Structural and Multidisciplinary Optimization (2018) 57:1815–1827
https://doi.org/10.1007/s00158-018-1912-8

EDUCATIONAL ARTICLE

Open-source coupled aerostructural optimization using Python

John P. Jasa[1] · John T. Hwang[2] · Joaquim R. R. A. Martins[1]

Received: 2 August 2017 / Revised: 28 November 2017 / Accepted: 15 January 2018 / Published online: 7 February 2018

Abstract
To teach multidisciplinary design optimization (MDO) to students effectively, it is useful to have accessible software that runs quickly, allowing hands-on exploration of coupled systems and optimization methods. Open-source software exists for low-fidelity aerodynamic or structural analysis, but there is no existing software for fast tightly coupled aerostructural analysis and design optimization. To address this need, we present OpenAeroStruct, an open-source low-fidelity aerostructural analysis and optimization tool developed in NASA's OpenMDAO framework. It uses the coupled adjoint method to compute the derivatives required for efficient gradient-based optimization. OpenAeroStruct combines a vortex lattice method and 1-D finite-element analysis to model lifting surfaces, such as aircraft wings and tails, and uses the coupled-adjoint method to compute the aerostructural derivatives. We use the Breguet range equation to compute the fuel burn as a function of structural weight and aerodynamic performance. OpenAeroStruct has proved effective both as an educational tool and as a benchmark for researching new MDO methods. There is much more potential to be exploited as the research community continues to develop and use this tool.

Keywords Aerostructural design optimization · Wing design · Multidisciplinary design optimization · Project-based learning · Python

1 Summary

In this paper, we discuss OpenAeroStruct,[1] an open-source coupled aerostructural analysis and design optimization tool. OpenAeroStruct couples the vortex-lattice method (VLM) and finite-element analysis (FEA) using six degree-of-freedom (DOF) spatial beam elements with axial, bending, and torsional stiffness. It is mostly implemented in Python, but some of the more intensive computations use Fortran.

[1] https://github.com/mdolab/openaerostruct

✉ John P. Jasa
johnjasa@umich.edu

John T. Hwang
john.hwang@epeerless.com

Joaquim R. R. A. Martins
jrram@umich.edu

[1] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, USA

[2] Peerless Technologies Corporation (contractor at NASA Glenn Research Center), Cleveland, OH, USA

OpenAeroStruct is developed within the OpenMDAO framework (Heath and Gray 2012), a NASA-developed open-source software framework for multidisciplinary design optimization (MDO). OpenMDAO facilitates derivative computation for gradient-based optimization using the modular analysis and unified derivatives (MAUD) architecture (Hwang and Martins 2018), which unifies the adjoint method with the chain rule and all other methods for computing discrete derivatives (Martins and Hwang 2013). OpenAeroStruct computes derivatives for the aerostructural system using the coupled adjoint method (Martins et al. 2005; Kenway et al. 2014). The aerodynamic forces and structural displacements are transferred between disciplines in a consistent and conservative manner. This process is simplified because the aerodynamic and structural meshes have the same spanwise discretization, so no interpolation is necessary to transfer the loads or displacements. A variety of solvers can be used to converge the coupled aerostructural system, including block Gauss–Seidel, GMRES, or LU decomposition for the linear system, and nonlinear block Gauss–Seidel or Newton for the nonlinear system. The standard aerostructural optimization problem is a fuel-burn minimization using the Breguet range equation, and the design variables consist of

twist distribution, spar thickness distribution, and planform variables. The constraints ensure that lift equals weight and the structural spar does not fail. The modular implementation in OpenMDAO makes it easy to reformulate the optimization problem, e.g., to minimize the sum of direct operating cost and acquisition cost, and to subsequently update the derivative computation.

OpenAeroStruct is unique because of its modular implementation and the efficient coupled gradient computation. The model is decomposed into many low-level computations. Each of these computations is implemented as an OpenMDAO *component*, and OpenMDAO computes the overall model-level derivatives given component-level derivatives of the outputs with respect to the inputs of each component. The components use symbolic differentiation or automatic differentiation (AD) to compute their partial derivatives (Mader et al. 2008). We include multiple visualization tools to examine the wing model and design variables during the optimization. Figure 1 shows a view from the included interactive GUI-based visualization tool.

The main objective of OpenAeroStruct is to provide a physically meaningful multidisciplinary model that can be used to obtain low-order approximations of aircraft performance or to compare solution algorithms. We evaluate on-design aerostructural performance using the Breguet range equation to approximate fuel burn. OpenAeroStruct is useful for a variety of educational areas, including aircraft design, MDO, uncertainty quantification, and numerical solution algorithms.

This paper caters to three different sets of users:

1. For educators, this paper presents OpenAeroStruct as a hands-on learning tool for the classroom. The modular implementation within a framework, the coupling within the disciplines, and the simplicity of the physics make it a practical teaching tool, as we have already demonstrated in the classroom setting at two universities. The individual aerodynamic and structural models are well known, but the way that they are coupled to provide analytic coupled derivatives is novel. Educators should read Section 4 for examples of classroom usage and may wish to read Section 3 for an understanding of the methodology.
2. For researchers, this paper presents OpenAeroStruct as a physics-based multidisciplinary model that is computationally inexpensive and uses the adjoint method to compute the coupled derivatives. This yields scalability that makes OpenAeroStruct useful for benchmarking in MDO research (Cook et al. 2017; Friedman et al. 2017; Bons et al. 2017; Chauhan et al. 2017), where it has been used to test solvers, MDO architectures, uncertainty quantification (UQ) methods, etc. Researchers should see Section 5 for examples of research insights gained from OpenAeroStruct.

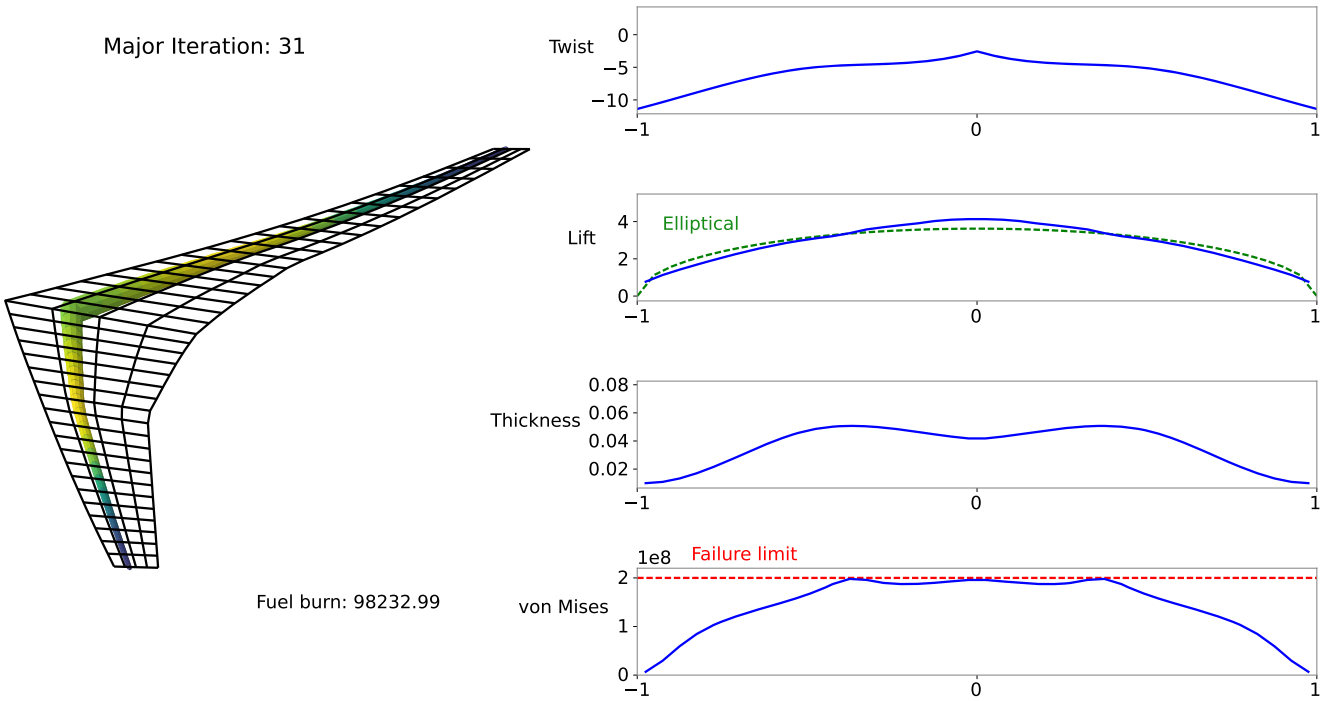

Fig. 1 Screenshot of interactive visualization tool included in OpenAeroStruct. Users can explore the optimization history by stepping through each iteration and examining the wing model, design variables, and output function values

Researchers may also be interested in Section 3, which explains the underlying system setup and capabilities.

3. For students, this paper serves as a reference for the aerodynamic and structural theory of OpenAeroStruct. We document the equations for the VLM and FEA analyzes in the model, the equations describing the consistent and conservative load and displacement transfer schemes, and the numerical solvers. Detailed code documentation, examples, and tutorials can be found in the online documentation linked in the GitHub repository. Students will find the background theory in Section 2 and the details of the coupled system setup in Section 3 helpful.

This paper is organized as follows. First, we detail the existing theory and our implementation for the aerodynamic and structural models (Section 2). Then we explain the coupled aerostructural solver implementation and the optimization problem formulation in Section 3. In Section 4, we present results from research applications of OpenAeroStruct and from its use in graduate courses. We review the research and educational outcomes achieved to date using OpenAeroStruct in Section 5.

2 Background

2.1 Motivation

In aircraft wing design, aerodynamics and structures are two of the most important disciplines. Increasing span reduces aerodynamic drag but increases structural weight, so a well-designed wing needs to balance these two trends. Changing the twist distribution to concentrate the lift inboard may hurt aerodynamic efficiency but improves structural efficiency. For wing design, it is important to account for the aerostructural coupling because the aerodynamic loads affect the structural deflections, which in turn affect the aerodynamic loads. The true deflected shape of the wing is thus different from the shape found by applying the aerodynamics loads computed based on the undeflected shape. In addition to finding the true shape of the wing, we must also consider design tradeoffs between wing shape and structural sizing. This is a quintessential multidisciplinary design problem that motivated the development of MDO in both low-fidelity (Haftka 1977) and high-fidelity studies (Kenway and Martins 2014; Liem et al. 2015).

The interactions between aerodynamics and structures can be captured with inexpensive physics-based models, such as the VLM for aerodynamics and 1-D FEA for structures. A coupled 1-D VLM–FEA model enables optimization of the spanwise distribution of the aerodynamic twist, planform, and structural thicknesses. However, computationally efficient aerodynamic and aerostructural optimization software is not freely available, even for low-fidelity models.

Existing free low-fidelity aerodynamic analysis tools have limited capabilities. XFOIL (Drela 1989) is restricted to 2-D problems, Tornado (Melin 2000) does not compute derivatives, and AVL (Drela and Youngren 2004) does not natively interface well with optimizers. Similarly, freely available tools for structural analysis, such as Frame3DD (Gavin 2010) and Calculix (Dhondt and Wittig 1998), do not provide derivatives and interface poorly with optimizers. High-fidelity tools for aerostructural design optimization have been developed, but they require the use of parallel high-performance computing platforms (Kenway et al. 2014; Kennedy and Martins 2014). Although software exists for lower fidelity aerodynamic or structural analysis alone, no open-source tool is able to perform coupled aerostructural analysis and design optimization.

In the gradient-based optimization of multidisciplinary systems, computing the coupled derivatives efficiently and accurately is critical and requires significant effort (Kenway et al. 2014). Gradient-based optimization scales well (often linearly) with the number of design variables because it uses the gradient information to find an efficient path from the initial point to the optimum. The adjoint method computes the gradient in a given optimization iteration at a cost that is comparable to that of the analysis, and that does not increase with the number of design variables. However, implementing the adjoint method is time- and effort-intensive because it requires modification of the source code for a model, in contrast to methods such as finite differences that treat the model as a black box.

We designed OpenAeroStruct as an MDO model that uses gradient-based optimization to solve a physically meaningful problem. The code is entirely open-source, so users can examine exactly what it is doing and expand its capabilities by building on the existing components. Analytic gradients are provided for each analysis component to enable efficient gradient-based optimization through the adjoint method, which has not previously been done for open-source wing design tools. The code is written using the OpenMDAO framework, so users can quickly and easily change the optimization formulation, the optimizer, and the system solvers. The fully functioning pure Python implementation means that the code can be used without worrying about dependencies or complicated installation procedures; OpenAeroStruct works on macOS, Windows, and Linux.

2.2 Aerodynamics model

The aerodynamics model uses a VLM (Anderson 1991) to compute the aerodynamic loads acting on the lifting surfaces. This combines multiple modern numerical

lifting-line theory (LLT) models, as described by Phillips and Snyder (2000). The VLM model is more general than the LLT model because it models low aspect ratio wings, swept wings, and delta wings more accurately (Anderson 1991). The theory of VLM is well established, and we create and use our own implementation so that we can easily obtain the relevant derivatives.

To set up the model, we follow the process outlined by Anderson (1991). A complete derivation of the theory is not provided here, but we summarize the important points below. Given a structured mesh defining a lifting surface, we can compute the aerodynamic properties by examining the circulation distribution. We do this by modeling the lifting surface using horseshoe vortices to represent the vortex system of a wing. Each horseshoe vortex consists of a bound vortex in the spanwise direction and two trailing vortices that extend into the freestream direction. Figure 2 shows a single horseshoe vortex on a lifting surface. We now examine the mathematical formulation used to calculate the wing circulation.

A vortex filament induces a flow field in the surrounding space. The strength of a vortex filament is its circulation, which produces lift on a surface. The Biot–Savart law relates the velocity of the flow field at an arbitrary point P caused by a segment $d\mathbf{l}$ of a vortex filament with circulation strength Γ via

$$d\mathbf{V} = \frac{\Gamma}{4\pi}\frac{d\mathbf{l}\times\mathbf{r}}{|\mathbf{r}|^3}.$$

Integrating over a semi-infinite straight vortex filament, we obtain

$$V = \frac{\Gamma}{4\pi h},$$

where h is the distance from point P to the finite start point of the vortex filament.

Helmholtz's vortex theorems (Anderson 1991) state that:

1. The strength of a vortex filament is constant along its length.
2. A vortex filament cannot end in a fluid; it must extend to the boundaries of the fluid (which can be $\pm\infty$) or form a closed path.

As Fig. 2 illustrates, a horseshoe vortex is comprised of semi-infinite vortices extending from point b to point a and from c to d, and a bound vortex extending from b to c. The circulations for each of these vortex segments are of equal magnitude and have consistent directions.

To model a lifting surface, we superimpose multiple horseshoe vortices over the span. This corresponds to a Weissinger LLT model, where there is only one horseshoe element in the chordwise direction. VLM is the Weissinger LLT method extended to allow multiple sets of horseshoe vortices in the chordwise direction.

The panel in Fig. 2 from the mesh of the lifting surface corresponds to the dashed lines, while the horseshoe vortex associated with that panel is drawn with solid lines. For a panel of length l, the bound vortex is at a distance of $\frac{1}{4}l$ from the front of the panel. Additionally, we define a control point at the centerline of the panel $\frac{3}{4}l$ from the front of the panel. We enforce a flow tangency condition at this control point that specifies that the velocity normal to the panel must be zero. Physically, this means that flow cannot go through the lifting surface. By imposing flow tangency conditions at each of the control points for all of the horseshoe vortices on the lifting surface, we obtain the linear system

$$\mathbf{A}\boldsymbol{\Gamma} = -\mathbf{V}_\infty \cdot \mathbf{n},$$

where $\mathbf{A}$ is the aerodynamic influence coefficients matrix, $\mathbf{V}_\infty$ is the freestream velocity, and $\mathbf{n}$ is the normal to the panel. We can solve this linear system to obtain the circulation strengths for each of the horseshoe vortices.

Now that we have the circulation strengths of the vortices, we can compute the aerodynamic forces acting on each individual panel using

$$\mathbf{F}_i = \rho\Gamma_i(\mathbf{V}_\infty + \mathbf{v}_i)\times\mathbf{l}_i,$$

where $\mathbf{v}_i$ is the induced velocity at the center of the bound vortex, and $\mathbf{l}_i$ is the bound vortex vector. The length and

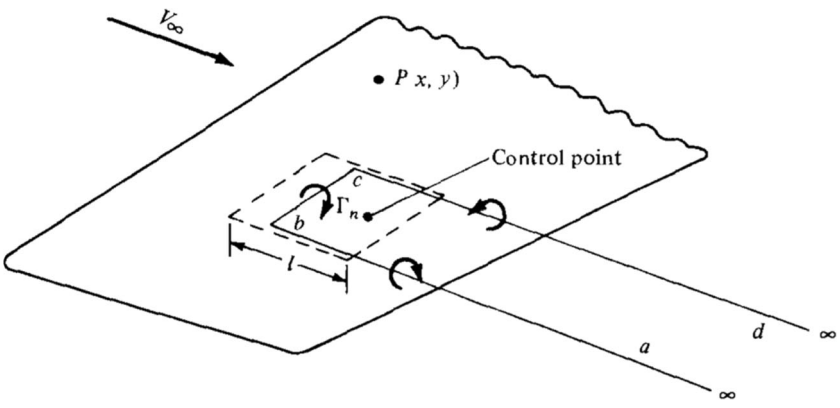

Fig. 2 Single horseshoe on a lifting surface (Anderson 1991)

direction of the bound vortex directly influence the forces and allow for different width horseshoe vortices across the span.

With the sectional panel forces we can compute the lift, drag, and other lifting surface metrics. The lift and drag correspond to the components of the force vector in the upward and freestream directions, respectively. We also compute the skin friction drag using flat-plate-based estimates (Raymer 2012, Sec. 12.5.3). This skin friction drag estimate is based on the airfoil thickness-to-chord ratio, the Reynolds number, and other aircraft and flow properties. The drag estimate is adjusted using a form factor, which accounts for pressure drag due to flow separation. The semi-empirical models used for drag estimation are considered valid up to the drag-divergence Mach number.

2.3 Structural model

For the structural model, we use a finite element method (FEM) approach that uses spatial beam elements, resulting in six DOFs per node. The spatial beam element is a combination of truss, beam, and torsion elements, which means that it simultaneously carries axial, bending, and torsional loads. We implement our own structural model using well-established theory so we can easily obtain the relevant derivatives.

Each spatial beam element has 12 DOFs in total, as shown in Fig. 3. At each end of the beam, there are three translational displacements in the x, y, and z directions, and three rotational DOFs with respect to the x, y, and z axes. The displacements and rotations shown in Fig. 3 are in the local coordinate frame of the element. The axial DOFs are u_1 and u_2; the z-plane bending DOFs are v_1, v_2, α_{z1}, and α_{z2}; the y-plane bending DOFs are w_1, w_2, α_{y1}, and α_{y2}; and the torsion DOFs are α_{x1} and α_{x2}. During the assembly of the global stiffness matrix, transformation matrices are used to convert the element stiffness matrix from the local frame to the global frame aligned with the x, y, and z axes. The stiffness matrix for a single element is given by

$$
[k]_e = \begin{bmatrix}
k_1 & 0 & 0 & 0 & 0 & 0 & -k_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 12k_2^z & 0 & 0 & 0 & 6k_2^z l & 0 & -12k_2^z & 0 & 0 & 0 & 6k_2^z l \\
0 & 0 & 12k_2^y & 0 & -6k_2^y l & 0 & 0 & 0 & -12k_2^y & 0 & -6k_2^y l & 0 \\
0 & 0 & 0 & k_3 & 0 & 0 & 0 & 0 & 0 & -k_3 & 0 & 0 \\
0 & 0 & -6k_2^y l & 0 & 4k_2^y l^2 & 0 & 0 & 0 & 6k_2^y l & 0 & 2k_2^y l^2 & 0 \\
0 & 6k_2^z l & 0 & 0 & 0 & 4k_2^z l^2 & 0 & -6k_2^z l & 0 & 0 & 0 & 2k_2^z l^2 \\
-k_1 & 0 & 0 & 0 & 0 & 0 & k_1 & 0 & 0 & 0 & 0 & 0 \\
0 & -12k_2^z & 0 & 0 & 0 & -6k_2^z l & 0 & 12k_2^z & 0 & 0 & 0 & -6k_2^z l \\
0 & 0 & -12k_2^y & 0 & 6k_2^y l & 0 & 0 & 0 & 12k_2^y & 0 & 6k_2^y l & 0 \\
0 & 0 & 0 & -k_3 & 0 & 0 & 0 & 0 & 0 & k_3 & 0 & 0 \\
0 & 0 & -6k_2^y l & 0 & 2k_2^y l^2 & 0 & 0 & 0 & 6k_2^y l & 0 & 4k_2^y l^2 & 0 \\
0 & 6k_2^z l & 0 & 0 & 0 & 2k_2^z l^2 & 0 & -6k_2^z l & 0 & 0 & 0 & 4k_2^z l^2
\end{bmatrix},
$$

where

$$
k_1 = \frac{EA}{L},\ k_2^z = \frac{EI_z}{L^3},\ k_2^y = \frac{EI_y}{L^3},\ k_3 = \frac{GJ}{L},
$$

and E is the Young's modulus, A is the beam cross-sectional area, L is the beam length, G is the shear modulus, J is the polar moment of inertia, and the Is are the second moments of area about the three local coordinate directions.

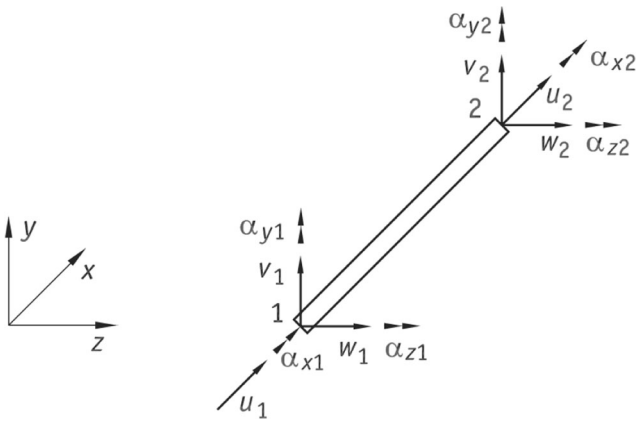

Fig. 3 6-DOF spatial beam element (Pesare 2016)

In OpenAeroStruct, spatial beam elements are always connected end-to-end in a single sequence, so the resulting global stiffness matrix exhibits a banded structure where stiffness submatrices are added on block diagonals. Once the stiffness matrix has been assembled, OpenAeroStruct solves the linear system $\mathbf{Ku} = \mathbf{f}$, where $\mathbf{K}$ is the global stiffness matrix, $\mathbf{u}$ is the vector of displacements and rotations at the nodes, and $\mathbf{f}$ are the forces and moments acting at the nodes.

3 Methodology

The aerodynamic and structural models described in the previous section are well known. In this section, we describe

our implementation of the coupled solver derivative computation for these two disciplines. The implementation relies on the OpenMDAO, so we include a brief overview of this framework. Then we explain the load and displacement coupling as well as the implementation of the aerostructural system. Finally, we detail the aerostructural optimization formulation.

3.1 OpenMDAO framework

We first provide an overview of the OpenMDAO framework since it is the computational modeling software within which OpenAeroStruct is built. OpenMDAO is an open-source NASA-developed framework for multidisciplinary design, analysis, and optimization with a focus on gradient-based approaches (Heath and Gray 2012). It is unique because it facilitates the efficient computation of the derivatives needed for optimization using methods such as the adjoint method.

As with other software frameworks, its primary function is to enable the modular construction of computational models, where a larger, more complex model is decomposed into smaller units of code, called components, that are simpler and more portable. The components are partitioned into *groups* that can in turn be part of other groups, forming a hierarchy tree. OpenMDAO performs data transfers between components, potentially across processors in parallel settings, and includes nonlinear and linear solvers with support for matrix-free or sparse linear algebra. Any nonlinear or linear solver can be applied in each group or component in the hierarchy tree. This means that OpenMDAO in general uses hierarchical solution approaches: e.g., a Newton solver can be used in a component while the group containing that component can use a nonlinear block Gauss–Seidel solver that calls its children in sequence, including the component with the Newton solver.

A unique and extremely useful feature of OpenMDAO is its ability to compute derivatives using a wide range of methods including the adjoint method. It does this via the MAUD architecture (Hwang and Martins 2018), which enables the computation of total (model) derivatives using a unified equation (Martins and Hwang 2013). This matrix equation unifies the chain rule, the direct method, the adjoint method, a coupled form of the chain rule, and hybrid methods. For OpenAeroStruct, the significance of the unification is that our task is limited to computing the partial derivatives of each component's outputs or residuals with respect to its inputs. If we then specify the appropriate linear solvers, OpenMDAO solves the unified equation to compute the needed derivatives. This amounts to the adjoint method if we set up a purely aerodynamic or purely structural OpenAeroStruct model, or the coupled adjoint method if we include both disciplines (Martins et al. 2005; Kenway et al. 2014). If, in a different setting, we set up a model that has no states and residuals, solving the unified equation would be equivalent to applying the chain rule to assemble the partial derivatives to form total derivatives.

3.2 Load and displacement transfer

In OpenAeroStruct, the load and displacement transfer is simplified by the assumption that the same spanwise discretization is used for the aerodynamic and structural models. The nodes of the structural mesh and the spanwise sections of the aerodynamic mesh are computed from the same wireframe mesh used for the wing. The structural nodes have the same spanwise discretization as the wireframe mesh and are placed a distance from the leading edge in the chordwise direction based on a specified percentage of the local chord. Likewise, each spanwise section of the aerodynamic mesh is obtained by uniformly splitting the corresponding section of the overall wireframe mesh into the desired number of edges.

The load and displacement transfer scheme used in OpenAeroStruct satisfies the requirements of being *consistent* and *conservative* (Martins et al. 2005). Consistency states that the sum of the nodal forces and moments obtained via the load transfer must be equal to the forces and moment resultants computed from the continuous pressure and shear force distributions on the element. Since the resultants are computed by integrating over a region, an infinite number of choices of nodal forces and moments provide consistency. Conservativeness states that the virtual work done by the forces over virtual displacements on the aerodynamic and structural meshes are equal. We now explain the load and displacement transfer schemes and then show that they are consistent and conservative.

For the load transfer, our objective is to transfer the traction $\mathbf{T}$ from each panel to the structural nodes, as shown in Fig. 4. The traction is assumed to be uniformly distributed over the panel, where the net force is equal to that computed by the VLM. Since the edges of the panels are aligned with the structural nodes, half of the traction on the panel is applied to each of the two structural nodes. The nodal force and moment vectors on either the left or right structural node are given by

$$\mathbf{F}_{s,i} = \int_{\text{panel}} \frac{1}{2}\mathbf{T} dS = \frac{1}{2}\mathbf{T}S \tag{1}$$

$$\mathbf{M}_{s,i} = \int_{\text{panel}} \mathbf{r}_i \times \frac{1}{2}\mathbf{T} dS = \frac{1}{2}\mathbf{r}_{cp,i} \times \mathbf{T}S, \tag{2}$$

where the subscript s indicates that the quantity is on the structural mesh, $i = 1, 2$ indicates the left or right node, S is the panel area, $\mathbf{r}_i$ is the vector pointing from the structural node to a point on the panel, and $\mathbf{r}_{cp,i}$ points from the

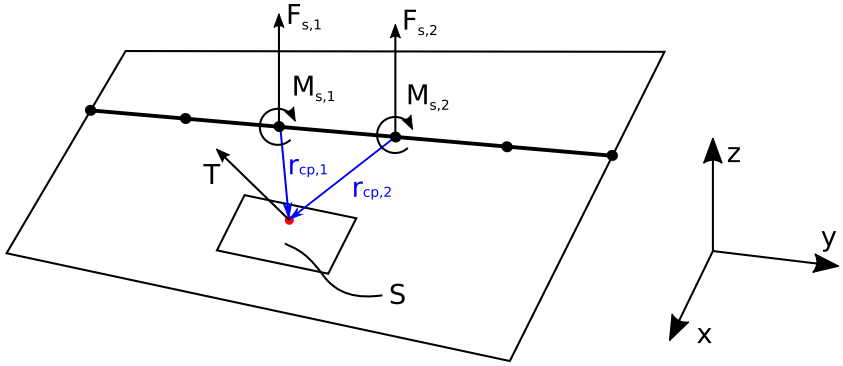

Fig. 4 Load transfer from a panel on the aerodynamic mesh to adjacent structural nodes. Since the force is assumed to be uniformly distributed on the panel, we can compute the resultant moments on the nodes using the vectors pointing to the aerodynamic centers of pressure

structural node to the aerodynamic center of pressure for the panel. We consider the center of pressure to be at the middle of the panel in the spanwise direction and a quarter-length of the panel from the front in the chordwise direction. With the $\mathbf{F}_{s,i}$ and $\mathbf{M}_{s,i}$ obtained using this load transfer scheme, consistency is satisfied by construction because the nodal forces and moments are defined to be the equivalent resultants from the traction field.

For the displacement transfer, our goal is to transfer the computed displacements on the structural mesh, $\mathbf{u}_{s,i}$, to deflections on the aerodynamic mesh, $\mathbf{u}_a$. The structural displacements are partitioned into translational displacements, $\mathbf{d}_{s,i}$, and rotations, $\boldsymbol{\theta}_{s,i}$. The displacement transfer is given by

$$\mathbf{u}_a = \frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{d}_{s,i} + \boldsymbol{\theta}_{s,i} \times \mathbf{r}_i\right), \tag{3}$$

where again, $\mathbf{r}_i$ is a vector pointing from the left or right structural node to the point on the aerodynamic mesh corresponding to $\mathbf{u}_a$. A constant factor of one-half is used, since we average the contributions from the left and right structural nodes no matter where we are in the panel. We can do this since we evaluate the aerodynamic mesh only at the midpoint in the spanwise direction.

We now verify that our load and displacement transfer scheme is conservative. The virtual work done on the structural mesh by the nodal forces and moments corresponding to a panel is

$$\delta W_s = \sum_{i=1}^{2}\left(\mathbf{F}_{s,i} \cdot \delta\mathbf{d}_{s,i} + \mathbf{M}_{s,i} \cdot \delta\boldsymbol{\theta}_{s,i}\right). \tag{4}$$

Inserting $\mathbf{F}_{s,i} = \frac{1}{2}\mathbf{T}S$ and $\mathbf{M}_{s,i} = \frac{1}{2}\mathbf{r}_{cp,i} \times \mathbf{T}S$, we obtain

$$\delta W_s = \frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{T} \cdot \delta\mathbf{d}_{s,i} + \mathbf{r}_{cp,i} \times \mathbf{T} \cdot \delta\boldsymbol{\theta}_{s,i}\right) S. \tag{5}$$

The virtual work done on the aerodynamic mesh by the traction is

$$\delta W_a = \int_{\text{panel}} \mathbf{T} \cdot \delta\mathbf{u}_a dS. \tag{6}$$

Inserting $\mathbf{u}_a$ from (3), we obtain

$$\delta W_a = \frac{1}{2}\sum_{i=1}^{2}\int\left(\mathbf{T} \cdot \delta\mathbf{d}_{s,i} + \mathbf{T} \cdot \delta\boldsymbol{\theta}_{s,i} \times \mathbf{r}_i\right) dS. \tag{7}$$

Since $\mathbf{T}$, $\mathbf{d}_{s,i}$, and $\boldsymbol{\theta}_{s,i}$ are constant over the panel, the integration yields

$$\delta W_a = \frac{1}{2}\sum_{i=1}^{2}\left(\mathbf{T} \cdot \delta\mathbf{d}_{s,i} + \mathbf{T} \cdot \delta\boldsymbol{\theta}_{s,i} \times \mathbf{r}_{cp,i}\right) S. \tag{8}$$

By vector algebra, we have

$$\begin{aligned}\mathbf{T} \cdot \delta\boldsymbol{\theta}_{s,i} \times \mathbf{r}_{cp,i} &= \delta\boldsymbol{\theta}_{s,i} \times \mathbf{r}_{cp,i} \cdot \mathbf{T} \\ &= \delta\boldsymbol{\theta}_{s,i} \cdot \mathbf{r}_{cp,i} \times \mathbf{T} = \mathbf{r}_{cp,i} \times \mathbf{T} \cdot \delta\boldsymbol{\theta}_{s,i}.\end{aligned} \tag{9}$$

Therefore, from (5), (8), and (9), we conclude that $\delta W_a = \delta W_s$, which proves that our load and displacement transfer scheme is conservative.

3.3 Aerostructural system

We now couple the aerodynamic and structural systems described above to solve the aerostructural system. We can think of the aerodynamics and structures as two separate groups that receive inputs and produce outputs. The aerodynamics group receives a mesh and outputs aerodynamic loads, whereas the structural group receives aerodynamic loads and outputs structural displacements.

The default setting within OpenAeroStruct uses Gauss–Seidel fixed-point iterations to converge the multidisciplinary analysis (MDA). This means that each analysis is run using the most recent output from the other analysis until a consistent set of state variables is returned. However, it is possible to use Newton's method to converge the coupled aerostructural system: the same partial derivatives used in implementing the adjoint method are used in computing the Newton step at every iteration.

One important feature of OpenMDAO is the ability to subdivide a problem into components that have a small number of inputs and outputs and contain relatively simple analyzes. An advantage of this decomposition is that the component-level partial derivatives are simpler to

symbolically differentiate if they are not done with AD. We can reorganize the components within the system, and OpenMDAO automatically computes the correct total coupled derivatives using the MAUD architecture (Hwang and Martins 2018). Additionally, we can simply add a few lines of code to obtain derivatives with respect to a new output.

Now that we have presented the underlying theory for the aerodynamic and structural models, we examine how the internal components within OpenAeroStruct pass data; see Fig. 5. The *prob_vars* component contains information about the airflow around the lifting surfaces. Just below this, we have the *wing* group, which contains geometric information about the lifting surface and structural spar. There is one such group for each lifting surface, with a user-defined name (the default name is *wing*).

We then have a *coupled* group that solves for the aerodynamic and structural state values. Within this *coupled* group, the *aero_states* group assembles the aerodynamic influence coefficient (AIC) matrix and computes the aerodynamic loads acting on each panel of the lifting surfaces. There is only one *aero_states* group no matter how many lifting surfaces are defined by the user. The *struct_states* group computes the structural displacements based on the aerodynamic loads.

Next, there is a *wing_perf* group, which calculates the lift and drag performance of the lifting surface. Again, there is one *_perf* group for each lifting surface defined by the user. Lastly, the *total_perf* group calculates aerostructural performance metrics for the entire aircraft, such as fuel burn and coefficient of moment.

We now offer some ideas for modifications to the model. Currently the structural mesh and aerodynamic mesh must have the same spanwise discretization, but it is possible to use interpolation to allow different spanwise discretizations. To do so, a consistent and conservative load and displacement transfer method must be used. Additionally, a different FEA model could be used in place of the tubular spar. A past user of OpenAeroStruct implemented a structural model that computes the cross-sectional properties of a wingbox to obtain more realistic wing deflection properties. Because of the code modularity, the user was able to do this cleanly and quickly by replacing the existing tubular spar with his wingbox model.

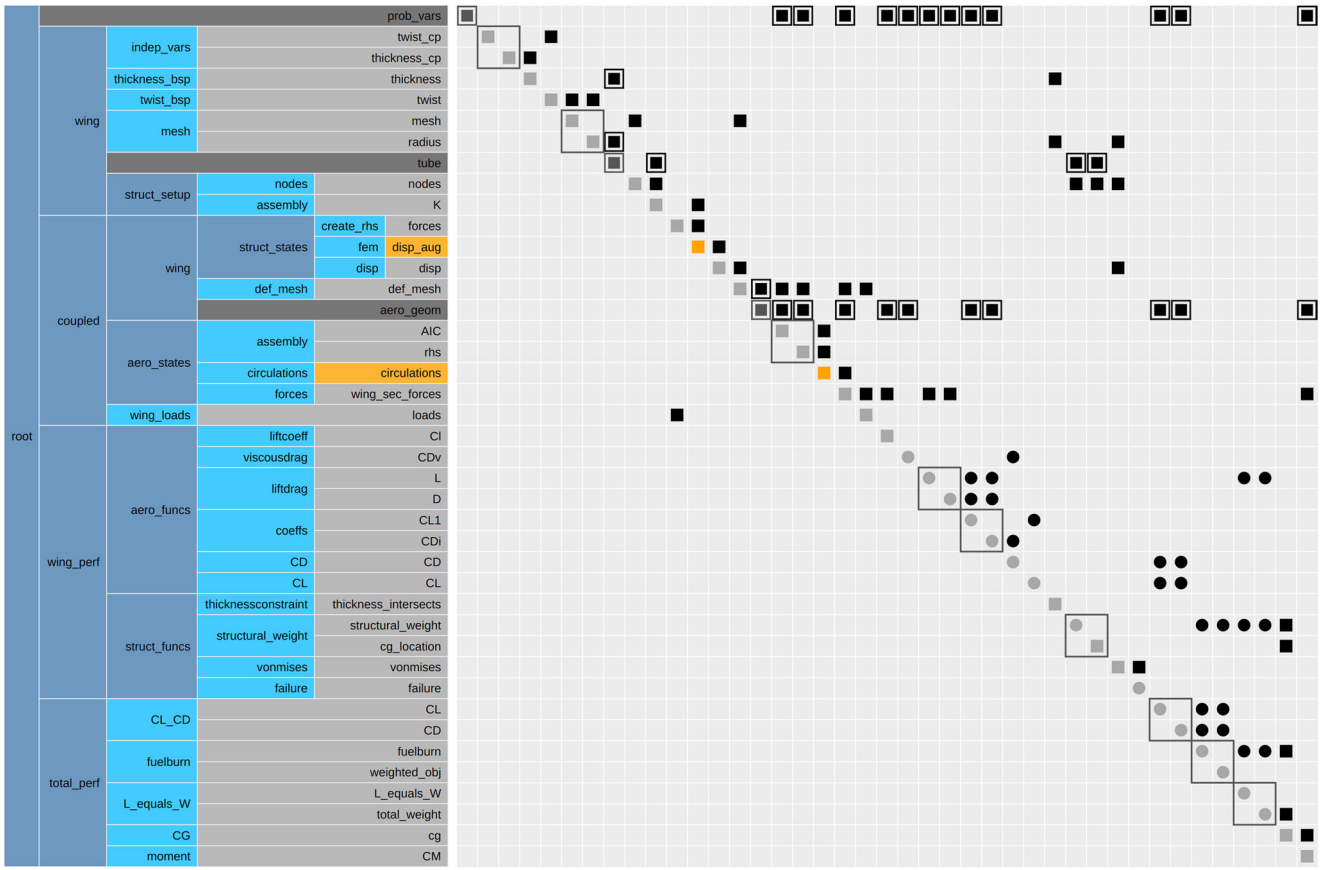

Fig. 5 Component layout for aerostructural analysis and optimization; this hierarchy and data-passing diagram is automatically produced by OpenMDAO for debugging and educational purposes

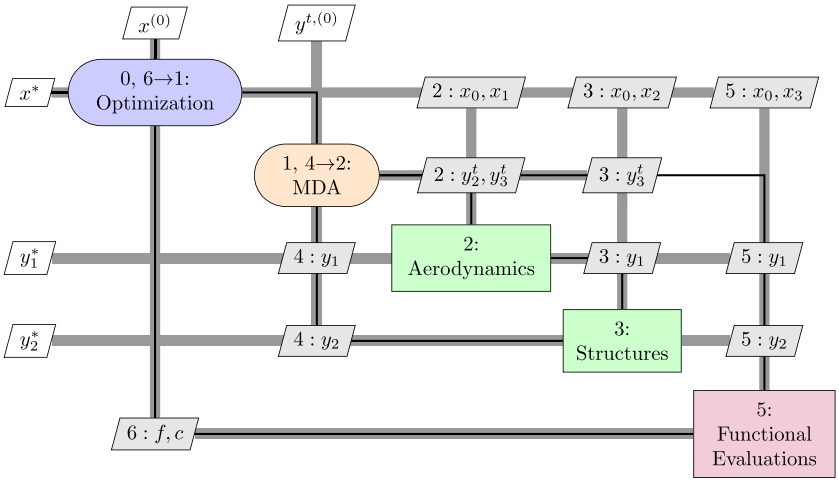

Fig. 6 XDSM diagram (Lambe and Martins 2012) for default aerostructural optimization. The x vectors are design variables and the y vectors are states, where * represents the values at the design optimum. The default solver for the MDA is Gauss–Seidel within OpenAeroStruct, shown here

3.4 Aerostructural optimization

For aerostructural optimization, we can use any combination of the previously discussed design variables. We also have two aerostructural outputs: fuel burn computed via the Breguet range equation, and a constraint ensuring that lift equals weight. The Breguet range equation is given by

$$W_f = (W_0 + W_s)\left[\exp\left(\frac{R \cdot \mathrm{SFC}}{V}\left(\frac{L}{D}\right)^{-1}\right) - 1\right],$$

where W_f is the fuel weight, W_0 is the aircraft empty weight, W_s is the structural weight, R is the range, V is the velocity, and SFC is the specific fuel consumption.

Users can easily add their own objective functions or constraints by creating an OpenMDAO component that computes the desired quantities. Because the total derivatives are computed using the unified chain rule and the adjoint method, the user simply needs to supply partial derivatives for the calculations done in the new component, which makes introducing new outputs a straightforward process. Since the software is open-source, these components can be added to the main repository to expand the capabilities of OpenAeroStruct. Figure 6 shows the XDSM diagram corresponding to aerostructural optimization. Here, the input and output vectors, x and y, respectively, vary depending on the design variables, objectives, and constraints selected.

OpenMDAO can use a number of optimization algorithms through the pyOptSparse interface (Perez et al. 2012).[2] The optimization algorithm used to solve the aerostructural design optimization problem is SNOPT, a sequential quadratic programming approach that efficiently solves large sparse nonlinear constrained optimization problems (Gill et al. 2002).

4 Education

OpenAeroStruct has been used in courses at ISAE-SUPAERO (the University of Toulouse's Institute for Aerospace Engineering) and the University of Michigan. These were graduate-level MDO courses taken by students whose previous exposure to Python ranged from none to using it in their daily research. Most of the students had experience only with Matlab or C/C++, but they were able to easily use OpenAeroStruct because of its simple run scripts and detailed documentation. The course prerequisites were not demanding: basic calculus, linear algebra, and programming skills. The students appreciated being able to test theories and hypotheses quickly by running tightly coupled aerostructural optimizations in minutes on their personal computers.

4.1 ISAE-SUPAERO

At ISAE-SUPAERO, the students used OpenAeroStruct to learn about structural optimization, MDA, and MDO over the span of a few hours. They worked in small groups and examined the effects of mesh size, algorithmic options, and component grouping on the overall optimization performance. The students were given problem sets instructing them to run the existing OpenAeroStruct model in various configurations and draw conclusions based on the results. To aid in their understanding of the different models and the problem hierarchy, we developed an interactive program to visualize the optimization history and wing model. A sample visualization for an optimized aerostructural model was shown in Fig. 1. In both years in which OpenAeroStruct was used, the problem sets consisted

[2] https://bitbucket.org/mdolab/pyoptsparse

of a progression from structural optimization to MDA and finally to MDO.

In the structural optimization section, the students interpreted the optimization problem, discussed the optimal thickness distributions, and looked at the effect of increasing the mesh size. The goal here was to have the students work with a simple hands-on numerical optimization problem that they could interpret and evaluate using their knowledge of physics.

In the MDA section, the students ran aerostructural analyzes, assigning their own nonlinear and linear solvers to converge the coupled systems. They were instructed to compare fixed-point iteration (nonlinear block Gauss–Seidel), Newton, and a hybrid approach where fixed-point iteration is used as a start-up strategy for the Newton iterations, effectively providing globalization. They were also instructed to look at various linear solvers for Newton's method, including direct methods, linear block Gauss–Seidel, Krylov subspace methods, and Krylov subspace methods with preconditioning.

In the MDO section, the students performed aerostructural optimization. They were instructed to compare the effect of the derivative computation method (finite differences and the adjoint method) on the computation time and the number of optimization iterations. They interpreted the optimized thickness and twist distributions, and they were encouraged to experiment with different solver choices and options to minimize the total optimization time.

A theme throughout the assignments was the emphasis on the benefits of modularity, especially when it comes to gradient computation using the adjoint method. Using OpenMDAO's model structure visualization, shown earlier in Fig. 5, the students were able to see how the overall OpenAeroStruct model decomposes into geometry, aerodynamics, structures, load and displacement transfer, and performance. In fact, the students were asked to add their own component in the MDO section. The aerostructural optimization was given to them as a fuel-burn minimization problem, but they were instructed to write a new component, one that computes range for a given fuel burn, enabling them to reformulate the fuel-burn minimization for a fixed range into a range maximization for a fixed amount of fuel. They were responsible for deriving and computing the partial derivatives of their new component, but, as they saw first-hand, the modular approach can update the total derivative computation without any additional effort from the user.

4.2 University of Michigan

OpenAeroStruct was also used as a basis for the final project in the MDO course at the University of Michigan. The students worked individually to set up an aircraft of their choice within OpenAeroStruct and performed aerodynamic, structural, and aerostructural analysis and optimization. They produced Pareto fronts of the fuel burn and aircraft weight, and they examined how sequential optimization compared to the multidisciplinary design feasible (MDF) architecture (Cramer et al. 1994; Martins and Lambe 2013). The students evaluated aircraft ranging from small-scale UAVs to the AN-225 and compared the results from OpenAeroStruct with real-world values. This allowed them to combine their aircraft design knowledge with hands-on experience of optimization methods, leading to a more intuitive understanding of MDO. Previous assignments in this course asked the students to implement optimization methods from scratch. Using OpenAeroStruct allowed them to explore different multidisciplinary solvers and design variables much more rapidly.

Figure 7 appeared in a student's final report. Because OpenAeroStruct uses OpenMDAO, the students can easily change the nonlinear solver used to converge the coupled aerostructural system to investigate convergence trends and computational costs, as shown in Fig. 7. Here, the student compares the time to converge the coupled system using different nonlinear solvers for decreasing spar thicknesses, which increases wing flexibility and thus coupling strength. Because of the short wall times, the students can perform many aerostructural optimizations with different formulations to compare relative aircraft performance. We see that NLBGS generally requires less time to solve the coupled system, but it cannot converge the most strongly coupled systems that Newton can handle. As

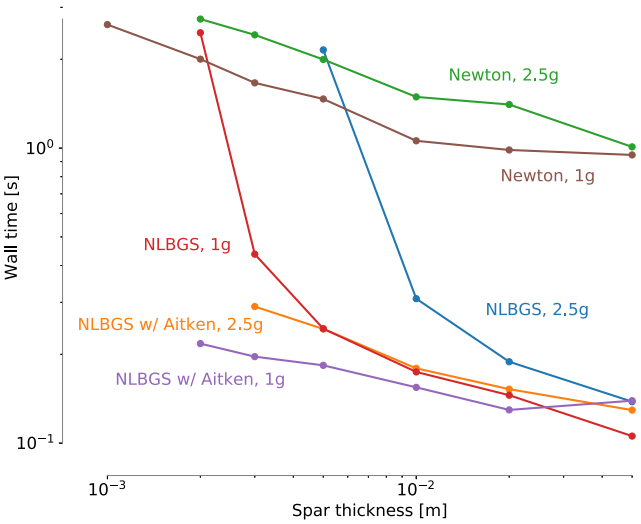

Fig. 7 Comparison of different solvers for the solutions of the coupled aerostructural system for level flight (1 g) and a pull-up maneuver (2.5 g). As spar thickness decreases, the system becomes more strongly coupled, and nonlinear block Gauss–Seidel (NLBGS) without Aitken relaxation cannot converge the coupled system as well as Newton's method (Chauhan et al. 2017)

the spar thickness decreases, the coupling strength decreases and all solvers require less time to converge the system.

For the MDO course project, the first author compared designs produced by sequential optimization and MDO for a small-scale flying-wing unmanned aerial vehicle. Figure 8 shows that the Pareto front for the MDO optima always dominates the sequential optimization results. For the sequential optimization, we follow the method presented by Chittick and Martins (2008). First, we size the structure based on fixed aerodynamic loads to obtain a structural weight estimate, and then we calculate the optimal twist distribution through an aerodynamic optimization based on the fixed structural weight. We repeat this process until the results converge. We run sequential optimizations at multiple fixed span values to obtain the different points on the Pareto front. The optimizer in the MDO approach can control span, twist, and structural spar thickness.

5 Research applications

OpenAeroStruct has also proved useful for research and has been used as a realistic testbed application by Bons et al. (2017), Cook et al. (2017), and Friedman et al. (2017). Because all of the internal analysis and derivative computations are exposed to the user, invasive methods can be implemented, including advanced methods for robust optimization, reliability-based design optimization, and conditional value-at-risk optimization.

Cook et al. (2017) used OpenAeroStruct to benchmark a new method for optimization under uncertainty called horsetail matching, which is a flexible approach to optimization under a mix of probabilistic and interval uncertainties. Horsetail matching minimizes the difference between the expected values of a quantity of interest and a desired target. This allows designers to perform robust optimization to try to minimize the chance of failure for the majority of operating cases. Cook et al. used OpenAeroStruct to easily and quickly compare their proposed method with other optimization-under-uncertainty methods using a physical problem representative of larger systems of interest. Compared to a deterministic optimization, their robust optimization resulted in a higher angle of attack and a less aggressive aerodynamic twist for better performance under uncertain conditions.

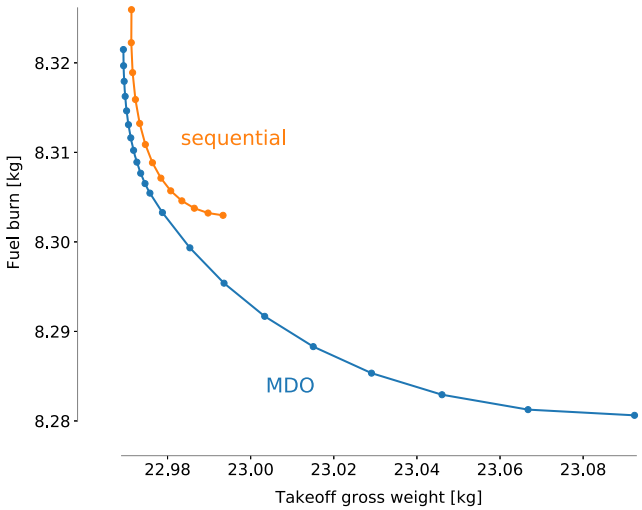

Fig. 8 The MDO-obtained Pareto front always dominates the sequential-optimization Pareto front

Friedman et al. (2017) used OpenAeroStruct as a test case to quantify the effects of model discrepancy (uncertainty associated with the fact that no model is perfect). Complex multidisciplinary systems often consist of multiple pre-existing physics-based models, which each have their own associated uncertainty. Friedman et al. compared different formulations of model discrepancy in coupled systems and performed a pattern search optimization to minimize the difference between each variable's marginal and conditional distributions. The coupled aerostructural model in OpenAeroStruct was evaluated thousands of times using a Gibbs sampler, which would not be tractable with a more expensive coupled model.

As another example of OpenAeroStruct's usefulness in research, we present an aerostructural optimization study that shows some of the fundamental tradeoffs in wing aerostructural design. Table 1 shows the problem formulation for the optimization problems in this study, where β is a fixed weighting parameter that combines the functions of interest. For each β value, the optimized result is the same regardless of the initial planform.

Figure 9 shows three optimized wing planform shapes and the corresponding structural thickness, twist, and lift distributions. The leftmost column shows the initial planform for the optimizations and the legend for the plots. The next three columns show the optimization results for different objective functions, where the leftmost is a fuel-burn minimization, the rightmost is a structural spar weight minimization, and the middle one corresponds to an equally weighted optimization. Because fuel burn and spar weight are of the same order and have the same units, we do not nondimensionalize the weighted objective function.

For the fuel burn minimization, the optimizer increases the span and decreases the root chord to produce a more aerodynamically efficient wing. Additionally, the optimized wing twist is positive for most of the wing, except at the tip. Aerodynamic considerations dominate the design of this specific wing, so the lowest fuel burn results in a lift distribution that is close to elliptical. The optimized thickness is greatest at the root and gradually decreases as we approach the tip, as expected. This optimization results in a lower total aircraft weight than that obtained from

Table 1 Sample aerostructural optimization problem formulation within OpenAeroStruct

	Function/variable	Description	Size
Minimize	$\beta \text{FB} + (1-\beta) W_{\text{struct}}$	Objective function	1
w.r.t.	Thickness	Structural spar thickness	n_{cp}
	Twist	Aerodynamic twist	n_{cp}
	α	Angle of attack	1
	Root chord	Root chord	1
	Taper	Taper ratio	1
Subject to	$L = W$	Lift equals weight	1
	$\text{KS}\left(\sigma_{2.5g}\right) \leq \sigma_{\text{yield}}$	Aggregated spar failure	1

FB stands for fuel burn. n_{cp} corresponds to the number of control points for the B-spline interpolation that controls the spanwise distribution of the variables. The numbers of thickness and twist control points do not necessarily need to be the same. The structural failure constraint is aggregated using a Kreisselmeier–Steinhauser (KS) function (Kreisselmeier and Steinhauser 1979; Lambe et al. 2017)

the structural spar weight minimization. When minimizing structural spar weight, the optimizer reduces the span and tries to minimize the thickness of the spar throughout. To avoid structural failure, it twists down the outboard section of the wing to unload the tip, as shown in the lift distribution. In this case, the optimizer does not consider the aerodynamic performance, except for its effect on the lift constraint.

We have observed these same trends when using high-fidelity analyzes in aerostructural optimization (Kenway and Martins 2014), but we can explore the design space much more quickly using OpenAeroStruct. This ability to rapidly explore the wing design space was especially valuable in the work of Bons et al. (2017), where OpenAeroStruct enabled the rapid exploration of multimodality in aerodynamic planform optimization, helped explain the physics of this multimodality, and provided promising starting points for much more costly design optimizations based on the solution of the Reynolds-averaged Navier–Stokes equations.

6 Conclusions

We have presented OpenAeroStruct, an educational open-source low-fidelity aerostructural analysis and optimization tool that uses NASA's OpenMDAO framework. Its modular implementation and efficient derivative computation make it a unique tool for teaching the adjoint method and solution methods for MDA and MDO.

OpenAeroStruct has already proved useful in both educational and research settings. It is straightforward to install and use because it is open-source and well-documented. Students can learn MDO techniques through realistic aircraft design problems while experimenting with optimization formulation and problem size. Additionally, it

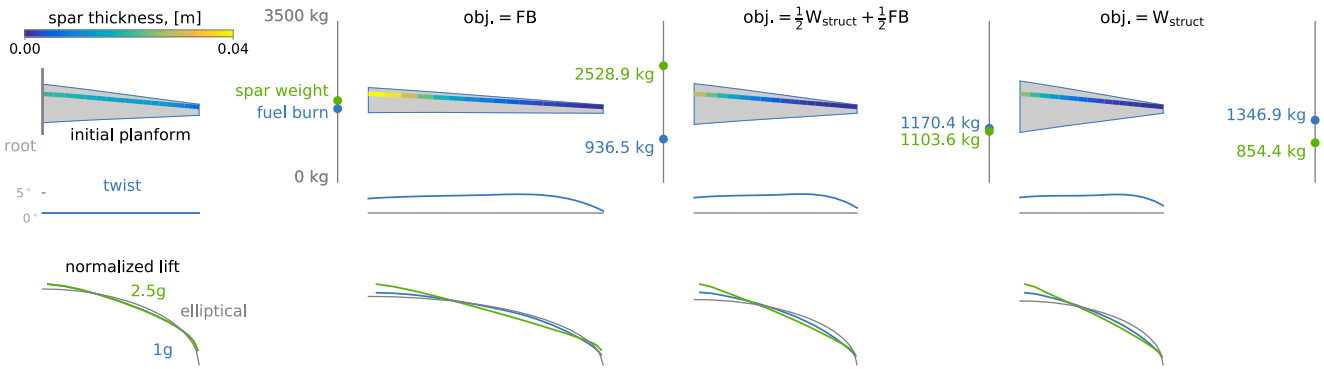

Fig. 9 Optimized planforms, twist, lift, and thickness distributions for three aerostructural optimizations with three different objective functions. The first column shows the initial shape and provides a legend for the plot. The second column is the result from a fuel burn minimization and the rightmost column corresponds to a structural weight minimization. The column between them shows the optimized result for an equally-weighted combined objective function. Each case took under a minute to run on a desktop computer. Multiple different starting points were tested, but each starting point converged to the same optimum for a given objective function, so we only show the results from one starting point here

produces simple MDO problems with variable levels of fidelity that allow state-of-the-art MDA solvers and uncertainty quantification methods to be tested quickly. Because OpenAeroStruct captures some of the same trends as high-fidelity analyzes, it can also be used to explore the design space before resorting to more computationally expensive methods for design refinement and better accuracy.

The modular nature of OpenAeroStruct encourages the addition of more features through collaboration. Going forward, we will expand OpenAeroStruct's capabilities for a variety of purposes, including stability-constrained UAV optimization, multifidelity optimization, and trajectory optimization.

Acknowledgements The authors are grateful for support from the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1256260 and from the AFOSR MURI on multi-information sources of multi-physics systems under Award Number FA9550-15-1-0038, program manager Jean-Luc Cambier. The authors would like to thank Shamsheer Chauhan for contributing his figures from the MDO course project, as well as Joseph Morlier and Nathalie Bartoli for their support in the ISAE-SUPAERO course.

References

Anderson JD (1991) Fundamentals of aerodynamics. McGraw–Hill, New York

Bons NP, He X, Mader CA, Martins JRRA (2017) Multimodality in aerodynamic wing design optimization. In: 18th AIAA/ISSMO multidisciplinary analysis and optimization conference

Chauhan SS, Hwang JT, Martins JRRA (2017) Benchmarking approaches for the multidisciplinary analysis of complex systems using a Taylor series-based scalable problem. In: 12th world congress on structural and multidisciplinary optimization. Braunschweig

Chittick IR, Martins JRRA (2008) Aero-structural optimization using adjoint coupled post-optimality sensitivities. Struct Multidiscip Optim 36:59–70. https://doi.org/10.1007/s00158-007-0200-9

Cook LW, Jarrett JP, Willcox KE (2017) Horsetail matching for optimization under probabilistic, interval and mixed uncertainties. In: 19th AIAA non-deterministic approaches conference, p 0590

Cramer EJ, Dennis JE, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization. SIAM J Optim 4(4):754–776

Dhondt G, Wittig K (1998) Calculix: a free software three-dimensional structural finite element program. MTU Aero Engines GmbH, Munich

Drela M (1989) XFOIL: an analysis and design system for low Reynolds number airfoils. In: Low Reynolds number aerodynamics. Springer, pp 1–12

Drela M, Youngren H (2004) Athena vortex lattice. Software Package Ver 3

Friedman S, Ghoreishi SF, Allaire DL (2017) Quantifying the impact of different model discrepancy formulations in coupled multidisciplinary systems. In: 19th AIAA non-deterministic approaches conference, p 1950

Gavin HP (2010) Frame3DD structural analysis code. Duke University, Durham

Gill PE, Murray W, Saunders MA (2002) SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J Optim 12(4):979–1006. https://doi.org/10.1137/S1052623499350013

Haftka RT (1977) Optimization of flexible wing structures subject to strength and induced drag constraints. AIAA J 15(8):1101–1106. https://doi.org/10.2514/3.7400

Heath C, Gray J (2012) OpenMDAO: framework for flexible multidisciplinary design, analysis and optimization methods. In: Proceedings of the 53rd AIAA structures, structural dynamics and materials conference. Honolulu. AIAA-2012-1673

Hwang JT, Martins JRRA (2018) A computational architecture for coupling heterogeneous numerical models and computing coupled derivatives. ACM Transactions on Mathematical Software. (In press)

Kennedy GJ, Martins JRRA (2014) A parallel aerostructural optimization framework for aircraft design studies. Struct Multidiscip Optim 50(6):1079–1101. https://doi.org/10.1007/s00158-014-1108-9

Kenway GKW, Martins JRRA (2014) Multipoint high-fidelity aerostructural optimization of a transport aircraft configuration. J Aircr 51(1):144–160. https://doi.org/10.2514/1.C032150

Kenway GKW, Kennedy GJ, Martins JRRA (2014) Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and derivative computations. AIAA J 52(5):935–951. https://doi.org/10.2514/1.J052255

Kreisselmeier G, Steinhauser R (1979) Systematic control design by optimizing a vector performance index. In: International federation of active controls symposium on computer-aided design of control systems, Zurich

Lambe AB, Martins JRRA (2012) Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. Struct Multidiscip Optim 46:273–284. https://doi.org/10.1007/s00158-012-0763-y

Lambe AB, Martins JRRA, Kennedy GJ (2017) An evaluation of constraint aggregation strategies for wing box mass minimization. Struct Multidiscip Optim 55(1):257–277. https://doi.org/10.1007/s00158-016-1495-1

Liem RP, Kenway GKW, Martins JRRA (2015) Multimission aircraft fuel burn minimization via multipoint aerostructural optimization. AIAA J 53(1):104–122. https://doi.org/10.2514/1.J052940

Mader CA, Martins JRRA, Alonso JJ, van der Weide E (2008) ADjoint: an approach for the rapid development of discrete adjoint solvers. AIAA J 46(4):863–873. https://doi.org/10.2514/1.29123

Martins JRRA, Hwang JT (2013) Review and unification of methods for computing derivatives of multidisciplinary computational models. AIAA J 51(11):2582–2599. https://doi.org/10.2514/1.J052184

Martins JRRA, Lambe AB (2013) Multidisciplinary design optimization: a survey of architectures. AIAA J 51(9):2049–2075. https://doi.org/10.2514/1.J051895

Martins JRRA, Alonso JJ, Reuther JJ (2005) A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. Optim Eng 6(1):33–62. https://doi.org/10.1023/B:OPTE.0000048536.47956.62

Melin T (2000) A vortex lattice MATLAB implementation for linear aerodynamic wing applications. Master's Thesis, Department of Aeronautics, Royal Institute of Technology (KTH), Stockholm

Perez RE, Jansen PW, Martins JRRA (2012) pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization. Struct Multidiscip Optim 45(1):101–118. https://doi.org/10.1007/s00158-011-0666-3

Pesare G (2016) Modeling of unsteady vortex lattice method for wing flutter control in OpenMDAO. Master's thesis, Sapienza - Università di Roma

Phillips W, Snyder D (2000) Modern adaptation of Prandtl's classic lifting-line theory. J Aircr 37(4):662–670

Raymer DP (2012) Aircraft design: a conceptual approach, 5th edn. AIAA