Parallel Finite Element Framework for Rotorcraft Multibody Dynamics and Discrete Adjoint Sensitivities

Komahan Boopathy* and Graeme J. Kennedy†

Georgia Institute of Technology, Atlanta, GA, 30332-0150, USA.

A parallel finite element framework for high-fidelity structural dynamic analysis and gradient evaluation using the discrete adjoint method is presented. The framework is intended to be used for gradient-based design optimization of flexible multibody dynamic systems such as rotorcraft. The formulation of governing equations, treatment of kinematic constraints, and the evaluation of functionals of interest and their derivatives are addressed. A minimal set of routines needed to implement the discrete adjoint method are proposed. The governing equations are integrated in time using a diagonally implicit Runge–Kutta method for second-order systems of equations. The formulation of the corresponding time dependent discrete adjoint equations are presented and are numerically verified using the complex-step method. A verification of the dynamics, an assessment of parallel scalability of the analysis and derivative evaluation techniques, and a demonstration of the design capability are presented.

*Ph.D. Candidate, School of Aerospace Engineering, komahan@gatech.edu, AIAA Student Member.
†Assistant Professor, School of Aerospace Engineering, graeme.kennedy@aerospace.gatech.edu, AIAA Member.

Nomenclature

a, b, c	first-order diagonally implicit Runge–Kutta coefficients
A, B	second-order diagonally implicit Runge–Kutta coefficients
E	Young's modulus of the material
f	functional of interest
F	integrand for functional of interest
$\bar{g}$	optimization problem constraints
$\mathbf{g}$	kinematic constraints
$\mathbf{A}$	Jacobian of kinematic constraints
h	time step size
$\mathscr{L}$	Lagrangian for equations of motion
L	Lagrangian for discrete adjoint
m	mass
N	number of time steps
$\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$	state variables and their time derivatives
$\mathbf{R}$	residual of constrained equations of motion for multibody system
s	number of diagonally implicit Runge–Kutta stages
$\mathbf{S}$, $\mathbf{T}$	residual of diagonally implicit Runge–Kutta step update equations
t	time
T	kinetic energy
$\mathbf{u}$, $\dot{\mathbf{u}}$, $\ddot{\mathbf{u}}$	state variables and their time derivatives at each stage
V	potential energy
$\mathbf{x}$	design variables

α, β, γ	scalar coefficients
δ	complex-step perturbation size
ε_r, ε_a	relative and absolute tolerance
λ	stage adjoint variables
μ	Lagrange multiplier for holonomic constraints
ν	Poisson's ratio of the material
ρ	density of material
ρ_{ks}	aggregation parameter
σ_{ks}	KS aggregation of von Mises stress
σ_{vm}	von Mises stress
σ_{max}	maximum von Mises stress
ϕ, ψ	step adjoint variables
Ω	spatial domain

I. Introduction

Accurate prediction of rotorcraft performance requires coupling multiple disciplines together to form an integrated multidisciplinary rotorcraft analysis. Depending on the performance metrics of interest, the disciplines needed for rotorcraft analysis include aerodynamics, structures, and structural dynamics at a minimum, and may also include control systems, acoustics, and propulsion systems. Comprehensive rotorcraft simulation tools provide good performance prediction capabilities by integrating low- and medium-fidelity models that capture the most important disciplinary physics. These comprehensive codes, such as RCAS [1], CAMRAD [2] and the general purpose flexible multibody dynamics analysis tool Dymore [3], have been used throughout industry and academia in a wide range of applications. However, as more advanced rotorcraft configurations are considered, such as tilt-rotors or co-axial rotor concepts, comprehensive analysis tools may not provide sufficient accuracy. Furthermore, due to the tightly coupled physics in rotorcraft problems, poor structural dynamics prediction can lead to poor aerodynamic predictions or vice-versa. Thus, high-fidelity disciplinary models are needed to accurately predict rotorcraft performance, especially when novel concepts are considered. In addition to accurate performance estimates, rotorcraft design tools should also provide a means to systematically improve designs. Such a capability is not available in existing comprehensive codes. Gradient-based design tools offer the potential to identify the design modifications necessary to achieve demanding rotorcraft performance goals such as higher forward flight speeds, higher operational altitudes, and longer endurance limits. However, the potential of gradient-based optimization can only be achieved if the gradient of a performance metric can be evaluated efficiently. An efficient adjoint-based gradient evaluation method makes optimization afford-

able since it can compute the gradient with a computational cost that is roughly the same as the original analysis.

The focus of this work is on parallel high-fidelity simulation techniques and the implementation of a discrete adjoint-based derivative evaluation method for time-accurate flexible multibody dynamic simulations. These are incorporated within a modular, extensible framework implemented using object-oriented software design principles and intended to work with other disciplines. These capabilities are implemented within the Toolkit for Analysis of Composite Structures (TACS), a parallel framework for finite element analysis [4], that is available as open source software[a]. These developments represents a step towards a high-fidelity multidisciplinary simulation tool for rotorcraft simulation and gradient-based design optimization. While the focus of this work is on addressing analysis and adjoint-based gradient evaluation techniques for structural and multibody dynamics, the proposed framework can be coupled to other disciplines, such as aerodynamics, using the FUNtoFEM aeroelastic interface-layer [5–7].

Structural models for rotorcraft range in complexity from simple beam models to complex 3D models that combine either shell elements or full 3D elements. Beam models can often provide accurate predictions of deflection, twist, and stresses in rotor blades for preliminary design purposes [8, 9]. However, beam models may not be adequate for modeling rapid variation in blade cross-section, local variation of blade sweep, localized structural damage assessment, or the analysis of load bearing components. Several authors have developed higher-fidelity structural and dynamics models that combine shell or 3D elements [10–13] in order to improve modeling accuracy. While these techniques have more sophisticated modeling capability, they also require many more degrees of freedom than even the most advanced nonlinear beam formulations. Typically, the use of shell or 3D elements requires between 2 to 4 orders of magnitude more degrees of freedom compared to beam models. As a result, higher-fidelity structural models naturally call for the use of parallel solution techniques.

While the size of structural models for rotorcraft simulations grow, the need for gradient-based design capabilities necessitates the implementation of adjoint-based gradient evaluation methods [4, 14–26]. A key property of the adjoint method is that the computational cost of evaluating the gradient of a single functional of interest is nearly independent of the number of design variables. This property is critical for high-fidelity multibody dynamics models that may require detailed cross-sectional design parametrizations for the rotor blade geometry. However, the overall gradient evaluation cost grows linearly with the number of functionals in the design formulation. As a result, in cases where the number of functionals is large, the adjoint method can become expensive. This is especially a concern in structural design with strength criteria where a large number of stress constraints may be required. In such cases, constraint aggregation methods [27, 28] can be used to reduce the number of functionals, thereby reducing the gradient evaluation cost. The adjoint method has been applied to structural [4, 14–17], aerodynamic [18–20], coupled aeroelastic [21, 22] and flexible multibody dynamics cases [23–26]. Cao et al. [23] presented general adjoint methods for differential algebraic equations in first-order systems, or systems that have been reduced to first-order form, with applications to multibody dynamics. Nachbagauer et al. [26] presented a continuous adjoint method for multibody dynamics with

[a]TACS is available at https://github.com/gjkennedy/tacs

a focus on applications for inverse dynamics and parameter identification for rigid body problems. Dopico et al. [25] developed an approach to the sensitivity analysis of multibody systems based on Maggi's formulation of the governing equations, using the direct sensitivity approach and the adjoint method. Ding et al. [24] presented an adjoint method for computing the second derivative of functionals of interest. The papers presented in the literature contain either serial adjoint implementations for dynamics that consist of rigid bodies and flexible beam elements [23–26] or feature parallel high-fidelity analysis capabilities that lack an adjoint method [10, 11]. Therefore, high-fidelity analysis, adjoint-based gradient evaluation, and parallelism are not entirely available within the same computational framework.

To address this deficiency, we present a high-fidelity finite-element based analysis and optimization framework that is suitable for rotorcraft computations. This work differs from previous mathematical formulations of the adjoint method for flexible multibody dynamics by utilizing a second-order descriptor form of the governing equations in combination with a diagonally implicit Runge–Kutta time-marching method. The key aspects of the implementation of the adjoint that enable an extensible and modular framework are described. The parallel scalability of the implementation is also demonstrated for large-scale flexible multibody models.

The remainder of the paper is structured as follows. Section II describes the flexible multibody dynamics equations, their solution strategy and the functional aggregation method used within this work. Section III details the development of time-marching techniques specialized to the form of governing equations and the formulation of the corresponding adjoint equations. Section IV describes the implementation aspects of the discrete adjoint method. Section V presents verification studies for the dynamics and discrete adjoint formulation and demonstrates the framework on rotor hub dynamic analysis and optimization, along with parallel performance evaluation. Section VII summarizes the conclusions from this work.

II. Background and Methodology

This section presents an overview of the governing equations of motion, the treatment of kinematic constraints, the solution of differential algebraic equations, and the evaluation of functionals of interest.

A. Governing Equations of Motion

The equations governing the motion of flexible multibody systems can be derived using a number of different methods [29, 30]. This work employs an approach based on the constrained Euler–Lagrange equations that leads to a system of differential algebraic equations (DAEs). The system of DAEs consists of both a set of differential equations and a set of algebraic constraints that restricts the kinematics using Lagrange multipliers. One advantage of using the Euler–Lagrange equations is that they can be numerically verified for consistency with the kinetic and potential energy expressions and the constraint equations using finite-difference or complex-step methods. The Lagrangian for

the equations of motion is defined as

$$\mathscr{L}(\dot{\mathbf{w}},\mathbf{w}) \triangleq T(\dot{\mathbf{w}},\mathbf{w}) - V(\mathbf{w}) \tag{1}$$

where $\mathbf{w}$ is a vector that contains the displacements and Euler parameters for rotation matrix parametrization, and $T(\dot{\mathbf{w}},\mathbf{w})$ and $V(\mathbf{w})$ are the kinetic and potential energy of the system, respectively. The kinetic energy and potential energies are computed as integrals over each finite element within the rotorcraft assembly based on the element type [29, 31, 32]. In this work, the kinematics of the flexible bodies are restricted through a set of holonomic constraints of the form $\mathbf{g}(\mathbf{w}) = 0$, where the extension to nonholonomic constraints is straightforward. The Jacobian of the kinematic constraints is $\mathbf{A} = \partial\mathbf{g}/\partial\mathbf{w}$. With these definitions, the governing equations of motion in second-order descriptor form are

$$\mathbf{R}(\ddot{\mathbf{q}},\dot{\mathbf{q}},\mathbf{q},\mathbf{x},t) \triangleq \begin{bmatrix} \dfrac{d}{dt}\left(\dfrac{\partial\mathscr{L}}{\partial\dot{\mathbf{w}}}\right) - \dfrac{\partial\mathscr{L}}{\partial\mathbf{w}} - \mathbf{A}^T\mu \\ \mathbf{g}(\mathbf{w}) \end{bmatrix} = 0. \tag{2}$$

Here the vector $\mathbf{q} = (\mathbf{w},\ \mu)$ includes both the degrees of freedom $\mathbf{w}$ and the Lagrange multipliers μ. Note that the vector of design variables, $\mathbf{x}$, is included to reflect the dependence of the system of equations on design variables. In the following sections, it will be necessary to compute the Jacobian of the governing equations with respect to the state variables and their derivatives. These Jacobian matrices always appear as a linear combinations of the form

$$\mathbf{J} = \gamma\frac{\partial\mathbf{R}}{\partial\ddot{\mathbf{q}}} + \beta\frac{\partial\mathbf{R}}{\partial\dot{\mathbf{q}}} + \alpha\frac{\partial\mathbf{R}}{\partial\mathbf{q}},$$

where α, β, and γ are scalar coefficients.

Often authors convert the governing system of equations into first-order form so that they can be solved using existing numerical libraries for solving first-order systems [33–35]. In this work, however, the second-order descriptor form (2) is employed directly since it retains the underlying structure of Euler–Lagrange equations and leads to a system of adjoint equations that is simpler to implement and easier to interpret. Haug et al. [36] similarly solves the equations of motion in their natural form as second-order for rigid multibody dynamic formulations. The descriptor form of the governing equations can be solved using implicit time integration schemes that are required to solve the numerically stiff governing equations associated with flexible multibody dynamics. The descriptor form (2) provides the basis for different element types implemented in the framework based on the finite element method. The elements within the framework, at present, consist of rigid bodies, flexible quadratic beam elements employing a Timoshenko beam formulation, and flexible bi-quadratic shell elements employing a Reissner–Mindlin formulation. To avoid shear locking, the beam and shell elements employ a mixed interpolation of tensorial components (MITC) formulation [31, 32]. In addition, kinematic constraints are implemented within the same element hierarchy, including the lower kinematic pairs [29, 37].

B. Differential Algebraic Equations

DAEs present additional computational challenges compared with ODEs. In particular, DAEs are numerically stiff and require implicit time integration techniques to attain computationally practical integration strategies. While implicit integration methods necessitate the solution of nonlinear system of equations at each time step, this additional computational cost is offset by the improved stability properties and the resulting larger time steps enabled by implicit methods. This work uses a diagonally implicit Runge–Kutta method for time integration. Runge–Kutta methods belong to the class of multistage methods for solving differential equations, which employ intermediate time steps known as stages. Explicit Runge–Kutta methods are not well-suited for the stiff systems that arise in flexible multibody dynamic simulations. However, Butcher [38] extended Runge–Kutta methods to include a class of Implicit Runge–Kutta (IRK) methods that are well suited for stiff problems. As a further refinement, Alexander [39] and subsequently Cash [40] developed Diagonally Implicit Runge–Kutta (DIRK) methods that achieve computational savings compared to IRK methods. These computational savings arise from the lower triangular structure of the matrix of DIRK coefficients. Various DIRK methods have been developed by Alexander [39], including one-stage second-order, two-stage third-order, and three-stage fourth-order schemes. A study of the mathematical properties of DIRK methods is presented by Kennedy and Carpenter [41]. A detailed description of the DIRK method developed to solve the governing second-order systems is provided in Section III.

C. Functionals of Interest

In the context of the optimization of flexible multibody systems, it is necessary to define the objective and constraint functionals of interest. In this work, the adjoint equations are derived based on a discrete approximation of the integral functional

$$f(\ddot{\mathbf{q}},\dot{\mathbf{q}},\mathbf{q},\mathbf{x}) = \int_0^{t_f} F(\ddot{\mathbf{q}},\dot{\mathbf{q}},\mathbf{q},\mathbf{x},t)\, dt, \tag{3}$$

where the integrand F depends on the state variables and their time derivatives as well as the design variables. This functional form enables a straightforward derivation of the discrete adjoint equations; however, not all functionals fit this simple form. For instance, aggregation functionals, which are designed to provide a smooth estimate of the maximum or minimum quantity of interest over space and time, do not fit this integral functional form. In this work, we use the continuous Kreisselmeier–Steinhauser (KS) [27, 28] which is designed to estimate the maximum stress within the structure and can be evaluated as

$$f(\ddot{\mathbf{q}},\dot{\mathbf{q}},\mathbf{q},\mathbf{x}) = c_{ks} + \frac{1}{\rho_{ks}} \ln\left[\int_0^{t_f}\int_\Omega e^{\rho_{ks}(\sigma_{vm}-c_{ks})}\, d\Omega\, dt\right], \tag{4}$$

where ρ_{ks} is a parameter that controls the accuracy of the estimate, σ_{vm} is the von Mises stress, c_{ks} is a constant that can be used to avoid numerical issues [28], and Ω represents the spatial domain of interest. While the KS functional (4) does not take the same form as the functional (3), it is a function of an integral functional and the techniques described

below can be extended to this case as well.

III. Time Integration and Discrete Adjoint

This section presents the numerical time integration procedure for the governing equations using DIRK methods, and the development of the corresponding discrete adjoint equations used to evaluate the derivative of functionals of interest. The mathematical form of the governing equations (2) and the functional of interest (3) enable a concise derivation of adjoint equations for a broad class of problems.

A. Time Integration

In an implicit Runge–Kutta time integration scheme, the governing equations are integrated forward in time by solving a coupled system of implicit stage equations, followed by an explicit update that computes the state variables at the next time step [38]. A DIRK scheme [39–41] is distinguished from general IRK schemes based on the fact that each stage equation is coupled only to the previous stages, enabling a sequential solution procedure for each subsequent stage variable. This decoupling is due to the lower triangular structure of the DIRK coefficients for generic s-stage DIRK scheme shown in Table 1, referred to as the Butcher tableau [38]. In this work, we use one-, two-, and three-stage DIRK schemes that are second-, third-, and fourth-order accurate in time, respectively [39]. The coefficients for the two- and three-stage schemes are shown in the Butcher tableaus shown in Table 2. Other two- and three-stage DIRK schemes with lower accuracy can provide stronger numerical stability properties, but we have found that for the range of problems presented within this paper, these schemes are adequately stable.

Table 1: Butcher's tableau structure for diagonally implicit Runge–Kutta methods.

c_1	a_{11}	0	0	0
c_2	a_{21}	a_{22}	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	0
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

Table 2: Two and three-stage DIRK methods that are 3rd and 4th order accurate from Alexander [39]. For the three-stage DIRK method $\alpha = 2(\cos(\pi/18))/\sqrt{3}$.

$\frac{1}{2}+\frac{1}{2\sqrt{3}}$	$\frac{1}{2}+\frac{1}{2\sqrt{3}}$	0
$\frac{1}{2}-\frac{1}{2\sqrt{3}}$	$-\frac{1}{\sqrt{3}}$	$\frac{1}{2}+\frac{1}{2\sqrt{3}}$
	$\frac{1}{2}$	$\frac{1}{2}$

(a) Two-stage DIRK tableau

$(1+\alpha)/2$	$(1+\alpha)/2$	0	0
$1/2$	$-\alpha/2$	$(1+\alpha)/2$	0
$(1-\alpha)/2$	$1+\alpha$	$-(1+2\alpha)$	$(1+\alpha)/2$
	$1/(6\alpha^2)$	$1-1/(3\alpha^2)$	$1/(6\alpha^2)$

(b) Three-stage DIRK tableau

1. DIRK stage variables

In this work, a constant time interval, h, is used for each time step. The state variables and their first time derivatives are approximated at time $t_k = hk$, as $\mathbf{q}_k \approx \mathbf{q}(t_k)$, and $\dot{\mathbf{q}}_k \approx \dot{\mathbf{q}}(t_k)$. The initial state variable values at time $t = 0$, given by $\mathbf{q}_0$ and $\dot{\mathbf{q}}_0$, are obtained from the configuration and kinematics of the multibody system. The values of $\mathbf{q}_k$, and $\dot{\mathbf{q}}_k$ for $k = 1 \ldots N$ are obtained incrementally at each new time step as a function of the time derivatives of the state variable values at s intermediate stages. The state variable values and their time derivatives at intermediate stage i, at time $t_{ki} = t_k + c_i h$, are denoted $\mathbf{u}_{ki} \approx \mathbf{q}(t_{ki})$, $\dot{\mathbf{u}}_{ki} \approx \dot{\mathbf{q}}(t_{ki})$, and $\ddot{\mathbf{u}}_{ki} \approx \ddot{\mathbf{q}}(t_{ki})$, respectively. In order to solve the second-order governing equations, the second time derivatives of the state variables at each stage, $\ddot{\mathbf{u}}_{ki}$, are treated as the unknown variables. All remaining state variables and their time derivatives are expressed in terms of $\ddot{\mathbf{u}}_{ki}$, based on the coefficients in the Butcher tableau 1. At stage i, at time step k, the first time derivatives, $\dot{\mathbf{u}}_{ki}$, are obtained using the DIRK stage relationships

$$\dot{\mathbf{u}}_{ki} = \dot{\mathbf{q}}_{k-1} + h \sum_{j=1}^{i} a_{ij} \ddot{\mathbf{u}}_{kj}, \tag{5}$$

while the variables $\mathbf{u}_{ki}$, are obtained using

$$\begin{aligned} \mathbf{u}_{ki} &= \mathbf{q}_{k-1} + h \sum_{j=1}^{i} a_{ij} \dot{\mathbf{u}}_{kj}, \\ &= \mathbf{q}_{k-1} + h \sum_{j=1}^{i} a_{ij} \left(\dot{\mathbf{q}}_{k-1} + h \sum_{l=1}^{j} a_{jl} \ddot{\mathbf{u}}_{kl} \right), \\ &= \mathbf{q}_{k-1} + h c_i \dot{\mathbf{q}}_{k-1} + h^2 \sum_{j=1}^{i} A_{ij} \ddot{\mathbf{u}}_{kj}. \end{aligned} \tag{6}$$

Here we have applied the property of the DIRK coefficients, $\sum_{j=1}^{s} a_{ij} = c_i$, and have defined second-order DIRK coefficients such that

$$A_{ij} = \sum_{l=1}^{s} a_{il} a_{lj}.$$

Note that the second-order DIRK coefficients A_{ij} share the same lower-triangular structure as the first-order DIRK coefficients and that the diagonal entries are $A_{ii} = a_{ii}^2$.

Once all stage variables are computed, the variables, $\mathbf{q}_k$, and their first time derivatives, $\dot{\mathbf{q}}_k$, can be obtained. The values of the first time derivatives, $\dot{\mathbf{q}}_k$, can be obtained as follows

$$\dot{\mathbf{q}}_k = \dot{\mathbf{q}}_{k-1} + h \sum_{i=1}^{s} b_i \ddot{\mathbf{u}}_{ki}, \tag{7}$$

while the variables, $\dot{\mathbf{q}}_k$ can be found as

$$\begin{aligned}\mathbf{q}_k &= \mathbf{q}_{k-1} + h\sum_{i=1}^{s} b_i \dot{\mathbf{u}}_{ki}, \\ &= \mathbf{q}_{k-1} + h\sum_{i=1}^{s} b_i \left(\dot{\mathbf{q}}_{k-1} + \sum_{j=1}^{i} a_{ij} \ddot{\mathbf{u}}_{kj} \right), \\ &= \mathbf{q}_{k-1} + h\dot{\mathbf{q}}_{k-1} + h^2 \sum_{i=1}^{s} B_i \ddot{\mathbf{u}}_{ki}.\end{aligned} \tag{8}$$

Here we again use a property of the DIRK coefficients, $\sum_{i=1}^{s} b_i = 1$, and again define second-order DIRK coefficients as

$$B_i = \sum_{j=1}^{s} b_j a_{ji}.$$

Note that all Runge–Kutta schemes that achieve higher than second-order accuracy satisfy the condition $\sum_{i=1}^{s} B_i = 1/2$.

2. *DIRK stage equations*

The values of second derivatives of the state variables at each stage, $\ddot{\mathbf{u}}_{ki}$, are determined by solving the governing equations of motion (2). At stage i of time step k, the governing equations (2) take the form

$$\mathbf{R}_{ki} \triangleq \mathbf{R}(\ddot{\mathbf{u}}_{ki}, \dot{\mathbf{u}}_{ki}, \mathbf{u}_{ki}, \mathbf{x}, t_{ki}) = 0, \tag{9}$$

where the full dependence of the state variables can be written as $\dot{\mathbf{u}}_{ki} = \dot{\mathbf{u}}_{ki}(\dot{\mathbf{q}}_{k-1}, \ddot{\mathbf{u}}_{k1}, \ldots, \ddot{\mathbf{u}}_{ki})$, and the full depencence of the time derivative of the state variables can be written as $\mathbf{u}_{ki} = \mathbf{u}_{ki}(\mathbf{q}_{k-1}, \dot{\mathbf{q}}_{k-1}, \ddot{\mathbf{u}}_{k1}, \ldots, \ddot{\mathbf{u}}_{ki})$ via the approximations (5) and (6), respectively. As a result of the depencence of $\dot{\mathbf{u}}_{ki}$ and $\mathbf{u}_{ki}$ on $\ddot{\mathbf{u}}_{kj}$, there is ambiguity about the notation for the partial derivative terms. Here, we use $\partial \mathbf{R}_{ki}/\partial \ddot{\mathbf{u}}_{ki}$ to denote the partial derivative of the stage equation residual (9), including any implicit dependence on $\ddot{\mathbf{u}}_{ki}$. Therefore, the matrix $\partial \mathbf{R}_{ki}/\partial \ddot{\mathbf{u}}_{ki}$ is a Jacobian matrix formed from a linear combination of the derivative of the governing equation (2) with respect to the state variables and their derivatives, $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$. To distinguish the components of this linear combination, we use the notation $\partial \mathbf{R}_{ki}/\partial \ddot{\mathbf{q}}$, with the subscript ki, to denote the derivative of the governing equations with respect to $\ddot{\mathbf{q}}$, i.e. $\partial \mathbf{R}/\partial \ddot{\mathbf{q}}$, evaluated at the state variables $\ddot{\mathbf{u}}_{ki}$, $\dot{\mathbf{u}}_{ki}$, and $\mathbf{u}_{ki}$. We also use analogous definitions for derivatives with respect to $\dot{\mathbf{q}}$, or $\mathbf{q}$. As a result, the Jacobian of the governing equations can be expressed as follows

$$\frac{\partial \mathbf{R}_{ki}}{\partial \ddot{\mathbf{u}}_{ki}} = \frac{\partial \mathbf{R}_{ki}}{\partial \ddot{\mathbf{q}}} + h a_{ii} \frac{\partial \mathbf{R}_{ki}}{\partial \dot{\mathbf{q}}} + h^2 A_{ii} \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{q}}.$$

An analogous notation is also employed for the integrand of the functional of interest (3).

The nonlinear system of equations at each stage (9) is solved using a Newton–Raphson method. At each Newton–

Raphson iteration, indexed by n, linearizing equation (9) with respect to $\ddot{\mathbf{u}}_{ki}$ results in the following linear system

$$\left[\frac{\partial \mathbf{R}_{ki}^{n}}{\partial \ddot{\mathbf{u}}_{ki}}\right] \Delta\ddot{\mathbf{u}}_{ki}^{n} = \left[\frac{\partial \mathbf{R}_{ki}^{n}}{\partial \ddot{\mathbf{q}}} + h a_{ii} \frac{\partial \mathbf{R}_{ki}^{n}}{\partial \dot{\mathbf{q}}} + h^2 A_{ii} \frac{\partial \mathbf{R}_{ki}^{n}}{\partial \mathbf{q}}\right] \Delta\ddot{\mathbf{u}}_{ki}^{n} = -\mathbf{R}_{ki}^{n}. \tag{10}$$

As an initial guess for the Newton–Raphson method, we set $\ddot{\mathbf{u}}_{ki}^{0} = 0$, and compute initial values for $\dot{\mathbf{u}}_{ki}^{0}$, and $\mathbf{u}_{ki}^{0}$, using the stage equations (5) and (6). The residual and Jacobian matrix are recomputed and the Jacobian matrix is completely refactored at each iteration. The linear system (10) is solved in parallel using a direct matrix factorization method described in Kennedy and Martins [4]. Once the solution, $\Delta\mathbf{u}_{ki}^{n}$ is obtained, the stage variables and their time derivatives are updated as follows:

$$\begin{aligned}
\ddot{\mathbf{u}}_{ki}^{n+1} &= \ddot{\mathbf{u}}_{ki}^{n} + \Delta\ddot{\mathbf{u}}_{ki}^{n}, \\
\dot{\mathbf{u}}_{ki}^{n+1} &= \dot{\mathbf{u}}_{ki}^{n} + h a_{ii} \Delta\ddot{\mathbf{u}}_{ki}^{n}, \\
\mathbf{u}_{ki}^{n+1} &= \mathbf{u}_{ki}^{n} + h^2 A_{ii} \Delta\ddot{\mathbf{u}}_{ki}^{n}.
\end{aligned} \tag{11}$$

The iterative updates are continued until the residual norm of the governing equations falls below either a relative tolerance, ε_r, such that $\|\mathbf{R}_{ki}^{n}\|_2 \leq \varepsilon_r \|\mathbf{R}_{ki}^{0}\|_2$ or an absolute tolerance, ε_a, such that $\|\mathbf{R}_{ki}^{n}\|_2 \leq \varepsilon_a$. In this work, we have used values of $\varepsilon_r = 10^{-9}$ and $\varepsilon_a = 10^{-12}$, respectively. After the forward problem has been solved, the functionals of interest can be evaluated using an integration scheme consistent with DIRK

$$f(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}, \mathbf{x}) \approx \sum_{k=1}^{N} h \sum_{i=1}^{s} b_i F(\ddot{\mathbf{u}}_{ki}, \dot{\mathbf{u}}_{ki}, \mathbf{u}_{ki}, \mathbf{x}, t_{ki}),$$

where b_i are the DIRK integration coefficients.

B. Time-Dependent Discrete Adjoint Method

The process of obtaining the derivative of a functional of interest with respect to the design variables requires two main steps: (1) the computation of the adjoint variables by solving the adjoint equations, and (2) the evaluation of the total derivative based on the adjoint variables. The form of the adjoint equations derived here is similar to the DIRK integration scheme: s stage-adjoint equations are solved followed by an explicit update to obtain the new adjoint variables. Like all time-dependent adjoint methods, the DIRK adjoint equations march backwards in time.

The adjoint equations and the total derivative are derived in this section based on a discrete approximation of a continuous Lagrangian. This approximate Lagrangian consists of a combination of the functional of interest, the inner product of the adjoint and the residual of the DIRK residuals at each stage, as well as the inner product of the state update equations with associated adjoint vectors. To form the discrete Lagrangian, the state update equations (7)

and (8) are first reformulated in residual form

$$\begin{aligned}\mathbf{S}_k &\triangleq \dot{\mathbf{q}}_{k-1} + h\sum_{i=1}^{s} b_i \ddot{\mathbf{u}}_{ki} - \dot{\mathbf{q}}_k = 0,\\ \mathbf{T}_k &\triangleq \mathbf{q}_{k-1} + h\dot{\mathbf{q}}_{k-1} + h^2\sum_{i=1}^{s} B_i \ddot{\mathbf{u}}_{ki} - \mathbf{q}_k = 0.\end{aligned} \tag{12}$$

Next, the adjoint variables, λ_{ki}, ψ_k and ϕ_k, are introduced, which are associated with the stage residuals, $\mathbf{R}_{ki}$, and the state update equations, $\mathbf{S}_k$, and $\mathbf{T}_k$, respectively, for each time step, k. With these definitions, the Lagrangian for the adjoint equations is

$$\mathsf{L} = \sum_{k=1}^{N} h\sum_{i=1}^{s} b_i(F_{ki} + \lambda_{ki}^T \mathbf{R}_{ki}) + \sum_{k=1}^{N} \psi_k^T \mathbf{S}_k + \sum_{k=1}^{N} \phi_k^T \mathbf{T}_k. \tag{13}$$

Note that the first term in the Lagrangian is a discrete approximation of the sum of the integral functional and the product of the adjoint variables with the governing equations.

The adjoint equations are obtained by finding the stationary point of the Lagrangian (13) with respect to (1) the second time derivatives of the state variables at each stage, $\ddot{\mathbf{u}}_{ki}$, and (2) the state variables and their first time derivatives at each step, $\mathbf{q}_k$, and $\dot{\mathbf{q}}_k$, respectively. The equation for the adjoint variables, ϕ_k, associated with the first stage equation (5), is obtained by imposing the condition $\partial\mathsf{L}/\partial\mathbf{q}_{k-1} = 0$, yielding the following equation

$$\frac{\partial \mathsf{L}}{\partial \mathbf{q}_{k-1}} = \frac{\partial \mathbf{T}_k}{\partial \mathbf{q}_{k-1}}^T \phi_k + \frac{\partial \mathbf{T}_{k-1}}{\partial \mathbf{q}_{k-1}}^T \phi_{k-1} + h\sum_{i=1}^{s} b_i \left(\frac{\partial F_{ki}}{\partial \mathbf{q}_{k-1}}^T + \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{q}_{k-1}}^T \lambda_{ki} \right) = 0,$$

which reduces to the following explicit update after the evaluation of partial derivatives and the application of chain rule of differentiation

$$\phi_{k-1} = \phi_k + h\sum_{i=1}^{s} b_i \left(\frac{\partial F_{ki}}{\partial \mathbf{q}}^T + \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{q}}^T \lambda_{ki} \right). \tag{14}$$

In an analogous manner, the equation for the adjoint variables, ψ_k, associated with the second stage equation (12) is obtained by setting $\partial\mathsf{L}/\partial\dot{\mathbf{q}}_{k-1} = 0$. This yields the following equation

$$\frac{\partial \mathsf{L}}{\partial \dot{\mathbf{q}}_{k-1}} = \frac{\partial \mathbf{S}_{k-1}}{\partial \dot{\mathbf{q}}_{k-1}}^T \psi_{k-1} + \frac{\partial \mathbf{S}_k}{\partial \dot{\mathbf{q}}_{k-1}}^T \psi_k + \frac{\partial \mathbf{T}_k}{\partial \dot{\mathbf{q}}_{k-1}}^T \phi_{k+1} + h\sum_{i=1}^{s} b_i \left(\frac{\partial F_{ki}}{\partial \dot{\mathbf{q}}_{k-1}}^T + \frac{\partial \mathbf{R}_{ki}}{\partial \dot{\mathbf{q}}_{k-1}}^T \lambda_{ki} \right) = 0.$$

This equation can be simplified to obtain an update for the adjoint variable ψ_k:

$$\psi_{k-1} = \psi_k + h\phi_k + h\sum_{i=1}^{s} b_i \left(\frac{\partial F_{ki}}{\partial \dot{\mathbf{q}}}^T + \frac{\partial \mathbf{R}_{ki}}{\partial \dot{\mathbf{q}}}^T \lambda_{ki} \right) + h^2 \sum_{i=1}^{s} b_i c_i \left(\frac{\partial F_{ki}}{\partial \mathbf{q}}^T + \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{q}}^T \lambda_{ki} \right). \tag{15}$$

At the last time step $k = N$, $\psi_N = \phi_N = 0$. Note that the adjoint equations (14) and (15) are explicit relations similar in form to the forward update equations (7) and (8).

Finally, the equation for the adjoint variables, λ_{ki}, associated with the residual at each stage can be found by setting

$\partial \mathsf{L}/\partial \ddot{\mathbf{u}}_{ki} = 0$. This gives the following relationship

$$\frac{\partial \mathsf{L}}{\partial \ddot{\mathbf{u}}_{ki}} = h \sum_{j=i}^{s} b_j \left(\frac{\partial F_{kj}}{\partial \ddot{\mathbf{u}}_{ki}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \ddot{\mathbf{u}}_{ki}}^T \lambda_{kj} \right) + \frac{\partial \mathbf{S}_k}{\partial \ddot{\mathbf{u}}_{ki}}^T \psi_k + \frac{\partial \mathbf{T}_k}{\partial \ddot{\mathbf{u}}_{ki}}^T \phi_k = 0.$$

Expanding the partial derivatives gives the following relationship

$$h \sum_{j=i}^{s} b_j \left[\left(\frac{\partial F_{kj}}{\partial \ddot{\mathbf{q}}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \ddot{\mathbf{q}}}^T \lambda_{kj} \right) + h a_{ji} \left(\frac{\partial F_{kj}}{\partial \dot{\mathbf{q}}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \dot{\mathbf{q}}}^T \lambda_{kj} \right) + h^2 A_{ji} \left(\frac{\partial F_{kj}}{\partial \mathbf{q}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \mathbf{q}}^T \lambda_{kj} \right) \right] + h b_i \psi_k + h^2 B_i \phi_k = 0.$$

Finally, dividing by hb_i and rearranging for the unknown adjoint variable λ_{ki} results in a linear system of the form:

$$\left[\frac{\partial \mathbf{R}_{ki}}{\partial \ddot{\mathbf{q}}} + h a_{ii} \frac{\partial \mathbf{R}_{ki}}{\partial \dot{\mathbf{q}}} + h^2 A_{ii} \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{q}} \right]^T \lambda_{ki} = -\mathbf{b}_{ki}, \tag{16}$$

where the right hand side is

$$\begin{aligned} \mathbf{b}_{ki} = & \left(\frac{\partial F_{ki}}{\partial \ddot{\mathbf{q}}}^T + h a_{ii} \frac{\partial F_{ki}}{\partial \dot{\mathbf{q}}}^T + h^2 A_{ii} \frac{\partial F_{ki}}{\partial \mathbf{q}}^T \right) \\ & + h \sum_{j=i+1}^{s} \frac{b_j a_{ji}}{b_i} \left(\frac{\partial F_{kj}}{\partial \dot{\mathbf{q}}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \dot{\mathbf{q}}}^T \lambda_{kj} \right) + h^2 \sum_{j=i+1}^{s} \frac{b_j A_{ji}}{b_i} \left(\frac{\partial F_{kj}}{\partial \mathbf{q}}^T + \frac{\partial \mathbf{R}_{kj}}{\partial \mathbf{q}}^T \lambda_{kj} \right) \\ & + \psi_k + h \frac{B_i}{b_i} \phi_k. \end{aligned} \tag{17}$$

There are three groups of terms that contribute to the right-hand-side (17) of the stage adjoint equation (16). The first group of terms represent the contributions from the derivative of the functional with respect to the state variables and their time derivatives. The second group of terms provide contributions from the future stage adjoint variables at the same time step. The last group of terms carry forward contributions from one time step to the next.

Once the adjoint variables λ_{ki} have been determined, the total derivative can be computed as follows:

$$\frac{df}{d\mathbf{x}} \triangleq \frac{d\mathsf{L}}{d\mathbf{x}} = \sum_{k=1}^{N} h \sum_{i=1}^{s} b_i \left(\frac{\partial F_{ki}}{\partial \mathbf{x}} + \lambda_{ki}^T \frac{\partial \mathbf{R}_{ki}}{\partial \mathbf{x}} \right). \tag{18}$$

Note that only the adjoint variables for the stage equations, λ_{ki}, contribute to the total derivative computation.

IV. Implementation of the Adjoint Method

General flexible multibody dynamics simulation tools contain a large library of flexible and rigid elements, joints, dampers, and a wide variety of kinematic constraints that can be used to model multibody systems. The implementation of the discrete adjoint imposes additional requirements on each component of the simulation. These additional requirements must be handled carefully in order to maintain an efficient and accurate adjoint implementation. This section presents the organization and implementation of the proposed adjoint sensitivities, that is designed to be mod-

ular and extensible to facilitate an expanding library of flexible and rigid elements in TACS [4]. The adjoint equations presented in section III contain:

1. The derivatives of the *functional* of interest and the *governing equations* with respect to the *state variables*,
2. The derivatives of the *functional* of interest and the *governing equations* with respect to the *design variables*, and
3. The products of the *adjoint variables* with the derivatives the *governing equations* respect to the *state variables*.

These three primary terms are implemented using a library that contains four interfaces: `Element`, `Function`, `Assembler` and `Integrator`. The organization and relationships between these four interfaces are shown in Figure 1. This organization allows for the separation of functionality that enables the underlying element and function library to be extended without having to change the adjoint implementation. The functionality of these interfaces are explained in the remainder of this section.

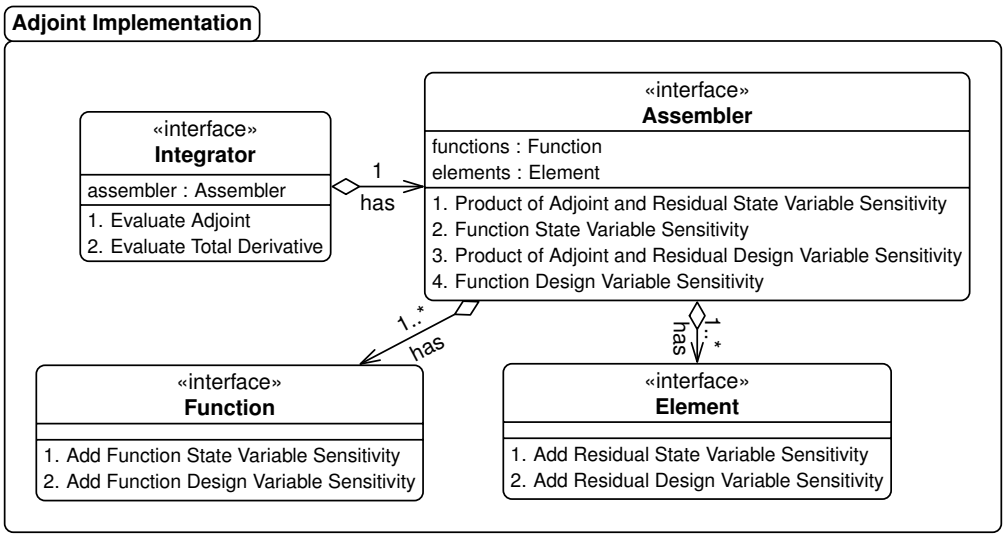


Figure 1: Class diagram illustrating the architecture of adjoint-based gradient implementation.

A. Element Interface

The element library contains beam, shell, and rigid-body elements as well as kinematic constraints including the lower kinematic pairs. These elements implement a common `Element` interface by providing specific implementations for the abstract prototypes based on the governing equations of motion. This interface contains two routines required for the adjoint implementation:

1. The computation of element-wise Jacobian matrices that are used by the `Assembler` to evaluate the global transpose Jacobian in the linear adjoint system.

2. The evaluation of the derivative of element-wise product of the residuals and the adjoint variables with respect to the design variables. This routine is used to evaluate the total derivative.

As new elements are added to the multibody dynamics library, they are required to implement these two routines so that they can be seamlessly merged in the existing framework.

B. Function Interface

The `Function` interface contains similar prototypes as the `Element` interface. The interface provides the derivatives of functionals of interests for design optimization which include two primary function-level routines:

1. The evaluation of the element-wise derivative of the functional integrand with respect to the state variables and their time derivatives.

2. The element-wise computation of the derivative of the functional integrand with respect to the design variables, required for the total derivative.

The functionals are evaluated over all or a subset of elements in the domain. For example, when evaluating the the KS functional to estimate the maximum stress (4), the rigid elements in the simulation model can be omitted.

C. Assembler Interface

The `Assembler` interface is designed to operate on a collection of `Element` and `Function` instances. The routines provided in this interface assemble the partial derivative terms necessary for sensitivity analysis and place them in global matrices and vectors. These operations depend only on the prototypes defined in `Element` and `Function` interfaces, rather than on the specific implementations of element and function types. This dependency of the `Assembler` on `Element` and `Function` interfaces is shown in Figure 1.

1. Solving the Adjoint Equations

The first set of `Assembler` routines are required for the solution of the adjoint equations (14), (15), and (16). These functions compute the transpose Jacobian-vector product of the governing equations with respect to the state variables

$$\chi \leftarrow \chi + \left[\gamma\frac{\partial \mathbf{R}}{\partial \ddot{\mathbf{q}}} + \beta\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{q}}} + \alpha\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^T \chi, \tag{19}$$

and the derivative of the functional integrand with respect to state variables

$$\chi \leftarrow \chi + \left[\gamma\frac{\partial F}{\partial \ddot{\mathbf{q}}} + \beta\frac{\partial F}{\partial \dot{\mathbf{q}}} + \alpha\frac{\partial F}{\partial \mathbf{q}}\right]. \tag{20}$$

Here χ is a place-holder for a state vector determined from the context of the adjoint equations. The inputs to these routines are scalar constants for each partial derivative (α, β, and γ) and the state variables, $\mathbf{q}$, and their time deriva-

tives, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$. The state variables and their time derivatives are stored to disk during the solution phase and reloaded when marching backwards in time during the adjoint solution process. This reduces the amount of memory required when the number of time steps is large. The routines (19) and (20) are used frequently in the discrete adjoint implementation of DIRK. The number of invocations of each routine for an s-stage DIRK adjoint implementation is listed in Table 3. Both the number of calls at each stage i and the total number of calls per time step are tabulated.

Table 3: Number of transpose Jacobian-vector products and functional integrand derivative computations required to form the right-hand-sides in the corresponding adjoint equations.

	$[\partial \mathbf{R}/\partial \mathbf{q}]^T \chi$	$\partial F/\partial \mathbf{q}$
Equation (14) for ϕ_k	s	s
Equation (15) for ψ_k	2s	2s
Equation (16) for λ_{ki}	2(s-i)+1	2(s-i)+1
Total per time step	4s + s(s+1)/2	4s + s(s+1)/2

2. Evaluating the Total Derivative

The second set of assembly-level routines are needed to evaluate the total derivative of the functional of interest based on equation (18). These routines compute the product of the adjoint variables with the derivative of the governing equations with respect to design variables

$$\chi \leftarrow \chi + \alpha \frac{\partial \mathbf{R}}{\partial \mathbf{x}}^T \chi, \tag{21}$$

and the derivative of the functional integrand with respect to design variables

$$\chi \leftarrow \chi + \alpha \frac{\partial F}{\partial \mathbf{x}}^T. \tag{22}$$

Here the inputs consist of a scalar α, the design variables $\mathbf{x}$, and the state variables, $\mathbf{q}$, and their time derivatives, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$. The output for both of these routines is a vector with the same dimension as the design variable vector. The routines (21) and (22) are used once at each stage to accumulate the contributions to the total derivative (18).

D. Integrator Interface

The class implementing the `Integrator` interface completes the evaluation the adjoint variables and the computation of the total derivative and provides it to the optimizer. The `Integrator` interface contains an instance of `Assembler`, which enables it to evaluate the partial derivative terms from the governing equations and the functionals of interest, and scale them with the appropriate coefficients, as dictated by the adjoint equations. Note that the `Integrator` does not interact directly with `Element` and `Function` interfaces, but instead uses the `Assembler` interface, as shown in Figure 1. This class contains routines that implement DIRK specific operations and is used repeatedly in a time loop starting from the final time step and ending at the initial time step. The `Assembler` set of routines used by

the `Integrator` are designed to work for any adjoint method corresponding to other time marching schemes such as Backwards Difference Formula, Adams-Bashforth-Moulton, or Newmark's method. The implementation of other time-integration methods requires only a new implementation of the Integrator interface. The authors have used utilized these proposed routines extensively for other discrete adjoint implementations [42].

V. Verification and Scalability Studies

This section presents a series of studies that verify the implementation of the governing equations and evaluates the scalability of the solution and adjoint-based derivative evaluation techniques.

A. Dynamics Verification

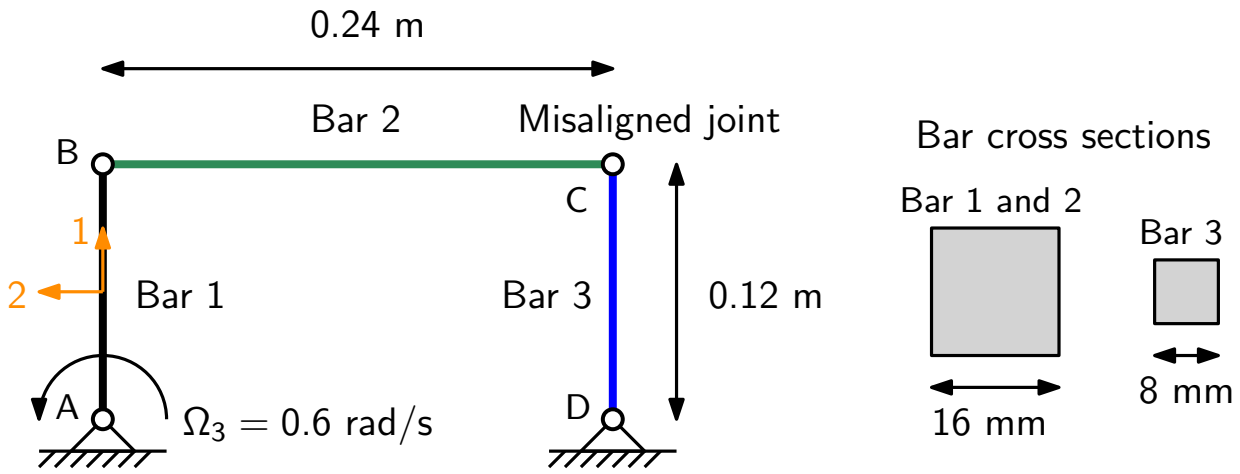


Figure 2: The four-bar mechanism problem used for dynamics verification of TACS.

A partial verification of the governing equations implemented within the framework is performed against the four-bar mechanism problem reported by Bauchau et al. [43]. Figure 2 illustrates the setup of the problem. Three bars *AB*, *BC* and *CD* of the mechanism are joined together using revolute connections. An imaginary fourth bar exists in the mechanism between the points *A* and *D*. The revolute joints at the points *A*, *B*, and *D*, have an axis of rotation that is perpendicular to the plane of the mechanism. The revolute joint at point *C* is misaligned by an angle of 5°, rotated about the direction of the bar *CD*. Bars *AB* and *BC* are of the same cross-section, while bar *CD* has a smaller, more flexible cross section. The bars in the mechanism are modeled using quadratic beam elements that are derived from Timoshenko beam theory. The beam element is implemented using a geometrically exact formulation that captures full rigid rotations and translations, and provides discretely objective strains. Shear locking is avoided through the use of a mixed interpolation of tensorial components (MITC) treatment of the transverse shear strains. The rotation of bar *AB* about point *A* of the mechanism is driven at an angular rate of $\Omega_3 = 0.6\,\text{rad/s}$.

When the bars are rigid, the four-bar mechanism locks and motion is inhibited. When the bars are modeled as flexible, motion becomes possible since the bars can bend to overcome the locking behavior. The motion of the four

bar mechanism is illustrated in Figure 3 as a time lapse. If joint C were not misaligned, the bars would rotate in phase. However, due to the misalignment, the third bar never completes a full rotation, while the first bar drives the motion.

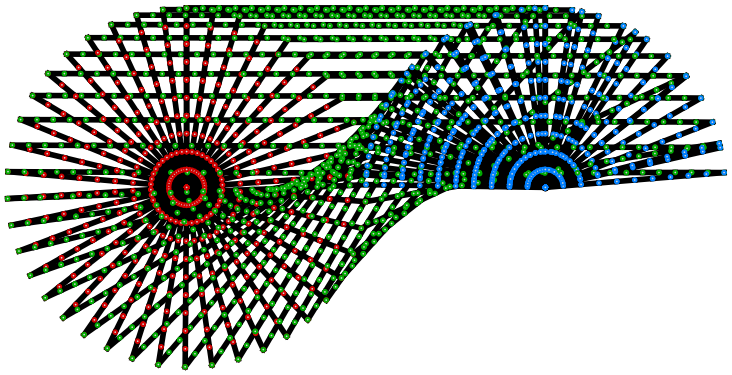


Figure 3: The time evolution of flexible four-bar mechanism simulated using TACS.

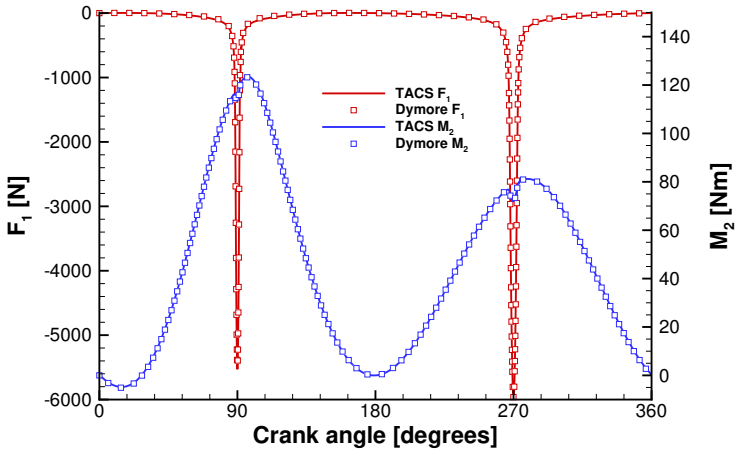


Figure 4: Comparison of TACS and Dymore [43] predictions of force and bending moment in bar AB at mid-span.

Figure 4 shows a comparison of the axial force and bending moment in bar AB at the mid-span compared with the same predictions using Dymore [43]. The force and moment predictions can be seen to be in excellent agreement for this benchmark problem.

B. Parallel Scalability Assessment

This section presents a study of the parallel scalability of the forward time integration, and the adjoint solution and total derivative evaluation methods described in Section III. Two models of a plate pendulum are used in this study: a medium model with 192,000 degrees of freedom, and a larger model with 2 million degrees of freedom. The pendulum consists of a $25 \times 5 \times 5$ cm beam, modeled using shell elements of 1 cm thickness. The material properties are aluminum with $E = 70$ GPa, $\nu = 0.3$, and $\rho = 2500\,\mathrm{kg/m^3}$. The functional evaluated for this simulation is the KS aggregation of von Mises failure ratio in space-time domain and the design variables are chosen to be the the shell thicknesses. The time evolution of the configuration of flexible pendulum is carried out using the two-stage third-

order DIRK method. During the time integration, the nonlinear system is solved using the Newton method described in Section A with relative and absolute tolerances of $\varepsilon_r = 10^{-9}$ and $\varepsilon_a = 10^{-12}$, respectively. The linear system arising from each Newton iteration is solved using a direct factorization of the Jacobian. After the forward simulation is complete, the time history is used to evaluate the KS functional and the adjoint method is used to evaluate its gradient. The computational time taken for each main component of these operations is recorded and forms the basis of this study.

The study consists of three parts: first, a high-level comparison of the overall performance of the forward analysis and adjoint-method. Second, a detailed low-level comparison of the computational time spent in the individual operations in the forward analysis and adjoint, respectively. And finally, an evaluation of the parallel scalability for cases involving additional functionals for fixed problem size. These studies were performed on a cluster with 2.50GHz Intel Xeon CPU E5-2680-v3 compute nodes. Each node has 24 processor cores with a total of 128 GB of RAM per node. The number of processor cores used to solve the medium sized problem are 1, 2, 4, 8, 12, 16, 20, and 24. The number of processor cores used to solve the larger problem are 12, 24, 48, and 72. The element-based domain decomposition is assigned based on the graph partitioning algorithms in METIS-5.0 [44], where the elements are assigned to both balance the number of elements per process while minimizing the size of the interface between domains. The number of iterations taken to solve the nonlinear problem does not vary with the number of processors used to solve the problem.

1. *High-level Operations*

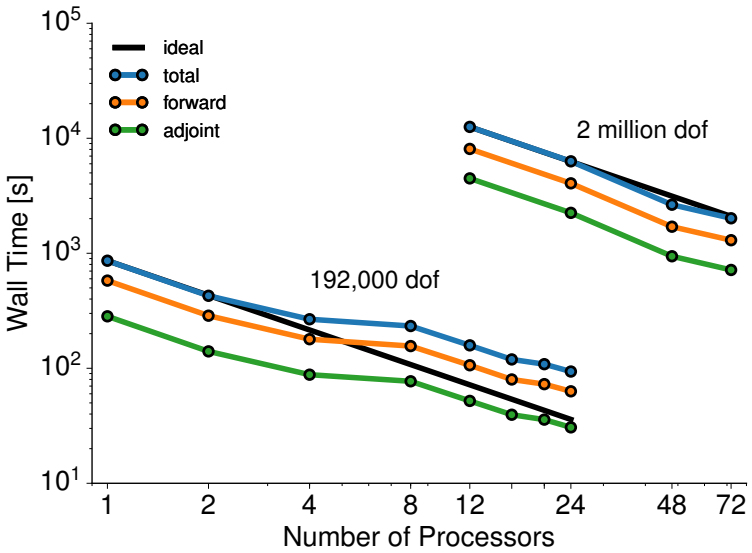


Figure 5: Parallel scalability evaluation of high-level operations such as the forward and adjoint modes.

The high-level operations consist of solving the forward problem for the state variables, computing the adjoint variables and evaluating the total derivative. The total time, consisting of the sum of the forward and adjoint times,

reflects the time required for a typical optimization iteration. The parallel scalability of the flexible pendulum plate problem is shown in Figure 5 for the medium and large problem sizes. Overall, good scalability is achieved. For the medium problem, the scalability between 4 to 8 processors suffers due to memory bandwidth limitations, since all cores are allocated to the same node.

2. *Forward Mode Operations*

Next, the computational time for the forward solution procedure is broken down into specific *Assembler*- and *Integrator*-level operations, to assess their overall contribution to the computational cost. The operations considered for this study are the ones directly involved in solving the nonlinear system (9) at each stage of a time step:

1. the *assembly operations* for evaluating the Jacobian matrix and right-hand-side (10),
2. the *factorization* of Jacobian matrix,
3. the *application of factorization* to solve for incremental state variable updates.

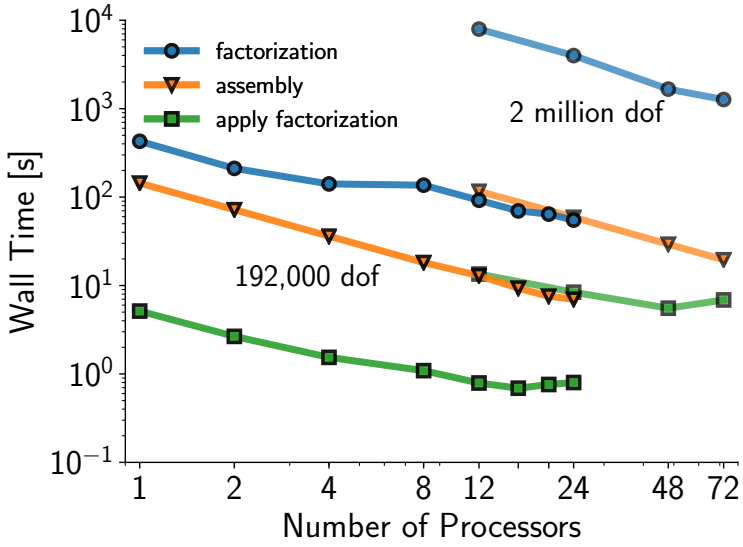


Figure 6: Parallel scalability evaluation of specific forward mode operations on medium and large problems.

Figure 6 shows the parallel scalability in terms of the total time taken for operations involved in the forward solution for two problem sizes considered. The matrix and residual assembly operations scale ideally for both cases. The matrix factorization and the application of matrix factorization scale well. The scalability of the application of the factorization stagnates noticeably for more processors. This bottleneck did not penalize the total performance as the order of magnitude of this operation is the least among those considered. On the other hand, matrix factorization is the most computationally expensive operation and its scalability is directly reflected in the overall time.

3. *Adjoint Mode Operations*

The scalability of adjoint mode operations are considered next for a single functional of interest. The adjoint-based gradient computation involves the solution of a linear system at each stage for all time steps to determine the adjoint variables for each functional of interest. The adjoint mode operations considered are:

1. the *factorization* of the transposed Jacobian matrices (16) in the adjoint linear system,
2. the *assembly operations* for setting up the transposed Jacobian matrices and accumulating contributions to the right-hand-side of the adjoint linear system (16),
3. the *application of factorizations* to solve for the adjoint variables for each functional of interest,
4. the *Jacobian-vector products* involved in equation (18) for the evaluation of the total derivative.

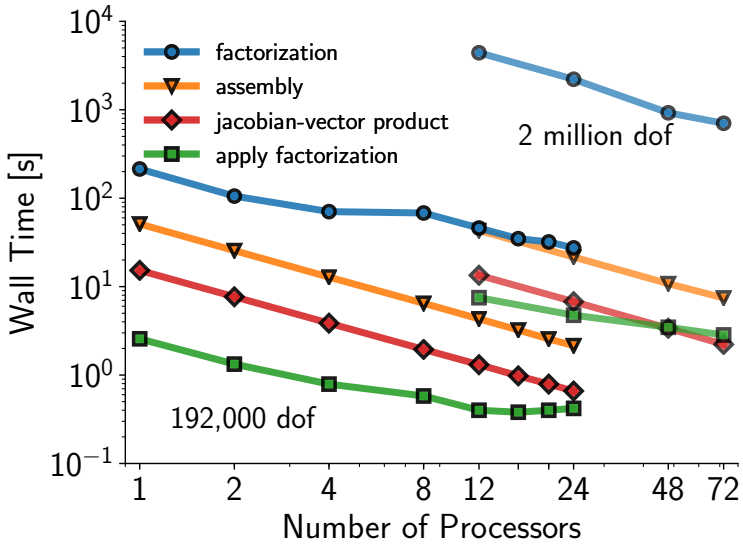


Figure 7: Parallel scalability evaluation of specific adjoint mode operations on medium and large problems.

The parallel scalability of these operations are shown in Figure 7. Trends similar to the forward mode are apparent for factorization and assembly operations. The additional Jacobian-vector product operations scale ideally. This computation consists of computing the element contributions to the Jacobian matrix and multiplying by the input vector without assembling a global matrix in memory. Note that the Jacobian-vector products are more computationally demanding than applying factorizations on the adjoint right hand sides.

4. Scaling with Number of Functionals

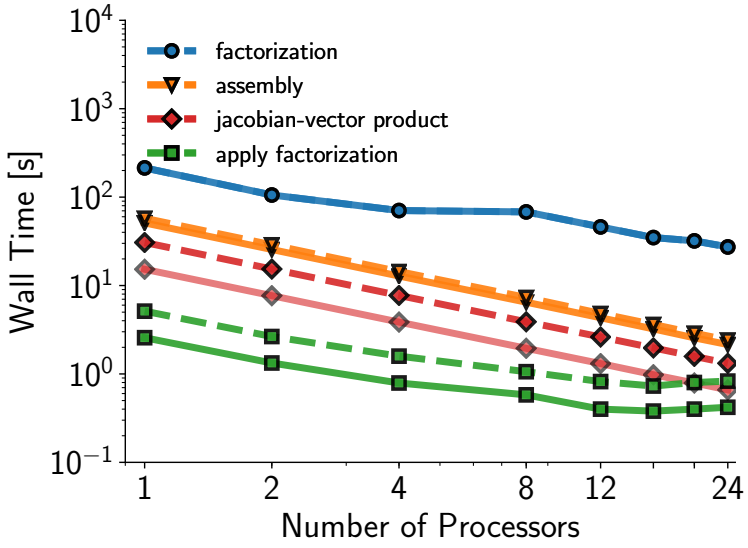


Figure 8: Parallel scalability evaluation for increasing number of functionals of interest for problem with 192,000 degrees of freedom. The dashed lines correspond to the two-functional case whereas the solid lines correspond to the one-functional case.

The effect of increasing the number of functionals on the computational cost of the adjoint mode operations is considered next. This study is performed on the medium model with $192,000$ degrees of freedom, with two functionals of interest. Figure 8 shows the additional costs of the adjoint mode operations in dashed-lines. The lines are shifted upwards without any perceivable change in the scalability trends shown before for the single functional case. Note that the application of matrix factorization and Jacobian-vector products see proportional increases in computational time. The factorization time did not increase as the same matrix factorization of the transposed Jacobian is reused for the additional functional. Overall, the total adjoint solution time increases by less than 5% for the additional functional.

In summary, the computational procedures fundamental to the presented flexible multibody dynamics design framework scale well. The largest single contribution to the computational cost for both the forward and adjoint solutions is the matrix factorization.

VI. Rotorcraft Demonstration and Optimization

The intent of this section is to demonstrate the suitability of the proposed flexible multibody dynamics framework for analysis and design of rotorcraft models that combine rigid and flexible components as well as kinematic constraints. For this purpose, a representative rotor hub configuration is considered for study. Typical rotorcraft hub assemblies fall under teetering, fully-articulated, hingeless and bearingless categories. These types differ in the mechanism used to achieve desired flight modes, such as hover or forward flight, and maneuvers, such as roll, pitch and yaw. To achieve these desired flight modes, the control mechanism must impart collective and cyclic control inputs

to the blades through the swashplate driven by the push rods, as illustrated in Figure 9. The hub and control chain dynamics are a central part of the rotorcraft flight control system and must be accurately modeled to achieve good performance prediction.

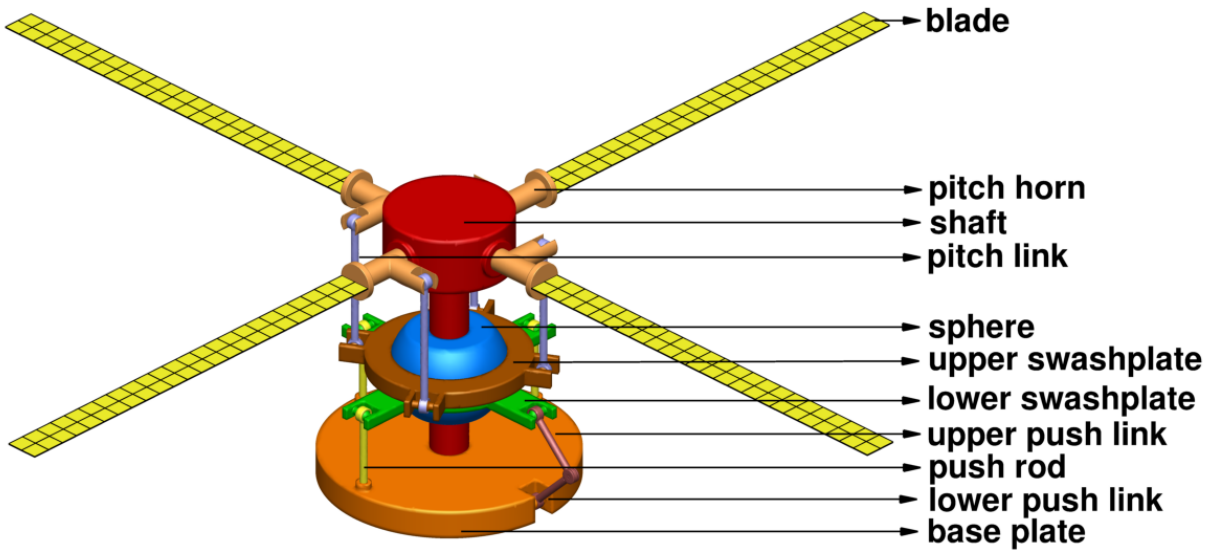


Figure 9: The baseline structural model of rotorcraft hub assembly with its parts labeled.

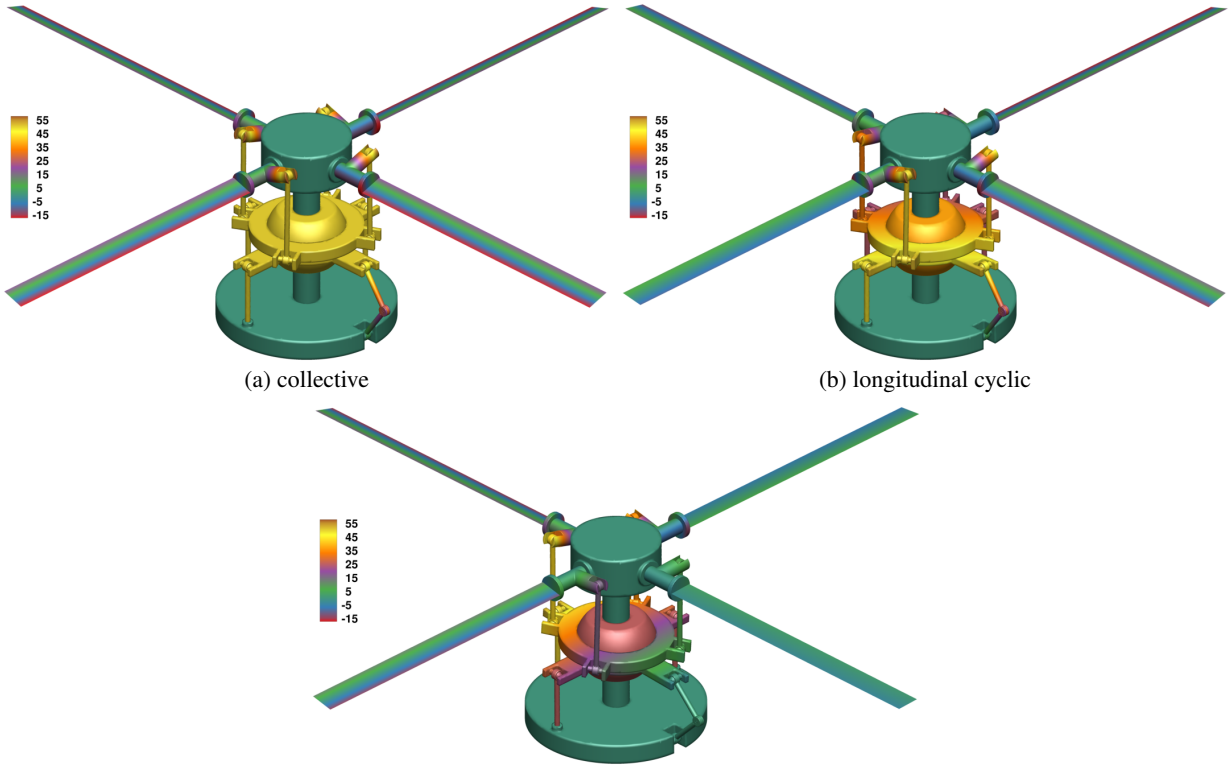
The control chain used for changing the pitch of rotor blades via appropriate inputs to the swashplate is investigated as the motion of interest. The representative four-bladed rotor hub assembly model used for this application is shown in Figure 9. The model consists of rigid bodies, kinematic constraints, flexible bodies and actuators. The four rotor blades are modeled as flexible using shell elements whereas the remaining parts are modeled as rigid bodies. The kinematic constraints and actuators used in the rotor assembly are listed in Table 4.

Table 4: List of constraint types and motion actuators associated with different bodies in hub assembly model.

Constraint/Actuator	**Part 1**	**Part 2**
Rotational actuator	shaft	–
Translational actuator	push rod	–
Spherical constraint	lower swashplate	sphere
Spherical constraint	upper swashplate	pitch link
Spherical constraint	pitch link	pitch horn
Spherical constraint	lower swashplate	pitch rod
Spherical constraint	lower swashplate	upper push link
Revolute constraint	lower swashplate	upper swashplate
Revolute constraint	shaft	pitch horn
Revolute constraint	baseplate	lower push link
Revolute constraint	lower push link	upper push link
Cylindrical constraint	sphere	shaft
Fixed constraint	baseplate	–

The push rods are connected to translational actuators that feed time dependent periodic motions, given by $u(t) =$

$u_0 \sin(\Omega_t t)$, where Ω_t is the assumed translational control signal frequency. This driven motion will be used as the basis for the study of different rotorcraft simulation scenarios in the examination of the rotor hub dynamics. The central shaft is connected to a rotational driver with a angular rate of $\Omega_r = 109.12$ rad/s. This structural model contains a total of $28,640$ degrees of freedom. The geometric modeling and meshing parametrization of rotor hub parts is performed using the open-source program GMSH [45]. The inertial properties are obtained directly from the geometry of each part.



(a) collective

(b) longitudinal cyclic

(c) lateral cyclic

Figure 10: Contour plots showing the vertical displacement of bodies during the motion in millimeters.

The rotor hub apparatus is studied for (a) collective (b) longitudinal cyclic and (c) lateral cyclic pitch control imparted through the three push rods at $90°, 180°$ and $270°$ from a horizontal reference axis. These conditions are summarized in Table 5. The corresponding time evolution of the configuration of the rotor assembly is simulated using two-stage, third-order diagonally implicit Runge–Kutta method detailed in Section III, employing a time step size corresponding to $1°$ per step at the angular rate 109.12 rad/s.

Table 5: Sinusoidally modulated control amplitudes supplied to the push rods to produce different flight scenarios.

Control	**Motion**	**Push rod 1**	**Push rod 2**	**Push rod 3**
Collective blade pitch control	vertical	$0.050\sin(\Omega_t t)$	$0.050\sin(\Omega_t t)$	$0.050\sin(\Omega_t t)$
Longitudinal cyclic blade pitch control	forward/pitch	$0.025\sin(\Omega_t t)$	$0.025\sin(\Omega_t t)$	$0.050\sin(\Omega_t t)$
Lateral cyclic blade pitch control	sideways/roll	$0.025\sin(\Omega_t t)$	$0.050\sin(\Omega_t t)$	$0.025\sin(\Omega_t t)$

Figure 10 presents the hub assembly at different time instances for each flight scenario listed in Table 5. The contours illustrate the tilting of the swashplate mechanism that produces a pitching motion for each of the blades. In the collective case, the blades attain equal blade pitch, which would produce a net upward aerodynamic force during flight. In the longitudinal and lateral cyclic cases, the pitch of the blades varies as a function of the azimuthal angle and would produce longitudinal and lateral aerodynamic moments during flight. Therefore, this model combined with the control actuation inputs, can represent different flight scenarios and readily lends itself to a multiscenario optimization case which will be demonstrated later in this section.

A. Adjoint Gradient Verification

The verification of adjoint formulation presented in Section III is performed using the complex-step derivative approximation technique. The complex-step estimate is obtained by perturbing the input to a function by a complex perturbation as follows

$$\frac{df}{d\mathbf{x}_i} = \frac{\mathrm{Im}\left(f(\mathbf{x}_i + i\delta)\right)}{\delta} + \mathcal{O}(\delta^2), \tag{23}$$

where the design variable component $\mathbf{x}_i$ is perturbed by $i\,\delta$. The complex-step method (23) is second order accurate, such that the truncation error decreases quadratically with a reduction in perturbation size [46, 47]. However, unlike finite-difference method, the complex-step method does not suffer from subtractive cancellation, which enables the use of very small perturbation step sizes to produce highly accurate derivative estimates. The functionals used for this

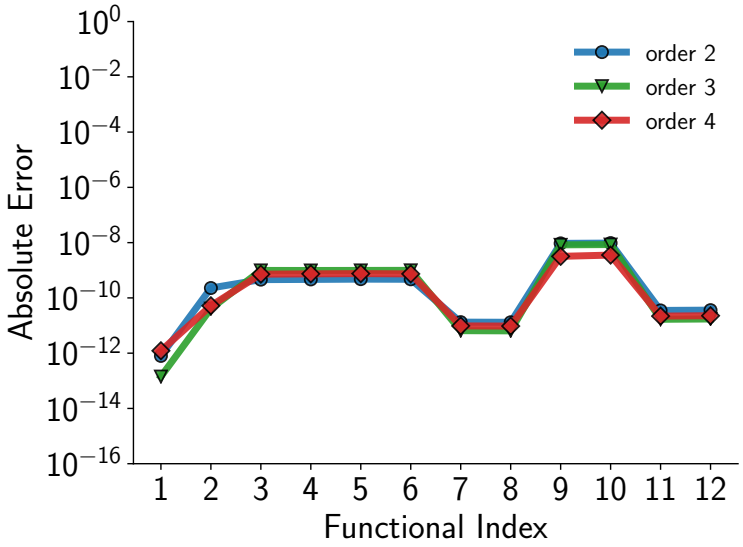


Figure 11: Complex-step verification of the DIRK adjoint formulation of different orders of accuracy with 12 functionals with a perturbation size $\delta = 10^{-16}$.

verification are the structural mass (index 1), the average structural compliance (index 2), the KS aggregates of the von Mises failure criterion (indices 3 and 4), and the induced exponential aggregates [28] of the von Mises failure criterion (indices 5 to 12). Table 6 shows the magnitude of discrete adjoint sensitivities along with corresponding complex-step

sensitivities. The digits in boldface represent the entries differing from the complex-step method.

Table 6: Comparison of complex-step and discrete adjoint derivatives for fourth-order DIRK with a perturbation size $\delta = 10^{-16}$.

Functional	**Complex-step**	**Adjoint**
Structural Mass	250.0000000000000	249.9999999999999
Compliance	-0.008903780405108	-0.00890378040**57068**
KS (discrete)	-2.510549172940552	-2.51054917**3663148**
KS (continuous)	-2.505178929745161	-2.50517893**0486006**
Induced (exponential)	-2.511154336312865	-2.51115433**7069819**
Induced (discrete exponential)	-2.516692488940226	-2.51669248**9679126**
Induced (discrete exponential squared)	-0.002788026920568	-0.00278802693**0265**
Induced (exponential squared)	-0.002762476355788	-0.00276247636**5353**
Induced (power)	-4.676554025570486	-4.67655402**8759453**
Induced (discrete power)	-4.715417147892726	-4.71541715**1435161**
Induced (power squared)	-0.006574319319522	-0.00657431931**41451**
Induced (discrete power squared)	-0.006679489586803	-0.00667948**9609466**

Figure 11 shows the absolute difference between the adjoint derivatives and the complex-step derivatives on the vertical axis for twelve test functionals indexed on the horizontal axis, for different orders of the DIRK time integration method. From Table 6 and Figure 11 it can be seen that the adjoint-based derivative exhibits an accuracy of 8 to 14 significant digits for different functionals. Thus, the adjoint-based gradient evaluation capability achieves sufficient accuracy for gradient-based optimization.

B. Rotorcraft Optimization Demonstration

The proposed parallel computational framework is applied to rotorcraft optimization in the following section. All three analysis scenarios described previously are included in optimization.

1. Optimization Problem

The objective of this optimization case is to minimize the mass of the of the rotor blades subject to stress constraints such that the von Mises failure ratio aggregated in space and time domains does not exceed 25% of maximum allowable value. A mass objective is chosen since more realistic rotor design objectives require multidisciplinary design criteria that cannot be evaluated without a coupled aeroelastic simulation. The design variables consist of the thicknesses of 48 spanwise panels modeling the rotor blades. Note that the thickness is constant in the chordwise direction. The cross-sectional geometry is held constant throughout the span for this demonstration case. Smoothness requirements

are imposed such that thicknesses of successive spanwise panels do not change more than 1 mm. The optimization problem is stated mathematically as follows:

$$
\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & \text{mass} = m(\mathbf{x}), \\
\text{subject to} \quad & \bar{g}^k(\mathbf{x}) = x_k - x_{k+1} \leq 1\,\text{mm}, \quad \forall k = 1,\dots 47, \quad \text{(smoothness requirement)} \\
& \bar{g}^k(\mathbf{x}) = x_{k+1} - x_k \leq 1\,\text{mm}, \quad \forall k = 1,\dots 47, \quad \text{(smoothness requirement)} \\
& \bar{g}^k(\mathbf{x}) = 1 - 4.0\frac{\sigma^k_{ks}}{\sigma^k_{max}} \geq 0, \qquad \forall k = 1,\dots 3, \quad \text{(stress constraint)} \\
\text{bounds} \quad & 10\,\text{mm} \leq \mathbf{x} \leq 20\,\text{mm}.
\end{aligned}
\tag{24}
$$

Each blade is assigned the same set of design variables so that all blades are identical during design. The time-dependent analysis and gradient evaluation for each of the three cases are performed in parallel using five processors. Both the mass objective and the smoothness constraints are independent of the structural state variables and their gradients are obtained analytically using straightforward differentiation. The time dependent adjoint formulation developed in this work is used to evaluate the three stress constraint gradients. The optimization problem with 48 design variables, 3 stress constraints from each flight scenario, and 94 smoothness constraints, is solved using the SLSQP optimizer within the python package pyOpt [48].

2. *Optimization Results*

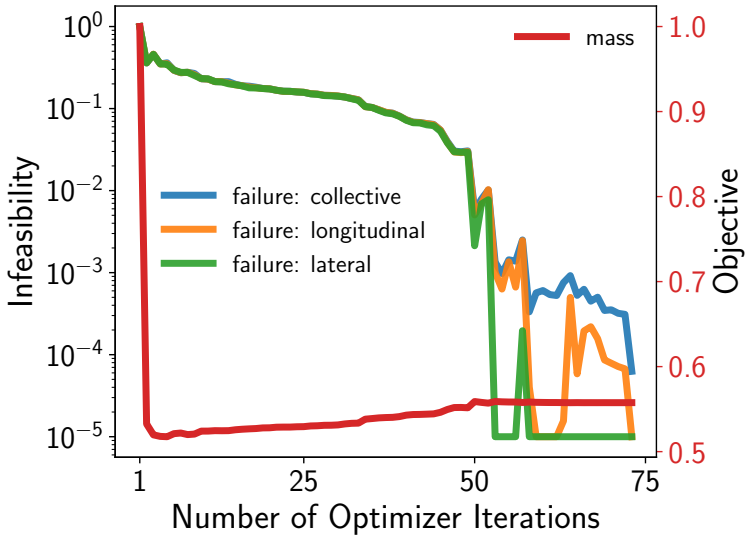


Figure 12: History of optimization showing the changes in normalized constraint infeasibility and objective values.

Figure 12 shows the optimization convergence history. The optimization took 73 iterations starting from an initial design of 2 cm thickness throughout, to converge to infeasibility and optimality tolerance of 10^{-4}. Note that the mass and stress constraint infeasibilities are normalized with respect to their values at the initial design. The mass of the

blades (shown in red) decreases immediately from the starting point and stays nearly constant throughout the rest of optimization. The stress constraint imposed on the lateral blade pitch flight scenario (shown in green) becomes feasible after about 50 iterations. Finally the stress constraints on longitudinal cyclic (shown in orange) and collective pitch (shown in blue) are satisfied near the termination of optimization algorithm. The optimization produced a design that has all three stress constraints active. The optimizer required 222 function evaluations and 88 gradient evaluations. On average, the forward analysis and gradient evaluation took 33 and 8 minutes, respectively, on five processors.

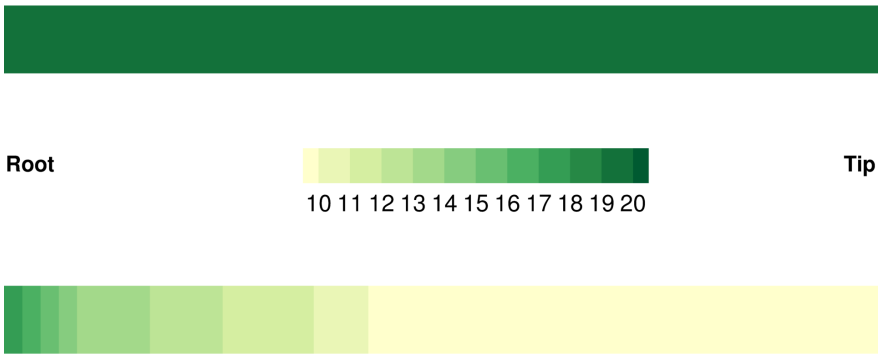


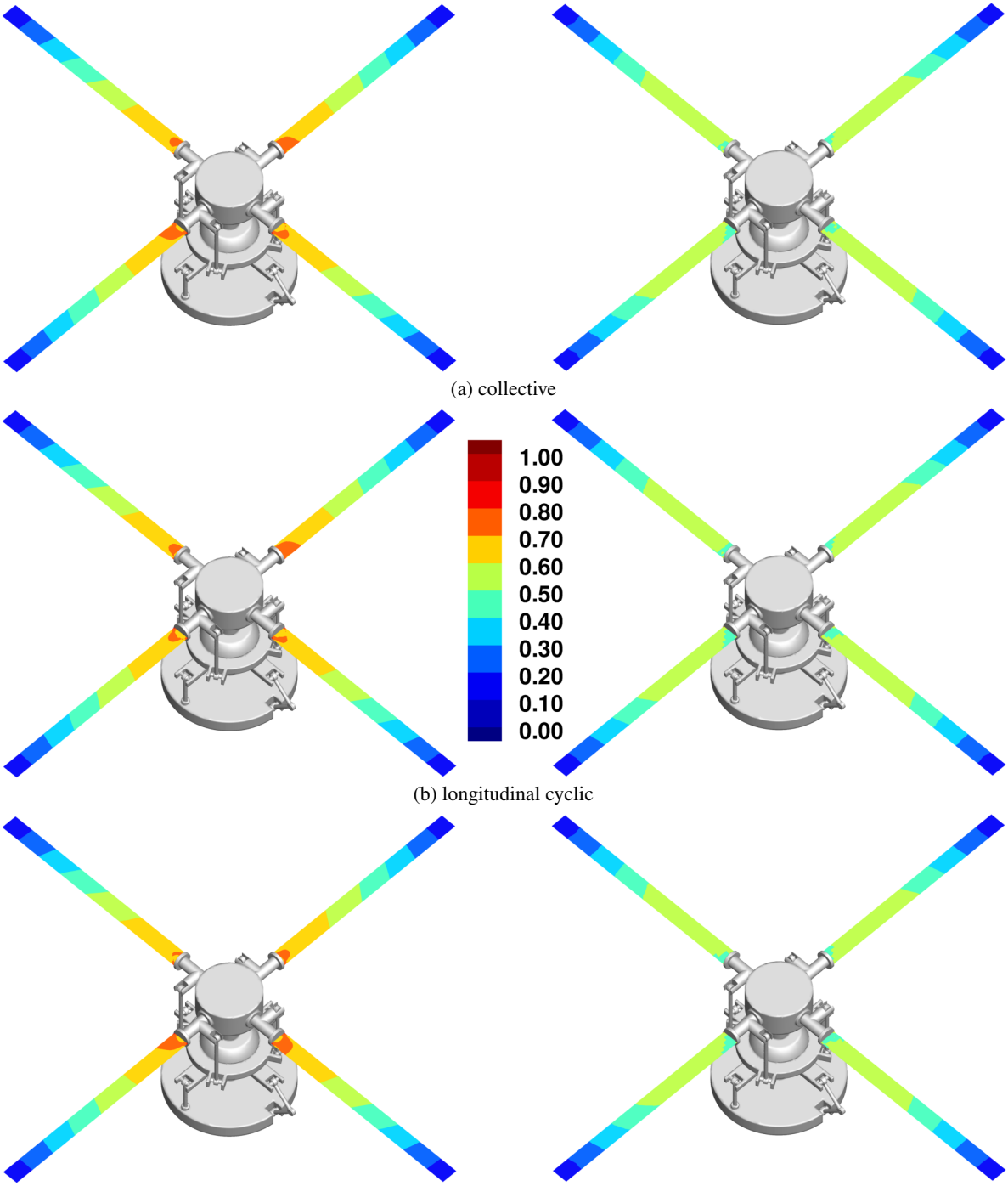
Figure 13: Comparison of thickness of initial (top) and optimized (bottom) designs in millimeters.

The final blade design thickness distribution produced by the optimizer is shown in Figure 13. Since the blades have identical design variables, only the optimized thickness from one blade is shown. The optimizer reduced the mass by decreasing the blade thicknesses towards the tip. The panel thickness gradually decreases along the span until it reaches the lower bound at the tip, thereby reducing the stress at the root. The gradual variation of panel thickness is a result of enforcing the smoothness constraints in the problem formulation.

Figure 14 compares the instantaneous stress normalized by design stress in the blades after one full rotation. The optimized stresses are lower for all three flight scenarios. The optimizer thickens the shell elements near the root of the blade that experience higher stress to comply with the stress requirements. In addition, there are notable differences in stress distribution patterns between each flight scenario, which is anticipated from the differences in dynamics. In this rotorcraft optimization application, high-fidelity structural dynamics, efficient sensitivity analysis using the adjoint method, along with the parallel processing capabilities have effectively reduced the optimization time.

VII. Conclusions

This paper presented a parallel flexible multibody dynamics design optimization framework, motivated by the need for a gradient-based design tool for next-generation rotorcraft design. This analysis and design tool will be an integral part of a broader multidisciplinary optimization framework for rotorcraft systems. The flexible multibody dynamics were implemented within the Toolkit for Analysis of Composite Structures (TACS) where the governing equations were derived based on constrained Euler–Lagrange equations. The governing equations were integrated directly in



(a) collective

(b) longitudinal cyclic

(c) lateral cyclic

Figure 14: Comparison of normalized stress failure ratios of initial and optimized blades for different flight scenarios at 360° azimuth.

their second-order form using a diagonally implicit Runge–Kutta method which enabled a consistent evaluation of the integral functionals of interest. The discrete adjoint equations were derived using the second-order descriptor form and a technique was described to facilitate a modular and extensible adjoint implementation. The structural dynamic analysis and adjoint-based derivative evaluation was shown to exhibit good parallel scalability for larger problems.

The structural dynamic analysis used was verified against an established four-bar benchmark problem and the time dependent discrete adjoint derivatives were verified against the complex-step derivatives. Finally, the optimization capabilities were demonstrated on a structural dynamics model of a rotor assembly, illustrating the applicability of the proposed framework for rotorcraft design applications.

Acknowledgments

The funding provided by NASA Langley Research Center through grant number NNL13AA08B with technical monitor Eric Nielsen is gratefully acknowledged.

References

[1] H. Saberi, M. Khoshlahjeh, R. Ormiston, and M. J. Rutkowski. Overview of RCAS and Application to Advanced Rotorcraft Problems. In *4th AHS Decennial Specialists' Conference on Aeromechanics*, San Francisco, CA, January 21-23 2004.

[2] W. Johnson. Rotorcraft Dynamic Models for a Comprehensive Analysis. In *American Helicopter Society Annual Forum 54*, Alexandria, VA, May 20-22 1998.

[3] O. Bauchau, C. Bottasso, and Y. Nikishkov. Modeling rotorcraft dynamics with finite element multibody procedures. *Mathematical and Computer Modelling*, 33(10):1113 – 1137, 2001. ISSN 0895-7177. doi:http://dx.doi.org/10.1016/S0895-7177(00)00303-4.

[4] G. J. Kennedy and J. R. R. A. Martins. A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures. *Finite Elements in Analysis and Design*, 87(0):56 – 73, 2014. ISSN 0168-874X. doi:10.1016/j.finel.2014.04.011.

[5] J. F. Kiviaho, K. Jacobson, M. J. Smith, and G. Kennedy. Application of a Time-Accurate Aeroelastic Coupling Framework to Flutter-Constrained Design Optimization. In *2018 Multidisciplinary Analysis and Optimization Conference*, AIAA Aviation Forum. AIAA, June 2018. doi:10.2514/6.2018-2932.

[6] K. Jacobson, J. F. Kiviaho, M. J. Smith, and G. Kennedy. An Aeroelastic Coupling Framework for Time-accurate Analysis and Optimization. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA SciTech Forum. AIAA, Jan 2018. doi:10.2514/6.2018-0100.

[7] J. F. Kiviaho, K. E. Jacobson, M. J. Smith, and G. J. Kennedy. A Robust and Flexible Coupling Framework for Aeroelastic Analysis and Optimization. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Denver, CO, June 2017. doi:10.2514/6.2017-4144.

[8] D. H. Hodges and E. H. Dowell. Nonlinear Equations of Motion for the Elastic Bending and Torsion of Twisted Nonuniform Rotor Blades. Technical Report NASA-TN-D-7818, NASA Ames Research Center and U.S. Army Air Mobility Research and Development Laboratory, 1974.

[9] O. A. Bauchau, C. Bottasso, and Y. Nikishkov. Modeling Rotorcraft Dynamics with Finite Element Multibody Procedures. *Mathematical and Computer Modeling*, 33(10-11):1113–1137, 2001.

[10] A. Datta and W. Johnson. Three-dimensional Finite Element Formulation and Scalable Domain Decomposition for High Fidelity Rotor Dynamic Analysis. *Journal of the American Helicopter Society*, 56(2):022003–1:14, 2011.

[11] A. Datta. X3D – A 3D Solid Finite Element Multibody Dynamic Analysis for Rotorcraft. In *American Helicopter Society Technical Meeting on Aeromechanics Design for Vertical Lift*, San Francisco, CA, January 2016.

[12] A. Ibrahimbegovic, R. L. Taylor, and H. Lim. Non-linear Dynamics of Flexible Multibody Systems. *Computers and Structures*, 81(12):1113–1132, 2003.

[13] J. Gerstmayr and J. Schöberl. A 3D Finite Element Method for Flexible Multibody Systems. *Multibody System Dynamics*, 14 (4):305–320, 2006.

[14] M. A. Akgun, R. T. Haftka, K. C. Wu, J. L. Walsh, and J. H. Garcelon. Efficient Structural Optimization for Multiple Load Cases Using Adjoint Sensitivities. *AIAA Journal*, 39(3):511–516, 2001. doi:10.2514/2.1336.

[15] A. D. Belegundu and J. S. Arora. A sensitivity interpretation of adjoint variables in optimal design. *Computer Methods in Applied Mechanics and Engineering*, 48(1):81–89, 1985. ISSN 00457825. doi:10.1016/0045-7825(85)90068-4.

[16] H. M. Adelman and R. T. Haftka. Sensitivity Analysis of Discrete Structural Systems. *AIAA Journal*, 24(5):823–832, 1986. doi:10.2514/3.48671.

[17] R. Haftka and H. Adelman. Recent developments in structural sensitivity analysis. *Structural optimization*, 1(3):137–151, 1989. ISSN 0934-4373. doi:10.1007/BF01637334.

[18] G. W. Burgreen and O. Baysal. Three-Dimensional Aerodynamic Shape Optimization Using Discrete Sensitivity Analysis. *AIAA Journal*, 34, No. 9:1761–1770, 1996. doi:10.2514/3.13305.

[19] W. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids*, 28(45):443 – 480, 1999. ISSN 0045-7930. doi:http://dx.doi.org/10.1016/S0045-7930(98)00041-3.

[20] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988. ISSN 08857474. doi:10.1007/BF01061285.

[21] B. Grossman, Z. Gürdal, R. T. Haftka, G. J. Strauch, and W. M. Eppard. Integrated aerodynamic/structural design of a sailplane wing. *Journal of Aircraft*, 25(9):855–860, 2013/09/28 1988. doi:10.2514/3.45980.

[22] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. A coupled–adjoint sensitivity analysis method for high–fidelity aero–structural design. *Optimization and Engineering*, 6:33–62, 2005. doi:10.1023/B:OPTE.0000048536.47956.62.

[23] Y. Cao, S. Li, L. Petzold, and R. Serban. Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint DAE System and Its Numerical Solution. *SIAM Journal on Scientific Computing*, 24(3):1076–1089, 2003. doi:10.1137/S1064827501380630.

[24] J.-Y. Ding, Z.-K. Pan, and L.-Q. Chen. Second order adjoint sensitivity analysis of multibody systems described by differential–algebraic equations. *Multibody System Dynamics*, 18(4):599–617, 2007. ISSN 1573-272X. doi:10.1007/s11044-007-9080-4.

[25] D. Dopico, Y. Zhu, A. Sandu, and C. Sandu. Direct and Adjoint Sensitivity Analysis of Ordinary Differential Equation Multibody Formulations. *Journal of Computational and Nonlinear Dynamics*, 10(1):1–7, 2014. doi:10.1115/1.4026492.

[26] K. Nachbagauer, S. Oberpeilsteiner, K. Sherif, and W. Steiner. The Use of the Adjoint Method for Solving Typical Optimization Problems in Multibody Dynamics. *Journal of Computational and Nonlinear Dynamics*, 10(6):1–10, 2015. ISSN 1555-1415. doi:10.1115/1.4028417.

[27] G. Kreisselmeier and R. Steinhauser. Systematic control design by optimizing a vector performance index. In *International Federation of Active Controls Symposium on Computer-Aided Design of Control Systems*, Zurich, Switzerland, 1979.

[28] G. J. Kennedy and J. E. Hicken. Improved constraint-aggregation methods. *Computer Methods in Applied Mechanics and Engineering*, 289:332 – 354, 2015. ISSN 0045-7825. doi:10.1016/j.cma.2015.02.017.

[29] O. A. Bauchau. *Flexible Multibody Dynamics*. Springer Netherlands, 2011. ISBN 978-94-007-0334-6. doi:10.1007/978-94-007-0335-3.

[30] C. Lanczos. *The Variational Principles of Mechanics*. Dover, New York, 4th edition, 2015. ISBN 978-0-486-65067-8.

[31] E. N. Dvorkin and K.-J. Bathe. A continuum mechanics based four-node shell element for general nonlinear analysis. *Engineering Computations*, 1:77–88, 1984.

[32] M. L. Bucalem and K.-J. Bathe. Higher-order MITC general shell elements. *International Journal for Numerical Methods in Engineering*, 36:3729–3754, 1993. ISSN 1097-0207. doi:10.1002/nme.1620362109.

[33] C. W. Gear. The Automatic Integration of Ordinary Differential Equations. *Commun. ACM*, 14(3):176–179, Mar. 1971. ISSN 0001-0782. doi:10.1145/362566.362571.

[34] C. W. Gear. Simultaneous Numerical Solution of Differential-Algebraic Equations. *IEEE Transactions on Circuit Theory*, 18 (1):89–95, Jan 1971. ISSN 0018-9324. doi:10.1109/TCT.1971.1083221.

[35] K. Brenan, S. Campbell, and L. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1995. doi:10.1137/1.9781611971224.

[36] E. J. Haug, D. Negrut, and M. Iancu. *Implicit Integration of the Equations of Multibody Dynamics*, pages 242–267. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-662-03729-4. doi:10.1007/978-3-662-03729-4_11.

[37] G. J. Kennedy and K. Boopathy. A Scalable Adjoint Method for Coupled Flexible Multibody Dynamics. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech*, 2016. doi:10.2514/6.2016-1907.

[38] J. C. Butcher. Implicit Runge–Kutta Processes. *Mathematics of Computation*, 18(85):50–64, 1964. ISSN 00255718, 10886842. doi:10.1090/S0025-5718-1964-0159424-9.

[39] R. Alexander. Diagonally Implicit Runge–Kutta Methods for Stiff O.D.E.'s. *SIAM J. Numer. Anal.*, 14(6):1006–1021, 1977. doi:10.1137/0714068.

[40] J. Cash. Diagonally implicit Runge–Kutta formulae for the numerical integration of nonlinear two-point boundary value problems. *Computers & Mathematics with Applications*, 10(2):123 – 137, 1984. ISSN 0898-1221. doi:10.1016/0898-1221(84)90043-9.

[41] C. A. Kennedy and M. H. Carpenter. Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review. Technical Report NASA/TM-2016-219173, NASA Langley Research Center, 2016.

[42] K. Boopathy and G. Kennedy. Adjoint-based derivative evaluation methods for flexible multibody systems with rotorcraft applications. In *55th AIAA Aerospace Sciences Meeting*, Grapevine, TX, January 2017. doi:10.2514/6.2017-1671.

[43] O. A. Bauchau, P. Betsch, A. Cardona, J. Gerstmayr, B. Jonker, P. Masarati, and V. Sonneville. Validation of flexible multibody dynamics beam formulations using benchmark problems. *Multibody System Dynamics*, 37(1):29–48, may 2016. ISSN 1573-272X. doi:10.1007/s11044-016-9514-y.

[44] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[45] C. Geuzaine and J. F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. ISSN 00295981. doi:10.1002/nme.2579.

[46] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, 1998. doi:10.1137/S003614459631241X.

[47] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, Sept. 2003. doi:10.1145/838250.838251.

[48] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins. pyOpt: a Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, 2012. doi:10.1007/s00158-011-0666-3.