol{\lambda}}_j$; $j \in \{1,2,3\}$ for quadratic interpolation/extrapolation, or $j \in \{1,2\}$ for linear interpolation/extrapolation.

3. Calculate the interpolated or extrapolated DCM at time t:

$$\mathbf{\Lambda}_t = \exp\left[\widetilde{\boldsymbol{\lambda}}_t\right], \tag{11}$$

   where the matrix exponential is calculated as follows:

   (a) Calculate the angle of rotation as

$$\theta_t = \left\|\boldsymbol{\lambda}_t\right\|, \tag{12}$$

   where $\|\cdot\|$ is the Euclidean norm.

   (b) Calculate the matrix exponential as

$$\mathbf{\Lambda}_t = \exp\left[\widetilde{\boldsymbol{\lambda}}_t\right] = \begin{cases} \mathbf{I}, & \theta_t = 0, \\ \mathbf{I} + \frac{\sin\theta_t}{\theta_t}\widetilde{\boldsymbol{\lambda}}_t + \frac{1-\cos\theta_t}{\theta_t^2}\widetilde{\boldsymbol{\lambda}}_t^2, & \theta_t > 0\,. \end{cases} \tag{13}$$

D. Non-Matching Temporal Meshes: Time-Step Subcycling

In our time-update algorithm, each partition is time advanced with a time increment $\Delta t^{(i)}$ that is either an integer multiple of, or an integer division of, Δt^I, the interaction time increment. We assume that the first M modules have $\Delta t^{(i)} > \Delta t^I$, and the remaining $N - M$ modules have $\Delta t^{(i)} \leq \Delta t^I$. Partition time increments are $\Delta t^{(i)} = q^{(i)} \Delta t^I$ for $i \in \{1, \ldots, M\}$ and $\Delta t^{(i)} = \Delta t^I / q^{(i)}$, for $i \in \{M+1, \ldots, N\}$, and where $q^{(i)}$ is an integer for all $i \in \{1, \ldots, N\}$.

Below is our time-advancement algorithm for a single interaction time increment Δt^I. Partition states in $\{1, \ldots, M\}$ may or not be updated depending on their time increment and the last time they were updated. Partition states in $\{M+1, \ldots, N\}$ will be updated one or more times over one interaction time increment. In the following, $t_n = t_0 + n\Delta t^I$, where t_0 is the time at initialization, and $n \in \{0, \ldots, n_{\max}\}$.

Known data at beginning of step

Let $\overline{n}^{(i)} = n + q^{(i)} - \overline{q}^{(i)}$ for all $i \in \{1, \ldots, M\}$ where $\overline{q}^{(i)}$ is an integer tracking the number of interaction time steps occurring since the last update; $\overline{q}^{(i)} \in \{0, \ldots, q^{(i)}\}$. Let $P \leq M$ be the number of partitions for which $\overline{q}^{(i)} = q^{(i)}$ (i.e., those with $\overline{n}^{(i)} = n$) and let $\{k_1, \ldots, k_P\}$ be the set of all partition numbers for which $\overline{q}^{(i)} = q^{(i)}$; the remaining $Q = M - P$ partition numbers are stored in the set $\{\ell_1, \ldots, \ell_Q\}$. We assume that the following data are known at time t_n:

$$\mathbf{x}^{(i)}_{\overline{n}^{(i)}}, \dot{\mathbf{x}}^{(i)}_{\overline{n}^{(i)}}, \dot{\mathbf{x}}^{(i)}_{\overline{n}^{(i)}-q^{(i)}}, \ldots, \dot{\mathbf{x}}^{(i)}_{\overline{n}^{(i)}-mq^{(i)}}, \mathbf{x}^{d,(i)}_{\overline{n}^{(i)}}, \mathbf{z}^{(i)}_{\overline{n}^{(i)}}, \mathbf{y}^{(i)}_{\overline{n}^{(i)},(-1)}, \mathbf{y}^{(i)}_{\overline{n}^{(i)}-q^{(i)}}, \mathbf{u}^{(i)}_{\overline{n}^{(i)},(-1)}, \mathbf{u}^{(i)}_{\overline{n}^{(i)}-q^{(i)}}, \mathbf{u}^{(i)}_{\overline{n}^{(i)}-2q^{(i)}}, \tag{14}$$

for $i \in \{\ell_1, \ldots, \ell_Q\}$,

$$\mathbf{x}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_{n-q^{(i)}}, \ldots, \dot{\mathbf{x}}^{(i)}_{n-mq^{(i)}}, \mathbf{x}^{d,(i)}_n, \mathbf{z}^{(i)}_n, \mathbf{y}^{(i)}_n, \mathbf{y}^{(i)}_{n-q^{(i)}}, \mathbf{u}^{(i)}_n, \mathbf{u}^{(i)}_{n-q^{(i)}}, \mathbf{u}^{(i)}_{n-2q^{(i)}}, \tag{15}$$

for $i \in \{k_1, \ldots, k_P\}$, and

$$\mathbf{x}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_{n-\frac{1}{q^{(i)}}}, \ldots, \dot{\mathbf{x}}^{(i)}_{n-\frac{m}{q^{(i)}}}, \mathbf{x}^{d,(i)}_n, \mathbf{z}^{(i)}_n, \mathbf{y}^{(i)}_n, \mathbf{u}^{(i)}_n, \mathbf{u}^{(i)}_{n-1}, \mathbf{u}^{(i)}_{n-2}, \tag{16}$$

for $i \in \{M+1, \ldots, N\}$, where a "(-1)" subscript denotes dependence on a polynomial-extrapolated value (see below for details), the state-derivative RHS is calculated as

$$\dot{\mathbf{x}}^{(i)}_n = \mathbf{X}^{(i)}\left(t_n, \mathbf{x}^{(i)}_n, \mathbf{x}^{d,(i)}_n, \mathbf{z}^{(i)}_n, \mathbf{u}^{(i)}_n\right), \tag{17}$$

for each $i \in \{1, \ldots, N\}$; m indicates the number of previous history points required by the underlying multi-step DAE- or ODE-solver algorithm; $m = 0$ for a single-step integrator like Runge-Kutta. Also known is the value of $j_{\max}$, which is the number of correction iterations to be taken after a prediction step.

Advance certain large-step modules

Step $1L$: For all $i \in \{k_1, \ldots, k_P\}$, (i.e., those for which $\overline{q}^{(i)} = q^{(i)}$), let $\overline{q}^{(i)} = 0$ (reset counter) and calculate predicted inputs with polynomial extraction of known inputs:

$$\mathbf{u}^{(i)}_{n+q^{(i)},(-1)} = \text{polyfit}\left(t_{n+q^{(i)}}, \Delta t^{(i)}, \mathbf{u}^{(i)}_n, \mathbf{u}^{(i)}_{n-q^{(i)}}, \mathbf{u}^{(i)}_{n-2q^{(i)}}\right), \tag{18}$$

Here, extrapolation is accomplished by the function polyfit, which is evaluated at the time $t_{n+q^{(i)}}$; the function polyfit is based on a 2^{nd}-order polynomial fit to the 3 values that are separated in time by $\Delta t^{(i)}$, and it has an $O\left[(\Delta t^{(i)})^3\right]$ error. Polyfit can also provide a linear extrapolation when only provided two values.

Step $2L$: For all $i \in \{k_1, \ldots, k_P\}$, update module states to $t_{n+q^{(i)}}$:

$$\left\{ \begin{array}{l} \mathbf{x}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_n, \dot{\mathbf{x}}^{(i)}_{n-q^{(i)}}, \ldots, \dot{\mathbf{x}}^{(i)}_{n-mq^{(i)}}, \mathbf{x}^{d,(i)}_n, \mathbf{z}^{(i)}_n, \\ \mathbf{u}^{(i)}_{n+q^{(i)},(-1)}, \mathbf{u}^{(i)}_n, \mathbf{u}^{(i)}_{n-q^{(i)}}, \mathbf{X}^{(i)}, \mathbf{X}^{d,(i)}, \mathbf{Z}^{(i)} \end{array} \right\} \xrightarrow{\text{ADV}} \left\{ \begin{array}{l} \mathbf{x}^{(i)}_{n+q^{(i)}}, \dot{\mathbf{x}}^{(i)}_{n+q^{(i)}}, \\ \mathbf{x}^{d,(i)}_{n+q^{(i)}}, \mathbf{z}^{(i)}_{n+q^{(i)}} \end{array} \right\}, \tag{19}$$

where the notation on the left-hand side explicitly indicates the data accessible to the underlying integrator, and where $\mathbf{X}^{(i)}$, $\mathbf{X}^{d,(i)}$, and $\mathbf{Z}^{(i)}$ express access to the continuous-state, discrete-state, and constraint right-hand sides, respectively; ADV denotes the execution of the underlying m-step DAE- or ODE-solver algorithm

over a single time step $\Delta t^{(i)}$. In the new FAST modularization, data structures are equipped to pass up to 3 input values to time-advancement routines; this allows for the routines to use either a single input value at the preferred time (e.g., $\mathbf{u}^{(i)}_{n+q^{(i)},(-1)}$ for an implicit integrator; $\mathbf{u}^{(i)}_{n}$ for an explicit integrator) or to use multiple values to interpolate, e.g., using polyfit, to preferred time locations (as in high-order Runge-Kutta integrators).

Step $3L$: Output for the P modules are calculated as

$$\mathbf{y}^{(i)}_{n+q^{(i)},(-1)} = \mathbf{Y}^{(i)}\left(t_{n+q^{(i)}}, \mathbf{x}^{(i)}_{n+q^{(i)}}, \mathbf{x}^{d,(i)}_{n+q^{(i)}}, \mathbf{z}^{(i)}_{n+q^{(i)}}, \mathbf{u}^{(i)}_{n+q^{(i)},(-1)}\right), \tag{20}$$

for all $i \in \{k_1, \ldots, k_P\}$.

Advance all small-step modules

Step $1S$: Let $j = 0$, where j is the "correction" counter, and predict the input at t_{n+1} for all small-step partitions through extrapolation (over constant Δt^I):

$$\mathbf{u}^{(i)}_{n+1,(j-1)} = \text{polyfit}\left(t_{n+1}, \Delta t^I, \mathbf{u}^{(i)}_{n}, \mathbf{u}^{(i)}_{n-1}, \mathbf{u}^{(i)}_{n-2}\right), \tag{21}$$

for each $i \in \{M+1, \ldots, N\}$, where a subscript in parentheses indicates the correction iteration.

Step $2S$: Advance the solution of all small-step partitions to yield predicted state and constraint values, i.e.,

$$\left\{ \begin{array}{l} \mathbf{x}^{(i)}_{n}, \dot{\mathbf{x}}^{(i)}_{n}, \dot{\mathbf{x}}^{(i)}_{n-\frac{1}{q^{(i)}}}, \ldots, \dot{\mathbf{x}}^{(i)}_{n-\frac{m}{q^{(i)}}}, \mathbf{x}^{d,(i)}_{n}, \mathbf{z}^{(i)}_{n}, \\ \mathbf{u}^{(i)}_{n+1,(j-1)}, \mathbf{u}^{(i)}_{n}, \mathbf{u}^{(i)}_{n-1}, \mathbf{X}^{(i)}, \mathbf{X}^{d,(i)}, \mathbf{Z}^{(i)} \end{array} \right\} \xrightarrow{\text{ADV}} \left\{ \begin{array}{l} \mathbf{x}^{(i)}_{n+1,(j)}, \dot{\mathbf{x}}^{(i)}_{n+1,(j)}, \\ \mathbf{x}^{d,(i)}_{n+1,(j)}, \mathbf{z}^{(i)}_{n+1,(j)} \end{array} \right\}, \tag{22}$$

for each $i \in \{M+1, \ldots, N\}$.

Step $3S$: Determine the inputs and outputs concurrently for the N partitions through a global solve of the input-output equations at t_{n+1}; i.e., solve for $\mathbf{u}^{(i)}_{n+1,(j)}, \mathbf{y}^{(i)}_{n+1,(j)}$ for all $i \in \{1, \ldots, N\}$ by solving the nonlinear system composed of the following $2N$ expressions:

$$\begin{aligned} \mathbf{0} = \mathbf{U}^{M,(i)}\Bigg(& t_{n+1}, \mathbf{M}^{1i}_{\mathbf{u}}\left(\mathbf{u}^{(1)}_{n+1,(j)}\right), \ldots, \mathbf{u}^{(i)}_{n+1,(j)}, \ldots, \mathbf{M}^{Ni}_{\mathbf{u}}\left(\mathbf{u}^{(N)}_{n+1,(j)}\right), \\ & \mathbf{M}^{1i}_{\mathbf{y}}\left(\mathbf{y}^{(1)}_{n+1,(j)}\right), \ldots, \mathbf{M}^{Mi}_{\mathbf{y}}\left(\mathbf{y}^{(N)}_{n+1,(j)}\right)\Bigg), \end{aligned} \tag{23}$$

for $i \in \{1, \ldots, N\}$,

$$\mathbf{y}^{(i)}_{n+1,(j)} = \left\{ \begin{array}{ll} \mathbf{Y}^{(i)}\left(t_{n+1}, \mathbf{x}^{(i)}_{n+1}, \mathbf{x}^{d,(i)}_{n+1}, \mathbf{z}^{(i)}_{n+1}, \mathbf{u}^{(i)}_{n+1,(j)}\right), & \text{if} \quad \overline{q}^{(i)} = q^{(i)} - 1, \\ \text{polyfit}\left(t_{n+1}, \Delta t^{(i)}, \mathbf{y}^{(i)}_{\overline{n}^{(i)}+q^{(i)},(-1)}, \mathbf{y}^{(i)}_{\overline{n}^{(i)}}, \mathbf{y}^{(i)}_{\overline{n}^{(i)}-q^{(i)}}\right), & \text{else}, \end{array} \right. \tag{24}$$

for $i \in \{1, \ldots, M\}$, and

$$\mathbf{y}^{(i)}_{n+1,(j)} = \mathbf{Y}^{(i)}\left(t_{n+1}, \mathbf{x}^{(i)}_{n+1,(j)}, \mathbf{x}^{d,(i)}_{n+1,(j)}, \mathbf{z}^{(i)}_{n+1,(j)}, \mathbf{u}^{(i)}_{n+1,(j)}\right), \tag{25}$$

for $i \in \{M+1, \ldots, N\}$.

If $j = j_{\max}$, skip to Step 4; otherwise, let $j = j + 1$ and repeat Step $2S$.

Step 4: Save all of the final variables:

$$\begin{array}{lll} \mathbf{x}^{(i)}_{n+1} = \mathbf{x}^{(i)}_{n+1,(j_{\max})}, & \mathbf{x}^{d,(i)}_{n+1} = \mathbf{x}^{d,(i)}_{n+1,(j_{\max})}, & \mathbf{z}^{(i)}_{n+1} = \mathbf{z}^{(i)}_{n+1,(j_{\max})}, \\ \mathbf{y}^{(i)}_{n+1} = \mathbf{y}^{(i)}_{n+1,(j_{\max})}, & \mathbf{u}^{(i)}_{n+1} = \mathbf{u}^{(i)}_{n+1,(j_{\max})}, & \dot{\mathbf{x}}^{(i)}_{n+1} = \dot{\mathbf{x}}^{(i)}_{n+1,(j_{\max})}, \end{array} \tag{26}$$

for each $i \in \{M+1, \ldots, N\}$, and, for $i \in \{\ell_1, \ldots, \ell_Q\}$, let $\mathbf{y}^{(i)}_{n+1} = \mathbf{y}^{(i)}_{n+1,(j_{\max})}$ if $\overline{q}^{(i)} = q^{(i)} - 1$. Finally, let $\overline{q}^{(i)} = \overline{q}^{(i)} + 1$ for each $i \in \{1, \ldots, M\}$, which completes solution advancement to time t_{n+1}.

E. Sparse-Matrix Storage and Sparse-Linear-System Solve

The Jacobians created for the solution of the input-output equations that govern the coupling of modules, Eqs. (23)-(25), can be large and very sparse. In order to exploit that sparseness, FAST has been equipped to work with the MUltifrontal Massively Parallel Solver[16–18] (MUMPS), a sparse direct solver. MUMPS is public domain software that is written in Fortran 90. It is capable of running in serial processing as well as in shared-memory and distributed-memory parallel processing. In our implementation, we employed METIS[19] as the external ordering package. With the inclusion of MUMPS, individual FAST modules also have access to the sparse direct solver. While MUMPS can handle several input formats for sparse matrices, we chose the "coordinate form", which, for a given sparse matrix, requires (i) the size of the matrix, (ii) the number of nonzero entries NZ, (iii) two integer arrays of size NZ with the row and column indices of each nonzero entry, and (iv) a real-number array of size NZ with the nonzero entries of the matrix.

III. Illustrative Examples and Results

A. Mapping Examples

In Sprague et al.,[11] we showed two examples demonstrating load and motion mapping between Point and Line2 elements in several configurations. Here, we demonstrate the new rotation interpolation for the Line2_to_Line2 mapping algorithm. An arbitrary sign wave has been selected for illustrative purposes. Figure 4 shows two Line2 discretizations of an interface in their reference configurations; the meshes have 10 and 30 Line 2 elements. Figure 5(a) shows how the less-refined source mesh deforms through a "flattening"; Figure 5(b) shows how the source-mesh displacements and orientations are mapped onto the destination mesh. Figure 6(b) shows applied forces on the more-refined source mesh; Figure 6(a) shows how those loads are mapped onto the less-refined destination mesh. We see that our new mapping algorithm provides a smooth mapping of both loads, motions, and orientations. However, the results are not fully intuitive. For example, there are "dips" at the peak and trough of the sine wave in the destination mesh. These "dips" are the result of the rigid-body rotation being included in the mesh mapping and will reduce when the mesh refinement is more similar. Further, the mapping is such that the total force and moment are balanced between the source and destination meshes.

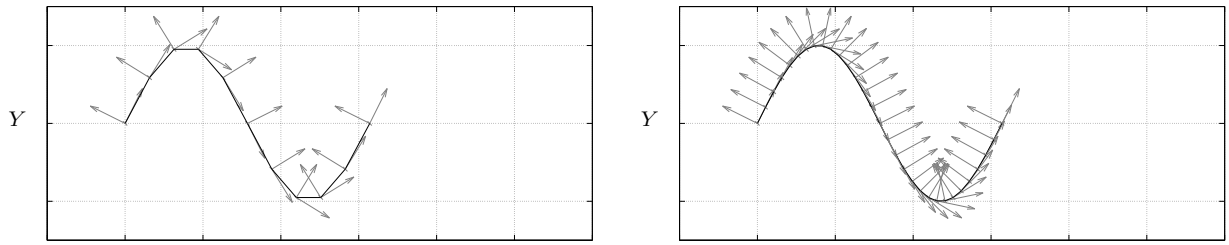
B. Time-Step Subcycling Examples

1. Uncoupled Models

In order to explore the accuracy and stability of our time-stepping algorithm with "large steps", we examine two coupling scenarios between three partitions described below. These are the same partitions examined in Sprague et al.[11] However, the partition properties have been modified to make them appropriate for the large-step algorithm.

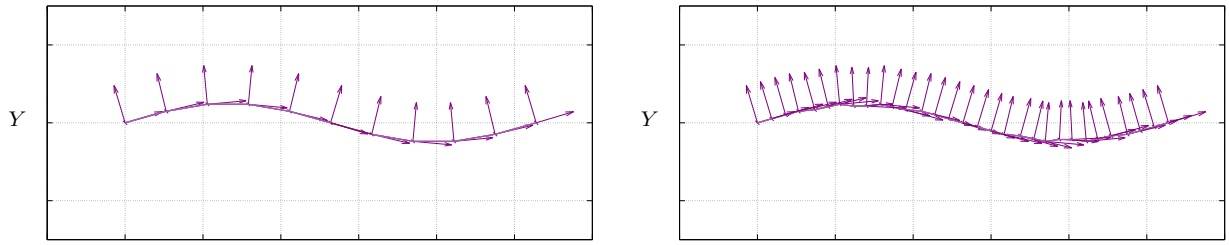
Figure 7 shows three stand-alone model partitions, each being a variation on the standard damped mechanical oscillator having some combination of inertial, damping, and stiffness terms. We examine the numerical solution of Partition 1 coupled to Partition 2 and Partition 1 coupled to Partition 3 below, and test the time-advancement algorithm described in Section II.D. Details for each partition, including its governing equation (GE), states, inputs, and outputs, are listed in the figure. We note that the outputs of all three partitions have direct feed-through of their inputs. Partition 3 has no states, but is defined solely through its input-output relationship. Dimensionless numerical values for system parameters are listed in Table 3. These parameters are different than those used in Sprague et al.[11] in that these were chosen to give Partition 2 a significantly smaller uncoupled frequency than Partition 1, which gives us an appropriate test case for the "large-step" time-step algorithm described above. The undamped natural frequencies of uncoupled Partitions 1 and 2 are 1.7 and 0.45 (dimensionless), respectively.

In Sprague et al.,[11] we examined closely the accuracy and stability of the predictor-corrector coupling algorithm where each module was time advanced with a time increment equal to or less than the interaction time increment. We examined different time increment sizes and the use of different fourth-order time integrators (Adams-Bashforth, Adams-Bashforth-Moulton, and Runge-Kutta). We showed how the use of second-order extrapolation and interpolation of input data as well as corrector steps could significantly improve the accuracy of coupled simulations. Readers are referred to Sprague et al.[11] for details. In our numerical experiments below, we use the fourth-order Adams-Bashforth-Moulton (ABM4) algorithm for



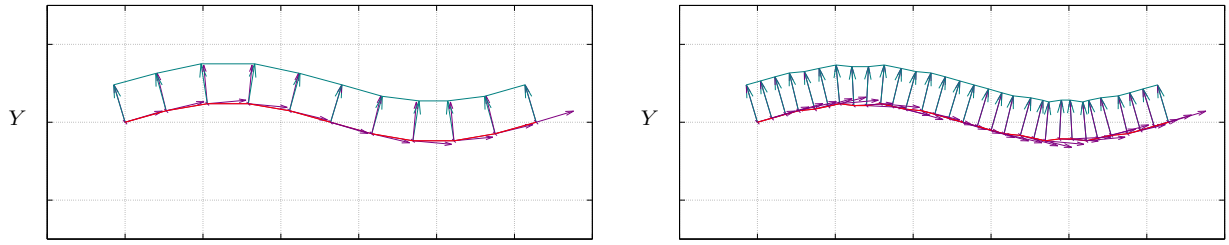
(a) Reference: Line2 element (b) Reference: Line2 elements

Figure 4. Demonstration of the Line2_to_Line2 mapping algorithm: Reference positions and orientations. (a) Reference position and orientation of 10 Line2 interface elements, which have motion output and force/moment input. (b) Reference positions and orientations of nodes along an assembly of 30 Line2 interface elements, which have motion inputs and force/moment outputs. The grey arrows are shown as orthogonal pairs illustrating orientations defined by a DCM at each node.



(a) Source: Line2 element (b) Destination: Line2 elements

Figure 5. Demonstration of the Line2_to_Line2 mapping algorithm: Motion mapping. (a) The source Line2 elements are "flattened" in the positive X-direction as shown. (b) Destination Line2 elements after motions have been mapped. Orientations are shown as orthogonal pairs of purple vectors.

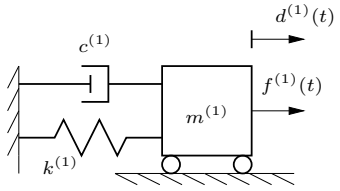


(a) Destination: Line2 element (b) Source: Line2 elements

Figure 6. Demonstration of the Line2_to_Line2 mapping algorithm: Load mapping. (b) Nodes of Source Line2 elements show values of distributed forces (turquoise vectors). (a) Destination Line2 distributed forces after mapping. Also shown are the orientations (orthogonal pairs of purple vectors) from Figure 5.

Table 3. Parameters associated with the test simulations.

Partition 1			Partition 2					Partition 3
$m^{(1)}$	$c^{(1)}$	$k^{(1)}$	$m^{(2)}$	$c^{(2)}$	$k^{(2)}$	$c_c^{(2)}$	$k_c^{(2)}$	$m^{(3)}$
1.0	0.1	3.0	1.0	0.01	0.1	0.01	0.1	2.0



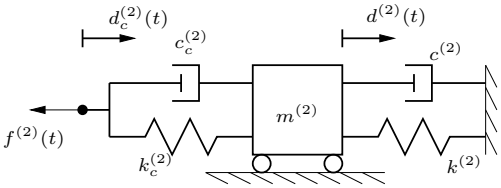
$$\text{GE}: m^{(1)}\ddot{d}^{(1)} + c^{(1)}\dot{d}^{(1)} + k^{(1)}d^{(1)} = f^{(1)}$$

$$\mathbf{x}^{(1)} = \begin{bmatrix} d^{(1)} & \dot{d}^{(1)} \end{bmatrix}^T, \quad \mathbf{x}^{d,(1)} = \emptyset, \quad \mathbf{z}^{(1)} = \emptyset,$$

$$\mathbf{u}^{(1)} = \begin{bmatrix} f^{(1)} \end{bmatrix},$$

$$\mathbf{y}^{(1)} = \begin{bmatrix} d^{(1)} & \dot{d}^{(1)} & \ddot{d}^{(1)} \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}^{(1)} \\ \left(\mathbf{u}^{(1)} - \begin{bmatrix} k^{(1)} & c^{(1)} \end{bmatrix} \mathbf{x}^{(1)}\right)/m^{(1)} \end{bmatrix}$$

(a) Partition 1



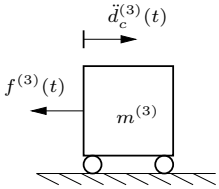
$$\text{GE}: m^{(2)}\ddot{d}^{(2)} + \left(c^{(2)} + c_c^{(2)}\right)\dot{d}^{(2)} + \left(k^{(2)} + k_c^{(2)}\right)d^{(2)} = c_c^{(2)}\dot{d}_c^{(2)} + k_c^{(2)}d_c^{(2)}$$

$$\mathbf{x}^{(2)} = \begin{bmatrix} d^{(2)} & \dot{d}^{(2)} \end{bmatrix}^T, \quad \mathbf{x}^{d,(2)} = \emptyset, \quad \mathbf{z}^{(2)} = \emptyset,$$

$$\mathbf{u}^{(2)} = \begin{bmatrix} d_c^{(2)} & \dot{d}_c^{(2)} \end{bmatrix}^T,$$

$$\mathbf{y}^{(2)} = \begin{bmatrix} f^{(2)} \end{bmatrix} = \begin{bmatrix} k_c^{(2)} & c_c^{(2)} \end{bmatrix} \left(\mathbf{x}^{(2)} - \mathbf{u}^{(2)}\right)$$

(b) Partition 2



$$\text{GE}: m^{(3)}\ddot{d}_c^{(3)} = -f^{(3)}$$

$$\mathbf{x}^{(3)} = \emptyset, \quad \mathbf{x}^{d,(3)} = \emptyset, \quad \mathbf{z}^{(3)} = \emptyset,$$

$$\mathbf{u}^{(3)} = \begin{bmatrix} \ddot{d}_c^{(3)} \end{bmatrix}, \quad \mathbf{y}^{(3)} = \begin{bmatrix} f^{(3)} \end{bmatrix} = -\mathbf{u}^{(3)}m^{(3)}$$

(c) Partition 3

Figure 7. Schematics of three uncoupled model partitions. Governing equations (GEs), states, constraints, inputs, and outputs are shown.

time advancing the solutions to the Partition 1 and Partition 2 governing equations. ABM4 is a multi-step predictor-corrector algorithm that has an implicit dependence on other-partition data in coupled configurations.

2. Partition 1 Coupled to Partition 2

We examine here results for Partition 1 coupled to Partition 2 under free-vibration conditions. Coupling is defined by the following input-output relationships:

$$\mathbf{u}^{(1)} = \mathbf{y}^{(2)}, \tag{27}$$

$$\mathbf{u}^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{y}^{(1)}. \tag{28}$$

The analytically determined benchmark solutions $d_b^{(1)}(t)$ and $d_b^{(2)}(t)$ (in a monolithically coupled form) are shown in Figure 8 for $0 \leq t \leq 50$ and for initial conditions

$$d^{(1)}(0) = 1, \quad \dot{d}^{(1)}(0) = d^{(2)}(0) = \dot{d}^{(2)}(0) = 0. \tag{29}$$

One can see that the magnitude of the response $q^{(2)}$ is significantly smaller than that of $q^{(1)}$, which is due to the small spring constant coupling the two masses.

Numerical experiments with ABM4 simulations of the uncoupled partitions were performed to determine the critical time increments for numerical stability. These were found to $\Delta t_c^{(1)} \approx 0.53$ and $\Delta t_c^{(2)} \approx 2.0$

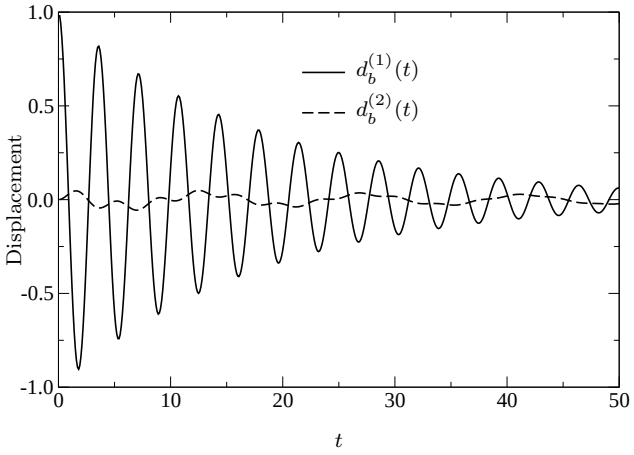


Figure 8. Benchmark solutions for Partition 1 and Partition 2 displacements when coupled.

for Partitions 1 and 2, respectively. We examine coupled simulations where $\Delta t^{(2)} = q^{(2)}\Delta t^I$ and $\Delta t^{(1)} = \Delta t^I$, where $q^{(2)} \geq 1$. In lock-step simulation ($\Delta t^{(1)} = \Delta t^{(2)}$), the critical time increment was found to be $\Delta t_c^{\text{lock}} \approx 0.52$, which is slightly smaller than $\Delta t_c^{(1)}$. Figure 9 shows the critical time increment Δt_c^I normalized by the critical time increment in a coupled lock-step simulation as a function of the time-increment multiplier $q^{(2)}$. We see that updating Partition 2 only every other step ($q^{(2)} = 2$) has virtually no effect on stability. However, increasing $q^{(2)}$ further clearly decreases the maximum allowable time increment for stability. However, updating Partition 2 every fifth step only decreases the critical time increment by half.

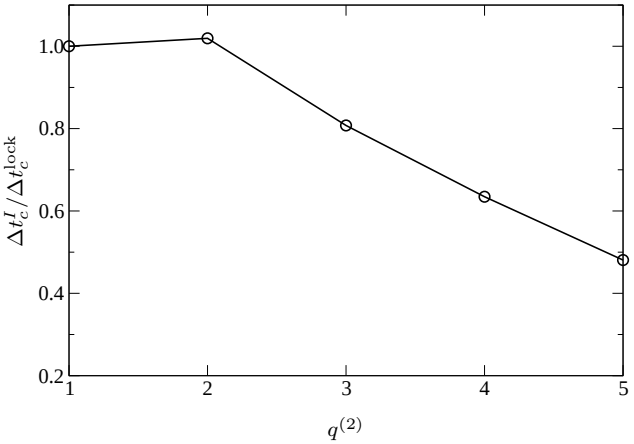


Figure 9. Normalized critical time increments for numerically stable coupled simulations where Partition 1 is coupled to Partition 2 and both were time integrated with ABM4. Simulations were performed with $\Delta t^{(1)} = \Delta t^I$ and $\Delta t^{(2)} = q^{(2)}\Delta t^I$. For lock-step time integration, the critical time increment was $\Delta t_c^{\text{lock}} = 0.52$.

Numerical experiments with ABM4 simulations of the coupled partitions were also performed to examine the effect of the Partition 2 time-increment multiplier $q^{(2)}$ on simulation accuracy. Figure 10 shows the normalized root-mean-square (RMS) error ϵ of simulations calculated over $0 \leq t \leq 50$, where

$$\epsilon = \sqrt{\frac{\sum_{k=0}^{n_{\max}} \left[d_k - d_b(t_k)\right]^2}{\sum_{k=0}^{n_{\max}} \left[d_b(t_k)\right]^2}}, \tag{30}$$

and where d_k is a displacement solution known at $(n_{\max} + 1)$ evenly spaced time stations and $d_b(t)$ is a benchmark solution. Simulations were calculated with $q^{(2)}$ =1 (lock-step), 2, and 4. We see that, with decreasing Δt^I, all methods exhibit approximate third-order accuracy, despite the underlying integrators being fourth-order accurate. The accuracy is limited by the quadratic interpolation/extrapolation of input

and output data (via the function polyfit), which is only third-order accurate. These results are consistent with our earlier studies[11] with Partition 1 coupled to Partition 2; there we found that only by performing a correction step (over the partition coupling) could we restore the fourth-order accuracy of the underlying integrators. We also see that by increasing $q^{(2)}$ beyond unity, we degrade significantly the accuracy (but not the order of the method). For example, in going from $q^{(2)} = 1$ to $q^{(2)} = 2$ with $\Delta t^I = 10^{-2}$, the error increases by a factor of 4. Further, in going from $q^{(2)} = 1$ to $q^{(2)} = 4$, the error increases by a factor of 40.

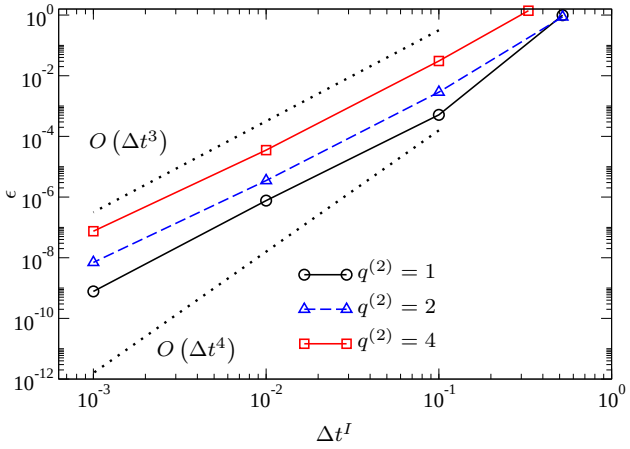


Figure 10. Normalized RMS error of $d^{(1)}(t)$ displacement histories for Partition 1 coupled to Partition 2 for $0 \le t \le 50$. Partitions 1 and 2 were both time integrated with ABM4; $\Delta t^{(1)} = \Delta t^I$ and $\Delta t^{(2)} = q^{(2)} \Delta t^I$. Dotted lines show ideal third-order and fourth-order convergence rates.

3. Partition 1 Coupled to Partition 3

We examine here results for Partition 1 coupled to Partition 3 under free-vibration conditions. Coupling is defined by the following input-output relationships:

$$\mathbf{u}^{(1)} = \mathbf{y}^{(3)}\,, \tag{31}$$

$$\mathbf{u}^{(3)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{y}^{(1)}\,. \tag{32}$$

This coupling is effectively that of a rigid-body connection between the two masses. As such, rigorous solution of the input-output equations is necessary for numerical stability and accuracy. The analytically determined benchmark solution $d_b^{(1)}(t)$ (in a monolithically coupled form) is shown in Figure 11 for $0 \le t \le 100$ and for initial conditions

$$d^{(1)}(0) = 1\,, \quad \dot{d}^{(1)}(0) = 0\,. \tag{33}$$

We performed numerical experiments to examine the numerical stability of the coupled system as in the previous subsection. In lock-step time integration we found the critical time increment to be $\Delta t_c^{\text{lock}} = 0.36$. Figure 12 shows how increasing the time-increment multiplier $q^{(3)}$ decreases the allowable time increment for stable solutions (this decrease goes approximately as $1/q^{(3)}$). The effect is much more significant in this system, which has a rigid connection, compared to the previous system that was connected with a "soft" spring.

Figure 13 shows the normalized RMS error ϵ of $d^{(1)}$ for simulations calculated over $0 \le t \le 100$. Again we see that all simulations exhibit third-order convergence and we see how increasing $q^{(3)}$ decreases significantly the accuracy with trends similar to those shown in Figure 10 for Partition 1 coupled to Partition 2.

C. Sparse-matrix solver

We examined simulation times for three of the FAST certification tests. Jacobians for the input-output equations governing the coupling of modules were calculated with finite differences; the linear systems were either stored in full and solved with LAPACK,[20] or were stored sparsely and solved with MUMPS. Each Jacobian was only factored once for each simulation. For our CPU timing tests, we ran the three fixed-bottom

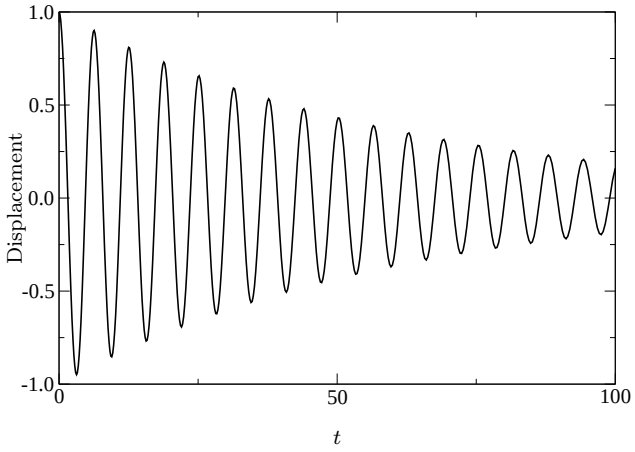


Figure 11. Benchmark solution for Partition 1 when coupled to Partition 3.

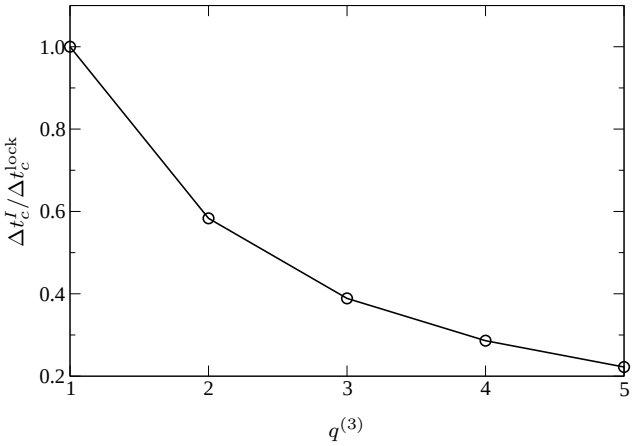


Figure 12. Normalized critical time increment for numerically stable coupled simulations where Partition 1 is coupled to Partition 3 and both were time integrated with ABM4. Simulations were performed with $\Delta t^{(1)} = \Delta t^I$ and $\Delta t^{(3)} = q^{(3)} \Delta t^I$. For lock-step time integration, the critical time increment was $\Delta t_c^{\text{lock}} = 0.36$.

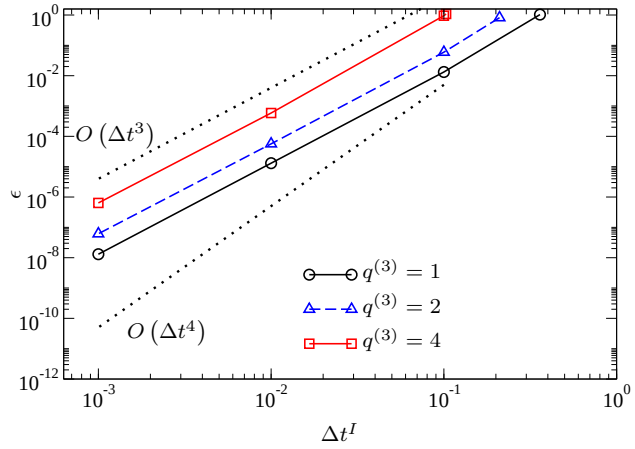


Figure 13. Normalized RMS error of $u^{(1)}(t)$ displacement histories for Partition 1 coupled to Partition 3 for $0 \leq t \leq 100$. Partitions 1 and 3 were both time integrated with ABM4; $\Delta t^{(1)} = \Delta t^I$ and $\Delta t^{(3)} = q^{(3)} \Delta t^I$. Dotted lines show ideal third-order and fourth-order convergence rates.

offshore models (Tests 19-21) that are distributed with FAST v8.09.00. Test 19 is a shortened version of load case 5.2 from OC3 Phase I,[21] using a different turbulent wind file. Test 20 is a slightly shortened version of load case 5.1 from OC3 Phase III.[21] Test 21 is similar to load case 5.7 from OC4 Phase I,[22] but using a 12 m/s turbulent wind file (instead of 18 m/s) and a significant wave height of 8 m (instead of 6 m).

Table 4 shows matrix information and timing results for the three tests cases for one minute of simulated time. For the smallest system, Test 19, there was little timing difference between the dense LAPACK solve and the sparse MUMPS solve. For Test 20, we see that MUMPS factorization was actually longer than that for LAPACK, whereas those times were similar for Test 21. Regardless, MUMPS provided solve times and and total CPU times that were signficantly less than required by LAPACK.

Table 4. Matrix and timing data for three FAST certification tests for one minute of simulated time; calculations were performed in serial processing with LAPACK (full matrix storage and solution) and MUMPS (sparse matrix storage and solution).

	FAST Certification Tests		
	19: OC3 Monopile	20: OC3 Tripod	21: OC4 Jacket
Simulation CPU Time (LAPACK)	0.5 min	4.2 min	10.0 min
Simulation CPU Time (MUMPS)	0.5 min	3.3 min	7.6 min
Jacobian Rank	498	4860	9618
Sparsity (% full)	1%	13%	8%
MUMPS Factorization Time	0.02 s	5.2 s	13.7 s
LAPACK Factorization Time	0.00 s	1.6 s	12.9 s
MUMPS Solve Time per simulated sec	0.03 s	0.8 s	2.8 s
LAPCK Solve Time per simulated sec	0.03 s	1.8 s	5.2 s

IV. Conclusion and Future Work

In this paper we described methods for numerical time integration of multi-physics systems composed of loosely coupled partitions in the FAST modular wind turbine CAE tool. The methods described are appropriate for time integration of modules that have non-matching spatial and temporal grids. We described a suite of element types for defining the discrete spatial interface of a module, which is required to interact with other modules. We provided detailed descriptions of the load and motion algorithms between 2-node line elements and point elements. A significant feature of these algorithms is in their ability to couple highly disparate spatial meshes. For non-matching temporal grids, we expanded the algorithm first described in Sprague et al.[11] The updated time-stepping algorithm allows modules to be time advanced with an increment that is smaller or larger than a given "interaction" time increment. For modules with increments equal to or smaller than the interaction time increment, a predictor-corrector option is provided. For those simulations, a correction step can significantly increase accuracy and stability. For modules with increments larger than the interaction increment, we do not allow correction of states. Numerical experiments showed that partitions with significantly different critical increments could be stably integrated with large sub-cycling ratios. However, the most accuracy was achieved when modules were time integrated in lock step. Whether or not it is advisable to perform simulations with mis-matched time increments depends on the particular modules, their computational cost relative to other modules, and their stiffness/damping/inertia characteristics. Planned future work includes finalization of the mapping algorithms for all of the element types shown in Figure 3. We will also examine the combined use of our algorithms for more complex systems, where non-matching spatial and temporal grids are present.

Appendix: Equations for Mapping Algorithms

A. Introduction

This appendix describes in detail the theoretical formulation of the mapping algorithms for Point_to_Point, Line2_to_Line2, Point_to_Line2, and Line2_to_Point, based on the algorithms outlined in Tables 1 and 2. The mapping search algorithms are presented first in Sections B though D, and are followed by mapping transfer algorithms in Sections E through G. Separate subsections are given for the algorithms of motion and scalar quantities and the algorithms of load quantities. Please note the following conventions used throughout this appendix:

- All vectors are 3×1 and are denoted with an over arrow. Variables without an over arrow denote scalars or 3×3 matrices.
- All vectors are expressed in the global inertial-frame coordinate system (not in a local coordinate system).
- All nodal position and motion quantities are absolute (expressed in the global inertial frame), except for the translational displacements of a node, which are expressed relative to the node reference position.
- All nodal rotations are expressed as 3×3 direction cosine matrices (containing the three components of each of three orthogonal unit vectors of a local coordinate system) and large rotations are permitted without a loss of accuracy (no small-angle assumptions are employed). While direction cosine matrices take nine dependent values to express three independent angles, direction cosine matrices are chosen to express nodal rotations because (1) they uniquely represent a given rotation unlike other rotational parameterizations, such as Euler angles, which depend on the rotation sequence, and (2) they can be used directly through matrix multiplication in the mapping transfer without computationally expensive trigonometric operations. All direction cosine matrices are absolute and orthonormal, and are defined such that premultiplication with a vector expressed in the global inertial-frame transforms the vector to a local coordinate system. Because a direction cosine matrix is orthonormal, the matrix inverse is the transpose of the matrix, so, premultiplication of the matrix-transpose with a vector expressed in a local coordinate system transforms the vector to the global inertial frame.
- No Line2 elements of the destination or source meshes can connect collocated nodes to avoid division-by-zero errors in the mapping-search equations; see Eqs. (36), (37), and (39) below.

Notation

$\vec{a}^D$ and $\vec{a}^S$:	Translational acceleration (absolute) of a node of the destination and source meshes, respectively
d:	distance
$\vec{F}^D$ and $\vec{F}^S$:	Concentrated (lumped) force of a node of the destination and source meshes, respectively
$\vec{f}^D$ and $\vec{f}^S$:	Distributed force (per unit length) of a node of a Line2 element of the destination and source meshes, respectively
I:	Identity matrix
$\overline{\ell}^D$ and $\overline{\ell}^S$:	Normalized location within a Line2 element of the destination and source meshes, respectively
$\vec{M}^D$ and $\vec{M}^S$:	Concentrated (lumped) moment of a node of the destination and source meshes, respectively
$\vec{m}^D$ and $\vec{m}^S$:	Distributed moment (per unit length) of a node of a Line2 element of the destination and source meshes, respectively
$\vec{p}^D$ and $\vec{p}^S$:	Displaced position (absolute) of a node of the destination and source meshes, respectively

$\vec{p}^{DR}$ and $\vec{p}^{SR}$: Reference position (absolute) of a node of the destination and source meshes, respectively

S^D and S^S: Scalar quantity of a node of the destination and source meshes, respectively

$\vec{u}^D$ and $\vec{u}^S$: Translational displacement (relative) of a node of the destination and source meshes, respectively

$\vec{v}^D$ and $\vec{v}^S$: Translational velocity (absolute) of a node of the destination and source meshes, respectively

XYZ: Axes of the global inertial frame

$\vec{\alpha}^D$ and $\vec{\alpha}^S$: Rotational acceleration (absolute) of a node of the destination and source meshes, respectively

$\mathbf{\Lambda}^D$ and $\mathbf{\Lambda}^S$: Displaced rotation (absolute orientation; direction cosine matrix) of a node of the destination and source meshes, respectively

$\mathbf{\Lambda}^{DR}$ and $\mathbf{\Lambda}^{SR}$: Reference rotation (absolute orientation; direction cosine matrix) of a node of the destination and source meshes, respectively

$\vec{\omega}^D$ and $\vec{\omega}^S$: Rotational velocity (absolute) of a node of the destination and source meshes, respectively

B. Point_to_Point Mapping Search

1. Motion and Scalar Quantities

For each Point-element node of the destination mesh, a nearest-neighbor Point-element node of the source mesh is found in the reference configuration. A source-mesh Point-element node may be associated with multiple destination-mesh Point-element nodes. That is, for each node of the destination mesh, the node of the source mesh that is the minimum distance away is found, calculated as distance in the reference configuration, $d \geq 0$, where

$$d = \left\| \vec{p}^{SR} - \vec{p}^{DR} \right\|_2 , \tag{34}$$

$\| \cdot \|_2$ denotes the vector two-norm (vector magnitude), and $\vec{p}^{DR}$ and $\vec{p}^{SR}$ contain the X, Y, Z coordinates (absolute, relative to the global inertial-frame origin) of a node of the destination mesh and source mesh in its reference (undisplaced) position, respectively.

2. Load Quantities

For each Point-element node of the source mesh, a nearest-neighbor Point-element node of the destination mesh is found in the reference configuration. A destination-mesh Point-element node may be associated with multiple source-mesh Point-element nodes. That is, for each node of the source mesh, the node of the destination mesh that is the minimum distance away is found, calculated as distance $d \geq 0$ from Eq. (34) above, is found in the reference configuration.

C. Line2_to_Line2 Mapping Search

1. Motion and Scalar Quantities

For each Line2-element node of the destination mesh, a nearest-neighbor Line2 element of the source mesh is found in the reference configuration, for which the destination Line2-element node projects orthogonally onto the source Line2-element domain. A source-mesh Line2 element may be associated with multiple destination-mesh Line2-element nodes. That is, for each node of the destination mesh, an orthogonal projection is made onto all possible Line2 elements of the source mesh and the Line2 element of the source mesh that is the minimum distance away is found, calculated as distance in the reference configuration, $d \geq 0$, where

$$d = \left\| \vec{p}^{DR} - \vec{p}_1^{SR} \left(1 - \overline{\ell}^S \right) - \vec{p}_2^{SR} \overline{\ell}^S \right\|_2 . \tag{35}$$

Here, subscripts 1 and 2, respectively, denote the 1st and 2nd nodes of the Line2 element of the source mesh, $0 \leq \overline{\ell}^S \leq 1$ is the normalized location within the Line2 element of the source mesh where the orthogonal projection intersects, where

$$\overline{\ell}^S = \frac{\left(\vec{p}_2^{SR} - \vec{p}_1^{SR}\right) \cdot \left(\vec{p}^{DR} - \vec{p}_1^{SR}\right)}{\left(\vec{p}_2^{SR} - \vec{p}_1^{SR}\right) \cdot \left(\vec{p}_2^{SR} - \vec{p}_1^{SR}\right)}, \tag{36}$$

and $\vec{p}_1^{SR}\left(1 - \overline{\ell}^S\right) + \vec{p}_2^{SR}\overline{\ell}^S$ is the absolute position of the intersection point in the reference configuration. (Only projections between and including a Line2 element's two nodes are considered.) The terms $\left(1 - \overline{\ell}^S\right)$ and $\overline{\ell}^S$ in Eq. (35) are the linear shape functions of the 1st and 2nd nodes, respectively, of the Line2 element of the source mesh evaluated at the intersection point. In Eq. (36), $\cdot$ denotes the vector dot product. If there is no Line2 element of the source mesh that a given node of the destination mesh projects onto, whereby $0 \leq \overline{\ell}^S \leq 1$, the mapping is aborted with an error.

2. Load Quantities

An augmented Line2-element source mesh is first formed by splitting the original Line2-element source mesh at each location where a destination-mesh Line2-element node projects orthogonally from the destination mesh. That is, for each Line2 element of the destination mesh and for both nodes of that element, a plane normal to the element and including the node is formed and all possible points where Line2 elements of the source mesh intersect this plane are found in the reference configuration. The normalized locations within a Line2 element of the source mesh where this plane is intersected, $0 < \overline{\ell}^S < 1$, are given by

$$\overline{\ell}^S = \frac{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}^{DR} - \vec{p}_1^{SR}\right)}{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}_2^{SR} - \vec{p}_1^{SR}\right)}. \tag{37}$$

The augmented source mesh is formed by introducing a new node in the original source mesh at each intersection point with reference position $\vec{p}_1^{SR}\left(1 - \overline{\ell}^S\right) + \vec{p}_2^{SR}\overline{\ell}^S$ and splitting the associated element in the original source mesh in two. (Only intersections between a Line2 element's two nodes are considered. To avoid introducing collocated nodes, new nodes are not introduced at the original node reference positions, $\overline{\ell}^S = 0$ or $\overline{\ell}^S = 1$.) No new nodes are introduced if no Line2 element of the source mesh intersect this plane.

For each Line2-element node of the augmented source mesh, a nearest-neighbor Line2 element of the destination mesh is found in the reference configuration, for which the source Line2-element node projects orthogonally onto the destination Line2-element domain. A destination-mesh Line2 element may be associated with multiple source-mesh Line2-element nodes. That is, for each node of the augmented source mesh, an orthogonal projection is made onto all possible Line2 elements of the destination mesh and the Line2 element of the destination mesh that is the minimum distance away is found, calculated as distance in the reference configuration, $d \geq 0$, where

$$d = \left\| \vec{p}^{SR} - \vec{p}_1^{DR}\left(1 - \overline{\ell}^D\right) - \vec{p}_2^{DR}\overline{\ell}^D \right\|_2, \tag{38}$$

and where $0 \leq \overline{\ell}^D \leq 1$ is the normalized location within the Line2 element of the destination mesh where the orthogonal projection intersects, which is calculated as

$$\overline{\ell}^D = \frac{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}^{SR} - \vec{p}_1^{DR}\right)}{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right)}. \tag{39}$$

(Only projections between and including a Line2 element's two nodes are considered.) If there is no Line2 element of the destination mesh that a given node of the augmented source mesh projects onto, whereby $0 \leq \overline{\ell}^D \leq 1$, the mapping is aborted with an error.

D. Point_to_Line2 and Line2_to_Point Mapping Search

1. Motion and Scalar Quantities

In the Point_to_Line2 mapping search for motion and scalar quantities, for each node of the Line2-element destination mesh, a nearest-neighbor Point-element node of the source mesh is found in the reference configuration in a manner identical to the Point_to_Point motion-mapping search (see Section B.1).

In the Line2_to_Point mapping search for motion and scalar quantities, for each Point-element node of the destination mesh, a nearest-neighbor Line2 element of the source mesh is found in the reference configuration in a manner identical to the Line2_to_Line2 motion-mapping search (see Section C.1).

2. Load Quantities

In the Point_to_Line2 mapping search for load quantities, for each Point-element node of the source mesh, a nearest-neighbor Line2 element of the destination mesh is found in the reference configuration in a manner identical to the Line2_to_Line2 load-mapping search (see Section C.2, but without augmentation of the source mesh).

In the Line2_to_Point mapping search for load quantities, an augmented Line2-element source mesh is first formed by splitting the original Line2-element source mesh at each location where a destination-mesh Point-element node projects orthogonally onto the Line2-element source mesh. That is, for each node of the destination mesh, an orthogonal projection is made onto all possible Line2 elements of the source mesh in the reference configuration. The normalized locations within a Line2 element of the source mesh where the orthogonal projection intersects, $0 < \overline{\ell}^S < 1$, are given by Eq. (36). The augmented source mesh is formed by introducing a new node in the original source mesh at each intersection point with reference position $\vec{p}_1^{\,SR}\left(1-\overline{\ell}^S\right)+\vec{p}_2^{\,SR}\overline{\ell}^S$ and splitting the associated element in the original source mesh in two. (Only projections between a Line2 element's two nodes are considered. To avoid introducing collocated nodes, new nodes are not introduced at the original node reference positions, $\overline{\ell}^S = 0$ or $\overline{\ell}^S = 1$.) No new nodes are introduced if there are no orthogonal projections onto a Line2 element of the source mesh.

For each node of the augmented Line2-element source mesh, a nearest-neighbor Point-element node of the destination mesh is found in the reference configuration in a manner identical to the Point_to_Point load-mapping search (see Section B.2).

E. Point_to_Point Mapping Transfer

1. Motion and Scalar Quantities

For each destination-mesh Point-element node, motion and scalar quantities are transferred from its mapped source Point-element node. In the case that the source and destination Point-element nodes are not coincident in the current configuration, rotations and moment arms (including displacements) are used to augment transferred translations such that overall motion is maintained.

The mapping transfer of translational displacement is given by

$$\vec{u}^D = \vec{u}^S + \left[I - \left(\mathbf{\Lambda}^S\right)^T \mathbf{\Lambda}^{SR}\right]\left(\vec{p}^{\,SR} - \vec{p}^{\,DR}\right) , \tag{40}$$

where $\vec{u}^D$ and $\vec{u}^S$ are 3×1 vectors containing the X, Y, Z translational displacements (relative to the node reference position) of a node of the destination mesh and mapped node of the source mesh, respectively,[a] I is the 3×3 identity matrix, $\mathbf{\Lambda}^{SR}$ is the reference rotation (direction cosine matrix) of the mapped node of the source mesh, $\mathbf{\Lambda}^S$ is the displaced rotation (direction cosine matrix) of the mapped node of the source mesh, and ^{T} denotes the transpose of a matrix. The second term on the right-hand side (RHS) of Eq. (40) represents the translation displacement of a node of the destination mesh due to rigid-body rotation of the mapped node of the source mesh from its reference orientation and is zero if the reference and displaced rotations of the mapped node of the source mesh are coincident (no rotational displacement), whereby $\left(\mathbf{\Lambda}^S\right)^T \mathbf{\Lambda}^{SR} = I$, or if the reference positions of the node of the destination mesh and mapped node of the source mesh are coincident, whereby $\vec{p}^{\,DR} = \vec{p}^{\,SR}$.

The mapping transfer of displaced rotation is given by

$$\mathbf{\Lambda}^D = \mathbf{\Lambda}^{DR}\left(\mathbf{\Lambda}^{SR}\right)^T \mathbf{\Lambda}^S , \tag{41}$$

where $\mathbf{\Lambda}^{DR}$ is the reference rotation (direction cosine matrix) of a node of the destination mesh and $\mathbf{\Lambda}^D$ is the displaced rotation (direction cosine matrix) of that node. The node of the destination mesh need not have the same reference orientation of the mapped node of the source mesh, but the node of the destination

[a]The absolute displaced positions of a node of the destination and source meshes, respectively, are related to the reference position and relative translational displacement of the node by $\vec{p}^{\,D} = \vec{p}^{\,DR} + \vec{u}^D$ and $\vec{p}^{\,S} = \vec{p}^{\,SR} + \vec{u}^S$, respectively.

mesh will still rotate the same amount as the mapped node of the source mesh (as a rigid body). The node of the destination mesh is not rotated if the reference and displaced rotations of the mapped node of the source mesh are coincident (no rotational displacement).

The mapping transfer of translational and rotation velocities is given by

$$\vec{v}^D = \vec{v}^S + \left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{\omega}^S \,, \tag{42}$$

$$\vec{\omega}^D = \vec{\omega}^S \,, \tag{43}$$

where $\vec{v}^D$ and $\vec{v}^S$ are the translational velocities of a node of the destination mesh and mapped node of the source mesh, respectively, and $\vec{\omega}^D$ and $\vec{\omega}^S$ are the rotational velocities of a node of the destination mesh and mapped node of the source mesh, respectively, and $\times$ denotes the vector cross product. The second term on the RHS of Eq. (42) represents the translation velocity of a node of the destination mesh due to the displaced offset between the node of the destination mesh and mapped node of the source mesh and the rotational velocity of the mapped node of the source mesh.[b] The node of the destination mesh will rotate the same as the mapped node of the source mesh (as a rigid body).

The mapping transfer of translational and rotation accelerations is given by

$$\vec{a}^D = \vec{a}^S + \left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{\alpha}^S + \vec{\omega}^S \times \left\{\left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{\omega}^S\right\} \,, \tag{44}$$

$$\vec{\alpha}^D = \vec{\alpha}^S \,, \tag{45}$$

where $\vec{a}^D$ and $\vec{a}^S$ are the translational accelerations of a node of the destination mesh and mapped node of the source mesh, respectively, and $\vec{\alpha}^D$ and $\vec{\alpha}^S$ are the rotational accelerations of a node of the destination mesh and mapped node of the source mesh, respectively. The second term on the RHS of Eq. (44) represents the tangential acceleration of a node of the destination mesh due to the displaced offset between the node of the destination mesh and mapped node of the source mesh and the rotational acceleration of the mapped node of the source mesh. The third term on the RHS of Eq. (44) represents the centripetal acceleration of a node of the destination mesh due to the displaced offset between the node of the destination mesh and mapped node of the source mesh and the rotational velocity of the mapped node of the source mesh. The node of the destination mesh will rotate the same as the mapped node of the source mesh (as a rigid body).

The mapping transfer of scalar quantities is given by

$$S^D = S^S \,, \tag{46}$$

where S^D and S^S are arrays of one or more scalar quantities of a node of the destination mesh and mapped node of the source mesh, respectively.

2. Load Quantities

For each source-mesh Point-element node, forces and moments are transferred to its mapped destination Point-element node; forces and moments are superposed when a destination element has more than one source element. In the case that the source and destination Point-element nodes are not coincident in the current configuration, forces and moment arms (including displacements) are used to augment transferred moments such that the overall load balance is maintained.

The mapping transfer of concentrated (lumped) forces and moments,

$$\vec{F}^D = \sum \vec{F}^S \,, \tag{47}$$

$$\vec{M}^D = \sum \left\{\vec{M}^S + \left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{F}^S\right\} \,, \tag{48}$$

where $\vec{F}^S$ and $\vec{F}^D$ are the concentrated forces of a node of the source mesh and mapped node of the destination mesh, respectively, and $\vec{M}^S$ and $\vec{M}^D$ are the concentrated moments of a node of the source mesh and mapped node of the destination mesh, respectively. The second term on the RHS of Eq. (48) represents the additional moment of the mapped node of the destination mesh due to the displaced offset (moment

[b] Rearrangement of Eq. (40) reveals that $\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right) = \left(\mathbf{\Lambda}^S\right)^T \mathbf{\Lambda}^{SR} \left(\vec{p}^{SR} - \vec{p}^{DR}\right)$, so, the term involving translational displacements of the nodes of the destination and source meshes in Eq. (42) and other mapping-transfer equations could alternatively be written in terms of rotations of the source mesh.

arm) between the node of the source mesh and mapped node of the destination mesh and the concentrated force of the node of the source mesh.[c] The summations in Eqs. (47) and (48) denote the superposition of loads when a destination element has more than one mapped source element.

F. Line2_to_Line2 Mapping Transfer

1. *Motion and Scalar Quantities*

For each destination-mesh Line2-element node, motion and scalar quantities are interpolated (based on projection) and are transferred from its mapped source Line2 element. In the case that the destination Line2-element node does not lie in its source Line2-element domain in the current configuration, interpolated rotations and moment arms (including displacements) are used to augment transferred translations such that overall motion is maintained.

The mapping transfer of all motion and scalar quantities (now including orientation as described in Section II.C), via interpolation based on the projected location in the source Line2 element, is given by

$$\left(\,\cdot\,\right) = \left(\,\cdot\,\right)_1 \left(1 - \overline{\ell}^S\right) + \left(\,\cdot\,\right)_2 \overline{\ell}^S \,, \tag{49}$$

where the $\left(\,\cdot\,\right)$ on the left-hand side (LHS) of Eq. (49) is a placeholder for $\vec{u}^D$, $\mathbf{\Lambda}^D$, $\vec{v}^D$, $\vec{\omega}^D$, $\vec{a}^D$, $\vec{\alpha}^D$, or S^D from the LHS of Eqs. (40) through (46), respectively, and $\left(\,\cdot\,\right)$ on the RHS of Eq. (49) is a placeholder for the corresponding RHS of Eqs. (40) through (46), with subscripts 1 and 2, respectively, denoting the 1st and 2nd nodes of the mapped Line2 element of the source mesh. $\overline{\ell}^S$ used in Eq. (49) was solved via Eq. (36) from the Line2_to_Line2 mapping search for motion and scalar quantities (see Section C.1). The motion quantities are not interpolated if the projection lies on a node, whereby $\overline{\ell}^S = 0$ or $\overline{\ell}^S = 1$.

2. *Load Quantities*

The fields of the new nodes of the augmented source mesh are first populated via interpolation of the fields from the original nodes of the source mesh. That is, Eq. (49) (where $\left(\,\cdot\,\right)$ is a placeholder) is used to calculate $\vec{u}^S$, $\vec{f}^S$, and $\vec{m}^S$ at the new nodes of the augmented source mesh, where $\overline{\ell}^S$ was solved via Eq. (37) from the Line2_to_Line2 mapping search for load quantities (see Section C.2), $\vec{f}^S$ is the distributed force (per unit length) of a node of a Line2 element of the source mesh, and $\vec{m}^S$ is the distributed moment (per unit length) of a node of a Line2 element of the source mesh.

For each Line2 element of the augmented source mesh, distributed loads are lumped as point loads at the two nodes (of the source Line2 element) such that the lumped loads maintain the overall load balance with the Line2-element distributed loads; lumped loads are superposed at nodes shared by multiple elements. For each Line2-element node of the augmented source mesh, the lumped load is split based on its projected location in the mapped destination Line2 element, and is transferred as two point loads at the destination Line2-element nodes. Forces and moments are superposed when a destination Line2-element node has more than one source element. In the case that the source Line2-element node does not lie in its destination Line2-element domain in the current configuration, forces and moment arms (including displacements) are used to augment transferred moments such that the overall load balance is maintained. The transferred point loads are transformed to distributed loads that maintain the overall load balance.

The lumping of distributed forces and moments to concentrated point forces and moments at the two nodes of each Line2 element of the augmented source mesh is given by

$$\vec{F}_1^S = \sum \frac{\left\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\right\|_2}{6} \left(2\vec{f}_1^S + \vec{f}_2^S\right) , \tag{50}$$

$$\vec{F}_2^S = \sum \frac{\left\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\right\|_2}{6} \left(\vec{f}_1^S + 2\vec{f}_2^S\right) , \tag{51}$$

[c] While it is rare for a single mesh to contain both motion and load quantities, the mapping transfer for moments uses both load and translational displacement quantities. In the actual source-code implementation, this is achieved by sending multiple meshes (one with load quantities and another with motion quantities) to the mapping-transfer routine for load quantities.

$$\vec{M}_1^S = \sum \frac{\left\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\right\|_2}{6} \left\{ 2\vec{m}_1^S + \vec{m}_2^S + \left[\left(\vec{p}_2^{SR} + \vec{u}_2^S\right) - \left(\vec{p}_1^{SR} + \vec{u}_1^S\right)\right] \times \left(\frac{\vec{f}_1^S + \vec{f}_2^S}{2}\right) \right\}, \tag{52}$$

$$\vec{M}_2^S = \sum \frac{\left\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\right\|_2}{6} \left\{ \vec{m}_1^S + 2\vec{m}_2^S - \left[\left(\vec{p}_2^{SR} + \vec{u}_2^S\right) - \left(\vec{p}_1^{SR} + \vec{u}_1^S\right)\right] \times \left(\frac{\vec{f}_1^S + \vec{f}_2^S}{2}\right) \right\}, \tag{53}$$

where the second term on the RHS of Eqs. (52) and (53) represents the additional lumped moment of a node of the source mesh due to the distributed force. As shown by Eqs. (50)–(53), both nodal distributed loads of a given Line2 element contribute to the lumped load of each node of that element. The summations in Eqs. (50)–(53) denote the superposition of loads when a given node of the source mesh is connected to multiple elements. Equations (50)–(53) were derived by integrating the nodal linear shape functions multiplied by the distributed loads (including moment arms for moments) along each source element.

The mapping transfer of lumped load quantities via splitting based on the projected location in the mapped destination Line2 element is given by

$$\left(\,\cdot\,\right)_1 = \sum \left(\,\cdot\,\right) \left(1 - \overline{\ell}^D\right), \tag{54}$$

$$\left(\,\cdot\,\right)_2 = \sum \left(\,\cdot\,\right) \overline{\ell}^D, \tag{55}$$

where $\left(\,\cdot\,\right)$ on the LHS of Eqs. (54) and (55) is a placeholder for $\vec{F}^D$ and $\vec{M}^D$ from the LHS of Eqs. (47) and (48), respectively, and $\left(\,\cdot\,\right)$ on the RHS of Eqs. (54) and (55) is a placeholder for the corresponding RHS of Eqs. (47) and (48) (but without the summations), with subscripts 1 and 2, respectively, denoting the 1st and 2nd nodes of the mapped Line2 element of the destination mesh. $\overline{\ell}^D$ used in Eqs. (54) and (55) was solved via Eq. (39) from the Line2_to_Line2 mapping search for load quantities (see Section C.2). The load quantities are not split if the projection lies on a node, whereby $\overline{\ell}^D = 0$ or $\overline{\ell}^D = 1$. The summations in Eqs. (54) and (55) denote the superposition of loads when a destination element has more than one mapped source element.

To transform the lumped nodes of the destination mesh to distributed loads,

$$\vec{F}_1^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left(2\vec{f}_1^D + \vec{f}_2^D\right), \tag{56}$$

$$\vec{F}_2^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left(\vec{f}_1^D + 2\vec{f}_2^D\right), \tag{57}$$

$$\vec{M}_1^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left\{ 2\vec{m}_1^D + \vec{m}_2^D + \left[\left(\vec{p}_2^{DR} + \vec{u}_2^D\right) - \left(\vec{p}_1^{DR} + \vec{u}_1^D\right)\right] \times \left(\frac{\vec{f}_1^D + \vec{f}_2^D}{2}\right) \right\}, \tag{58}$$

$$\vec{M}_2^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left\{ \vec{m}_1^D + 2\vec{m}_2^D - \left[\left(\vec{p}_2^{DR} + \vec{u}_2^D\right) - \left(\vec{p}_1^{DR} + \vec{u}_1^D\right)\right] \times \left(\frac{\vec{f}_1^D + \vec{f}_2^D}{2}\right) \right\}, \tag{59}$$

are solved inversely, where $\vec{f}^D$ is the distributed force (per unit length) of a node of a Line2 element of the destination mesh, and $\vec{m}^D$ is the distributed moment (per unit length) of a node of a Line2 element of the destination mesh. Akin to Eqs. (50)–(53) for the source mesh, Eqs. (56)–(59) express the lumping of distributed forces and moments to concentrated point forces and moments at the two nodes of each Line2 element of the destination mesh. (As implied by the summations in Eqs. (56)–(59), which denote the superposition of loads when a given node of the destination mesh is connected to multiple elements, the inverse of Eqs. (56)–(59) depends on the element connectivity; the inverse of Eqs. (56)–(59) cannot be expressed in closed form until the element connectivity is defined.)

G. Point_to_Line2 and Line2_to_Point Mapping Transfer

1. Motion and Scalar Quantities

In the Point_to_Line2 mapping transfer for motion and scalar quantities, for each destination-mesh Line2-element node, motion and scalar quantities are transferred from its mapped source Point-element node in a manner identical to Point_to_Point motion-mapping transfer (see Section E.1).

In the Line2_to_Point mapping transfer for motion and scalar quantities, for each destination-mesh Point-element node, motion and scalar quantities are interpolated (based on projection) and are transferred from its mapped source Line2 element in a manner identical to the Line2_to_Line2 motion-mapping transfer (see Section F.1).

2. Load Quantities

In the Point_to_Line2 mapping transfer for load quantities, for each source-mesh Point-element node, the point load is split based on its projected location in the mapped destination Line2 element, and is transferred as two point loads at the destination Line2-element nodes and transformed to distributed loads in a manner identical to the Line2_to_Line2 load-mapping transfer (see Section F.2, but without augmentation and lumping of the source mesh).

In the Line2_to_Point mapping transfer for load quantities, the fields of the new nodes of the augmented source mesh are first populated via interpolation of the fields from the original nodes of the source mesh. That is, Eq. (49) (where $\left(\cdot\right)$ is a placeholder) is used to calculate $\vec{u}^S$, $\vec{f}^S$, and $\vec{m}^S$ at the new nodes of the augmented source mesh, where $\overline{\ell}^S$ was solved via Eq. (36) as discussed in the Line2_to_Point mapping search for load quantities (see Section D.2).

For each Line2 element of the augmented source mesh, distributed loads are lumped as point loads at the two nodes (of the source Line2 element) such that the lumped loads maintain the overall load balance with the Line2-element distributed loads; lumped loads are superposed at nodes shared by multiple elements in a manner identical to lumping in the Line2_to_Line2 load mapping (see Section F.2). The lumped nodal loads from each Line2-element node of the augmented source mesh are transferred to its mapped destination Point-element node in a manner identical to Point_to_Point load mapping (see Section E.2).

Acknowledgments

This work was performed at NREL in support of the U.S. Department of Energy under contract number DE-AC36-08GO28308 and under a project funded through topic area 1.1 of the Funding Opportunity Announcement (FOA) number DE-FOA-0000415.

References

[1]Jonkman, J. M. and Buhl, M. L., "FAST User's Guide," Tech. Rep. NREL/EL-500-38230, National Renewable Energy Laboratory, Golden, CO, August 2005.

[2]Laino, D. J. and Hansen, A. C., "Users Guide to the Wind Turbine Dynamics Aerodynamics Computer Software AeroDyn," December 2002, Windward Engineering LLC. Prepared for the National Renewable Energy Laboratory under Subcontract No. TCX-9-29209-01.

[3]Moriarty, P. J. and Hansen, A. C., "AeroDyn Theory Manual," Tech. Rep. NREL/EL-500-36881, National Renewable Energy Laboratory, Golden, CO, December 2005.

[4]Jonkman, J. M., *Dynamics Modeling and Loads Analysis of an Offshore Floating Wind Turbine*, Ph.D. thesis, Department of Aerospace Engineering Sciences, University of Colorado, Boulder, CO, 2007, also Tech. Rep. NREL/TP-500-41958, National Renewable Energy Laboratory, Golden, CO.

[5]Jonkman, J. M., "Dynamics of Offshore Floating Wind Turbines–Model Development and Verification," *Wind Energy*, Vol. 12, 2009, pp. 459–492, also Tech. Rep. NREL/JA-500-45311 National Renewable Energy Laboratory, Golden, CO.

[6]Wang, Q., Johnson, N., Sprague, M. A., and Jonkman, J., "BeamDyn: A High-Fidelity Wind Turbine Blade Solver in the FAST Modular Framework," *Proceedings of SciTech 2015*, 2015, submitted.

[7]Jonkman, J. M., "The New Modularization Framework for the FAST Wind Turbine CAE Tool," *Proceedings of the 51st AIAA Aerospace Sciences Meeting*, 2013, also Tech. RepNREL/CP-5000-57228, National Renewable Energy Laboratory, Golden, CO.

[8]Schiesser, W. E., *The Numerical Method of Lines: Integration of Partial Differential Equations*, Academic Press, San Diego, 2001.

[9]Gasmi, A., Sprague, M. A., Jonkman, J. M., and Jones, W. B., "Numerical Stability and Accuracy of Temporally Coupled Multi-Physics Modules in Wind-Turbine CAE Tools," *Proceedings of the 51st Aerospace Sciences Meeting*, 2013, also published in tech. report NREL/CP-2C00-57298, National Renewable Energy Laboratory, Golden, CO.

[10]Felippa, C. A., Park, K. C., and Farhat, C., "Partitioned Analysis of Coupled Mechanical Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 190, 2001, pp. 3247–3270.

[11]Sprague, M. A., Jonkman, J. M., and Jonkman, B. J., "FAST Modular Wind Turbine CAE Tool: Nonmatching Spatial and Temporal Meshes," *Proceedings of the AIAA SciTech 2014*, National Harbor, Maryland, 2014, also published in tech. report NREL/CP-2C00-60742, National Renewable Energy Laboratory, Golden, CO.

[12]Jonkman, B. J., Michalakes, J., Jonkman, J. M., Buhl, M. L., Platt, A., and Sprague, M. A., "NWTC Programmer's

Handbook: A Guide for Software Development within the FAST Computer-Aided Engineering Tool," Tech. rep., National Renewable Energy Laboratory, Golden, CO, 2013, to be published.

[13]Farhat, C., Lesoinne, M., and LeTallec, P., "Load and Motion Transfer Algorithms for Fluid/Structure Interaction Problems with Non-Matching Discrete Interfaces: Momentum and Energy Conservation, Optimal Discretization and Application to Aeroelasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 157, 1998, pp. 95–114.

[14]Sprague, M. A. and Geers, T. L., "A Spectral-Element Method for Modeling Cavitation in Transient Fluid-Structure Interaction," *International Journal for Numerical Methods in Engineering*, Vol. 60, 2004, pp. 2467–2499.

[15]Mota, A., Sun, W. C., Ostien, J., III, J. F., and Long, K., "Lie-group interpolation and variational recovery for internal variables," *Computational Mechanics*, Vol. 52, 2013, pp. 1281–1299.

[16]Amestoy, P. R., Duff, I. S., Koster, J., and LExcellent, J.-Y., "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal of Matrix Analysis and Applications*, Vol. 23, 2001, pp. 15–41.

[17]Amestoy, P. R., Guermouche, A., LExcellent, J.-Y., and Pralet, S., "Hybrid scheduling for the parallel solution of linear systems," *Parallel Computing*, Vol. 32, 2006, pp. 136–156.

[18]*MUltifrontal Massively Parallel Solver (MUMPS 4.10.0) Users guide*, 2011.

[19]Karypis, G. and Kumar, V., *METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices - Version 4.0*, University of Minnesota, 1998.

[20]Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd ed., 1999.

[21]Jonkman, J. and Musial, W., "Offshore Code Comparison Collaboration (OC3) for IEA Task 23 Offshore Wind Technology and Deployment," Tech. Rep. NREL/TP-500-48191, National Renewable Energy Laboratory, Golden, CO, 2010.

[22]Popko, W., Vorpahl, F., Zuga, A., Kohlmeier, M., Jonkman, J., Robertson, A., Larsen, T. J., Yde, A., Saetertro, K., Okstad, K. M., Nichols, J., Nygaard, T. A., Gao, Z., Manolas, D., Kim, K., Yu, Q., Shi, W., Park, H., and Vasquez-Rojas, A., "Offshore Code Comparison Collaboration Continuation (OC4), Phase I - Results of Coupled Simulations of an Offshore Wind Turbine with Jacket Support Structure," Tech. Rep. CP-5000-54124, National Renewable Energy Laboratory, Golden, CO, 2012.