2022-1

A General Coupling Methodology for Unsteady Aerostructural Optimization with Analytic Derivatives

Taylor McDonnell
Brigham Young University

Adam Cardoza
Brigham Young University

Denis-Gabriel Caprace
Brigham Young University

Andrew Ning
Brigham Young University, aning@byu.edu

A General Coupling Method for Unsteady Aerostructural Optimization with Analytic Derivatives

Taylor McDonnell*, Adam Cardoza† , Denis-Gabriel Caprace‡, and Andrew Ning§
Brigham Young University, Provo, UT, 84602

Multidisciplinary design optimization offers a comprehensive approach for designing complex aerostructural systems such as wind turbines. Gradient-based multidisciplinary design optimization can be used to optimize complex systems using large numbers of design variables. However, the convergence of gradient-based optimization methods to an optimal solution is highly dependent on the accuracy of the provided derivatives. In this paper, we propose a general method for creating unsteady coupled systems for steady-state, eigenvalue, and time-domain analyses. We then describe how highly accurate derivatives may be obtained from these analyses using a combination of automatic differentiation and analytic methods (including the unsteady adjoint). Finally, we use our coupling method to assemble and test several aerostructural models relevant for wind turbine design.

I. Introduction

Wind turbines are complex multidisciplinary systems. Wind turbine designers must be careful to consider all aspects of the wind turbine design problem simultaneously in order to design optimal turbines [1]. Blades must be aerodynamically efficient. Blade and tower costs should be minimized, often by reducing structural material, while still being able to avoid failure during their lifespan due to stress or fatigue. The optimal design for a wind turbine is also highly dependent on the turbine's operating conditions, which may vary from location to location. Additionally, if a turbine is part of an offshore system, then the number of conditions that must be considered further increases. The large number of aspects that must be considered and their interactions make it difficult to design optimized wind turbines.

In order to reduce costs and more quickly create innovative wind turbine designs, designers are increasingly turning to multidisciplinary design analysis and optimization (MDAO) frameworks. Multidisciplinary design analysis (MDA) frameworks, in which the interactions between disciplines are explicitly modeled, are often better able to predict real-world performance than isolated disciplinary models [2]. This allows designers to more quickly and accurately quantify the impact of proposed changes on the real-world performance of systems, saving time and money. MDAO frameworks provide the additional advantage relative to MDA frameworks of being designed for optimization so that the wind turbine design process can be efficiently automated. They also allow the design variables for various components of the wind turbine design problem to be optimized simultaneously rather than sequentially, which leads to better designs.

To facilitate the development of MDA/MDAO frameworks, several coupling methods and programs have been created which attempt to reduce the amount of effort required to assemble MDA/MDAO frameworks. A number of open source wind turbine MDA/MDAO coupling frameworks have also been created. Dymola [3], based on the Modelica language, was one of the first integrated approaches for modeling coupled systems. It, however, provided no native support for optimization. Other MDA/MDAO frameworks include Dassault Systemes' Isight/SEE, Pheonix Integration's ModelCenter/CenterLink, Esteco's modeFRONTIER, TechnoSoft's AML Suite, Noesis Solutions' Optimus, SORCER [4], and Vanderplaats' VisualDOC [5]. Open source MDA frameworks developed specifically for wind turbine design include OpenFast [6], QBlade [7] and SHARPy [8]. The first two provide a comprehensive array of models which cover most turbine systems and their interaction using various physics models. SHARPy is a higher fidelity model which mainly focuses on aerostructural interactions. Noteworthy MDAO frameworks for wind turbines include WISDEM [9], HAWTopt2 [10], and Cp-Max [11]. All provide comprehensive capabilities for the optimization of wind turbines, although the level of fidelity of the built-in models vary from one framework to another.

MDAO relies on the formulation of an optimization problem. The objective function and the constraints depend on functions of interests (FOIs). Since gradient-based optimization algorithms are the only computationally feasible

*Ph.D. Candidate, Department of Mechanical Engineering, AIAA Student Member.
†Ph.D. Candidate, Department of Mechanical Engineering, adam.cardoza@byu.edu, AIAA Student Member.
‡Post-doctoral Research Fellow, Department of Mechanical Engineering, AIAA Member.
§Associate Professor, Department of Mechanical Engineering, AIAA Associate Fellow.

optimization methods when large numbers of design variables are used [12], a key consideration when developing MDAO frameworks is how the gradients of the FOIs are obtained with respect to the design variables (DVs). The most computational expensive part of a gradient-based optimization is typically the gradient computations, so gradients should be calculated as efficiently as possible. Additionally, since gradients are a central component of gradient-based optimization algorithms, inaccuracies in the gradient computations due to errors or numerical noise have the potential to reduce the efficiency of the optimization algorithm or make optimization problems unsolvable. Despite the importance of accurate gradients, the majority of MDAO frameworks compute gradients using finite differencing, which is highly susceptible to numerical noise due to subtractive cancellation. This is especially apparent for long strings of computations, such as in unsteady time-marching simulations, since the computational noise associated with each computation can accumulate until the resulting error has a non-negligible impact on the accuracy of the gradients. Transient simulations are therefore often ill-suited for use in gradient-based optimization frameworks when finite differencing is used to compute gradients.

There are situations, however, when transient simulations are necessary in order to fully capture the design requirements of a MDAO problem. Wind turbines, in particular, should be designed to be compliant with the standards and design load cases (DLCs) set by the International Electrotechnical Commission [13]. Included in these standards are maximum loadings, deflections, and fatigue limits that turbines must be able to withstand. While it is common practice to consider only a subset of these design conditions in order to reduce computational costs during an optimization [11, 12, 14, 15], omitting critical design constraints can lead to the adoption of infeasible wind turbine designs. It is therefore important to be able to show compliance with all of these standards in a MDAO framework, including fatigue constraints which require the use of transient simulations. To allow for transient simulations to be included in MDAO frameworks, more exact methods for obtaining derivatives than finite-differencing are required.

Recently, OpenMDAO [16] was developed to provide a general framework for coupling multidisciplinary systems. One of the key advantages of OpenMDAO relative to other modeling approaches is its ability to compute analytic sensitivities for complex coupled systems. This allows it to provide exact derivatives to gradient-based optimizers, which increases the accuracy and efficiency of gradient-based optimization [17]. These capabilities make multidisciplinary design optimization possible and more reliable for a larger number of coupled systems than is possible with finite differencing.* Dymos [18], an optimal control library built on top of OpenMDAO, extends OpenMDAO to be able to model unsteady simulations using an implicit pseudo-spectral time integration approach. This approach is efficient for optimal control problems, but a time-marching approach may be less computationally complex and memory intensive when applied to initial value problems.

In this paper, we present a method for coupling multiple unsteady models which allows for the derivatives of steady and unsteady simulations to be obtained accurately and efficiently. This reduces the computational costs associated with including steady and unsteady simulations in MDAO frameworks, which in turn allows for additional design criteria to be included, such as wind turbine fatigue constraints. We then present the theory for how gradients may be obtained from steady state, eigenvalue, and unsteady analyses. The discrete adjoint method which we present, in particular, has the potential to provide significant improvements to both the accuracy and efficiency of sensitivity computations relative to finite differencing. Finally, we present several examples of aerostructural models created using our method in the Julia programming language [19].

II. Methods

In this section, we present our method for coupling multiple steady and unsteady models together in a modular framework. We then provide a brief overview of the analyses which may be performed using the resulting coupled system of equations. Finally, we present how exact derivatives may be obtained from steady state, frequency domain, and time domain analyses for use with gradient-based optimization.

A. Generic Framework for Modeling Coupled Systems

The governing equations for many unsteady systems may be described by the first-order implicit differential equation

$$0 = \boldsymbol{f}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{p}, t) \tag{1}$$

where $\boldsymbol{f}$ is a vector-valued residual function, $\dot{()}$ denotes the time derivative, $\boldsymbol{x}$ is a vector of state variables, $\boldsymbol{y}$ is a vector of time-varying inputs, $\boldsymbol{p}$ is a vector of time-invariant parameters, and t is the current time. If we assume that $\boldsymbol{f}$ is

*WISDEM is an excellent application of OpenMDAO that allows the developers to write modules with exact gradients

continuous and differentiable, then $\frac{\partial f}{\partial \dot{x}}$, $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial p}$, and $\frac{\partial f}{\partial t}$ exist and may be defined analytically or by using automatic differentiation.

The equations of motion for any number of unsteady models may be described as a single combined model by concatenating the governing equations and variables associated with each model. Assuming that each model may be described by eq. (1), the combined model is then given by

$$\mathbf{0} = \mathcal{F}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{P}, t) \tag{2}$$

where

$$\mathcal{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \quad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_n \end{bmatrix} \quad \boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \vdots \\ \boldsymbol{p}_n \end{bmatrix} \tag{3}$$

If the governing equations for each submodel in the combined model are continuous and differentiable, then the governing equations of the combined model are also continuous and differentiable with partial derivatives defined by the following expressions

$$\begin{gathered} \frac{\partial \mathcal{F}}{\partial \dot{\boldsymbol{X}}} = \begin{bmatrix} \frac{\partial f_1}{\partial \dot{x}_1} & & & \\ & \frac{\partial f_2}{\partial \dot{x}_2} & & \\ & & \ddots & \\ & & & \frac{\partial f_n}{\partial \dot{x}_n} \end{bmatrix} \quad \frac{\partial \mathcal{F}}{\partial \boldsymbol{X}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & & & \\ & \frac{\partial f_2}{\partial x_2} & & \\ & & \ddots & \\ & & & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad \frac{\partial \mathcal{F}}{\partial \boldsymbol{Y}} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & & & \\ & \frac{\partial f_2}{\partial y_2} & & \\ & & \ddots & \\ & & & \frac{\partial f_n}{\partial y_n} \end{bmatrix} \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{P}} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & & & \\ & \frac{\partial f_2}{\partial p_2} & & \\ & & \ddots & \\ & & & \frac{\partial f_n}{\partial p_n} \end{bmatrix} \quad \frac{\partial \mathcal{F}}{\partial t} = \begin{bmatrix} \frac{\partial f_1}{\partial t} \\ \frac{\partial f_2}{\partial t} \\ \vdots \\ \frac{\partial f_n}{\partial t} \end{bmatrix} \end{gathered} \tag{4}$$

The combined model described by eq. (2) is a decoupled model, since the states, inputs, and parameters of each submodel are not influenced in any way by the states, inputs, and parameters of the other submodels. It is this feature of the combined system which allows its partial derivatives to be defined as block diagonal matrices, as shown in eq. (4). To couple these models, we introduce the following coupling function, which defines the inputs of the combined model as functions of its rates, states, and parameters as well as the current time.

$$\boldsymbol{y} = \mathcal{G}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \boldsymbol{P}, t) \tag{5}$$

Combining eqs. (2) and (5) allows us to define the governing equations for a general coupled model.

$$\tilde{\mathcal{F}}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \boldsymbol{P}, t) = \mathcal{F}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \mathcal{G}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \boldsymbol{P}, t), \boldsymbol{P}, t) = \mathbf{0} \tag{6}$$

If we assume that $\mathcal{G}$ is continuous and differentiable, then $\frac{d\mathcal{G}}{d\dot{\boldsymbol{X}}}$, $\frac{d\mathcal{G}}{d\boldsymbol{X}}$, $\frac{d\mathcal{G}}{d\boldsymbol{P}}$, and $\frac{d\mathcal{G}}{dt}$ exist and may be defined analytically or using automatic differentiation. Assuming eq. (2) is also continuous and differentiable, then the partial derivatives of eq. (6) exist and may be defined analytically using the chain rule as

$$\begin{aligned} \frac{\partial \tilde{\mathcal{F}}}{\partial \dot{\boldsymbol{X}}} &= \frac{\partial \mathcal{F}}{\partial \dot{\boldsymbol{X}}} + \frac{\partial \mathcal{F}}{\partial \boldsymbol{Y}} \frac{\partial \mathcal{G}}{\partial \dot{\boldsymbol{X}}} \\ \frac{\partial \tilde{\mathcal{F}}}{\partial \boldsymbol{X}} &= \frac{\partial \mathcal{F}}{\partial \boldsymbol{X}} + \frac{\partial \mathcal{F}}{\partial \boldsymbol{Y}} \frac{\partial \mathcal{G}}{\partial \boldsymbol{X}} \\ \frac{\partial \tilde{\mathcal{F}}}{\partial \boldsymbol{P}} &= \frac{\partial \mathcal{F}}{\partial \boldsymbol{P}} + \frac{\partial \mathcal{F}}{\partial \boldsymbol{Y}} \frac{\partial \mathcal{G}}{\partial \boldsymbol{P}} \\ \frac{\partial \tilde{\mathcal{F}}}{\partial t} &= \frac{\partial \mathcal{F}}{\partial t} + \frac{\partial \mathcal{F}}{\partial \boldsymbol{Y}} \frac{\partial \mathcal{G}}{\partial t} \end{aligned} \tag{7}$$

This approach for creating coupled models presents several advantages. First, it is applicable to a wide variety of systems and therefore is amenable for use in a generic unsteady analysis framework. Second, it is highly modular. Separating the definitions for each model and their couplings makes it easier to define the differential equations associated with each model and to try out different combinations of models. Finally, by explicitly including parameters as part of the governing equations, derivatives with respect to these parameters can be more easily defined analytically or obtained using automatic differentiation. When these parameters are used as design variables in MDAO frameworks, this approach directly defines analytic derivatives for use with gradient-based optimization.

B. Applications of the Nonlinear Differential Equations of Coupled Systems

The first-order implicit differential equation for a coupled system defined in eq. (6) may be used to analyze the system's steady state response, perform eigenvalue analyses of small motions about steady state operating conditions, and simulate transient dynamic behavior. Each of these applications has the potential to be included as components of a design optimization framework. We provide a brief description of each of these analyses in this section.

1. Steady State Analyses

Steady state analyses are useful for determining and constraining a system's design operating conditions. They may be performed by neglecting time derivatives in the coupled system's governing equations (i.e., setting $\dot{\boldsymbol{X}} = \boldsymbol{0}$ and $t = \infty$) and solving the resulting nonlinear system of equations for the state variables corresponding to steady state operating conditions. A number of different nonlinear solution methods may be used to solve the resulting nonlinear system of equations, including, but not limited to, fixed-point iteration and the Newton-Raphson method. Many efficient implementations of these methods are available in open-source software packages, which reduces the development time associated with performing steady state analyses.

2. Eigenvalue Analyses

Eigenvalue analyses may be used to assess and constrain system stability for a variety of design operating conditions. To perform an eigenvalue analysis, we first linearize the governing differential equations in eq. (6) with respect to the states and state rates. Using a Taylor series expansion, the linearized equations of motion takes the form

$$\boldsymbol{0} = \tilde{\boldsymbol{\mathcal{F}}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t) + \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \dot{\boldsymbol{X}}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t)\left(\dot{\boldsymbol{X}} - \dot{\bar{\boldsymbol{X}}}\right) + \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \boldsymbol{X}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t)\left(\boldsymbol{X} - \bar{\boldsymbol{X}}\right) \tag{8}$$

where $\bar{()}$ denotes the value of a variable used for the linearization. Assuming steady state operating conditions and introducing the new variable $\hat{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$, these equations of motion may be expressed as

$$\boldsymbol{0} = \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \dot{\boldsymbol{X}}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t)\dot{\hat{\boldsymbol{x}}} + \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \boldsymbol{X}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t)\hat{\boldsymbol{x}} \tag{9}$$

If we now assume the solution takes the form $\hat{\boldsymbol{x}} = \boldsymbol{v}e^{\lambda t}$, then we obtain the generalized eigenvalue problem

$$(K + \lambda M)\,\boldsymbol{v} = \boldsymbol{0} \tag{10}$$

where

$$K = \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \boldsymbol{X}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t) \quad M = \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}}{\partial \dot{\boldsymbol{X}}}(\dot{\bar{\boldsymbol{X}}}, \bar{\boldsymbol{X}}, \boldsymbol{P}, t)) \tag{11}$$

If the mass matrix M is invertible, the generalized eigenvalue problem can be reformulated as a standard eigenvalue problem

$$\left(M^{-1}K + I\lambda\right)\boldsymbol{v} = \boldsymbol{0} \tag{12}$$

and solved using a standard eigenvalue solver. Otherwise a generalized eigenvalue problem solver may be used. If the system is large, an iterative solver such as ARPACK may be used to compute a subset of the eigenvalues of the full system.

3. Time Domain Analyses

Time domain analyses may be used to assess and constrain the transient behavior of a system. To simulate dynamic behavior, a time marching solution can be constructed from the governing differential equations using an appropriate integration scheme. A large number of possible integration schemes exist, each with their own advantages and disadvantages when applied to different problems. Among all these integration schemes, perhaps one of the most classic (and simple) is the forward Euler method. In this method, a nonlinear solver is first used to determine the state rates at the current time step $\dot{\boldsymbol{x}}_n$ given the current state variables $\boldsymbol{x}_n$, parameters $\boldsymbol{p}$, and time t. Then the state variables for the next time step are calculated using the expression

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \dot{\boldsymbol{x}}_n \Delta t \tag{13}$$

and the process is repeated. While the simplicity of the forward Euler method is appealing, in practice, more stable and efficient integration schemes should be used.

C. Derivatives of Coupled Systems

Obtaining exact derivatives of steady state, eigenvalue, and time-domain analyses can be tedious. Defining derivatives by hand can be a long, hard, and error prone process. Fortunately, automatic differentiation (AD) has made the task easier by introducing algorithms which automatically calculate derivatives using the chain rule[2]. Theoretically, AD may be applied to calculate exact derivatives for any program, but computational and practical limitations restrict its use.

Many complex iterative solvers are either incompatible with AD because they are implemented in a lower-level language with limited support for AD or too computationally expensive when used with AD. Examples of solvers for which AD is often unsupported or computationally expensive include the nonlinear solvers used for steady state analyses, the eigenvalue routines used for eigenvalue analyses, and unsteady time-integration schemes. For these solvers, it is often more appropriate to use a combination of automatic differentiation and analytic differentiation, in order to avoid the potential issues associated with applying AD to iterative processes.

When deriving expressions for the derivatives associated with iterative processes such as steady state, eigenvalue, and time domain analyses, it is convenient to introduce two types of derivatives: total and partial derivatives. Total derivatives, in the context of this work, are the derivatives of the functions of interest (FOIs). Partial derivatives are the derivatives of a function with respect to a variable while holding all other variables constant. In the following sections we present how we compute partial and total derivatives for steady state, eigenvalue, and time domain analyses in this paper.

1. Partial derivatives

Partial derivatives of the governing equations of a system are often used to solve nonlinear systems of equations. They can also be used to compute total derivatives. When exact expressions for these partial derivatives cannot be easily provided, one can use AD to compute them. AD consists of decomposing the entire code into basic operations with known derivatives. By applying the chain rule, one can then automatically compute the derivative of complex functions. Formally, AD can be seen as an augmented algebra which operates on primal and dual numbers. While the primal numbers perform regular floating point arithmetic, the dual numbers accumulate information about the derivatives. Julia, in particular, amenably allows for convenient and flexible AD implementations. Since the typing of variables is dynamic, one can simultaneously pass primal and dual numbers to most functions. Every computational operator can also be overloaded for dual numbers. Thus, AD capabilities are available to the user without modification to the original code.

The accumulation of derivative information in AD can be performed in several ways [20]. In forward mode, the differentiated variable is fixed and the derivative is obtained in the same order as the normal code execution. For vector-valued functions, the derivative of all outputs is obtained with respect to one input. This approach is relatively straightforward but the cost scales with the number of inputs. In reverse or backward mode, the summation in the chain rule happens in reverse. For vector-valued functions, the derivative of one output is obtained with respect to all inputs. Computationally, this is done by recording each computation in the original evaluation and then working backwards through each computation to obtain the derivatives. In the examples provided in this paper, forward AD is the default method to compute partial derivatives, based on the implementation provided in ForwardDiff.jl [21].

2. Total derivatives

Total derivatives are required in order to perform gradient-based optimization. In general, forward AD may be used to obtain these derivatives; however, this may be inefficient for models with large numbers of parameters, since

computational cost scales with the number of parameters. In this case, reverse AD can be used instead. The inefficiencies associated with using AD with steady state and eigenvalues analyses are typically small compared to the computational costs associated with time domain analyses. AD may therefore be reasonably used to compute total derivatives for steady state and eigenvalue analyses while analytic expressions are employed to reduce the computational costs associated with obtaining total derivatives from time-domain analyses.

We recommend using the discrete unsteady adjoint method for obtaining total derivatives from time-domain analyses. While this method is more involved than AD, it presents tremendous advantages when applied to optimization. To demonstrate how the discrete unsteady adjoint method may be implemented for a coupled system, let us first assume that the coupled system residual in eq. (6) may be expressed as

$$\tilde{\mathcal{F}}(\dot{\boldsymbol{X}}, \boldsymbol{X}, \boldsymbol{P}, t) = \boldsymbol{M}(\boldsymbol{X}, \boldsymbol{P}, t)\dot{\boldsymbol{X}} + \mathcal{R}(\boldsymbol{X}, \boldsymbol{P}, t) = \boldsymbol{0} \tag{14}$$

where $\boldsymbol{M}$ is the mass matrix, and $\mathcal{R}$ is a vector valued function. Note that the only restriction we place upon the governing equations presented in eq. (6) is that the residual function is linearly dependent on the state rates, which still allows this system to be applicable to a wide variety of coupled systems.

We use Euler's implicit method with N discrete time steps to demonstrate the discrete unsteady adjoint method. Using this integration scheme, an expression for the residual function $\tilde{\mathcal{F}}$ at each time step may be constructed, with values

$$\tilde{\mathcal{F}}^n = \boldsymbol{M}^n(\boldsymbol{X}^n, \boldsymbol{P}, t^n)\frac{\boldsymbol{X}^n - \boldsymbol{X}^{n-1}}{\Delta t} + \mathcal{R}^n(\boldsymbol{X}^n, \boldsymbol{P}, t^n) = \boldsymbol{0} \qquad n = 1, ..., N \tag{15}$$

where the superscript n denotes the discrete time index. Using this notation, $\boldsymbol{X}^0$ represents the initial conditions and $\boldsymbol{X}^N$ represents the final state vector. Note that the initial conditions must satisfy eq. (14).

As proposed by Kast [22], it is convenient to express the solution to the discrete time integration as the large matrix problem:

$$\mathbf{F} = \frac{1}{\Delta t}\begin{bmatrix} \boldsymbol{M}^1 & & \\ & \ddots & \\ & & \boldsymbol{M}^N \end{bmatrix}\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{X}^0 \\ \mathbf{X} \end{bmatrix} + \mathbf{R} = \mathbf{0}. \tag{16}$$

where

$$\mathbf{F} = \begin{bmatrix} \tilde{\mathcal{F}}^1 \\ \vdots \\ \tilde{\mathcal{F}}^N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \boldsymbol{X}^1 \\ \vdots \\ \boldsymbol{X}^N \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} \mathcal{R}^1 \\ \vdots \\ \mathcal{R}^N \end{bmatrix} \tag{17}$$

Let us also define a generic scalar FOI which is dependent on the value of the state variables throughout the simulation as well as the parameters:

$$h\left(\boldsymbol{P}, \mathbf{X}\right) \tag{18}$$

Common FOIs for time domain simulations which can be represented by eq. (18) include the value of a quantity at a given time step, the maximum of a quantity over a given time interval, and the integral of a quantity over a given time frame. Additionally, while this FOI is restricted to a scalar output, the analysis remains general, since any number of FOIs may be considered.

Our objective is to obtain the derivative of the FOI with respect to the parameters. Taking advantage of the notation introduced in eq. (16) the gradient of the FOI with respect to the parameters is

$$\frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{P}} = \frac{\partial h}{\partial \boldsymbol{P}} + \frac{\partial h}{\partial \mathbf{X}}\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\boldsymbol{P}} \tag{19}$$

Equivalently, the derivative of the residual expressions in $\mathbf{F}$ with respect to the parameters is

$$\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\boldsymbol{P}} = \frac{\partial \mathbf{F}}{\partial \boldsymbol{P}} + \frac{\partial \mathbf{F}}{\partial \mathbf{X}}\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\boldsymbol{P}} = 0 \tag{20}$$

Since the derivative of eq. (20) is equal to zero, we can rearrange this expression to obtain the total derivative of the states with respect to the parameters:

$$\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}\boldsymbol{P}} = -\left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}}\right)^{-1}\frac{\partial \mathbf{F}}{\partial \boldsymbol{P}} \tag{21}$$

Substituting eq. (21) into eq. (19) yields the total derivatives of the FOI with respect to the parameters, expressed in terms of partial derivatives:

$$\frac{\mathrm{d}h}{\mathrm{d}\boldsymbol{P}} = \frac{\partial h}{\partial \boldsymbol{P}} - \frac{\partial h}{\partial \mathbf{X}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \right)^{-1} \frac{\partial \mathbf{F}}{\partial \boldsymbol{P}} \tag{22}$$

To complete our derivation of the discrete unsteady adjoint, we define the adjoint vector $\boldsymbol{\Psi}$ such that

$$\boldsymbol{\Psi}^T = \frac{\partial h}{\partial \mathbf{X}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \right)^{-1} \tag{23}$$

Rather than attempting to invert $\frac{\partial \mathbf{F}}{\partial \boldsymbol{X}}$, we simply postulate it as a linear system,

$$\left(\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \right)^T \boldsymbol{\Psi} = \left(\frac{\partial h}{\partial \mathbf{X}} \right)^T \tag{24}$$

One way to obtain $\boldsymbol{\Psi}$ is by directly solving the large nonlinear system. However, to be computational efficient, we take advantage of the sparsity of the matrices in the discrete adjoint method. Crucially, we notice from eq. (16) that only the major diagonal and the first lower diagonal of $(\partial \mathbf{F}/\partial \mathbf{X})$ are non-zero.

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \frac{\partial \tilde{\boldsymbol{\mathcal{F}}}^j}{\partial \boldsymbol{X}^j} = \begin{cases} \dfrac{\partial \boldsymbol{M}^i}{\partial \boldsymbol{X}^j} \dfrac{\left(\boldsymbol{X}^i - \boldsymbol{X}^{i-1}\right)}{\Delta t} + \dfrac{\boldsymbol{M}^i}{\Delta t} + \dfrac{\partial \boldsymbol{\mathcal{R}}^i}{\partial \boldsymbol{X}^j} & i = j \\ -\dfrac{\boldsymbol{M}^i}{\Delta t} & i = j - 1 \end{cases} \tag{25}$$

Hence, eq. (24) can be solved efficiently. For example, using backward propagation, the elements of the adjoint vector with respect to the parameters may be computed using the following set of expressions, which may be solved starting from the final time step.

$$\begin{aligned} \left(\frac{\partial \boldsymbol{M}^n}{\partial \boldsymbol{X}^n} \frac{\left(\boldsymbol{X}^n - \boldsymbol{X}^{n-1}\right)}{\Delta t} + \frac{\boldsymbol{M}^n}{\Delta t} + \frac{\partial \boldsymbol{\mathcal{R}}^n}{\partial \boldsymbol{X}^n} \right) \boldsymbol{\psi}^n - \left(\frac{\boldsymbol{M}^{n+1}}{\Delta t} \right) \boldsymbol{\psi}^{n+1} &= \left(\frac{\partial h}{\partial \boldsymbol{X}^n} \right) \quad n = 1, \ldots, N-1 \\ \left(\frac{\partial \boldsymbol{M}^n}{\partial \boldsymbol{X}^n} \frac{\left(\boldsymbol{X}^n - \boldsymbol{X}^{n-1}\right)}{\Delta t} + \frac{\boldsymbol{M}^n}{\Delta t} + \frac{\partial \boldsymbol{\mathcal{R}}^n}{\partial \boldsymbol{X}^n} \right) \boldsymbol{\psi}^n &= \left(\frac{\partial h}{\partial \boldsymbol{X}^n} \right) \quad n = N \end{aligned} \tag{26}$$

Substituting $\boldsymbol{\Psi}$ into eq. (22) yields the total derivatives of h with respect to the parameters, where $\frac{\partial h}{\partial \boldsymbol{P}}$ and $\frac{\partial h}{\partial \mathbf{X}}$ may be defined using automatic differentiation and $\frac{\partial \mathbf{F}}{\partial \boldsymbol{P}}$ may be defined analytically using partial derivatives of eq. (5).

III. Results & Discussion

In this section, we present several coupled models which have been constructed using our framework. We first present several aerostructural models created by coupling various aerodynamic models with a typical section model. Then we present a unsteady aerodynamic model created by coupling the Risø dynamic stall model [23] with the blade element momentum (BEM) method. These examples showcase the modular nature of our coupling method.

A. Aeroelasticity of a Typical Section

For our first example, we model the aeroelasticity of a rigid, spring-restrained, two-dimensional airfoil section with plunging degree of freedom h and pitching degree of freedom θ, as depicted in fig. 1. This model is commonly referred to as a typical section model, and is a classic example in aeroelasticity textbooks due to its simplicity.

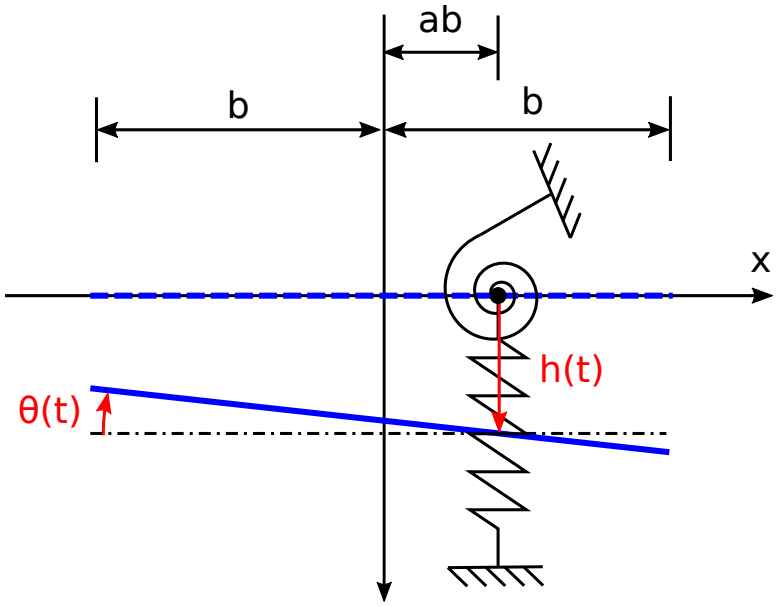

Fig. 1 Two degree of freedom typical section model

The equations of motion for this model are

$$\begin{bmatrix} m & S_\theta \\ S_\theta & I_\theta \end{bmatrix} \begin{bmatrix} \ddot{h} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} k_h & 0 \\ 0 & k_h \end{bmatrix} \begin{bmatrix} h \\ \theta \end{bmatrix} = \begin{bmatrix} -\mathcal{L} \\ \mathcal{M} \end{bmatrix} \tag{27}$$

where k_h is the linear spring constant, k_θ is the torsional spring constant, m is the mass per unit span, S_θ is the structural imbalance, I_θ is the mass moment of inertia, $\mathcal{L}$ is the lift per unit span, and $\mathcal{M}$ is the moment per unit span. The structural imbalance S_θ is defined as the x-displacement of the center of mass from the reference location multiplied by the mass per unit span. We define the state, input, and parameter vectors for the typical section model as

$$x_{\text{sec}} = \begin{bmatrix} h \\ \theta \\ \dot{h} \\ \dot{\theta} \end{bmatrix} \quad y_{\text{sec}} = \begin{bmatrix} \mathcal{L} \\ \mathcal{M} \end{bmatrix} \quad p_{\text{sec}} = \begin{bmatrix} k_h \\ k_\theta \\ m \\ S_\theta \\ I_\theta \end{bmatrix} \tag{28}$$

All properties and loads for this model are defined at the reference location, which is located ab aft of the semichord, as shown in fig. 1.

We use four different aerodynamic models to model the lift and moment of the typical section model. All four models are based on thin airfoil theory, but each make different assumptions when modeling unsteady airfoil behavior. The inputs to each aerodynamic model are the local freestream velocity in the chordwise direction u, the local freestream velocity in the normal direction v, the freestream angular velocity ω, and the time derivatives of these values. The outputs from each aerodynamic model are the normal force $\mathcal{N}$ and moment $\mathcal{M}$ at the reference location. These inputs and outputs are depicted in fig. 2.

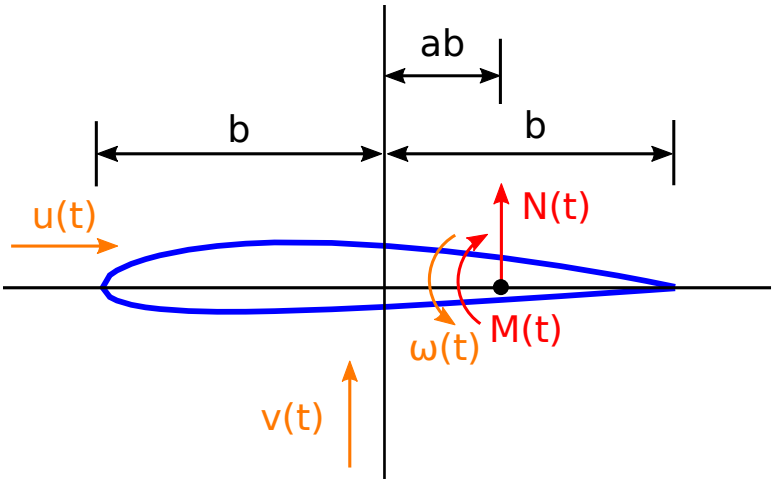

Fig. 2 Inputs and outputs for the two-dimensional aerodynamic models

1. Steady Thin Airfoil Theory

The first aerodynamic model we consider is the classical steady thin airfoil theory, which defines the normal force $\mathcal{N}$ and moment $\mathcal{M}$ at the reference location as

$$\begin{aligned} \mathcal{N} &= a_0 \rho_\infty u^2 b \alpha_{\text{eff}} \\ \mathcal{M} &= b\left(\frac{1}{2} + a\right)\mathcal{N} \end{aligned} \tag{29}$$

where a defines the reference location, b is the semi-chord, a_0 is the lift curve slope, ρ_∞ is the air density, and α_{eff} is the effective angle of attack, defined as

$$\alpha_{\text{eff}} = -\frac{v}{u} - \alpha_0 \tag{30}$$

where α_0 is the zero lift angle of attack.

To couple this model with the typical section model, we make use of a small angle approximation to define the local freestream velocity components as functions of the freestream velocity U_∞ and pitch angle θ.

$$\begin{aligned} u &\approx U_\infty \\ v &\approx U_\infty \theta \\ \omega &\approx 0 \end{aligned} \tag{31}$$

We also use a small angle assumption to define the lift at the reference location to be approximately equal to the normal force at the reference location.

$$\mathcal{L} \approx \mathcal{N} \tag{32}$$

2. Quasi-Steady Thin Airfoil Theory

The second aerodynamic model we consider is a quasi-steady thin airfoil theory model which is based on unsteady thin airfoil theory as derived by Theodorsen [24], but neglects unsteady aerodynamic effects. For this model, the normal force $\mathcal{N}$ and moment $\mathcal{M}$ at the reference location are defined as

$$\begin{aligned} \mathcal{N} &= a_0 \rho_\infty u^2 b \alpha_{\text{eff}} + \pi \rho b^2 \left(\dot{v} + u\omega - ab\dot{\omega}\right) \\ \mathcal{M} &= -\pi \rho_\infty b^3 \left[\frac{1}{2}\dot{v} + u\omega + b\left(\frac{1}{8} - \frac{a}{2}\right)\dot{\omega}\right] + b\left(\frac{1}{2} + a\right)\mathcal{N} \end{aligned} \tag{33}$$

The effective angle of attack for this model is defined as

$$\alpha_{\text{eff}} = \frac{v}{u} + \frac{b}{u}\left(\frac{1}{2} - a\right)\omega - \alpha_0 \tag{34}$$

To couple this model with the typical section model, we assume the freestream velocity components u and v are aligned with the undeflected chordwise and normal directions, respectively, so that

$$\begin{aligned} u &\approx U_\infty \\ v &\approx \dot{h} \\ \omega &\approx \dot{\theta} \end{aligned} \tag{35}$$

To capture the effect of twist on the circulatory lift (since it is not implicitly modeled by the $\frac{v}{u}$ quantity) twist is added to the effective angle of attack from the quasi-steady model so that the effective angle of attack is now given by

$$\alpha_{\text{eff}} = \theta - \frac{v}{u} + \frac{b}{u}\left(\frac{1}{2} - a\right)\omega - \alpha_0 \tag{36}$$

The original expression for the effective angle of attack may be used by defining the new variable $\bar{v} = u\theta + v$ such that

$$\alpha_{\text{eff}} = -\frac{\bar{v}}{u} + \frac{b}{u}\left(\frac{1}{2} - a\right)\omega - \alpha_0 \tag{37}$$

We also use a small angle assumption to define the lift at the reference location to be approximately equal to the normal force at the reference location,

$$\mathcal{L} \approx \mathcal{N} \tag{38}$$

Since the steady state and quasi-steady models introduce no new state variables into the coupled model's governing equations, they also introduce no additional residual expressions or inputs into the coupled model. The state variables, inputs, and parameters for the typical section model when coupled with either of these models is then

$$x_{\text{coupled}} = \begin{bmatrix} h \\ \theta \\ \dot{h} \\ \dot{\theta} \end{bmatrix} \quad y_{\text{coupled}} = \begin{bmatrix} \mathcal{L} \\ \mathcal{M} \end{bmatrix} \quad p_{\text{coupled}} = \begin{bmatrix} a \\ b \\ a_0 \\ \alpha_0 \\ k_h \\ k_\theta \\ m \\ S_\theta \\ I_\theta \\ U_\infty \\ \rho_\infty \end{bmatrix} \tag{39}$$

where we have introduced the aerodynamic parameters a, b, a_0, α_0 for the steady state and quasi-steady models and the additional parameters U_∞, ρ_∞ for the coupled model. The inputs L and M may be defined using the coupled model's input function, which may be derived using eqs. (29) to (32) for the steady model or eqs. (33), (35), (36) and (38) for the quasi-steady model.

3. Wagner's Function Unsteady Aerodynamics

The third aerodynamic model we consider is a unsteady aerodynamic model based on Wagner's function, which models the indicial response of aerodynamic loads under a sudden change in downwash w at the three-quarter's chord. The exact expression for Wagner's function is

$$\phi(t) = \frac{2}{\pi}\int_0^\infty \frac{Re(C)\sin(\omega(u/b)t)}{\omega} d\omega \tag{40}$$

where $C(\omega)$ is Theodorsen's function. In many cases, approximate versions of Wagner's function are used rather than the exact expression, of which one of the most common is the approximation of Wagner's function provided by R. T. Jones [25],

$$\phi(t) = 1 - C_1 e^{-\varepsilon_1 (u/b)t} - C_2 e^{-\varepsilon_2 (u/b)t} \tag{41}$$

where $C_1 = 0.165$, $C_2 = 0.335$, $\varepsilon_1 = 0.455$, and $\varepsilon_2 = 0.3$.

Wagner's function may be used to model arbitrary airfoil motion using Duhamel's integral. We start by modeling the increment in the circulatory normal force $d\mathcal{N}_c(t)$ at time t due to an increment in downwash $dw(t)$ at earlier time τ as

$$\frac{d\mathcal{N}_c(t)}{a_0\rho_\infty ub} = \phi(t-\tau)dw(\tau) \tag{42}$$

where $\phi(t)$ is the impulse response function, which in this case is defined in eq. (41). Superimposing all previous impulse responses using Duhamel's integral yields the following expression for the instantaneous circulatory normal force:

$$\frac{\mathcal{N}_c}{a_0\rho_\infty ub} = \int_{-\infty}^{t} dw(\tau)\phi(t-\tau)d\tau = w(0)\phi(t) + \int_0^t dw(\tau)\phi(t-\tau)d\tau \tag{43}$$

We can transform this equation using integration by parts, yielding

$$\frac{\mathcal{N}_c}{a_0\rho_\infty ub} = w(t)\phi(0) - \int_0^t w(\tau)d\phi(t-\tau)d\tau \tag{44}$$

The integral in this expression may be expressed as a the following function of the aerodynamic states λ_1 and λ_2

$$\lambda_1 + \lambda_2 = -\int_0^t w(\tau)d\phi(t-\tau)d\tau \tag{45}$$

where

$$\lambda_1 = C_1\varepsilon_1\frac{u}{b}\int_0^t w(\tau)e^{-\varepsilon_1(u/b)(t-\tau)}d\tau \tag{46}$$

and

$$\lambda_2 = C_2\varepsilon_2\frac{u}{b}\int_0^t w(\tau)e^{-\varepsilon_2(u/b)(t-\tau)}d\tau \tag{47}$$

The expression for the circulatory normal force then reduces to

$$\frac{\mathcal{N}_c}{a_0\rho_\infty ub} = w(t)\phi(0) + \lambda_1 + \lambda_2 \tag{48}$$

where the downwash at the three quarter's chord is given by

$$w(t) = v + b\left(\frac{1}{2} - a\right)\omega - u\alpha_0 \tag{49}$$

and the aerodynamic state variables λ_1 and λ_2 may be described by the ordinary differential equations

$$\dot{\lambda_1} = -\varepsilon_1\frac{u}{b}\lambda_1 + C_1\varepsilon_1\frac{u}{b}w(t) \tag{50}$$

and

$$\dot{\lambda_2} = -\varepsilon_2\frac{u}{b}\lambda_2 + C_2\varepsilon_2\frac{u}{b}w(t) \tag{51}$$

The same expressions for the normal force and moment at the reference location are used as in the quasi-steady model (see eq. (33)), but with the following effective angle of attack

$$\alpha_{\text{eff}} = \left(\frac{v}{u} + \frac{b}{u}\left(\frac{1}{2} - a\right)\omega - \alpha_0\right)\phi(0) + \frac{\lambda_1}{u} + \frac{\lambda_2}{u} \tag{52}$$

For the Wagner's function model, two state variables and corresponding governing equations are added to those for the typical section model. However, since all coupling occurs through the input variables, the governing equations for

the Wagner's function model may be used as previously defined. The state variables, inputs, and parameters for the typical section model when coupled with the Wagner's function model is

$$x_{\text{coupled}} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ h \\ \theta \\ \dot{h} \\ \dot{\theta} \end{bmatrix} \quad y_{\text{coupled}} = \begin{bmatrix} u \\ v \\ \omega \\ \mathcal{L} \\ \mathcal{M} \end{bmatrix} \quad p_{\text{coupled}} = \begin{bmatrix} a \\ b \\ a_0 \\ \alpha_0 \\ k_h \\ k_\theta \\ m \\ S_\theta \\ I_\theta \\ U_\infty \\ \rho_\infty \end{bmatrix} \tag{53}$$

The inputs u, v, ω, L, M may be defined using the coupled model's input function, which may be derived using eqs. (33), (35), (38) and (52).

4. Peters' Finite State Model Aerodynamics

The final aerodynamic model we consider is an unsteady aerodynamic model based on the finite state theory by Peters et al. [26]. For this model, an additional term λ_0 is added to the expression for the effective angle of attack from the quasi-steady model (see eq. (34)) to account for induced velocity:

$$\alpha_{\text{eff}} = \frac{v}{u} + \frac{b}{u}\left(\frac{1}{2} - a\right)\omega + \frac{\lambda_0}{u} - \alpha_0 \tag{54}$$

The induced velocity λ_0 is approximated from a set of N induced-flow states $\lambda_1, \lambda_2, \ldots, \lambda_N$ as

$$\lambda_0 \approx \frac{1}{2}\sum_{n=1}^{N} b_n \lambda_n \tag{55}$$

The set of N first-order ordinary differential equations which govern the N finite aerodynamic states are derived by Peters et al. as

$$\bar{A}\dot{\lambda} + \frac{u}{b}\lambda = \bar{c}\left[-\dot{v} + u\omega + b\left(\frac{1}{2} - a\right)\dot{\omega}\right] \tag{56}$$

where

$$\bar{A} = \bar{D} + \bar{d}\bar{b}^T + \bar{c}\bar{d}^T + \frac{1}{2}\bar{c}\bar{b}^T$$

$$\bar{D}_{nm} = \begin{cases} \frac{1}{2n} & n = m+1 \\ \frac{-1}{2n} & n = m-1 \\ 0 & n \neq m \pm 1 \end{cases} \quad \bar{b}_n = \begin{cases} (-1)^{n-1}\frac{(N+n-1)!}{(N-n-1)!}\frac{1}{(n!)^2} & n \neq N \\ (-1)^{n-1} & n = N \end{cases} \quad \bar{c}_n = \frac{2}{n} \quad \bar{d}_n = \begin{cases} \frac{1}{2} & n = 1 \\ 0 & n \neq 1 \end{cases} \tag{57}$$

Similar to the unsteady aerodynamic model based on Wagner's function, the normal force and moment at the reference location for this model is the same as that provided for the quasisteady model (see eq. (33)), but with the modified effective angle of attack defined in eq. (54).

For the Peters' finite state model, N state variables and corresponding governing equations are added to those for the typical section model. However, as with the Wagner's function model, since all coupling occurs through the input variables, the governing equations for Peters' finite state model may be used as previously defined. The state variables,

inputs, and parameters for the typical section model when coupled with the Peters' finite state model is

$$x_{\text{coupled}} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ h \\ \theta \\ \dot{h} \\ \dot{\theta} \end{bmatrix} \quad y_{\text{coupled}} = \begin{bmatrix} u \\ \omega \\ \dot{v} \\ \dot{\omega} \\ \mathcal{L} \\ \mathcal{M} \end{bmatrix} \quad p_{\text{coupled}} = \begin{bmatrix} a \\ b \\ a_0 \\ \alpha_0 \\ k_h \\ k_\theta \\ m \\ S_\theta \\ I_\theta \\ U_\infty \\ \rho_\infty \end{bmatrix} \tag{58}$$

The inputs $u, \omega, \dot{v}, \dot{\omega}, L, M$ for this model may be defined using the coupled model's input function, which may be derived using eqs. (33), (35), (38) and (54).

5. Typical Section Aeroelastic Analysis

Using each of the aerodynamic models coupled with the typical section model, we perform an eigenvalue analyses to determine the flutter speed for the two-degree of freedom typical section. We use the non-dimensional parameters

$$\begin{aligned} a &= -1/5 & e &= -1/10 \\ r^2 &= \frac{I_\theta}{mb^2} & \sigma &= \frac{\omega_h}{\omega_\theta} \\ \mu &= \frac{m}{\rho_\infty \pi b^2} & V_\infty &= \frac{U_\infty}{b\omega_\theta} \end{aligned} \tag{59}$$

where e is the x-displacement of the center of mass from the reference location normalized by b, V_∞ is the reduced velocity, and ω_h and ω_θ are the uncoupled natural frequencies of the typical section model, defined as

$$\omega_h = \sqrt{\frac{k_h}{m}} \quad \omega_\theta = \sqrt{\frac{k_\theta}{I_\theta}} \tag{60}$$

The results of the eigenvalue analyses are presented in fig. 3. The damping Γ and frequency Ω of each mode are the real and imaginary parts of the corresponding eigenvalue, respectively. With damping defined in this manner, a mode is unstable when its damping is positive. The flutter velocity is the smallest velocity that has at least one unstable mode.

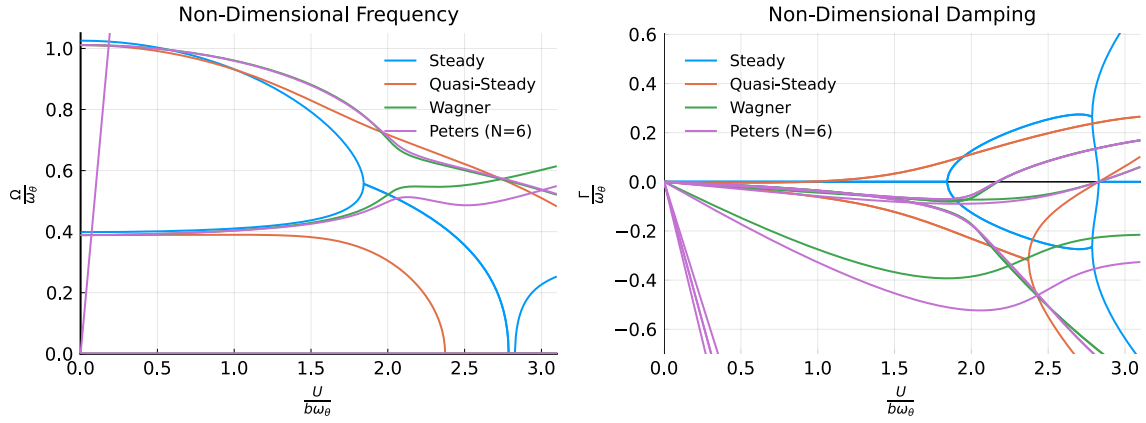

Fig. 3 Aeroelastic analyses for the steady state, quasi-steady, Wagner's function, and Peter's finite state model coupled with a two degree of freedom typical section model

Using the Wagner or Peters aerodynamic models yields a flutter reduced velocity around 2.2, while the steady and quasi-steady aerodynamic models predict lower flutter velocities. The aerodynamic state variables of the Wagner and Peters models allows these models to capture the impact of vortex shedding on the lift and drag of the profile, therefore we can expect these models to yield more accurate results than the steady state and quasi-steady models.

The non-dimensional parameters we use for this example match those used by Hodges and Pierce[27], who performed this analysis using a steady state model and Peter's finite state model with six state variables. The results presented here for the steady state and Peters' finite state models match the results presented by Hodges and Pierce exactly, which verifies our implementation of these models. Additionally, since the flutter speed predicted by the Wagner and Peters' models match, we can be reasonably confident that the Wagner unsteady aerodynamic model is also implemented correctly.

B. Unsteady Aerodynamics of a Wind Turbine

To demonstrate the capability of our coupling methodology to simulate coupled models across time, we couple the Risø dynamic stall model [23] with the blade element momentum (BEM) method.

1. The Risø Dynamic Stall Model

The Risø model is a simplified Beddoes-Leishman type dynamic stall model specifically designed for the aeroelastic modeling of wind turbines. The state space representation of the model is

$$\begin{aligned}
\dot{x}_1 &= -\left(b_1 + \frac{c\dot{U}}{2U^2}\right)\frac{x_1}{T_u} + \frac{b_1 A_1 \alpha}{T_u} \\
\dot{x}_2 &= -\left(b_2 + \frac{c\dot{U}}{2U^2}\right)\frac{x_2}{T_u} + \frac{b_2 A_2 \alpha}{T_u} \\
\dot{x}_3 &= -\frac{x_3}{T_p} + \frac{1}{T_p}\left(\left.\frac{\mathrm{d}C_l}{\mathrm{d}\alpha}\right|_{\alpha_0}(\alpha_{\text{eff}} - \alpha_0) + \pi T_u \dot{\alpha}\right) \\
\dot{x}_4 &= -\frac{x_4}{T_f} + \frac{f(\alpha_f)}{T_f}
\end{aligned} \tag{61}$$

where x_i are states, U is the time dependent freestream velocity, α is the time dependent angle of attack, α_{eff} is the effective angle of attack, α_f is the equivalent angle of attack, f is the separation point function, A_i and b_i are airfoil specific dynamic response coefficients, c is the chord length, $\left.\frac{\mathrm{d}C_l}{\mathrm{d}\alpha}\right|_{\alpha_0}$ is the slope of the lift curve in the linear region, T_u is a time varying time constant, T_p is the pressure response time constant, and T_f is the boundary layer response time constant. The effective angle of attack, equivalent angle of attack, separation point function, and time varying time constant are calculated as

$$\begin{gathered}
\alpha_{\text{eff}} = \alpha(1 - A_1 - A_2) + x_1 + x_2 \\
\alpha_f = \frac{x_3}{\mathrm{d}C_l/\mathrm{d}\alpha} + \alpha_0 \qquad f(\alpha) = \left(2\sqrt{\frac{C_{l,\text{static}}(\alpha)}{\mathrm{d}C_l/\mathrm{d}\alpha(\alpha - \alpha_0)}} - 1\right)^2 \qquad T_u = \frac{c}{2U}
\end{gathered} \tag{62}$$

where $C_{l,\text{static}}(\alpha)$ is a function representing the fit of the static lift curve. The lift and drag of the airfoil can easily be computed based on the states,

$$C_l^{dyn} = \left.\frac{\mathrm{d}C_l}{\mathrm{d}\alpha}\right|_{\alpha_0}(\alpha_{\text{eff}} - \alpha_0)x_4 + C_l^{fs}(\alpha_{\text{eff}})(1 - x_4) + \pi T_u \dot{\alpha} \tag{63}$$

$$C_d^{dyn} = C_d^{\text{static}}(\alpha_{\text{eff}}) + (\alpha - \alpha_{\text{eff}})C_L^{dyn} + \left(C_d^{\text{static}}(\alpha_{\text{eff}} - C_{d_0})\right)\left(\frac{\sqrt{f(\alpha_{\text{eff}})} - \sqrt{x_4}}{2} - \frac{f(\alpha_{\text{eff}}) - x_4}{4}\right) \tag{64}$$

Our implementation of the Risø model can be compared with Hansen's implementation [23]. We consider the pitching motion of a NACA 6315 airfoil with a 1 m chord, at a reduced frequency of $k = 0.1$ with a 1 m/s freestream velocity. The angle of attack for this test is:

$$\alpha(t) = 4^\circ \sin(\omega t) + 12^\circ \tag{65}$$

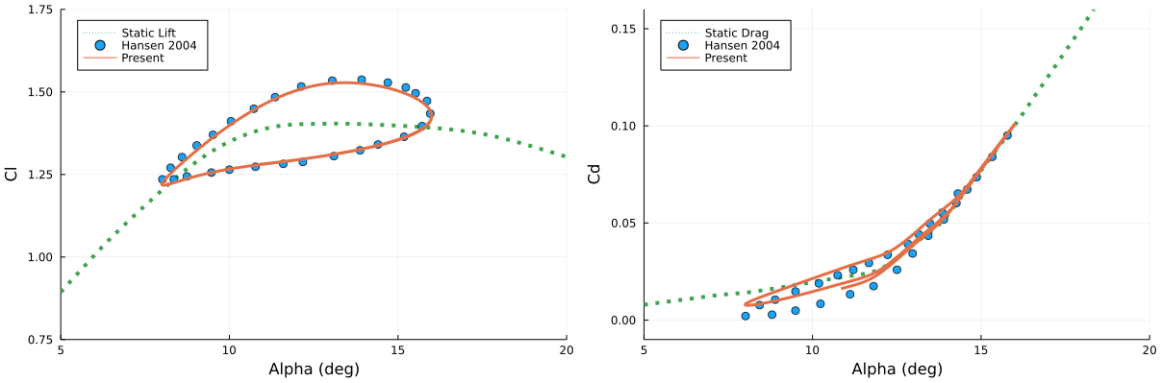

Fig. 4 The oscillatory pitching motion of a NACA 6315.

As can be seen in figure 4, there are small differences between our results and Hansen's. The maximum absolute error is 0.02 and 0.007 for the coefficients of lift and drag respectively. The average absolute error is 0.009 for the lift coefficient and 0.003 for the drag coefficient. These discrepancies may be the result of differences between the dynamic airfoil coefficients used by Hansen and the dynamic airfoil coefficients used in our analysis. Hansen's original paper does not provide dynamic airfoil coefficients, so we had to estimate these coefficients by minimizing the difference between the lift predicted by our analysis and Hansen's. Considering the potential discrepancies between the coefficients used in our analysis and Hansen's, we consider the current level of agreement to be a reasonable verification of our implementation.

2. Blade Element Momentum Method

Blade element momentum (BEM) theory is a common rotor aerodynamic analysis method that utilizes conservation of steady linear and angular momentum, and the forces on a blade element to resolve the induced inflow velocity. Typically the inflow velocity is resolved by converging two residuals simultaneously. We use Ning's [28, 29] formulation of the BEM equations, which condenses the two residuals to a single residual that is a function of the inflow angle (ϕ):

$$\mathcal{R}(\phi) = \frac{\sin(\phi)}{1 + a(\phi)} - \frac{u}{v}\frac{\cos(\phi)}{\left(1 - a'(\phi)\right)} \tag{66}$$

where u is the axial inflow velocity, v is the tangential inflow velocity (typically $v = \Omega r$), $a(\phi)$ is the axial induction factor, and $a'(\phi)$ is the tangential induction factor. The induction factors are defined as

$$a(\phi) = \frac{\sigma' c_n}{4F\sin^2(\phi) - \sigma' c_n} \qquad a'(\phi) = \frac{\sigma' c_t}{4F\sin(\phi)\cos(\phi) + \sigma' c_t}$$
$$\sigma' = \frac{Bc}{2\pi r} \tag{67}$$

where σ' is the local solidity, F is the tip correction factor, c_n and c_t are the normal and tangential force coefficients, B is the number of blades, and r is the radial distance from the center of rotation. Rather than re-implementing the BEM equations for this paper, we use the Julia package CCBlade.jl [29], which has been extensively validated.

3. Aerodynamic Coupling

The state rate equations of the dynamic stall model (eq. (61)) can be easily formed into a single residual equation for coupling with other models. We define the state, input, and parameter vectors for the dynamic stall model as

$$x_{\text{ds}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad y_{\text{ds}} = \begin{bmatrix} U \\ \dot{U} \\ \alpha \\ \dot{\alpha} \end{bmatrix} \qquad p_{\text{ds}} = \begin{bmatrix} c \\ \mathrm{d}C_l/\mathrm{d}\alpha \\ \alpha_0 \\ A \\ b \end{bmatrix} \tag{68}$$

where

$$U = \sqrt{u^2 + v^2} \quad \dot{U} = \frac{u\dot{u} + v\dot{v}}{\sqrt{u^2 + v^2}} \quad \alpha = \phi - (\theta + \beta) \tag{69}$$

and θ is the local twist angle and β is the pitch angle. We assume changes in the inflow angle are negligible, such that the derivative of the angle of attack is zero, $\dot{\alpha} = 0$.

We define the BEM method state, input, and parameter vectors as

$$x_{bem} = \begin{bmatrix} \phi \end{bmatrix} \quad y_{bem} = \begin{bmatrix} C_l \\ C_d \\ u \\ v \end{bmatrix} \quad p_{bem} = \begin{bmatrix} r \\ c \\ \theta \\ \beta \\ B \end{bmatrix} \tag{70}$$

The inputs C_l, C_d, u and v may be defined using the coupled model's input function, which may be derived using eqs. (63) and (64) and the local evaluation of the inflow and rotation velocities along the blade. Note that we chose to use the force coefficients as inputs, as it allows the model to be more general.

This coupling is then solved across time using the Julia package DifferentialEquations.jl [30]. Here we present a time-domain simulation of a simplified version of the NREL 5 MW reference wind turbine [31] (we omitted the shaft tilt from the model). We verify that the dynamic analysis, starting from arbitrary initial conditions, converges to the steady solution from CCBlade (see fig. 5). Future work will demonstrate the capability of the method to model unsteady conditions originating in rotor tilt, yaw, or turbulent inflow.

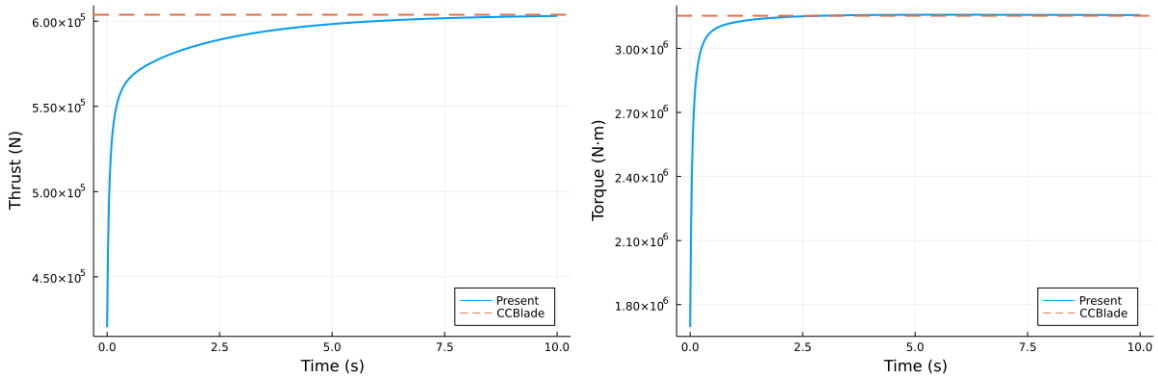

Fig. 5 Rotor plane forces converging to the steady state value (CCBlade) across time.

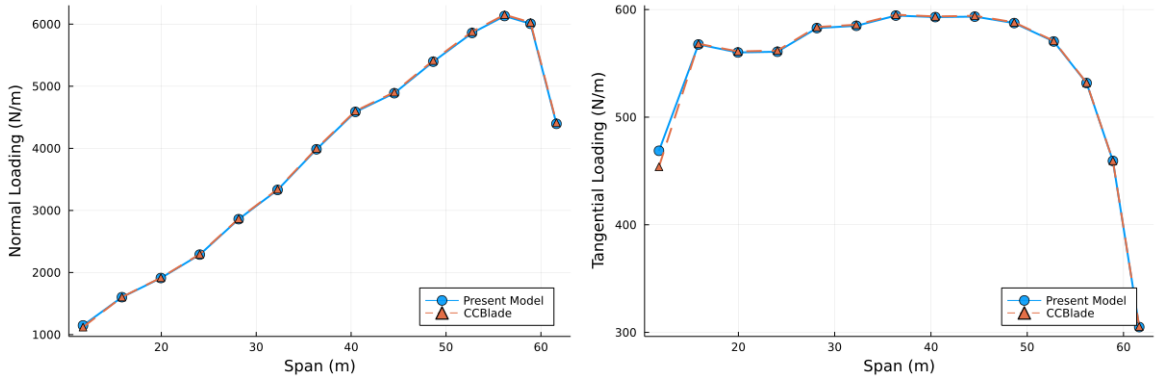

Fig. 6 Converged values of the normal and tangential loadings across the span of the turbine blade.

IV. Conclusion and Future Work

In this paper, we presented a general coupling method for creating unsteady models for use in multidisciplinary design analysis and optimization frameworks. This method handles the connections between collections of systems defined as implicit first-order differential equations. It also propagates derivatives through these connections so that they may be used for analysis and/or optimization. The result is a coupled set of first-order implicit differential equations with defined partial derivatives.

We described how coupled models created using our method may be used for a variety of engineering analyses including steady-state, eigenvalue, and time-domain analyses and how total derivatives may be obtained from these analyses using automatic differentiation and analytic methods. We also presented theory for how the unsteady adjoint method may be implemented in order to more efficiently calculate the derivatives of quantities derived from time marching analyses.

Through two examples, we showed how our method can facilitate constructing and solving aerostructural problems. First, we considered the aeroelastic behavior of a typical section model using various aerodynamic models. With this coupled system, we demonstrated that an accurate solution for the flutter velocity can be obtained. Then we considered a time-domain simulation of a wind turbine blade. For this simulation we coupled a dynamic stall model and a blade element momentum solver using our approach. We then verified the consistency between the system's long term response and expected steady state behavior.

The coupling method presented in this paper is a first step toward enabling the efficient and comprehensive MDAO of complex aerostructural systems. We plan to fully integrate this coupling method with analytic equations so that exact gradients (within machine precision error) from steady-state, eigenvalue, and time-domain analyses may be more easily obtained for use with gradient-based optimization. We also plan to use this coupling method to develop a complex aerostructural MDAO for the design and development of wind turbines similar to OpenFAST[6], but designed specifically for use with gradient-based optimization. By leveraging the capabilities of this coupling method to provide efficient and accurate derivatives, we expect the resulting MDAO framework to be able to easily handle the design and optimization of complex wind turbine systems parameterized with large numbers of variables.

Acknowledgments

This research is supported by the Department of Energy (DOE) Advanced Research Projects Agency-Energy (ARPA-E) Program award DE-AR0001186 entitled "Computationally Efficient Control Co-Design Optimization Framework with Mixed-Fidelity Fluid and Structure Analysis." The authors thank DOE ARPA-E Aerodynamic Turbines Lighter and Afloat with Nautical Technologies and Integrated Servo-control (ATLANTIS) Program led by Dr. Mario Garcia-Sanz. Special thanks to the entire ATLANTIS Team for their support.

References

[1] Ning, A., Damiani, N., and Moriarty, P., "Objectives and constraints for wind turbine optimization." *Journal of Solar Energy Engineering*, 2013. https://doi.org/10.2514/6.2013-201.

[2] Martins, J. R., and Ning, A., *Engineering design optimization*, Cambridge University Press, 2021.

[3] Brück, D., Elmqvist, H., Mattsson, S. E., and Olsson, H., "Dymola for multi-engineering modeling and simulation," *Proceedings of modelica*, Vol. 2002, Citeseer, 2002.

[4] Kolonay, R. M., and Sobolewski, M., "Service oriented computing environment (SORCER) for large scale, distributed, dynamic fidelity aeroelastic analysis," *Optimization, International Forum on Aeroelasticity and Structural Dynamics, IFASD2011*, 2011.

[5] Balabanov, V., Com, V., Charpentier, C., Ghosh, D., Quinn, G., Vanderplaats, G., and Venter, G., "VisualDOC: A Software System for General Purpose Integration and Design Optimization," *AIAA*, 2002, pp. 2002–5513. https://doi.org/10.2514/6.2002-5513.

[6] NREL, "OpenFAST v3.0.0," , June 23, 2021. URL https://github.com/OpenFAST/openfast.

[7] Marten, D., Wendler, J., Pechlivanoglou, G., Nayeri, C. N., and Paschereit, C. O., "QBLADE: an open source tool for design and simulation of horizontal and vertical axis wind turbines," *Int. J. Emerging Technol. Adv. Eng*, Vol. 3, No. 3, 2013, pp. 264–269.

[8] del Carre, A., Munoz-Simon, A., Goizueta, N., and Palacios, R., "SHARPy: A dynamic aeroelastic simulation toolbox for very flexible aircraft and wind turbines," *Journal of Open Source Software*, Vol. 4, No. 44, 2019, p. 1885. https://doi.org/10.21105/joss.01885, URL https://doi.org/10.21105/joss.01885.

[9] NREL, "Wind-Plant Integrated System Design and Engineering Model (WISDEM) v3.4.1," , Nov 4, 2021. URL https://github.com/WISDEM/WISDEM.

[10] Zahle, F., Tibaldi, C., Pavese, C., McWilliam, M. K., Blasques, J. P. A. A., and Hansen, M. H., "Design of an Aeroelastically Tailored 10 MW Wind Turbine Rotor," *Journal of Physics: Conference Series*, Vol. 753, 2016, p. 062008. https://doi.org/10.1088/1742-6596/753/6/062008.

[11] Bottasso, C., Bortolotti, P., Croce, A., and Gualdoni, F., "Integrated aero-structural optimization of wind turbines," *Multibody Syst Dyn*, Vol. 38, 2016, pp. 317–344. https://doi.org/10.1007/s11044-015-9488-1.

[12] Ning, A., and Petch, D., "Integrated design of downwind land-based wind turbines using analytic gradients," *Wind Energy*, Vol. 19, 2016. https://doi.org/10.1002/we.1972.

[13] *IEC Wind turbines Part 1: Design requirements*, International Electrotechnical Commission, 3rd ed., 08 2005.

[14] Ingersoll, B., "Efficient Incorporation of Fatigue Damage Constraints in Wind Turbine Blade Optimization," *Wind Energy*, Vol. 23, No. 4, 2020, pp. 1063–1076. https://doi.org/10.1002/we.2473.

[15] Caprace, D.-G., Cardoza, A., Ning, A., Mangano, M., He, S., and Martins, J. R. R. A., "Incorporating High-Fidelity Aerostructural Analyses in Wind Turbine Rotor Optimization," *AIAA Scitech Forum*, 2021.

[16] Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., and Naylor, B. A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization," *Structural and Multidisciplinary Optimization*, Vol. 59, No. 4, 2019, pp. 1075–1104. https://doi.org/10.1007/s00158-019-02211-z.

[17] Martins, J. R. R. A., and Hwang, J. T., "Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models," *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2582–2599. https://doi.org/https://doi.org/10.2514/1.J052184.

[18] Falck, R., Gray, J. S., Ponnapalli, K., and Wright, T., "dymos: A Python package for optimal control of multidisciplinary systems," *Journal of Open Source Software*, Vol. 6, No. 59, 2021, p. 2809.

[19] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A fresh approach to numerical computing," *SIAM review*, Vol. 59, No. 1, 2017, pp. 65–98. URL https://doi.org/10.1137/141000671.

[20] Naumann, U., *The art of differentiating computer programs: an introduction to algorithmic differentiation*, SIAM, 2011.

[21] Revels, J., Lubin, M., and Papamarkou, T., "Forward-Mode Automatic Differentiation in Julia," *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

[22] Kast, S. M., "An introduction to adjoints and output error estimation in computational fluid dynamics," Tech. rep., University of Michigan, 2017. URL https://arxiv.org/abs/1712.00693.

[23] Hansen, M. H., Gaunaa, M., and Madsen, H. A., "A Beddoes-Leishman type dynamic stall model in state-space and indicial formulations," Tech. Rep. Risø-R-1354, June 2004.

[24] Theodorsen, T., "General Theory of Aerodynamic Instability and the Mechanism of Flutter," Tech. Rep. 496, National Advisory Committee for Aeronautics, Washington, DC, 1934.

[25] Jones, R., "The Unsteady Lift of a Wing of Finite Aspect Ratio," Tech. Rep. 681, National Advisory Committee for Aeronautics, Washington, DC, 1940.

[26] Peters, D. A., Karunamoorthy, S., and Cao, W.-M., "Finite state induced flow models. I - Two-dimensional thin airfoil," *Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 313–322. https://doi.org/10.2514/3.46718, URL https://doi.org/10.2514/3.46718.

[27] Hodges, D. H., and Pierce, G. A., *Introduction to Structural Dynamics and Aeroelasticity*, 2nd ed., Cambridge University Press, New York, 2011.

[28] Ning, A., "A simple solution method for the blade element momentum equations with guaranteed convergence," *Wind Energy*, Vol. 17, 2014, pp. 1327–1345. https://doi.org/https://doi.org/10.1002/we.1636.

[29] Ning, A., "Using Blade Element Momentum Methods with Gradient-Based Design Optimization," *Structural and Multidisciplinary Optimization*, 2021. https://doi.org/10.1007/s00158-021-02883-6.

[30] Rackauckas, C., and Nie, Q., "Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia," *Journal of Open Research Software*, Vol. 5, No. 1, 2017. https://doi.org/http://doi.org/10.5334/jors.151.

[31] Jonkman, J. M., Butterfield, N. S., Musial, W., and Scott, G., "Definitino of a 5-MW Reference Wind Turbine for Offshore System Development," Tech. Rep. NREL/TP-500-38060, National Renewable Energy Laboratory, February 2009.