FAST Modular Wind Turbine CAE Tool: Nonmatching Spatial and Temporal Meshes

Preprint

M. A. Sprague, J. M. Jonkman, and
B. J. Jonkman

To be presented at SciTech2014
National Harbor, Maryland
January 13-17, 2014

Conference Paper
NREL/CP-2C00-60742
January 2014

Contract No. DE-AC36-08GO28308

Cover Photos: (left to right) photo by Pat Corkery, NREL 16416, photo from SunEdison, NREL 17423, photo by Pat Corkery, NREL 16560, photo by Dennis Schroeder, NREL 17613, photo by Dean Armstrong, NREL 17436, photo by Pat Corkery, NREL 17721.

Printed on paper containing at least 50% wastepaper, including 10% post consumer waste.

FAST Modular Wind Turbine CAE Tool: Nonmatching Spatial and Temporal Meshes[*]

Michael A. Sprague[†], Jason M. Jonkman[‡], and Bonnie J. Jonkman[§]

National Renewable Energy Laboratory, Golden, Colorado, 80401, USA

In this paper we propose and examine numerical algorithms for coupling time-dependent multi-physics modules relevant to computer-aided engineering (CAE) of wind turbines. In particular, we examine algorithms for coupling modules where spatial grids are non-matching at interfaces and module solutions are time advanced with different time increments and different time integrators. The new mesh-mapping algorithm supports mapping between spatial meshes that are highly disparate. Sharing of data between modules is accomplished with a predictor-corrector approach, which allows for either implicit or explicit time integration within each module. Algorithms are presented in a general framework, but are applied to simple problems that are representative of the systems found in a whole-turbine analysis. Numerical experiments are used to explore the stability, accuracy, and efficiency of the proposed algorithms. This work is motivated by an in-progress major revision of FAST, the National Renewable Energy Laboratory's (NREL's) premier aero-elastic CAE simulation tool. The algorithms described here will greatly increase the flexibility and efficiency of FAST.

I. Introduction

In this paper we examine the numerical stability and accuracy of several coupling methods for multi-physics modules relevant to computer-aided engineering (CAE) of wind turbines. Our focus is coupling of modules that have spatial grids that are, in general, non-matching at module interfaces and have different time increments and time integrators. The wind turbine system is composed of many physics that have significantly different characteristic time and length scales. The wind turbine industry relies heavily on CAE tools for analyzing wind turbine performance, loading, and mechanical 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]), platform hydrodynamics (HydroDyn[4,5]) for offshore systems, control and electrical systems (ServoDyn), and structural dynamics (ElastoDyn). 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. The wind turbine can be modeled on land or offshore on fixed-bottom or floating substructures.

This work is motivated by a major restructuring of the FAST tool suite, which is described in Jonkman.[6] FAST-restructuring goals include (1) improving the ability to read, implement, and maintain source code;

[*]The submitted manuscript has been offered by employees of the Alliance for Sustainable Energy, LLC (Alliance), a contractor of the U.S. Government under Contract No. DE-AC36-08GO28308. Accordingly, the U.S. Government and Alliance retain a nonexclusive royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

[†]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.

(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 non-linear, 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.[7]

In Gasmi et al.,[8] 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). This paper focuses on the modification and extension of the temporal coupling strategies described in Gasmi et al.[8] Namely, we enhance the numerical algorithm to handle non-matching temporal and spatial meshes. 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.[9] for a discussion of the benefits of loose coupling of partitions over tight coupling.

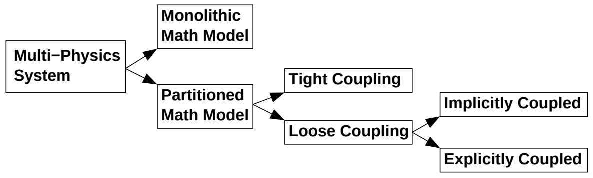


Figure 1. Schematic illustrating our "taxonomy" for models that describe a multi-physics systems.

In this paper we present our approach for loose coupling of modules where spatial and temporal meshes are, in general, non-matching. Numerical experiments with simple mechanical models are used to explore the stability, accuracy, and efficiency of the proposed algorithms. Individual modules are time integrated with standard explicit time integrators.

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.[6]

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[6] and Gasmi et al.[8] 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 $i \in \{1, 2, \ldots, N\}$ 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 $m^{(i)} = \{0, 1, \ldots\}$); 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[6] 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*. Simple examples of systems in this partitioned representation can be found in Gasmi et al.[8]

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

Here, 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 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.

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.

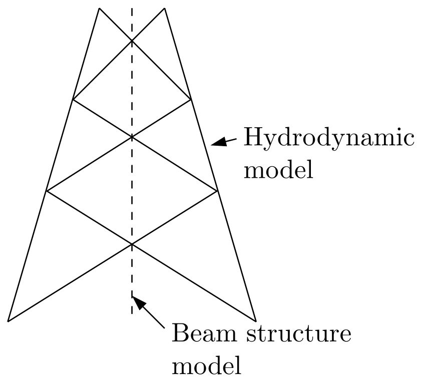


Figure 2. A multi-member hydrodynamics partition overlayed on a single-beam structural partition.

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.[10]

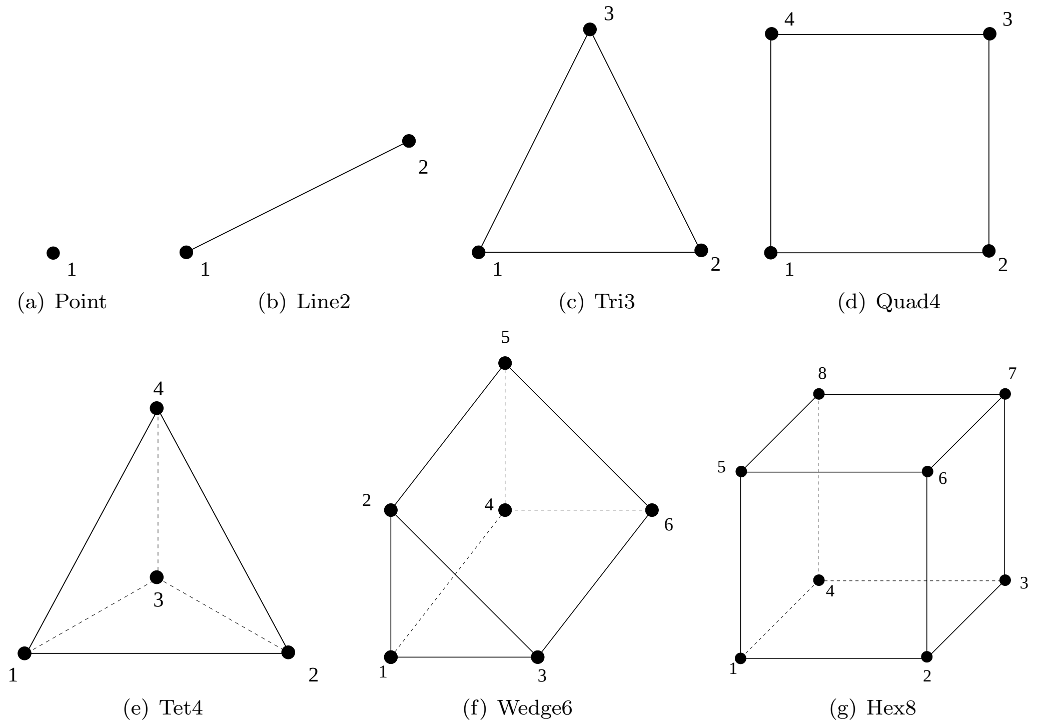


Figure 3. Interface-element types in the modularized FAST framework for coupling partitions.

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

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

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.
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*.[11,12] 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.

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

We are interested in solving initial-value problems associated with the partitioned-system representation given by Eqs. (1)–(5). Time-advancing the partitioned-system solution requires care, as it entails the manipulation and transfer of data between partitions at each time step. In this section we present in detail the coupling strategies employed in our numerical investigation. For clarity, we refer to the mathematical model for a given subsystem as a partition, while the numerical-algorithm implementation for time-advancement of a partition solution is called a *module*.

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; orientations are transferred from the nearest-neighbor node of the 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.

We consider here a multi-physics system that has been modeled as N coupled partitions. To enhance modularity, we assume that data may be communicated between modules only at discrete-time stations at "global" time increments; individual modules may be advanced with shorter increments (i.e., time-step subcycling). Time advancement of modules in our implicit loose coupling is based on a predictor-corrector (PC) approach. The approach described here deals with the communication of data between partitions, and is independent of the underlying ODE or DAE solver employed by each partition. This approach shares similarities with those described elsewhere.[9, 13]

Our algorithm can be summarized as follows. Consider the case where all partition solutions are to be time advanced from t_n to t_{n+1}, where $t_n = n\Delta t^g$, $n = \{0, 1, \ldots, n_{\max}\}$ is the global time-step number, and Δt^g is the constant global time increment. Module inputs (from other modules) are predicted at t_{n+1} based on polynomial extrapolation, and each module solution is advanced (independently) to t_{n+1}. These predicted solutions can then be taken as the accepted result (explicit loose coupling), or the step can be repeated using corrected input values. The approach described here differs from that described in Gasmi et al.[8] in several regards: (*i*) modules are updated in a "Jacobi" sense as they are updated in parallel, whereas in Gasmi et al.[8] modules are updated in a "Gauss-Seidel" sense[14] and each module uses the most up-to-date information available in its time advance, (*ii*) module inputs and outputs can be predicted with polynomial extrapolation of up-to second order rather than only first-order extrapolation, (*iii*) module solutions can be time advanced with different time increments, and (*iv*) modules can interact through non-matching spatial meshes at their interfaces.

The step-by-step procedure for advancing the solution from t_n to t_{n+1} is described below. We denote this algorithm PC($j_{\max}$), where j is the correction counter and $j_{\max} \geq 0$ is the user-defined number of corrector steps to be taken over each time step. A global time increment, Δt^g, is chosen and the individual-module time increments are $\Delta t^{(i)} = \Delta t^g / n^{(i)}$, where $n^{(i)}$ is an integer. This choice of time increment values is designed to accommodate modules with discrete states for which the time increment is fixed.

Known information at start of time step: We assume that the following data are known at time t_n:

$$\mathbf{x}_n^{(i)}, \dot{\mathbf{x}}_n^{(i)}, \dot{\mathbf{x}}_{n-\frac{1}{n^{(i)}}}^{(i)}, \ldots, \dot{\mathbf{x}}_{n-\frac{m}{n^{(i)}}}^{(i)}, \mathbf{x}_n^{d,(i)}, \mathbf{z}_n^{(i)}, \mathbf{y}_n^{(i)}, \mathbf{u}_n^{(i)}, \mathbf{u}_{n-1}^{(i)}, \ldots, \mathbf{u}_{n-p}^{(i)}, \tag{7}$$

where p is the number of previous inputs ($p = 0$, 1, or 2), the state-derivative RHS is calculated as

$$\dot{\mathbf{x}}_n^{(i)} = \mathbf{X}^{(i)}\left(t_n, \mathbf{x}_n^{(i)}, \mathbf{x}_n^{d,(i)}, \mathbf{z}_n^{(i)}, \mathbf{u}_n^{(i)}\right), \tag{8}$$

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.

Step 1 (Predict):
(Step 1.a) Let $j = 0$, where j is the "correction" counter, and predict the input at t_{n+1} for all partitions through extrapolation (over constant Δt^g):

$$\mathbf{u}_{n+1,(j-1)}^{(i)} = \text{polyfit}\left(t_{n+1}, \Delta t^g, \mathbf{u}_n^{(i)}, \mathbf{u}_{n-1}^{(i)}, \ldots, \mathbf{u}_{n-p}^{(i)}\right), \tag{9}$$

for each $i \in \{1, \ldots, N\}$, where a subscript in parentheses indicates the correction iteration. Here, extrapolation is accomplished by the function polyfit, which is evaluated at the time t_{n+1}. The function polyfit is based on a p^{th}-order polynomial fit to the $p+1$ values that are separated in time by Δt^g, and it has an $O\left[(\Delta t^g)^{p+1}\right]$ error.

(Step 1.b) Advance the solution of all partitions to yield predicted state and constraint values, i.e.,

$$\left\{ \begin{array}{l} \mathbf{x}_n^{(i)}, \dot{\mathbf{x}}_n^{(i)}, \dot{\mathbf{x}}_{n-\frac{1}{n^{(i)}}}^{(i)}, \ldots, \dot{\mathbf{x}}_{n-\frac{m}{n^{(i)}}}^{(i)}, \mathbf{x}_n^{d,(i)}, \mathbf{z}_n^{(i)}, \\ \mathbf{u}_{n+1,(j-1)}^{(i)}, \mathbf{u}_n^{(i)}, \ldots, \mathbf{u}_{n-p+1}^{(i)}, \mathbf{X}^{(i)}, \mathbf{X}^{d,(i)}, \mathbf{Z}^{(i)} \end{array} \right\} \xrightarrow{\text{ADV}} \left\{ \begin{array}{l} \mathbf{x}_{n+1,(j)}^{(i)}, \dot{\mathbf{x}}_{n+1,(j)}^{(i)}, \\ \mathbf{x}_{n+1,(j)}^{d,(i)}, \mathbf{z}_{n+1,(j)}^{(i)} \end{array} \right\}, \tag{10}$$

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 global time step; $n^{(i)}$ substeps will be taken by module i. In the new FAST modularization, data structures are equipped to pass up to $p+1 = 1$, 2, or 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., $\mathbf{u}_{n+1,(j)}^{(i)}$ for an implicit integrator; $\mathbf{u}_n^{(i)}$ 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, or when $n^{(i)} > 1$ and a module uses time-step subcycling).

(Step 1.c) Determine the inputs and outputs concurrently for all partitions through a global solve of the input-output equations at t_{n+1}; i.e., solve

$$\begin{aligned}\mathbf{0} = \mathbf{U}^{M,(i)}\Bigg(& t_{n+1}\, , \mathbf{M}_{\mathbf{u}}^{1i}\left(\mathbf{u}_{n+1,(j)}^{(1)}\right), \ldots, \mathbf{u}_{n+1,(j)}^{(i)}, \ldots, \mathbf{M}_{\mathbf{u}}^{Ni}\left(\mathbf{u}_{n+1,(j)}^{(N)}\right), \\ & \mathbf{M}_{\mathbf{y}}^{1i}\left(\mathbf{y}_{n+1,(j)}^{(1)}\right), \ldots, \mathbf{M}_{\mathbf{y}}^{ii}\left(\mathbf{y}_{n+1,(j)}^{(i)}\right), \ldots, \mathbf{M}_{\mathbf{y}}^{Ni}\left(\mathbf{y}_{n+1,(j)}^{(N)}\right)\Bigg), \end{aligned} \tag{11}$$

$$\mathbf{y}_{n+1,(j)}^{(i)} = \mathbf{Y}^{(i)}\left(t_{n+1}\, , \mathbf{x}_{n+1,(j)}^{(i)}\, , \mathbf{x}_{n+1,(j)}^{d,(i)}\, , \mathbf{z}_{n+1,(j)}^{(i)}\, , \mathbf{u}_{n+1,(j)}^{(i)}\right), \tag{12}$$

for $\mathbf{u}_{n+1(j)}^{(i)}$, $\mathbf{y}_{n+1(j)}^{(i)}$ and for all $i \in \{1, \ldots, N\}$. This approach requires, in general, a nonlinear-system solve, which, for a Newton-Raphson solver, requires the Jacobian of the input-output equations and output-equation right sides, i.e.,

$$\frac{\partial \mathbf{U}^{M,(i)}}{\partial \mathbf{u}_{n+1,(j)}^{(k)}}, \quad \frac{\partial \mathbf{U}^{M,(i)}}{\partial \mathbf{y}_{n+1,(j)}^{(k)}}, \quad \frac{\partial \mathbf{Y}^{(i)}}{\partial \mathbf{u}_{n+1,(j)}^{(i)}}, \tag{13}$$

for all $i, k \in \{1, \ldots, N\}$; we note $\frac{\partial \mathbf{Y}^{(i)}}{\partial \mathbf{y}_{n+1,(j)}^{(k)}} = \mathbf{0}$ for all $k \in \{1, \ldots, N\}$ and $\frac{\partial \mathbf{Y}^{(i)}}{\partial \mathbf{u}_{n+1,(j)}^{(k)}} = \mathbf{0}$ for $k \neq i$.

If $j_{\max} = 0$, skip to Step 3.

Step 2 (Correct):
(Step 2.a) Advance the solution to yield corrected states and constraints, i.e.,

$$\left\{\begin{array}{l} \mathbf{x}_n^{(i)}\, , \dot{\mathbf{x}}_n^{(i)}\, , \dot{\mathbf{x}}_{n-\frac{1}{n^{(i)}}}^{(i)}\, , \ldots, \dot{\mathbf{x}}_{n-\frac{m}{n^{(i)}}}^{(i)}\, , \mathbf{x}_n^{d,(i)}\, , \mathbf{z}_n^{(i)}\, , \\ \mathbf{u}_{n+1,(j)}^{(i)}\, , \mathbf{u}_n^{(i)}\, , \ldots, \mathbf{u}_{n-p+1}^{(i)}\, , \mathbf{X}^{(i)}\, , \mathbf{X}^{d,(i)}\, , \mathbf{Z}^{(i)} \end{array}\right\} \xrightarrow{\text{ADV}} \left\{\begin{array}{l} \mathbf{x}_{n+1,(j+1)}^{(i)}\, , \dot{\mathbf{x}}_{n+1,(j+1)}^{(i)}\, , \\ \mathbf{x}_{n+1,(j+1)}^{d,(i)}\, , \mathbf{z}_{n+1,(j+1)}^{(i)} \end{array}\right\}, \tag{14}$$

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

(Step 2.b) Update the outputs and inputs following Step 1.c, yielding $\mathbf{y}_{n+1,(j+1)}^{(i)}$ and $\mathbf{u}_{n+1,(j+1)}^{(i)}$, respectively, for $i \in \{1, \ldots, N\}$.

(Step 2.c) Let $j = j + 1$. If $j < j_{\max}$, repeat Step 2. If $j = j_{\max}$, proceed to Step 3.

Step 3: Save all of the final variables,

$$\begin{array}{lll} \mathbf{x}_{n+1}^{(i)} = \mathbf{x}_{n+1,(j_{\max})}^{(i)}, & \mathbf{x}_{n+1}^{d,(i)} = \mathbf{x}_{n+1,(j_{\max})}^{d,(i)}, & \mathbf{z}_{n+1}^{(i)} = \mathbf{z}_{n+1,(j_{\max})}^{(i)}, \\ \mathbf{y}_{n+1}^{(i)} = \mathbf{y}_{n+1,(j_{\max})}^{(i)}, & \mathbf{u}_{n+1}^{(i)} = \mathbf{u}_{n+1,(j_{\max})}^{(i)}, & \dot{\mathbf{x}}_{n+1}^{(i)} = \dot{\mathbf{x}}_{n+1,(j_{\max})}^{(i)}, \end{array} \tag{15}$$

for each $i \in \{1, \ldots, N\}$, which completes solution advancement to time t_{n+1}.

III. Illustrative Examples and Results

A. Mapping Examples

In this section we show examples that demonstrate Point_to_Point and Line2_to_Line2 mapping between non-matching meshes as described in Tables 1 and 2. Figure 4 shows two coupled Point-element meshes in their reference configurations in three-dimensional (X, Y, Z) space. The mesh in Figure 4(a) is composed of a single Point element, whose outputs are motions and whose inputs are loads. The mesh in Figure 4(b) is composed of five Point elements, whose outputs are loads and whose inputs are motions. Figure 5(a) shows the Point element moving two units in the positive X direction, with velocity of 0.5 units and no translational acceleration, and rotating 20° about the X axis with rotational velocity of 2 units and rotational acceleration of 2 units. Figure 5(b) shows how the motions are mapped to the destination Point-element mesh. We note that the translational velocity is tangential to the line of point elements, and that it linearly increases with distance from the point that is coincident with the source Point element. Also, the translational acceleration includes both centripetal and tangential components. Figure 6(b) shows the mesh with five point forces, and Figure 6(a) shows how those five point forces are mapped to a single point force and a single point moment on the destination mesh; the net load is balanced between the two meshes.

Figure 7 shows an example of the Line2_to_Line2 motion-mapping algorithm described in Table 1. Figure 7(a) shows the two non-matching meshes in their reference configurations. Figure 7(b) shows how each node in the destination Line2 mesh is mapped to the source Line2 mesh after the mapping search; interpolated motion values are taken from those mapped locations. Figure 8 shows an example of the Line2_to_Line2 load-mapping algorithm described in Table 2. Figure 8(a) shows the two non-matching meshes in their reference configurations. Figure 8(b) shows how an augmented source Line2 mesh is created by projecting from each node of the destination Line2 mesh. Note that not all destination-mesh nodes will, or need to, project onto the source Line2 mesh. Figure 8(c) illustrates how each node of the augmented source Line2 mesh transfers its lumped load onto the destination mesh, where they are transformed into distributed loads such that the overall load is balanced.

B. Time-Step Subcycling Examples

1. Uncoupled Models

Figure 9 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.C. 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.

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	1.0	1.0	0.1	5.0	0.01	1.0	2.0

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:

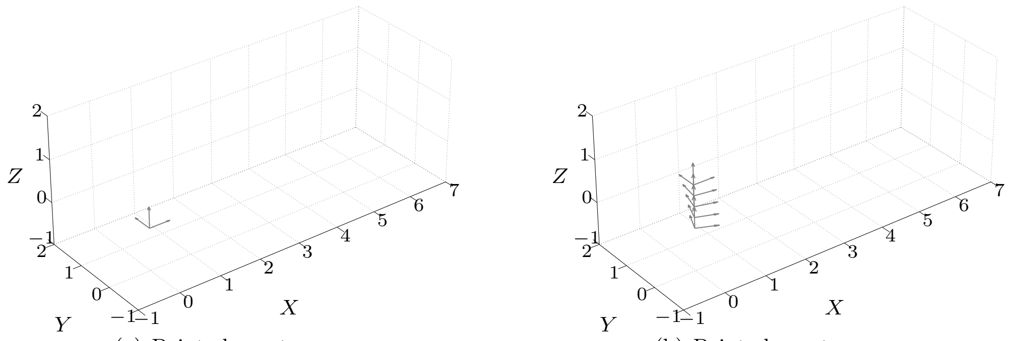
$$\mathbf{u}^{(1)} = \mathbf{y}^{(2)}\,, \tag{16}$$

$$\mathbf{u}^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{y}^{(1)}\,. \tag{17}$$

The analytically determined benchmark solutions $q_b^{(1)}(t)$ and $q_b^{(2)}(t)$ (in a monolithically coupled form) are shown in Figure 10 for $0 \le t \le 30$ and for initial conditions

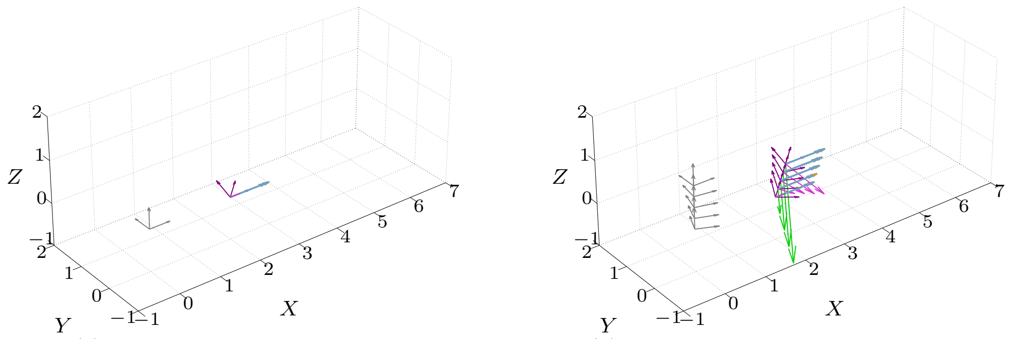
$$q^{(1)}(0) = 1\,, \quad \dot{q}^{(1)}(0) = q^{(2)}(0) = \dot{q}^{(2)}(0) = 0\,. \tag{18}$$

We consider numerical solution of the ODEs where Partition 1 is time integrated with either a fourth-order Adams-Bashforth-Moulton (ABM4) or Runge-Kutta (RK4) time integrator. ABM4 is an explicit multi-step predictor-corrector algorithm that has an implicit dependence on other-system data (through interaction forcing). For our numerical tests, the first three time steps are taken to be the benchmark solution for



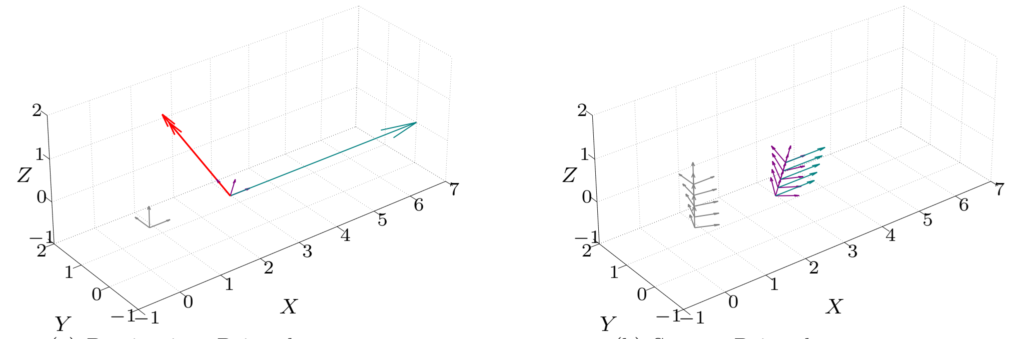
(a) Point element (b) Point elements

Figure 4. Demonstration of the Point_to_Point mapping algorithm: Reference positions and orientations. (a) Reference position and orientation of a Point interface element, which has motion output and point force/moment input. (b) Reference positions and orientations of nodes along an assembly of five Point interface elements, which have motion inputs and point force/moment outputs.



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

Figure 5. Demonstration of the Point_to_Point mapping algorithm: Motion mapping. (a) The Point element moves two units in the positive *X* direction with velocity of 0.5 units and no translational acceleration, and rotates 20° about the *X* axis with rotational velocity of 2 units and rotational acceleration of 2 units. (b) Point element after motions have been mapped from the Point element to the Point elements: translational velocity (light-purple arrows), translational acceleration (green arrows), rotational velocity (blue arrows), rotational acceleration (orange arrows hidden by blue arrows).



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

Figure 6. Demonstration of the Point_to_Point mapping algorithm: Load mapping. (b) Nodes of Point elements show values of forces (turquoise arrows). (a) Destination Point force and moment (red arrow) after mapping of point forces from assembly of source Point elements. The net force and moment in the destination Point element is identical to that of the source Point elements.

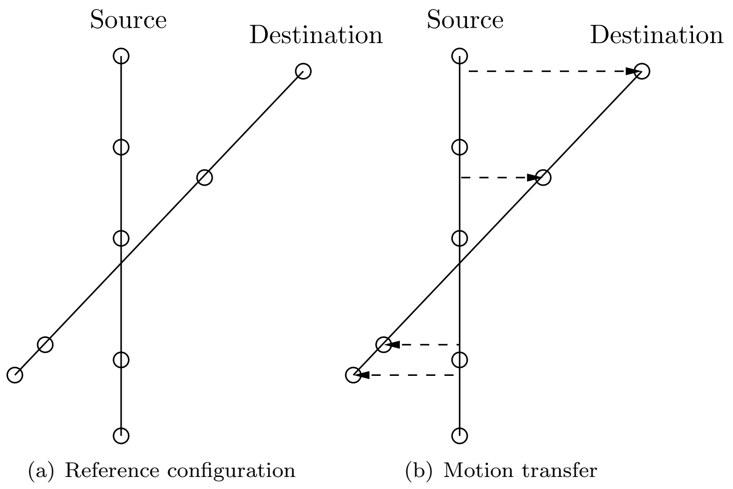


Figure 7. Line2_to_Line2 motion mapping: (a) Two Line2 non-matching meshes in their reference configurations; (b) each destination Line-element node is mapped to a source Line2 element, from which motions are interpolated.

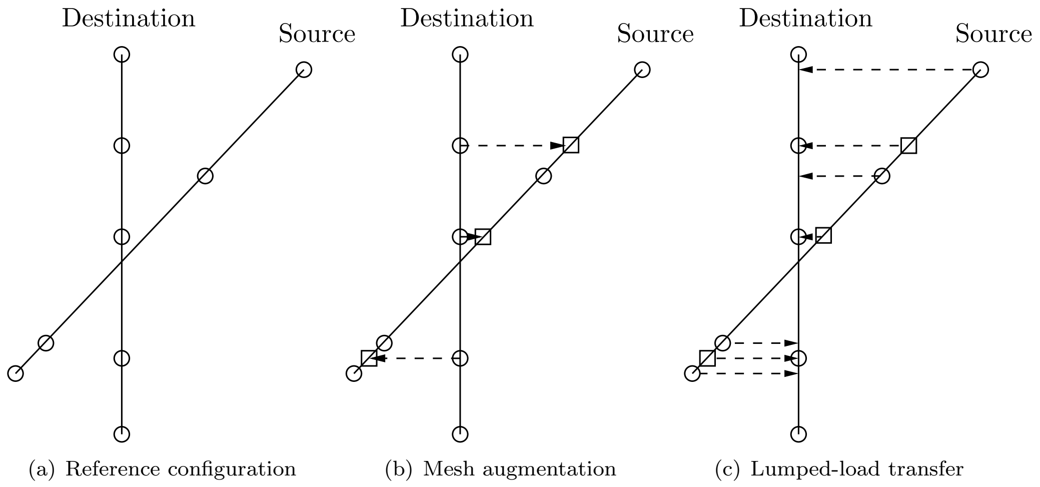
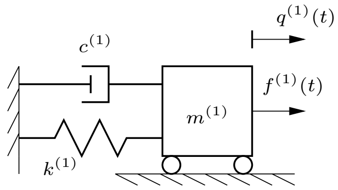
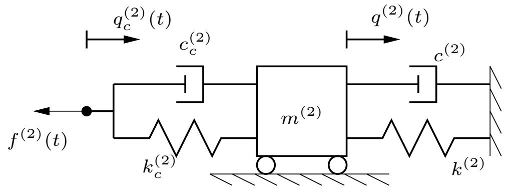


Figure 8. Line2_to_Line2 load mapping: (a) Two Line2 non-matching meshes in their reference configurations; (b) augmentation of the source mesh based on destination-mesh node locations (augmented nodes denoted as squares); (c) transfer of lumped nodal loads from the augmented source mesh to the destination mesh.



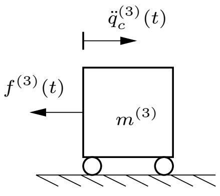
$$
\begin{aligned}
&\text{GE} : m^{(1)}\ddot{q}^{(1)} + c^{(1)}\dot{q}^{(1)} + k^{(1)}q^{(1)} = f^{(1)} \\
&\mathbf{x}^{(1)} = \begin{bmatrix} q^{(1)} & \dot{q}^{(1)} \end{bmatrix}^T , \quad \mathbf{x}^{d,(1)} = \emptyset , \quad \mathbf{z}^{(1)} = \emptyset , \\
&\mathbf{u}^{(1)} = \begin{bmatrix} f^{(1)} \end{bmatrix} , \\
&\mathbf{y}^{(1)} = \begin{bmatrix} q^{(1)} & \dot{q}^{(1)} & \ddot{q}^{(1)} \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}^{(1)} \\ \left(\mathbf{u}^{(1)} - \begin{bmatrix} k^{(1)} & c^{(1)} \end{bmatrix} \mathbf{x}^{(1)}\right) / m^{(1)} \end{bmatrix}
\end{aligned}
$$

(a) Partition 1



$$
\begin{aligned}
&\text{GE} : m^{(2)}\ddot{q}^{(2)} + \left(c^{(2)} + c_c^{(2)}\right)\dot{q}^{(2)} + \left(k^{(2)} + k_c^{(2)}\right) q^{(2)} = \\
&\qquad\qquad c_c^{(2)}\dot{q}_c^{(2)} + k_c^{(2)} q_c^{(2)} \\
&\mathbf{x}^{(2)} = \begin{bmatrix} q^{(2)} & \dot{q}^{(2)} \end{bmatrix}^T , \quad \mathbf{x}^{d,(2)} = \emptyset , \quad \mathbf{z}^{(2)} = \emptyset , \\
&\mathbf{u}^{(2)} = \begin{bmatrix} q_c^{(2)} & \dot{q}_c^{(2)} \end{bmatrix}^T , \\
&\mathbf{y}^{(2)} = \begin{bmatrix} f^{(2)} \end{bmatrix} = \begin{bmatrix} k_c^{(2)} & c_c^{(2)} \end{bmatrix} \left(\mathbf{x}^{(2)} - \mathbf{u}^{(2)}\right)
\end{aligned}
$$

(b) Partition 2



$$
\begin{aligned}
&\text{GE} : m^{(3)}\ddot{q}_c^{(3)} = -f^{(3)} \\
&\mathbf{x}^{(3)} = \emptyset , \quad \mathbf{x}^{d,(3)} = \emptyset , \quad \mathbf{z}^{(3)} = \emptyset , \\
&\mathbf{u}^{(3)} = \begin{bmatrix} \ddot{q}_c^{(3)} \end{bmatrix} , \quad \mathbf{y}^{(3)} = \begin{bmatrix} f^{(3)} \end{bmatrix} = -\mathbf{u}^{(3)}/m^{(3)}
\end{aligned}
$$

(c) Partition 3

Figure 9. Schematics of three uncoupled model partitions. Governing equations (GEs), states, constraints, inputs, and outputs are shown.

error-free initialization of ABM4. In productions runs, a single-step integrator (e.g., Runge-Kutta) would be used to initialize the multi-step integrator. Extrapolation and interpolation via the function polyfit is accomplished with $p = 2$. Partition 2 is time integrated with ABM4.

Numerical simulations with the *uncoupled* Partition 1 and Partition 2 and ABM4 integration showed that the critical time increments for numerically stable solutions were $\Delta t_c^{(1)} \approx 0.92$ and $\Delta t_c^{(2)} \approx 0.37$, respectively; Partition 2 is significantly more stiff than Partition 1 ($k^{(2)}/k^{(1)} = 5$), and thus has a more restrictive critical time increment. For lock-step coupled simulations ($\Delta t^{(1)} = \Delta t^{(2)}$; $n^{(1)} = n^{(2)} = 1$, PC(1)), the critical time increment was the same as $\Delta t_c^{(2)}$. However, for simulations with time-step subcycling with $n^{(1)} = 1$ and $n^{(2)} = 3$, the global critical time increment was about 0.77. For these subcycling ratios, $\Delta t^{(1)} = \Delta t^g = 0.77$, $\Delta t^{(2)} = \Delta t^{(1)}/3 = 0.26$. Thus, with time-step subcycling, stable solutions could be achieved where each model was time advanced with increments close to their uncoupled limits.

Figure 11 shows normalized root-mean-square (RMS) error of the numerical solutions as a function of global time increment for the displacement of Partition 1 and Partition 2 over the time interval $0 \le t \le 30$. Normalized RMS error for $n_{\max}$ numerical response values q_n were calculated as

$$
\epsilon\left(q\right) = \sqrt{\frac{\sum_{k=0}^{n_{\max}} \left[q_k - q_b(t_k)\right]^2}{\sum_{k=0}^{n_{\max}} \left[q_b(t_k)\right]^2}} . \tag{19}
$$

Also shown in Figure 11 are dotted lines indicating third-order $O\left(\Delta t^3\right)$ and fourth-order $O\left(\Delta t^4\right)$ convergence rates. Figure 11(a) shows error for lock-step time integration. For explicit coupling, PC(0), only third-order accuracy is achieved. This is because the interaction terms were extrapolated with third-order accuracy.

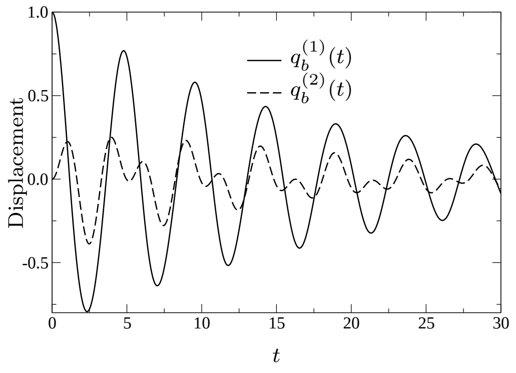


Figure 10. Benchmark solutions for Partition 1 and Partition 2 displacements when coupled.

However, with one correction per time step, PC(1), the fourth-order accuracy of the underlying algorithms is restored. Figure 11(b) shows error with time-step subcycling. Here, for both PC(0) and PC(1), only third-order convergence is obtained, though PC(1) provides significantly more accuracy. These data are limited to third-order convergence because, while Partition 2 is being updated at subintervals, it is relying on interpolated or extrapolated data from Partition 1. While time-step subcycling allowed for individual models to run with larger time increments, overall accuracy was roughly equivalent to that for PC(0) with lock-step time integration, while time-step subcycling was less accurate than lock-step integration with PC(1). Further, for a given Δt^g, time-step subcycling requires 3 times more ABM4 time-step updates than in lock-step integration.

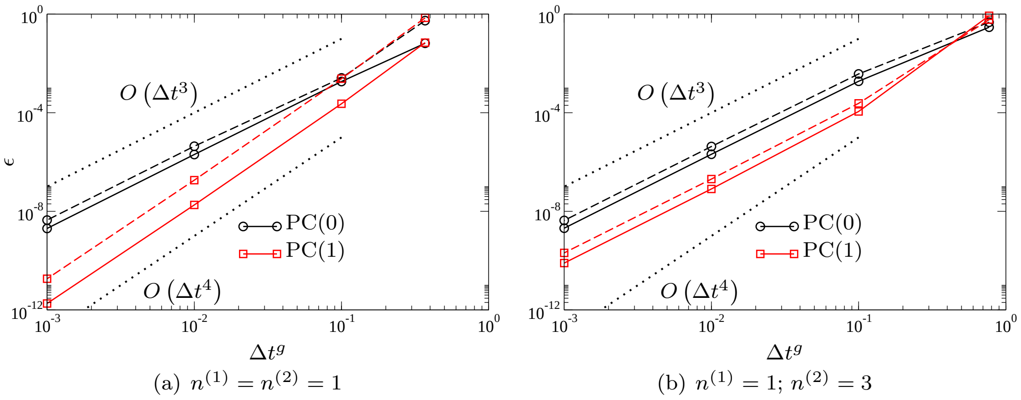


Figure 11. Normalized RMS error of $q^{(1)}(t)$ (solid lines) and $q^{(2)}(t)$ (dashed lines) displacement histories for Partition 1 coupled to Partition 2 for $0 \leq t \leq 30$. Partitions 1 and 2 were both time integrated with ABM4. Dotted lines show ideal third-order and fourth-order convergence rates.

Figure 12 shows normalized RMS for RK4 integration of Partition 1 and ABM4 integration of Partition 2. Here, the critical time increment for stable integration of the uncoupled Partition 1 was $\Delta t_c^{(1)} \approx 2.9$. Again, lock-step time integration of the coupled system was limited to $\Delta t_c^{(2)} \approx 0.37$, while time integration with time-step subcycling with $n^{(1)} = 1$ and $n^{(2)} = 8$ allowed stable time integration with $\Delta t^g = 1.9$. Figure 12 shows that for both lock-step and time-step-subcycling integration, numerical solutions are limited to third-order accuracy. This is because the RK4 integration is limited by the accuracy of extrapolation or interpolation of interaction terms. Going from PC(0) to PC(1) decreases the error, but does not affect the convergence rate. Here, while time-step subcycling allowed for individual models to run with larger time increments, overall accuracy was largely unchanged, while, for a given Δt^g, time-step subcycling requires 8 times more ABM4 time-step updates than in lock-step integration.

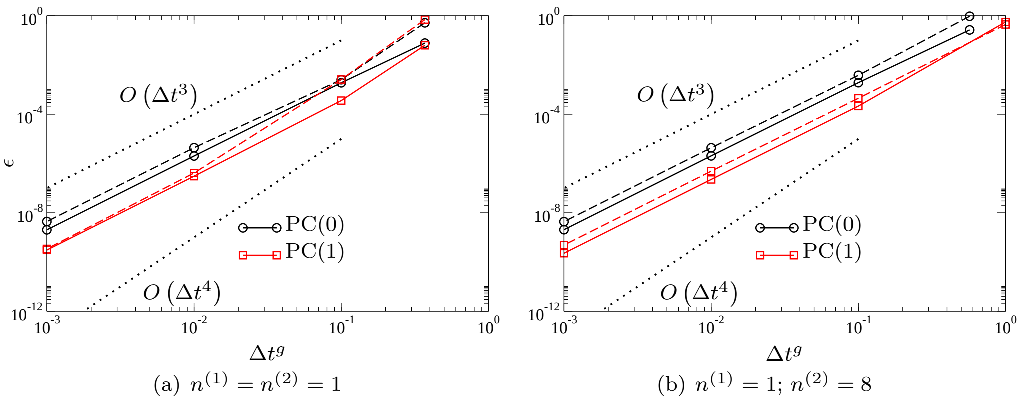


Figure 12. Normalized RMS error of $q^{(1)}(t)$ (solid lines) and $q^{(2)}(t)$ (dashed lines) displacement histories for Partition 1 coupled to Partition 2 for $0 \leq t \leq 30$. Partitions 1 and 2 were time integrated with RK4 and ABM4, respectively. Dotted lines show ideal third-order and fourth-order 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:

$$\mathbf{u}^{(1)} = \mathbf{y}^{(3)}\,, \tag{20}$$

$$\mathbf{u}^{(3)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{y}^{(1)}\,. \tag{21}$$

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 $q_b^{(1)}(t)$ (in a monolithically coupled form) is shown in Figure 13 for $0 \leq t \leq 30$ and for initial conditions

$$q^{(1)}(0) = 1\,, \quad \dot{q}^{(1)}(0) = 0\,. \tag{22}$$

Because Partition 3 is a pure inertial element, there is no associated stiffness or critical time increment. Numerical simulations were performed with ABM4 integration of Partition 1 and Partition 3 was calculated in lock step. Figure 14 shows normalized RMS error of $q^{(1)}(t)$ with PC(0) and PC(1). As expected, only third-order accuracy is achieved with PC(0) due to the accuracy limitation of our extrapolation. However, PC(1) achieves desired fourth-order accuracy, albeit PC(1) requires twice as much computational effort compared to PC(0) for a given time increment.

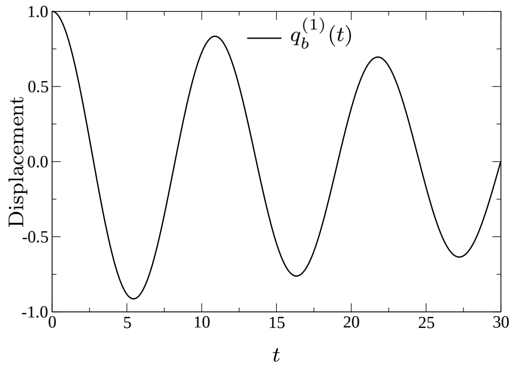


Figure 13. Benchmark solution for Partition 1 when coupled to Partition 3.

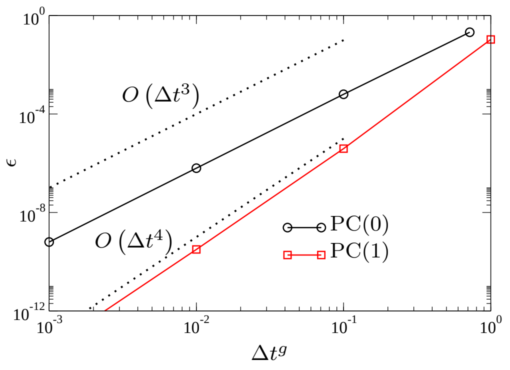


Figure 14. Normalized RMS error of $q^{(1)}(t)$ displacement histories for Partition 1 coupled to Partition 3 calculated with ABM4 (Partition 1) for $0 \leq t \leq 30$. Dotted lines show ideal third-order and fourth-order convergence rates.

IV. Conclusion and Future Work

In this paper we described methods for numerical time integration of mechanical 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 introduced a time-step subcycling algorithm with a predictor-corrector option, which allows each module to be time advanced with a time increment close to is uncoupled "preferred" increment. 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. 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. Future work will also study the numerical accuracy, stability, and efficiency of other coupling examples.

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. (25), (26), and (28) 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
θ^D and θ^S:	Displaced rotation (absolute orientation; direction cosine matrix) of a node of the destination and source meshes, respectively
θ^{DR} and θ^{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

$$d = \left\| \vec{p}^{SR} - \vec{p}^{DR} \right\|_2 \,, \tag{23}$$

$\| \cdot \|_2$ 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. (23) 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

$$d = \left\| \vec{p}^{DR} - \vec{p}_1^{SR} \left(1 - \overline{\ell}^S \right) - \vec{p}_2^{SR} \overline{\ell}^S \right\|_2 \,. \tag{24}$$

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

$$\overline{\ell}^S = \frac{\left(\vec{p}_2^{SR} - \vec{p}_1^{SR} \right) \cdot \left(\vec{p}^{DR} - \vec{p}_1^{SR} \right)}{\left(\vec{p}_2^{SR} - \vec{p}_1^{SR} \right) \cdot \left(\vec{p}_2^{SR} - \vec{p}_1^{SR} \right)} \,, \tag{25}$$

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. (24) 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. (25), $\cdot$ 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

$$\overline{\ell}^S = \frac{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}^{DR} - \vec{p}_1^{SR}\right)}{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}_2^{SR} - \vec{p}_1^{SR}\right)} . \tag{26}$$

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

$$d = \left\| \vec{p}^{SR} - \vec{p}_1^{DR}\left(1 - \overline{\ell}^D\right) - \vec{p}_2^{DR}\overline{\ell}^D \right\|_2 , \tag{27}$$

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

$$\overline{\ell}^D = \frac{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}^{SR} - \vec{p}_1^{DR}\right)}{\left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right) \cdot \left(\vec{p}_2^{DR} - \vec{p}_1^{DR}\right)} . \tag{28}$$

(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. (25). 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

$$\vec{u}^D=\vec{u}^S+\left[I-\left(\theta^S\right)^T\theta^{SR}\right]\left(\vec{p}^{SR}-\vec{p}^{DR}\right)\,, \tag{29}$$

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, θ^{SR} is the reference rotation (direction cosine matrix) of the mapped node of the source mesh, θ^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. (29) 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 $\left(\theta^S\right)^T\theta^{SR}=I$, 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

$$\theta^D=\theta^{DR}\left(\theta^{SR}\right)^T\theta^S\,, \tag{30}$$

where θ^{DR} is the reference rotation (direction cosine matrix) of a node of the destination mesh and θ^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 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

$$\vec{v}^D=\vec{v}^S+\left[\left(\vec{p}^{SR}+\vec{u}^S\right)-\left(\vec{p}^{DR}+\vec{u}^D\right)\right]\times\vec{\omega}^S\,, \tag{31}$$

$$\vec{\omega}^D=\vec{\omega}^S\,, \tag{32}$$

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. (31) 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).

[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.

[b]Rearrangement of Eq. (29) reveals that $\left(\vec{p}^{SR}+\vec{u}^S\right)-\left(\vec{p}^{DR}+\vec{u}^D\right)=\left(\theta^S\right)^T\theta^{SR}\left(\vec{p}^{SR}-\vec{p}^{DR}\right)$, so, the term involving translational displacements of the nodes of the destination and source meshes in Eq. (31) and other mapping-transfer equations could alternatively be written in terms of rotations of the source mesh.

The mapping transfer of translational and rotation accelerations is given by

$$\vec{a}^D = \vec{a}^S + \left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{\alpha}^S + \vec{\omega}^S \times \left\{\left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{\omega}^S\right\} , \tag{33}$$

$$\vec{\alpha}^D = \vec{\alpha}^S , \tag{34}$$

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. (33) 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. (33) 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

$$S^D = S^S , \tag{35}$$

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,

$$\vec{F}^D = \sum \vec{F}^S , \tag{36}$$

$$\vec{M}^D = \sum \left\{\vec{M}^S + \left[\left(\vec{p}^{SR} + \vec{u}^S\right) - \left(\vec{p}^{DR} + \vec{u}^D\right)\right] \times \vec{F}^S\right\} , \tag{37}$$

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. (37) represents the additional moment of the mapped node of the destination mesh due to the displaced offset (moment 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. (36) and (37) 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; orientations are transferred from the nearest-neighbor node of the 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, except displaced rotation, via interpolation based on the projected location in the source Line2 element, is given by

$$(\,\cdot\,) = (\,\cdot\,)_1 \left(1 - \overline{\ell}^S\right) + (\,\cdot\,)_2 \overline{\ell}^S , \tag{38}$$

[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.

where the $\left(\,\cdot\,\right)$ on the left-hand side (LHS) of Eq. (38) is a placeholder for $\vec{u}^D$, $\vec{v}^D$, $\vec{\omega}^D$, $\vec{a}^D$, $\vec{\alpha}^D$, or S^D from the LHS of Eqs. (29) and (31) through (35), respectively, and $\left(\,\cdot\,\right)$ on the RHS of Eq. (38) is a placeholder for the corresponding RHS of Eqs. (29) and (31) through (35), 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. (38) was solved via Eq. (25) 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 $\overline{\ell}^S = 1$.

A direction cosine matrix cannot be interpolated (because the resulting matrix would no longer be orthonormal), so instead, the displaced rotations are transferred from the nearest-neighbor node of the mapped source Line2 element per Eq. (39), which is based on Eq. (30), with NINT$\left(\,\cdot\,\right)$ denoting the nearest-integer function.[d]

$$\theta^D = \begin{cases} \theta^{DR}\left(\theta_1^{SR}\right)^T \theta_1^S\,, & \text{for NINT}\left(\overline{\ell}^S\right) = 0\,, \\ \theta^{DR}\left(\theta_2^{SR}\right)^T \theta_2^S\,, & \text{for NINT}\left(\overline{\ell}^S\right) = 1\,. \end{cases} \tag{39}$$

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. (38) (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. (26) 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

$$\vec{F}_1^S = \sum \frac{\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\|_2}{6}\left(2\vec{f}_1^S + \vec{f}_2^S\right)\,, \tag{40}$$

$$\vec{F}_2^S = \sum \frac{\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\|_2}{6}\left(\vec{f}_1^S + 2\vec{f}_2^S\right)\,, \tag{41}$$

$$\vec{M}_1^S = \sum \frac{\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\|_2}{6}\left\{2\vec{m}_1^S + \vec{m}_2^S + \left[\left(\vec{p}_2^{SR} + \vec{u}_2^S\right) - \left(\vec{p}_1^{SR} + \vec{u}_1^S\right)\right] \times \left(\frac{\vec{f}_1^S + \vec{f}_2^S}{2}\right)\right\}\,, \tag{42}$$

$$\vec{M}_2^S = \sum \frac{\|\vec{p}_2^{SR} - \vec{p}_1^{SR}\|_2}{6}\left\{\vec{m}_1^S + 2\vec{m}_2^S - \left[\left(\vec{p}_2^{SR} + \vec{u}_2^S\right) - \left(\vec{p}_1^{SR} + \vec{u}_1^S\right)\right] \times \left(\frac{\vec{f}_1^S + \vec{f}_2^S}{2}\right)\right\}\,, \tag{43}$$

where the second term on the RHS of Eqs. (42) and (43) represents the additional lumped moment of a node of the source mesh due to the distributed force. As shown by Eqs. (40)–(43), both nodal distributed loads of a given Line2 element contribute to the lumped load of each node of that element. The summations in Eqs. (40)–(43) denote the superposition of loads when a given node of the source mesh is connected to multiple elements. Equations (40)–(43) were derived by integrating the nodal linear shape functions multiplied by the distributed loads (including moment arms for moments) along each source element.

[d]Future work may consider deriving three independent rotational parameters from the direction cosine matrix of each node of the mapped Line2 element of the source mesh, interpolating the rotational parameters, forming a direction cosine matrix from the interpolated rotational parameters, and transferring this direction cosine matrix to the destination node.

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

$$\left(\,\cdot\,\right)_1 = \sum \left(\,\cdot\,\right) \left(1 - \bar{\ell}^D\right) \,, \tag{44}$$

$$\left(\,\cdot\,\right)_2 = \sum \left(\,\cdot\,\right) \bar{\ell}^D \,, \tag{45}$$

where $\left(\,\cdot\,\right)$ on the LHS of Eqs. (44) and (45) is a placeholder for $\vec{F}^D$ and $\vec{M}^D$ from the LHS of Eqs. (36) and (37), respectively, and $\left(\,\cdot\,\right)$ on the RHS of Eqs. (44) and (45) is a placeholder for the corresponding RHS of Eqs. (36) and (37) (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. $\bar{\ell}^D$ used in Eqs. (44) and (45) was solved via Eq. (28) 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 $\bar{\ell}^D = 0$ or $\bar{\ell}^D = 1$. The summations in Eqs. (44) and (45) 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,

$$\vec{F}_1^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left(2\vec{f}_1^D + \vec{f}_2^D\right) \,, \tag{46}$$

$$\vec{F}_2^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left(\vec{f}_1^D + 2\vec{f}_2^D\right) \,, \tag{47}$$

$$\vec{M}_1^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left\{2\vec{m}_1^D + \vec{m}_2^D + \left[\left(\vec{p}_2^{DR} + \vec{u}_2^D\right) - \left(\vec{p}_1^{DR} + \vec{u}_1^D\right)\right] \times \left(\frac{\vec{f}_1^D + \vec{f}_2^D}{2}\right)\right\} \,, \tag{48}$$

$$\vec{M}_2^D = \sum \frac{\left\|\vec{p}_2^{DR} - \vec{p}_1^{DR}\right\|_2}{6} \left\{\vec{m}_1^D + 2\vec{m}_2^D - \left[\left(\vec{p}_2^{DR} + \vec{u}_2^D\right) - \left(\vec{p}_1^{DR} + \vec{u}_1^D\right)\right] \times \left(\frac{\vec{f}_1^D + \vec{f}_2^D}{2}\right)\right\} \,, \tag{49}$$

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. (40)–(43) for the source mesh, Eqs. (46)–(49) 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. (46)–(49), which denote the superposition of loads when a given node of the destination mesh is connected to multiple elements, the inverse of Eqs. (46)–(49) depends on the element connectivity; the inverse of Eqs. (46)–(49) 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. (38) (where $(\,\cdot\,)$ 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 $\bar{\ell}^S$ was solved via Eq. (25) 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

The authors acknowledge useful discussions with John Michalakes. 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 published in tech. report 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 published in tech. report NREL/JA-500-45311 National Renewable Energy Laboratory, Golden, CO.

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

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

[8]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.

[9]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.

[10]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.

[11]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.

[12]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.

[13]Belytschko, T., Yen, H. J., and Mullen, R., "Mixed Methods for Time Integration," *Computer Methods in Applied Mechanics and Engineering*, Vol. 17, 1979, pp. 259–275.

[14]Matthies, H. G., Nekamp, R., and Steindorf, J., "Algorithms for Strong Coupling Procedures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, 2006, pp. 2028–2049.