NREL/CP-2C00-57298. Posted with permission. Presented at the 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition.

Numerical Stability and Accuracy of Temporally Coupled Multi-Physics Modules in Wind-Turbine CAE Tools*

Amir Gasmi†, Michael A. Sprague‡,
Jason M. Jonkman§, and Wesley B. Jones¶
National Renewable Energy Laboratory, Golden, Colorado, 80401, USA

In this paper we examine the stability and accuracy of numerical algorithms for coupling time-dependent multi-physics modules relevant to computer-aided engineering (CAE) of wind turbines. 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. We employ two simple examples as test systems, while algorithm descriptions are kept general. Coupled-system governing equations are framed in monolithic and partitioned representations as differential-algebraic equations. Explicit and implicit loose partition coupling is examined. In explicit coupling, partitions are advanced in time from known information. In implicit coupling, there is dependence on other-partition data at the next time step; coupling is accomplished through a predictor-corrector (PC) approach. Numerical time integration of coupled ordinary-differential equations (ODEs) is accomplished with one of three, fourth-order fixed-time-increment methods: Runge-Kutta (RK), Adams-Bashforth (AB), and Adams-Bashforth-Moulton (ABM). Through numerical experiments it is shown that explicit coupling can be dramatically less stable and less accurate than simulations performed with the monolithic system. However, PC implicit coupling restored stability and fourth-order accuracy for ABM; only second-order accuracy was achieved with RK integration. For systems without constraints, explicit time integration with AB and explicit loose coupling exhibited desired accuracy and stability.

I. Introduction

In this paper we examine the numerical stability and accuracy of several temporal coupling methods for multi-physics modules relevant to computer-aided engineering (CAE) of wind turbines. The wind turbine system is composed of many physics that have significantly different characteristic time and length scales. Figure 1 shows an illustration of a floating wind turbine with some possible physical interactions. 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 suite of tools is FAST,[1] which is a modular assembly of advanced CAE tools. FAST is an open-source, professional-grade software package. As shown in Figure 2, FAST encompasses modules for aerodynamics (AeroDyn[2,3]), platform hydrodynamics (HydroDyn[4,5]) for offshore systems, control and electrical systems (servo), and structural dynamics. The modules are coupled to allow for nonlinear analysis of aero-hydro-servo-elastic

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

‡Senior Scientist, Computational Science Center, 15013 Denver West Parkway, Golden, CO 80401.

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

¶Group Manager and Senior Scientist, Computational Science Center, 15013 Denver West Parkway, Golden, CO 80401.

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.

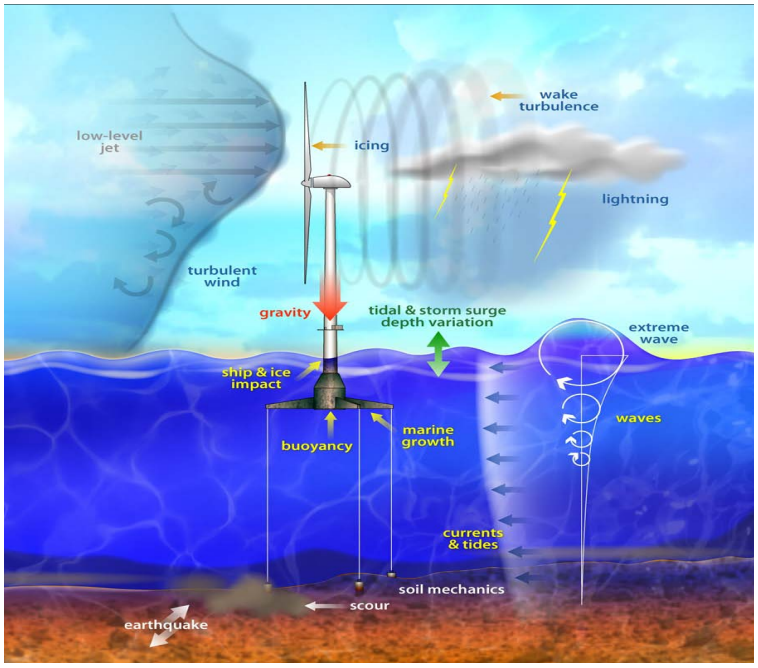


Figure 1. Illustration showing possible physical interactions experienced by an offshore floating wind turbine.

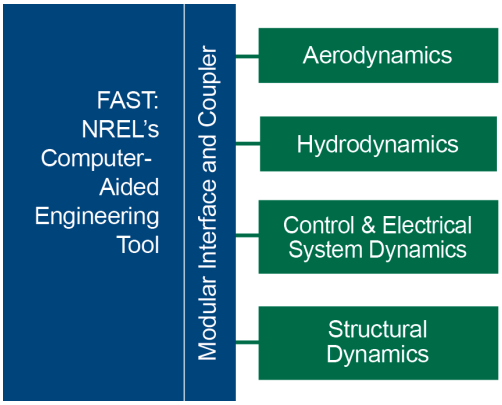


Figure 2. Schematic illustrating primary simulation modules of NREL's wind turbine CAE tool FAST.

This work is motivated by a major restructuring of the FAST tool suite, which is described in a companion paper.[6] 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. We note that the proposed modularization framework has some features similar to those of other model-coupling frameworks (See, e.g., Refs.[7–10]).

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-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 the method-of-lines approach in PDE analysis.

In the established version of FAST, the wind turbine system is *partitioned*[11] into its natural domains, and each partition is treated with a particular software module, e.g., aerodynamics are modeled with the package AeroDyn. The time-dependent ODEs or DAEs associated with each module are time integrated with various integrators, e.g., AeroDyn is integrated with a multi-step predictor-corrector algorithm (fourth-order Adams-Bashforth-Moulton). The modules interact in FAST in an explicit loosely coupled configuration, where partition solutions are updated from time t to time $t + \Delta t$ using coupling information known at time t, where Δt is the time increment of the numerical integrator.

Our taxonomy of models for a time-dependent multi-physics system is summarized schematically in Figure 3. On the left is a multi-physics system that can be modeled in either a monolithic or a partitioned approach. In a monolithic approach, where physics are naturally and inherently coupled, the system is represented by a single system of differential or differential-algebraic equations, and time integration is accomplished with a single solver; all quantities are advanced simultaneously. In a partitioned approach, the system is decomposed into an assembly of subsystems, each being represented by a partition that is subsequently coupled to other subsystem partitions through partition input-output relationships. We refer readers to Felippa et al.[11] for a comparison of monolithic and partitioned approaches. A partitioned system can be assembled into a tightly coupled system that is appropriate for time integration by a single integrator (likely as a system of DAEs). Alternatively, partitions can be loosely coupled, where they are time integrated in a *conventional serial staggered* procedure[12] (i.e., sequential integration). Loose coupling is appealing because it allows for flexibility in modeling each subsystem. This flexibility may allow for different time integrators, different time increments, and different spatial discretization. Further, it allows for the use of existing software modules such as dedicated fluid-dynamics and structural-dynamics software. However, linearization of loosely coupled partitioned systems can be problematic.[6] Finally, loosely coupled partitions can be coupled either explicitly or implicitly (right side of Figure 3). Under explicit loose coupling, as described above, advancement of all partition solutions from time t to $t + \Delta t$ is accomplished based on shared information at t. While appealing in its simplicity, loose explicit coupling may suffer numerical instabilities. Further, explicit coupling presents an inherent challenge for modules that employ a DAE or ODE solver based on implicit numerical time integration, or where a module employs a high-order single step method (like Runge-Kutta), which performs right-hand-side (RHS) evaluations in the semi-closed interval $(t, t + \Delta t]$. Alternatively, implicit partition coupling is formulated such that one or more partitions require other-module data in $(t, t+\Delta t]$ before the solution can be advanced; the partitions are thus tied together in a linear or nonlinear system that must be solved for time advancement. However, approximate solution to the implicitly coupled systems can be pursued through a predictor-corrector (PC) approach, where each partition is advanced from t to $t + \Delta t$ one or more times using predicted/corrected values from other partitions. A PC approach can maintain the modularity and flexibility of loose coupling.

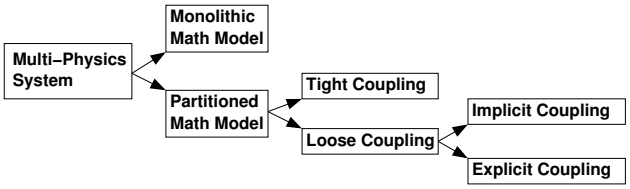


Figure 3. Schematic illustrating our "taxonomy" for models that describe a multi-physics systems.

In this paper, we examine time integration of monolithic systems and equivalent partitioned systems with explicit and implicit loose coupling. Our work is guided by a motivating question: How can we couple multi-physics modules of a partitioned system, such as those in FAST, in a manner that is accurate and numerically stable? Our application examples include a simple two-mass damped oscillator, and a damped oscillator coupled to a quasi-static nonlinear cable. We examine various partition-coupling strategies where the underlying fourth-order fixed-step time integrator is either Runge-Kutta, Adams-Bashforth, or Adams-Bashforth-Moulton.

II. Model Formulation

In this section we present a general framework for describing time-dependent monolithic and partitioned systems that is adopted by the new modular framework in FAST,[6] and we provide specifics of two simple example applications that will be used as test cases to explore the stability and accuracy of the proposed coupling methods described in Section III.

II.A. General Formulation

II.A.1. Monolithic System Representation

In a monolithic representation of a continuous-time-dependent physical system, there is a single system of inherently tightly coupled DAEs. Using a generic state-space representation, we can write the monolithic-system DAEs as follows:

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

$$\mathbf{0} = \mathbf{Z}\left(t, \mathbf{x}(t), \mathbf{z}(t)\right), \tag{2}$$

$$\mathbf{y} = \mathbf{Y}\left(t, \mathbf{x}(t), \mathbf{z}(t)\right), \tag{3}$$

where t is time, an overdot denotes time differentiation, $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{Y}$ are multi-variable vector functions, corresponding to the state, constraint, and output equations, respectively. As with the formulation described in the companion paper (which describes the new modular framework for FAST[6]), we restrict our formulation to semi-explicit DAEs of index 1. While not included here, the companion paper[6] includes discrete-time states. The associated vector dependent variables, $\mathbf{x}$, and $\mathbf{z}$, are the state and constraint, respectively; $\mathbf{y}$ is the output-vector variable. While in some systems it is natural to setup the monolithic equations in the form represented by Eqs. (1)–(3), often the complete multi-physics system involves independent spatial domains and the interaction is achieved through shared-boundary interfaces, which makes it well suited for partitioning. In the next subsection, we present the generalized formalism for such multi-physics-system representation.

II.A.2. Partitioned System Representation

Let us consider that the entire multi-physics system is subdivided into N subsystems; each of these subsystems can be represented in a general DAE form:

$$\dot{\mathbf{x}}_i = \mathbf{X}_i\left(t, \mathbf{x}_i(t), \mathbf{z}_i(t), \mathbf{u}_i(t)\right), \tag{4}$$

$$\mathbf{0} = \mathbf{Z}_i\left(t, \mathbf{x}_i(t), \mathbf{z}_i(t), \mathbf{u}_i(t)\right), \tag{5}$$

$$\mathbf{y}_i(t) = \mathbf{Y}_i\left(t, \mathbf{x}_i(t), \mathbf{z}_i(t), \mathbf{u}_i(t)\right), \tag{6}$$

where the subscript i corresponds to the i^{th} subsystem, and $\mathbf{u}_i$ is a vector of inputs derived from outputs (and, in general, inputs) of coupled partitions. Here, the input to partition i is determined through the additional implicit input-output relationship

$$\mathbf{0} = \mathbf{U}_i\left(\mathbf{u}_1(t), \ldots, \mathbf{u}_N(t), \mathbf{y}_1(t), \ldots, \mathbf{y}_N(t)\right). \tag{7}$$

Equations (4)–(7) for $i \in \{1, \ldots, N\}$ constitute what we consider a *partitioned-system represention.*

II.B. Application Examples

In this subsection we demonstrate how the above representations can be applied to two application examples. These examples will serve as our test systems for illustrating the essence of various coupling schemes.

II.B.1. Linear 2-DOF Damped Oscillator System

We consider first a linear two degrees-of-freedom (DOF) damped oscillator shown by Figure 4. This is a common system for examining numerical coupling algorithms (see, e.g., Ref.[13]). A monolithic-system representation is shown in Figure 4(a), and our partitioned-system representation is shown in Figure 4(b). The partitioning was accomplished by introducing a virtual interaction node between the System 1 mass, m_1, and spring-damper coupling, whose displacement and velocity are denoted $q_I(t)$ and $\dot{q}_I(t)$, respectively. System 1 output (related to System 2 input) is displacement $q_1(t)$ and velocity $\dot{q}_1(t)$ of mass 1, while System 1 input (System 2 output) is the coupling force f_c.

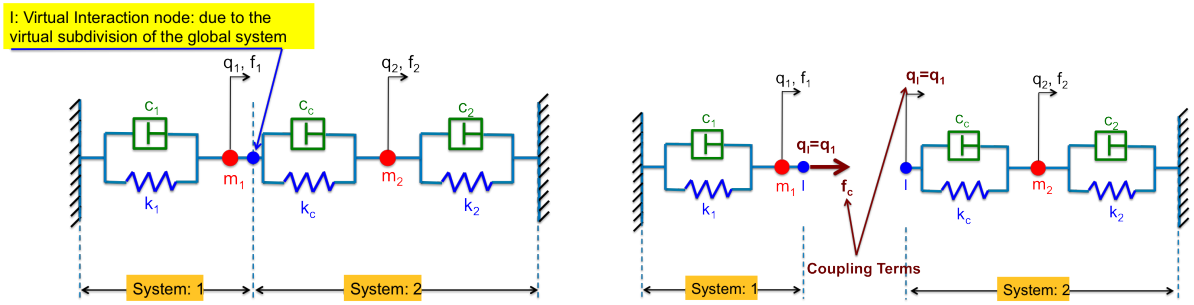


Figure 4. Schematic of the two-mass damped oscillator: (a) monolithic system, and (b) partitioned system.

The governing equation for displacement $q_1(t)$ of the point mass in System 1 is given by

$$m_1\ddot{q}_1 + c_1\dot{q}_1 + k_1 q_1 = f_c + f_1, \tag{8}$$

where f_c is the coupling force due to the relative motion between Systems 1 and 2; c_1 and k_1 are the damping and stiffness coefficients, respectively, and f_1 is a prescribed external force. Following the general formulation of the partitioned-system representation, we recast the System 1 governing equations as follows,

$$\dot{\mathbf{x}}_1(t) = \mathbf{X}_1\left(t, \mathbf{x}_1(t), \mathbf{u}_1(t)\right), \tag{9}$$
$$\mathbf{y}_1(t) = \mathbf{Y}_1\left(t, \mathbf{x}_1(t), \mathbf{u}_1(t)\right), \tag{10}$$

where $\mathbf{x}_1 = \begin{bmatrix} q_1 & \dot{q}_1 \end{bmatrix}^T$ is the state vector (a T-superscript denotes a transpose), $\mathbf{u}_1 = \begin{bmatrix} f_c \end{bmatrix}$ is the input vector, and $\mathbf{y}_1 = \begin{bmatrix} q_1 & \dot{q}_1 \end{bmatrix}^T$ is the output vector. We note that in the above formulation, we have neither a constraint equation nor an associated constraint-state vector. For this linear system, we can write the right-hand sides in common state-space form as

$$\mathbf{X}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_1\mathbf{u}_1 + \mathbf{f}_1, \tag{11}$$
$$\mathbf{Y}_1 = \mathbf{C}_1\mathbf{x}_1 + \mathbf{D}_1\mathbf{u}_1, \tag{12}$$

where $\mathbf{f}_1 = \begin{bmatrix} 0 & f_1/m_1 \end{bmatrix}^T$, and the state matrix $\mathbf{A}_1$, input matrix $\mathbf{B}_1$, output matrix $\mathbf{C}_1$, and feed-through matrix $\mathbf{D}_1$, are given by

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ -\frac{k_1}{m_1} & -\frac{c_1}{m_1} \end{bmatrix}, \qquad \mathbf{B}_1 = \begin{bmatrix} 0 \\ \frac{1}{m_1} \end{bmatrix}, \qquad \mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{13}$$

respectively.

Similarly, the governing equation for displacement $q_2(t)$ of the point mass m_2 in System 2 is given by

$$m_2\ddot{q}_2 + (c_c + c_2)\,\dot{q}_2 + (k_c + k_2)\,q_2 = c_c\dot{q}_I + k_c q_I + f_2, \tag{14}$$

for which a general state-space form may be written

$$\dot{\mathbf{x}}_2 = \mathbf{X}_2\left(t, \mathbf{x}_2, \mathbf{u}_2\right), \tag{15}$$
$$\mathbf{y}_2 = \mathbf{Y}_2\left(t, \mathbf{x}_2, \mathbf{u}_2\right). \tag{16}$$

Here, c_2 and k_2 are damping and stiffness coefficients, respectively, c_c and k_c are damping- and stiffness-coupling coefficients, respectively, f_2 is a prescribed force, $\mathbf{x}_2 = \begin{bmatrix} q_2 & \dot{q}_2 \end{bmatrix}^T$, $\mathbf{u}_2 = \begin{bmatrix} q_I & \dot{q}_I \end{bmatrix}^T$, and

$$\mathbf{X}_2 = \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}_2\mathbf{u}_2 + \mathbf{f}_2, \tag{17}$$
$$\mathbf{Y}_2 = \mathbf{C}_2\mathbf{x}_2 + \mathbf{D}_2\mathbf{u}_2, \tag{18}$$

where $\mathbf{f}_2 = \begin{bmatrix} 0 & f_2/m_2 \end{bmatrix}^T$,

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ -\frac{k_c+k_2}{m_2} & -\frac{c_c+c_2}{m_2} \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 0 & 0 \\ \frac{k_c}{m_2} & \frac{c_c}{m_2} \end{bmatrix}, \quad \mathbf{C}_2 = \begin{bmatrix} k_c & c_c \end{bmatrix}, \quad \mathbf{D}_2 = \begin{bmatrix} -k_c & -c_c \end{bmatrix}. \tag{19}$$

Hence, $\mathbf{y}_2 = \begin{bmatrix} c_c\left(\dot{q}_2 - \dot{q}_I\right) + k_c\left(q_2 - q_I\right) \end{bmatrix} = \begin{bmatrix} f_c \end{bmatrix}$. In this form, the input-output relationships governing the data dependence between Systems 1 and 2 are $\mathbf{U}_1\left(\mathbf{u}_1, \mathbf{y}_2\right) = \mathbf{0}$ and $\mathbf{U}_2\left(\mathbf{u}_2, \mathbf{y}_1\right) = \mathbf{0}$, respectively, where

$$\mathbf{U}_1 = \mathbf{u}_1 - \mathbf{y}_2\,, \qquad \mathbf{U}_2 = \mathbf{u}_2 - \mathbf{y}_1. \tag{20}$$

Alternatively, in the monolithic approach, the governing equations (8) and (14) can be written as a single state-space representation, e.g.,

$$\dot{\mathbf{x}}_{1-2} = \mathbf{X}_{1-2}\left(t, \mathbf{x}_{1-2}(t)\right), \tag{21}$$
$$\mathbf{y}_{1-2} = \mathbf{x}_{1-2}\,, \tag{22}$$

where

$$\mathbf{X}_{1-2} = \mathbf{A}\mathbf{x}_{1-2} + \mathbf{f}_{1-2}, \tag{23}$$

$\mathbf{x}_{1-2} = \begin{bmatrix} q_1 & \dot{q}_1 & q_2 & \dot{q}_2 \end{bmatrix}^T$, $\mathbf{f}_{1-2} = \begin{bmatrix} 0 & f_1/m_1 & 0 & f_2/m_2 \end{bmatrix}^T$, and

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_c+k_1}{m_1} & -\frac{c_c+c_1}{m_1} & \frac{k_c}{m_1} & \frac{c_c}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_c}{m_2} & \frac{c_c}{m_2} & -\frac{k_c+k_2}{m_2} & -\frac{c_c+c_2}{m_2} \end{bmatrix}. \tag{24}$$

We note that, here, we do not have constraints or inputs.

II.B.2. Linear 1-DOF Damped Oscillator Connected with a Nonlinear Quasi-Static Cable

We consider here the case where System 2 in the coupled system, as described above, is replaced with System 3 (shown in Figure 5), which is a quasi-static catenary cable where the right end is fixed and the left end is coupled to the displacement of the mass in System 1. The nonlinear relationship between cable horizontal force $H(t)$ and left-end displacement $q_3(t)$ is given by Jonkman and Buhl[14] as

$$0 = q_3 - D + \frac{HL}{AE} + \frac{H}{w}\left(\ln\left(\sqrt{\frac{L^2w^2}{4H^2}+1} + \frac{Lw}{2H}\right) - \ln\left(\sqrt{\frac{L^2w^2}{4H^2}+1} - \frac{Lw}{2H}\right)\right), \tag{25}$$

where D is the distance between the left and right cable ends when $q_3 = 0$, L is the unstretched length of the cable, w is the weight per unit length, E is the modulus of elasticity, and A is the cross-sectional area. In our framework, this partition may be represented as having zero state variables, $\mathbf{x}_3 = \emptyset$, one constraint variable, $\mathbf{z}_3 = \begin{bmatrix} H \end{bmatrix}$, with input $\mathbf{u}_3 = \begin{bmatrix} q_3 \end{bmatrix}$, and output $\mathbf{y}_3 = \mathbf{Y}_3 = \begin{bmatrix} H \end{bmatrix}$. Under the notation described above, the constraint equation $\mathbf{Z}_3\left(t, \mathbf{z}_3, \mathbf{u}_3\right)$ is given by the right side of Eq. (25). For System 1 coupled to System 3, the input-output relationships are $\mathbf{U}_1\left(\mathbf{u}_1, \mathbf{y}_3\right) = \mathbf{0}$ and $\mathbf{U}_3\left(\mathbf{u}_3, \mathbf{y}_1\right) = \mathbf{0}$, where

$$\mathbf{U}_1 = \mathbf{u}_1 - \mathbf{y}_3, \quad \mathbf{U}_3 = \mathbf{u}_3 - \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{y}_1. \tag{26}$$

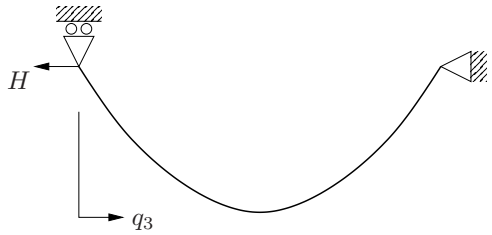


Figure 5. Schematic of System 3, a nonlinear quasi-static cable.

The monolithic representation of System 1 coupled to System 3 with $q_3 = q_1$ is given by

$$\dot{\mathbf{x}}_1 = \mathbf{X}_{1-3}\left(t, \mathbf{x}_1(t), \mathbf{z}_3(t)\right) , \tag{27}$$

$$\mathbf{0} = \mathbf{Z}_{1-3}\left(t, \mathbf{x}_1(t), \mathbf{z}_3(t)\right) , \tag{28}$$

$$\mathbf{y}_{1-3} = \mathbf{Y}_{1-3} , \tag{29}$$

where

$$\mathbf{X}_{1-3} = \mathbf{A}_1\mathbf{x}_1 + \begin{bmatrix} 0 & 1 \end{bmatrix}^T \mathbf{z}_3 + \mathbf{f}_1 , \tag{30}$$

$$\mathbf{Z}_{1-3} = \left[\; q_1 - D + \tfrac{HL}{AE} + \tfrac{H}{w}\left(\ln\left(\sqrt{\tfrac{L^2w^2}{4H^2}+1} + \tfrac{Lw}{2H}\right) - \ln\left(\sqrt{\tfrac{L^2w^2}{4H^2}+1} - \tfrac{Lw}{2H}\right)\right) \; \right] , \tag{31}$$

$$\mathbf{Y}_{1-3} = \begin{bmatrix} q_1 & \dot{q}_1 & H \end{bmatrix} . \tag{32}$$

III. Numerical Coupling Methods

We are interested in solving initial-value problems associated with the monolithic-system representation given by Eqs. (1)–(3) and the partitioned-system representation given by Eqs. (4)–(6). While it is straight forward to time-advance the solution to the monolithic DAEs, time-advancing the partitioned-system solution requires care, as it entails the manipulation and transfer of data between partitions at each time step. In the following subsections we present in detail the different 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*.

III.A. Monolithic System

The monolithic-system DAEs are well suited for implementation in a general DAE solver. Alternatively, in the absence of constraints, any number of first-order-ODE solvers can be employed, e.g, Runge-Kutta or Adams methods. Here, we denote the time advance from discrete-time station $t^n = t^0 + n\Delta t$ to t^{n+1} as

$$\mathbf{x}^n\,, \dot{\mathbf{x}}^n\,, \ldots\,, \dot{\mathbf{x}}^{n-m+1}\,, \mathbf{z}^n\,, \mathbf{X}(t)\,, \mathbf{Z}(t) \xrightarrow{\text{ADV}} \mathbf{x}^{n+1}, \mathbf{z}^{n+1}, \tag{33}$$

where t^0 is the initial time and a n superscript denotes the discrete-time station, where $n \in \{0, \ldots, n_{max}\}$. The notation on the left hand side (LHS) explicitly indicates the data accessible to the underlying integrator, and where $\mathbf{X}(t)$ and $\mathbf{Z}(t)$ express access to the continuous-time forms of 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.

In our formulation, we do not restrict how a particular module will handle its constraint-variable update. For example, a module may simply time lag its constraints. In that case, $\mathbf{z}^n$ is first calculated to be consistent with the states at t^n, and continuous states are then updated; $\mathbf{z}^{n+1}$ is calculated independently and is seen as a guess for the constraint at the next time step. Alternatively, in an implicit DAE solver, $\mathbf{z}^{n+1}$ and $\mathbf{x}^{n+1}$ are determined concurrently. For more discussion on DAE solvers, we refer interested readers to Brenan et al.[15]

III.B. Partitioned System: Explicit and Implicit Coupling

We consider here a multi-physics system that has been modeled as N coupled partitions following the representation shown in Eqs. (4)–(7). To enhance modularity, we assume that data may be communicated between modules only at discrete-time stations. Further, we restrict our analysis to lock-step time integration; all partitions are time advanced with the same time increment Δt. Two data-transfer schemes are considered: explicit and implicit partition coupling, where the latter system is solved and advanced in a predictor-corrector approach.

III.B.1. Explicit Loose

Explicit coupling and time-advancement of partitions is straight forward, as each partition is advanced based on information known at t^n. Our algorithm is summarized as follows for advancement of partition solutions from t^n to t^{n+1} using an m-step integrator. For each partition i, where $i \in \{1, \ldots, N\}$, we assume that we have previous-solution information required by the underlying time-integrator, as well as output data, and input data, i.e.,

$$\mathbf{x}_i^n \,, \dot{\mathbf{x}}_i^n \,, \ldots \,, \dot{\mathbf{x}}_i^{n-m+1} \,, \mathbf{z}_i^n \,, \mathbf{y}_i^n \,, \mathbf{u}_i^n . \tag{34}$$

In general, these input and output values (associated with $t = t^n$) are determined as a solution to the system

$$\begin{aligned}
\mathbf{0} &= \mathbf{U}_1 \left(\mathbf{u}_1^n \,, \ldots \,, \mathbf{u}_N^n \,, \mathbf{y}_1^n \,, \ldots \,, \mathbf{y}_N^n \right) \,, \\
&\cdots \\
\mathbf{0} &= \mathbf{U}_N \left(\mathbf{u}_1^n \,, \ldots \,, \mathbf{u}_N^n \,, \mathbf{y}_1^n \,, \ldots \,, \mathbf{y}_N^n \right) \,, \\
\mathbf{y}_1^n &= \mathbf{Y}_1 \left(t^n \,, \mathbf{x}_1^n \,, \mathbf{z}_1^n \,, \mathbf{u}_1^n \right) , \\
&\cdots \\
\mathbf{y}_N^n &= \mathbf{Y}_N \left(t^n \,, \mathbf{x}_N^n \,, \mathbf{z}_N^n \,, \mathbf{u}_N^n \right) ,
\end{aligned} \tag{35}$$

which, in general, is nonlinear, and the state time derivative at t^n is calculated as

$$\dot{\mathbf{x}}_i^n = \mathbf{X}_i \left(t^n, \mathbf{x}_i^n, \mathbf{z}_i^n, \mathbf{u}_i^n \right) , \tag{36}$$

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

We denote time update under explicit coupling as follows:

$$\mathbf{x}_i^n \,, \dot{\mathbf{x}}_i^n \,, \ldots \,, \dot{\mathbf{x}}_i^{n-m+1} \,, \mathbf{z}_i^n \,, \mathbf{u}_i^n \,, \mathbf{X}_i(t) \,, \mathbf{Z}_i(t) \xrightarrow{\text{ADV}} \mathbf{x}_i^{n+1} \,, \mathbf{z}_i^{n+1} \,, \tag{37}$$

where, again, the notation on the left hand side (LHS) explicitly indicates the data accessible to the underlying integrator, and where $\mathbf{X}_i(t)$ and $\mathbf{Z}_i(t)$ express access to the continuous-time forms of state and constraint right-hand sides, respectively.

III.B.2. Implicit Loose

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 is related to one described elsewhere,[11,13] and can be described as follows for a two-partition system: (1) *predict* the partition 1 state and constraint at t^{n+1} using known information; (2) *substitute* those predicted values into partition 2 through the coupling terms, and calculate a predicted value of the partition 2 state and constraint at t^{n+1}; (3) *correct* the partition 1 state by using the predicted partition 2 data. Steps (2) and (3) may then be repeated in order to improve the solution to the associated partitioned system. This approach is associated with the Gauss-Seidel iterative-solution method,[16] in that Steps (2) and (3) are applied to each partition with the most updated predicted values. If the PC procedure was modified such that each partition was updated with data known from the previous implicit-solve iteration, the method would be associated with the Jacobi iterative-solution method, which offers slower convergence rates but is more suitable to distributed-memory parallel computation. 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 $j_{max} \geq 1$ is the user-defined number of corrector steps to be taken over each time step.

Starting known information: We assume that the following is known at time t^n:

$$\mathbf{x}_i^n\,,\dot{\mathbf{x}}_i^n\,,\ldots\,,\dot{\mathbf{x}}_i^{n-m+1}\,,\mathbf{z}_i^n\,,\mathbf{y}_i^n\,,\mathbf{y}_i^{n-1}\,,\mathbf{u}_i^n\,,\mathbf{u}_i^{n-1}\,, \tag{38}$$

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

Step 1 (Predict): Let $j = 0$, and predict the constraint, state, and output of partition 1 at t^{n+1}.

(Step 1.a) Predict the input and output at t^{n+1} of all partitions through linear extrapolation (over constant Δt):

$$\mathbf{u}_i^{n+1(j)} = 2\mathbf{u}_i^n - \mathbf{u}_i^{n-1}, \tag{39}$$

$$\mathbf{y}_i^{n+1(j)} = 2\mathbf{y}_i^n - \mathbf{y}_i^{n-1}, \tag{40}$$

for each $i \in \{1,\ldots,N\}$, which is an $O(\Delta t^2)$ accurate prediction. Here, the superscript in parentheses indicates the correction iteration.

(Step 1.b) Solve the input-output equation for the predicted input at t^{n+1} for partition 1, i.e., solve

$$\mathbf{0} = \mathbf{U}_1\left(\mathbf{u}_1^{n+1(j+1)}\,,\mathbf{u}_2^{n+1(j)}\,,\ldots\,,\mathbf{u}_N^{n+1(j)}\,,\mathbf{y}_1^{n+1(j)},\ldots,\mathbf{y}_N^{n+1(j)}\right), \tag{41}$$

for $\mathbf{u}_1^{n+1(j+1)}$.

(Step 1.c) Let $\mathbf{u}_1^\alpha = \mathbf{L}\left(\alpha_1, \mathbf{u}_1^n, \mathbf{u}_1^{n+1(j+1)}\right)$ be a linearly interpolated input value based on known and predicted inputs, where

$$\mathbf{L}\left(\alpha,\ \mathbf{u}_1^n,\ \mathbf{u}_1^{n+1}\right) = (1-\alpha)\mathbf{u}_1^n + \alpha\mathbf{u}_1^{n+1}. \tag{42}$$

The value of α_1 is tied to the underlying ODE/DAE integrator. When the underlying ODE or DAE integrator requires other-partition data in the interval $t^n < t \leq t^{n+1}$, $\alpha_1 > 0$. For example, for implicit Adams-Moulton update, or predictor-corrector Adams-Bashforth-Moulton update, $\alpha_1 = 1$. However, for explicit Adams-Bashforth, $\alpha_1 = 0$

(Step 1.d) Advance the solution to yield predicted state and constraint values, i.e.,

$$\mathbf{x}_1^n\,,\dot{\mathbf{x}}_1^n\,,\ldots\,,\dot{\mathbf{x}}_1^{n-m+1}\,,\mathbf{z}_1^n\,,\mathbf{u}_1^\alpha\,,\mathbf{X}_1(t)\,,\mathbf{Z}_1(t) \xrightarrow{\text{ADV}} \mathbf{x}_1^{n+1(j+1)}\,,\mathbf{z}_1^{n+1(j+1)}\,, \tag{43}$$

while $\mathbf{u}_1^{n+1(j+1)}$ is used to calculated the output, i.e.,

$$\mathbf{y}_1^{n+1(j+1)} = \mathbf{Y}_1\left(t^{n+1}, \mathbf{x}_1^{n+1(j+1)}, \mathbf{z}_1^{n+1(j+1)}, \mathbf{u}_1^{n+1(j+1)}\right). \tag{44}$$

Step 2 (Substitute):

(Step 2.a) Solve the input-output equation for the input for partition 2, i.e., solve

$$\mathbf{0} = \mathbf{U}_2\left(\mathbf{u}_1^{n+1(j+1)}, \mathbf{u}_2^{n+1(j+1)}, \mathbf{u}_3^{n+1(j)}, \ldots, \mathbf{u}_N^{n+1(j)}, \mathbf{y}_1^{n+1(j+1)}, \mathbf{y}_2^{n+1(j)}, \mathbf{y}_3^{n+1(j)}, \ldots, \mathbf{y}_N^{n+1(j)}\right), \tag{45}$$

to yield $\mathbf{u}_2^{n+1(j+1)}$.

(Step 2.b) Advance the solution with input $\mathbf{u}_2^\alpha = \mathbf{L}\left(\alpha_2, \mathbf{u}_2^n, \mathbf{u}_2^{n+1(j+1)}\right)$ to yield state, constraint, and output values, i.e.,

$$\mathbf{x}_2^n\,,\dot{\mathbf{x}}_2^n\,,\ldots\,,\dot{\mathbf{x}}_2^{n-m+1}\,,\mathbf{z}_2^n\,,\mathbf{u}_2^\alpha\,,\mathbf{X}_2(t)\,,\mathbf{Z}_2(t) \xrightarrow{\text{ADV}} \mathbf{x}_2^{n+1(j+1)}\,,\mathbf{z}_2^{n+1(j+1)}\,, \tag{46}$$

$$\mathbf{y}_2^{n+1(j+1)} = \mathbf{Y}_2\left(t^{n+1}, \mathbf{x}_2^{n+1(j+1)}, \mathbf{z}_2^{n+1(j+1)}, \mathbf{u}_2^{n+1(j+1)}\right). \tag{47}$$

Again, the value of α_2 is related to the underlying ODE/DAE integrator.

(Step 2.c) In the same manner as in Step 2.b, advance the solution of the remaining $N-2$ partitions ($\mathbf{z}_i^{n+1(j+1)}$, $\mathbf{x}_i^{n+1(j+1)}$ for $i \in \{3,\ldots,N\}$), always calculating the input with the most updated outputs from other partitions, i.e., the input equation for partition $i \in \{3,\ldots,N\}$ is

$$\mathbf{0} = \mathbf{U}_i\left(\mathbf{u}_1^{n+1(j+1)},\ldots,\mathbf{u}_i^{n+1(j+1)},\mathbf{u}_{i+1}^{n+1(j)},\ldots,\mathbf{u}_N^{n+1(j)},\mathbf{y}_1^{n+1(j+1)},\ldots,\mathbf{y}_{i-1}^{n+1(j+1)},\mathbf{y}_i^{n+1(j)},\ldots,\mathbf{y}_N^{n+1(j)}\right), \tag{48}$$

which is solved to yield $\mathbf{u}_i^{n+1(j+1)}$.

Step 3 (Correct): Correct the solution for partition 1.

(Step 3.a) Solve the input-output equation for the input for partition 1, i.e., solve

$$\mathbf{0} = \mathbf{U}_1\left(\mathbf{u}_1^{n+1(j+2)},\mathbf{u}_2^{n+1(j+1)},\ldots,\mathbf{u}_N^{n+1(j+1)},\mathbf{y}_1^{n+1(j+1)},\ldots,\mathbf{y}_N^{n+1(j+1)}\right), \tag{49}$$

to yield $\mathbf{u}_1^{n+1(j+2)}$.

(Step 3.b) Advance the solution with input $\mathbf{u}_1^{\alpha} = \mathbf{L}\left(\alpha_1,\mathbf{u}_1^n,\mathbf{u}_1^{n+1(j+2)}\right)$ to yield state, constraint, and output values, i.e.,

$$\mathbf{x}_1^n,\dot{\mathbf{x}}_1^n,\ldots,\dot{\mathbf{x}}_1^{n-m+1},\mathbf{z}_1^n,\mathbf{u}_1^{\alpha},\mathbf{X}_1(t),\mathbf{Z}_1(t) \xrightarrow{\text{ADV}} \mathbf{x}_1^{n+1(j+2)},\mathbf{z}_1^{n+1(j+2)}, \tag{50}$$

$$\mathbf{y}_1^{n+1(j+2)} = \mathbf{Y}_1\left(t^{n+1},\mathbf{x}_1^{n+1(j+2)},\mathbf{z}_1^{n+1(j+2)},\mathbf{u}_1^{n+1(j+2)}\right). \tag{51}$$

Let $j = j+1$. If $j < j_{max}$, repeat Steps 2 and 3. If $j = j_{max}$, proceed to Step 4.

Step 4: Save all the final variables,

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^{n+1(j_{max})},\quad \mathbf{z}_i^{n+1} = \mathbf{z}_i^{n+1(j_{max})},\quad \mathbf{y}_i^{n+1} = \mathbf{y}_i^{n+1(j_{max})},\quad \mathbf{u}_i^{n+1} = \mathbf{u}_i^{n+1(j_{max})}, \tag{52}$$

for each $i \in \{2,\ldots,N\}$, and

$$\mathbf{x}_1^{n+1} = \mathbf{x}_1^{n+1(j_{max}+1)},\quad \mathbf{z}_1^{n+1} = \mathbf{z}_i^{n+1(j_{max}+1)},\quad \mathbf{y}_1^{n+1} = \mathbf{y}_1^{n+1(j_{max}+1)},\quad \mathbf{u}_1^{n+1} = \mathbf{u}_1^{n+1(j_{max}+1)}, \tag{53}$$

which completes solution advancement to time t^{n+1}. For use in the next time step, calculate the state-derivative RHS

$$\dot{\mathbf{x}}_i^{n+1} = \mathbf{X}_i\left(t^{n+1},\mathbf{x}_i^{n+1},\mathbf{z}_i^{n+1},\mathbf{u}_i^{n+1}\right), \tag{54}$$

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

PC(j_{max}), as described above, requires ($j_{max}N+1$) ADV evaluations per time step. Hence, for a given Δt, the PC approach requires more operations than explicit coupling. However, this additional cost per time step is negated if the PC-approach is significantly more numerically stable and/or more accurate than explicit coupling. The above algorithm could be modified to stop iterating after convergence to within some tolerance, e.g., iterations continue until $\left|\mathbf{x}_i^{n+1(j+1)} - \mathbf{x}_i^{n+1(j)}\right| < \epsilon$ and $\left|\mathbf{z}_i^{n+1(j+1)} - \mathbf{z}_i^{n+1(j)}\right| < \epsilon$ for each $i \in \{1,\ldots,N\}$, where ϵ is a user-specified tolerance.

IV. Application Example Results and Discussions

In this section, we examine the numerical performance of the three coupling approaches, described in Section III, using the two application examples, described in Section II.B. We examine the numerical stability and accuracy of three standard, fixed-time-increment methods: (1) fourth-order, explicit *Runge-Kutta* (RK4) which is a single-step (though "multi-stage") method, (2) fourth-order *Adams-Bashforth-Moulton* (ABM4), which is an explicit multi-step method with an implicit dependence on partition-interaction data, and (3) fourth-order explicit *Adams-Bashforth* (AB4). ABM4 also serves as a proxy for a multi-step implicit time integrator as it has the same implicit dependence on other-partition data.

We note that the RK4 method is self-starting, while ABM4 and AB4 are not. In our simulations with ABM4 and AB4, the first four time steps are initialized with benchmark solutions. In practice, ABM4

and AB4 could be initialized with RK4 integrated with sufficiently small time increments. In our PC implementation of RK4, numerical experiments showed that $\alpha = 0.5$ provides the best performance. RK4 performs four RHS evaluations over a time step; providing a midpoint average of the input seems prudent. For ABM4, PC coupling is accomplished with $\alpha = 1$, while $\alpha = 0$ is employed for AB4. As such, AB4 PC simulations for System 1 coupled to System 2 are equivalent to those under explicit coupling. However, in the PC simulations where System 1 is coupled to System 3, the constraint is updated with $\alpha = 1$. Importantly, PC-coupling (as described in Section III.B.2) was implemented with System 2 (or 3) as the first partition, and with System 1 as the second partition. Hence, System 2 (or 3) was *predicted* (Step 1), System 1 was *substituted* (Step 2), and then System 2 (or 3) was *corrected* (Step 3).

IV.A. Linear 2-DOF Damped Oscillator System

We implemented the three numerical-coupling algorithms described in Section III (monolithic, explicit loose, and PC implicit loose) for the linear 2-DOF damped oscillator system shown in Figure 4. Throughout this subsection, solutions were calculated for $0 \leq t \leq 30$ in the absence of external forcing ($f_1 = f_2 = 0$), with (dimensionless) initial conditions given by

$$q_1(0) = 1, \quad \dot{q}_1(0) = 0, \quad q_2(0) = 0, \quad \dot{q}_2(0) = 0, \tag{55}$$

and with baseline system parameters listed in Table 1. For accuracy assessment, a closed-form exact solution was calculated; Figure 6 shows the exact benchmark displacement histories.

Table 1. System 1 and 2 baseline parameters associated with the test simulations.

System 1			System 2				
m_1	c_1	k_1	m_2	c_2	k_2	c_c	k_c
1.0	0.1	1.0	1.0	0.1	1.0	0.01	1.0

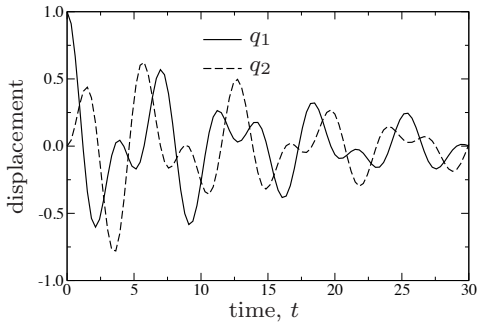


Figure 6. Benchmark displacement histories $q_1(t)$ and $q_2(t)$ for the system parameters listed in Table 1.

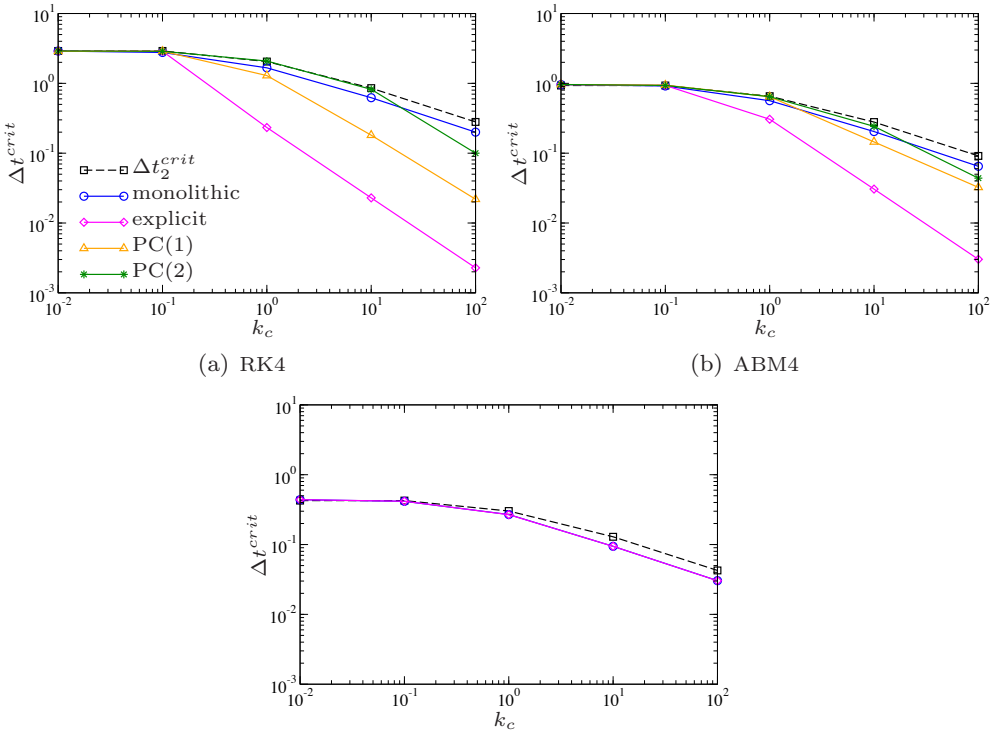
IV.A.1. Stability Analysis

Figure 7 shows numerically determined critical time increments Δt^{crit} as a function of coupling-spring stiffness (k_c) for the three coupling schemes with the three time integrators. Here, Δt^{crit} is the largest time increment for which bounded numerical solutions were produced. Also included are the critical time increments associated with uncoupled integration of System 2, Δt_2^{crit}, which was determined with the displacement of the interface node held fixed. For all cases shown, $\Delta t_2^{crit} < \Delta t_1^{crit}$, where Δt_1^{crit} is the critical time increment for the uncoupled System 1. In regard to the numerical-stability data shown in Figure 7, we make the following observations:

- $\Delta t^{max} = \min(\Delta t_1^{crit}, \Delta t_2^{crit})$ is an upper bound on stable time increments for all cases examined.
- For integration of the monolithic system, RK4 was the most stable, while ABM4 was significantly more stable than AB4. However, RK4 requires four RHS evaluations per time step, ABM4 requires

two, while AB4 requires only one. From an accuracy vs. computational-cost perspective, the three integrators are comparable performers for the monolithic system.

- Explicit coupling is prone to numerical instability for RK4 and ABM4 integration schemes, and can require $\Delta t \ll \Delta t^{max}$ for stable solutions. This is, perhaps, not surprising. RK4 and ABM4 both have an implicit dependence on input (from the other partition) over $t^n < t \leq t^{n+1}$, but the input is held constant in the implementation here, which introduces additional stiffness. Explicitly coupled AB4, however, exhibits stability characteristics indistinguishable from those of the AB4 monolithic treatment.
- For RK4 and ABM4 integration, PC coupling is significantly more stable than explicit coupling. The stability of PC(2) is significantly better than that of PC(1). For some RK4 and ABM4 cases, PC(2) exhibited critical time increments that exceeded those of the monolithic approach (but were still bounded by Δt^{max}).
- While not shown, PC(3) solutions exhibited stability characteristics virtually indistinguishable from those of PC(2).



(a) RK4 (b) ABM4

(c) AB4

Figure 7. Numerically determined critical time increments as a function of coupling-spring stiffness for (a) RK4, (b) ABM4, and (c) AB4 time integration and with the three coupling schemes. Only monolithic and explicit coupling is shown for AB4, for which PC-implicit and explicit are equivalent. Δt_2^{crit} corresponds to the critical increment associated with uncoupled integration of System 2.

IV.A.2. Accuracy Analysis

We examine here the accuracy of the three numerical coupling methods for the three numerical integrators, and with the baseline parameters listed in Table 1. Figure 8 shows normalized root-mean-square (RMS) error of the numerical solutions for the displacement of System 1 over the time interval $0 \leq t \leq 30$. Normalized RMS error for n_{max} numerical response values q_1^n, where $q_1^n \approx q_1(t^n)$, was calculated as

$$\varepsilon\left(q_1\right) = \sqrt{\frac{\sum_{k=0}^{n_{max}} \left[q_1^k - q_b(t^k)\right]^2}{\sum_{k=0}^{n_{max}} \left[q_b(t^k)\right]^2}}, \tag{56}$$

where $q_b(t)$ is a known benchmark solution; $q_b(t)$ is either known analytically or is interpolated from a highly resolved numerical solution. In regard to the numerical-accuracy data shown in Figure 8, we make the following observations:

- Integration of the monolithic systems exhibits expected accuracy (convergence rates) for the three fourth-order time integrators. RK4 and ABM4 exhibit similar accuracy that is better than that of AB4. However, RK4 requires four ODE-RHS evaluations per time step, ABM4 requires two, while AB4 requires only one. From this perspective, the three time integrators applied to the monolithic system exhibit similar computational cost for a given accuracy.
- Explicit coupling with RK4 and ABM4 exhibits only first-order accuracy, while fourth-order accuracy is maintained with AB4 integration. RK4 and ABM4 integrators are clearly hindered by their implicit dependence on other-partition data.
- For RK4 integration, PC(1) provides only first-order accuracy, while second-order accuracy can be achieved with PC(2). Numerical experiments showed that RK4 with PC($j > 2$) is no better than second-order accurate. This limitation in accuracy is attributed to the fact that input data is held constant over the time step, even though a high-order multi-step method like RK4 performs RHS evaluations within the time step.
- For ABM4, PC(1) exhibits second-order accuracy, while PC(2) exhibits the desired fourth-order accuracy of the underlying time integrator.
- For this simple partitioned test system, AB4 integration with explicit coupling is clearly the best performer. While ABM4 with PC(2) exhibits slightly better accuracy for a given Δt, it requires five "steps" with ABM4 (one prediction, two substitutions, and two correctors) per time step. Alternatively, AB4 with explicit coupling requires only two "steps" (one per module) per time step. However, this strong performance is largely tied to the simplicity of this non-stiff ODE system; AB4 and explicit coupling may well be a poor choice for stiff systems and/or DAE systems.

IV.B. A Linear 1-DOF Damped Oscillator Connected with a Nonlinear Quasi-Static Cable

We examine here numerical-accuracy results for System 1, a one-DOF damped oscillator, coupled to System 3, a nonlinear quasi-static cable, as described in Section II.B.2. Dimensionless property values for both systems are listed in Table 2. Benchmark solutions for the coupled System 1 and System 3 were generated with IDA, from the Sundials[17] software suite. IDA employs variable-step variable-order backwards differentiation formulas. Cable displacement and force benchmark histories are shown in Figure 9(a) and (b), respectively, for $0 \leq t \leq 20$ and for the following initial conditions: $q_1(0) = 1$, $\dot{q}_1(0) = 0$, $q_3(0) = 1$. For comparison, Figure 9(a) includes $q_1(t)$ calculated in the absence of force from the attached cable; the cable force clearly has a significant impact on the response of $q_1(t)$.

Table 2. System 1 and 3 dimensionless parameters associated with the test simulations.

System 1			System 3				
m_1	c_1	k_1	D	L	A	E	W
1.0	0.1	3.0	1.5	3.0	1.0	50000	2.9033

In our numerical implementation, System 1 is treated in the same manner as in the Section IV.A, where one of several time-integrators can be employed. The nonlinear constraint equation in System 3 is solved with a standard Newton-Raphson (NR) algorithm with a tolerance of 10^{-10}. The System 3 module is purely a root-finding algorithm; the output will correspond to the same time as that of the input.

Due to the limited performance exhibited by RK4 in our coupling schemes, we focus here on ABM4 and AB4 integrators coupled to the NR constraint solve. Figure 10 shows RMS errors of $q_1(t)$ histories produced with ABM4 and AB4 integration of System 1 and with explicit, PC(1), and PC(2) coupling to System 3. Explicit coupling yields only first-order convergence, as the force from the cable (a constraint variable) is time lagged by one step, which introduced $O(\Delta t)$ error. This first-order error can be reduced to $O(\Delta t^2)$ if the cable force constraint is calculated based on a linearly extrapolated input as in Eq. (39) in PC

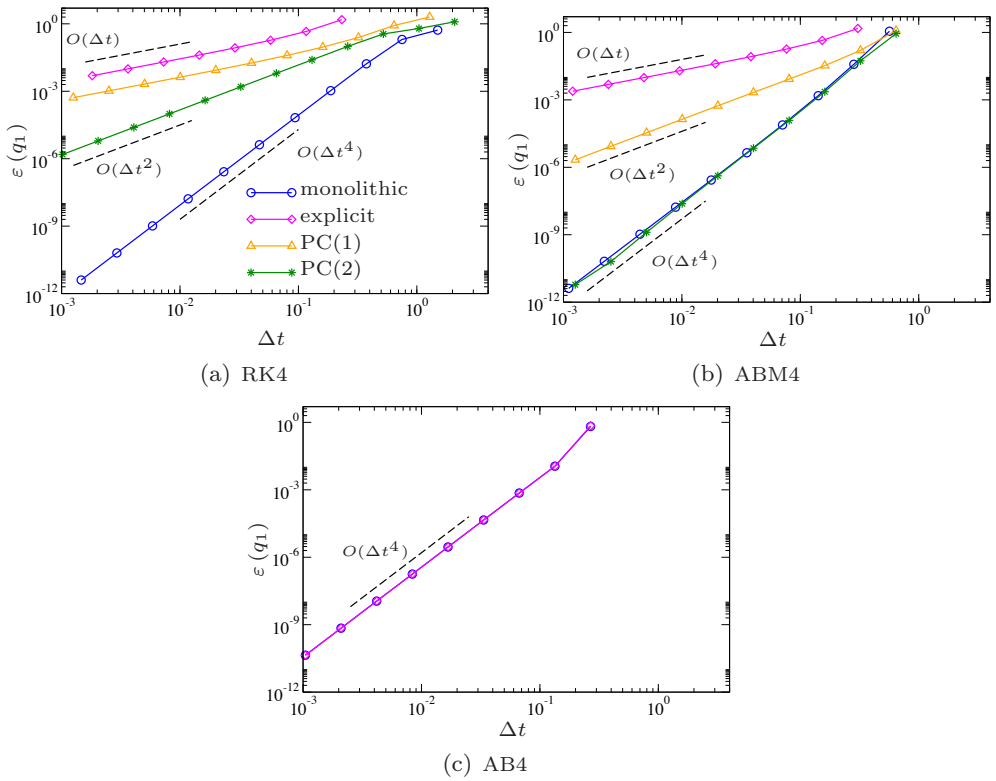


Figure 8. Normalized RMS error of System 1 displacement histories over $0 \leq t \leq 30$ as a function of time increment for (a) RK4, (b) ABM4, and (c) AB4 time integration and the three coupling schemes. Only monolithic and explicit coupling is shown for AB4, for which PC-implicit and explicit are equivalent. Dashed lines show reference convergence rates.

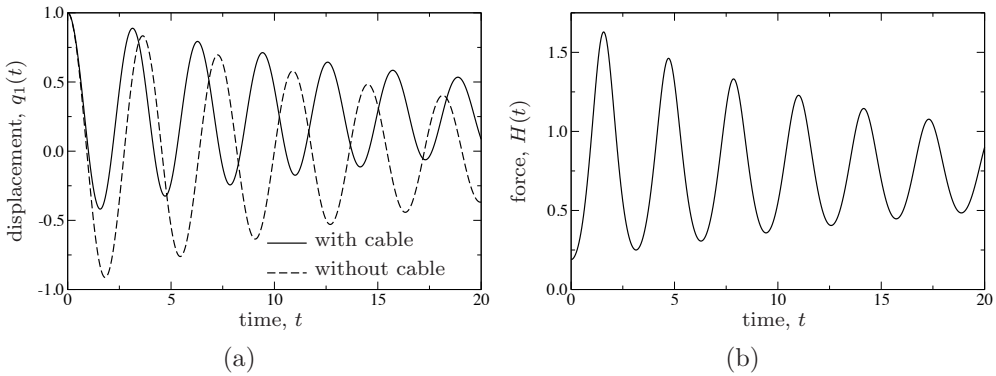


Figure 9. Numerically calculated benchmark response histories for (a) System 1 displacement, and (b) System 3 cable force. System 1 displacements are shown with and without the effects of the attached cable.

coupling. However, PC(1) coupling provides second-order accuracy for ABM4-NR and fourth-order accuracy for AB4-NR. AB4-NR achieves desired accuracy with only PC(1) because System 1 is advanced with the consistent value of the cable force (at t^n), and the System 3 cable force is then corrected to be consistent with displacement at t^{n+1}. For this coupled system, ABM4-NR required PC(2) coupling for fourth-order accuracy. This is because, in the first cycle (embodied in PC(1)), ABM4 is advanced with a $O(\Delta t^2)$ prediction of the constraint; a second correction cycle is required to obtain desired accuracy. Here, solutions converge to error of $O(10^{-6})$, rather than machine precision, due to the discrete nature of the benchmark solution and the interpolation of that solution for comparison.

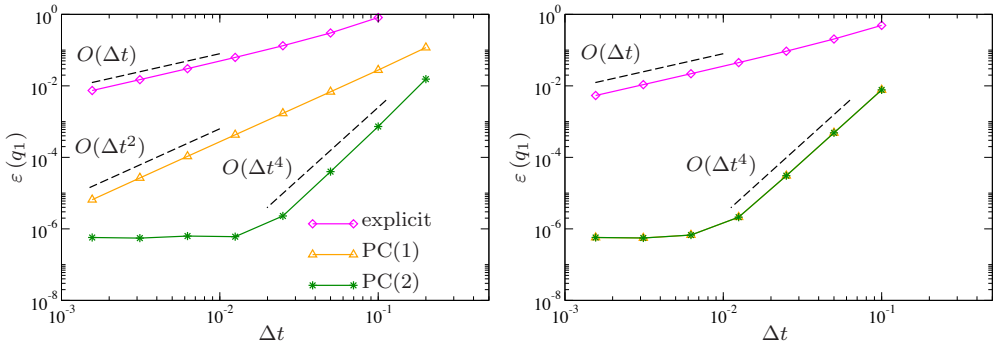


Figure 10. Normalized RMS errors of $q_1(t)$ histories calculated for $0 \leq t \leq 20$ with (a) ABM4-NR and (b) AB4-NR and with various coupling schemes; PC(1) and PC(2) results are indistinguishable for AB4-NR. Dashed lines show reference convergence rates.

V. Conclusions

This work was motivated by a major restructuring of the FAST wind turbine CAE tool suite; details of the restructuring are described in a companion paper.[6] Our goal was to guide FAST-module developers and to provide them with stable, accurate, and efficient numerical coupling strategies for multi-physics multi-module simulations. Using two simple example systems, we described possible partition choices and methods for coupling those partitions, under the restriction that data is communicated between partitions only at full time steps. We examined two loose-coupling schemes, explicit and implicit. Loose coupling implies that modules may be updated sequentially, rather than concurrently, which provides flexibility in modularization and parallelization. In explicit coupling, partitions are advanced in time using only known information. In implicit coupling, one or more partitions require data from other partitions at the next step. Here, implicit coupling is accomplished in loose-coupling manner through a predictor-corrector approach. Time integration was examined with three fixed-step fourth-order schemes: Runge-Kutta (explicit single-step method), Adams-Bashforth-Moulton (predictor-corrector multi-step method), and Adams-Bashforth (explicit multi-step method).

Through numerical experiments, it was shown that, if the underlying ODE integrator has an implicit dependence on other-partition data (as in RK and ABM), or if there is an inherent implicit dependence due to constraint equations (as in our System 3 coupled to System 1), explicit coupling can be dramatically less stable and less accurate than simulations performed with the monolithic system. However, predictor-corrector implicit coupling can restore desired accuracy and stability. For coupled partitions without constraints, explicit time integration with AB and explicit loose coupling exhibited desired accuracy and stability. Based on this work we make the following recommendations. First, for general multi-partition systems, we recommend implicit coupling through a predictor-corrector approach, which is significantly more robust than explicit coupling. Second, high-order single step methods such as Runge-Kutta should be avoided under the restriction that partitions may only communicate at whole time steps.

In future work, we will extend our analysis to include time-step subcycling,[13,18,19] where one or more partitions is updated with a time increment that is a fraction of the largest time increment used by other systems. Further, we will examine accuracy and stability associated with higher-order extrapolation of input and output data in our PC algorithm; e.g., Eqs. (39) and (40), respectively. Linear extrapolation was employed in this work; however, quadratic extrapolation may provide significant improvement in accuracy with limited stability degradation.[20] Regardless of the extrapolation accuracy, desired accuracy of the underlying multi-step time integrators appears attainable if a sufficient number of correction steps are applied. In our examples, fourth-order accuracy with multi-step methods was attained with two correction iterations (and only one in some cases). We will also compare the loose-coupling schemes described in this paper with tight implicit coupling, where the partitioned systems are assembled into a system suitable for concurrent time update with a single DAE solver, and where constraints are naturally introduced from the input-output relations. Future work may also include the examination of implicit numerical integrators, and models with discrete-time states.

Acknowledgments

The authors acknowledge useful discussions with Bonnie Jonkman and John Michalakes. Thanks also to Dr. Ray Grout and Dr. Peter Graf for reviewing the paper prior to submission.

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 TurbinesModel 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," 2012, To appear in the proceedings of the 32nd ASME Wind Energy Symposium, 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.

[7]Larson, J. W., Jacob, R. L., Foster, I., and Guo, J., "The Model Coupling Toolkit," *Proceedings of the 2001 International Conference on Computational Science*, 2001.

[8]Bettencourt, M. T., "Distributed Model Coupling Framework," *Proceedings of HPDC-11*, 2002.

[9]Elwasif, W. R., Bernholdt, D. E., Shet, A. G., Foley, S. S., Bramley, R., Batchelor, D. B., and Berry, L. A., "The design and implementation of the SWIM Integrated Plasma Simulator," *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010.

[10]Pawlowski, R., Bartlett, R., Belcourt, N., Hooper, R., and Schmidt, R., "A Theory Manual for Multi-Physics Code Coupling in LIME," Tech. Rep. SAND2011-2195, Sandial National Laboratories, Albuquerque, New Mexico and Livermore, California, March 2011.

[11]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.

[12]Farhat, C., van der Zee, K. G., and Geuzaine, P., "Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, 2006, pp. 1973–2001.

[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]Jonkman, J. M. and Buhl, M. L., "Development and Verification of a Fully Coupled Simulator for Offshore Wind Turbines," *Proceedings of the 45th AIAA Aerospace Sciences Meeting*, Reno, NV, 2007, also published in tech. report NREL/CP-500-40979 National Renewable Energy Laboratory, Golden, CO.

[15]Brenan, K. E., Campbell, S. L., and Petzold, L. R., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Society of Industrial and Applied Mathematics, 1996.

[16]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.

[17]Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward., C. S., "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software*, Vol. 31, 2005, pp. 363–396.

[18]Belytschko, T., "Partitioned and Adaptive Algorithms for Explicit Time Integration," *Nonlinear Finite Element Analysis in Structural Mechanics*, edited by W. Wunderlich, E. Stein, and K. J. Bath, Springer-Verlag, 1981, pp. 572–584.

[19]Hulbert, G. M. and Hughes, T. J. R., "Numerical Evaluation and Comparison of Subcycling Algorithms for Structural Dynamics," Tech. Rep. DNA-TR-88-8, Defense Nuclear Agency, Washington, D.C., 1988.

[20]Elliott, A. S., "A Highly Efficient, General-Purpose Approach for Co-Simulation with ADAMS®," *Proceedings of the 15th European ADAMS Users' Conference*, 2000.