Solver-Independent Aeroelastic Coupling For Large-Scale Multidisciplinary Design Optimization

Sebastiaan P. C. van Schie [*], Han Zhao [†], Jiayao Yan [‡], Ru Xiang [§], John T. Hwang [¶], David Kamensky [‖]
University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093

Recent advances in Multidisciplinary Design Optimization (MDO) have opened up the possibility of solving large-scale MDO problems with appropriate computational tools. One way in which design space exploration in MDO problems is made possible is by using solvers with varying levels of fidelity and computational cost. Our aim in this work is to make possible such an MDO approach by establishing a solver-independent aeroelastic coupling approach. We show that conservation of aeroelastic work requires an algebraic relation to be satisfied that involves the projection operators which map displacements and forces between solvers and states. Implementation examples are given for various combinations of fluid and solid solvers, and it is shown that these implementations indeed conserve all force components and aeroelastic work.

I. Introduction

Various new and innovative technologies, such as composite materials and electric propulsion, have emerged over the last couple of decades. As these technologies mature they are slowly being adopted by the aerospace industry. However, their adoption so far can be characterized as incremental improvement over the status quo, rather than opening up new design spaces. This is in part due to limitations in Multidisciplinary Design Optimization (MDO) approaches that inhibit efficient design space exploration. Additionally, the need for new, high-efficiency aircraft designs has opened up interest in new air vehicle concepts. The absence of existing baseline designs makes it difficult to confidently design such new concepts without relying on expensive prototyping and physical testing. The lack of efficient design space exploration tools furthermore forces designers to stick to tried-and-tested concepts; while these have shown to be effective and efficient for existing air vehicles through trial-and-error, it is unknown whether this is also the case for the aforementioned new air vehicle concepts.

Computational analysis capabilities have increased along with the rise of available computing power with Moore's law. Whereas nowadays large-scale computational simulations are used to inform engineers during the aircraft design process, their use in MDO algorithms has been limited due to their significant computational expense coupled with the large number of design evaluations that are necessary by traditional gradient-free optimization algorithms. In recent years gradient-based optimization has been on the rise within the MDO community, due to its superior performance for optimization problems with many design variables. Gradient-based optimization methods have been implemented in several open-source toolboxes. One example of such is NASA's OpenMDAO framework [1]. As pointed out by Martins and Hwang in [2], gradient-based optimization has found limited use outside of the MDO community since most computational methods do not contain the capabilities that are necessary to efficiently and accurately compute derivatives.

The focus of this work is to establish an aeroelastic coupling approach for use within a large-scale multifidelity MDO framework where such derivatives are available. This MDO approach consists of linked stages with different fidelity requirements. Different solvers are used to fulfill these fidelity requirements, and as such the coupling approach has to be solver-independent. Our goal is to implement a unified aeroelastic coupling formulation that conserves force & energy and allows for derivative computations in both steady-state and dynamic simulations. We start by going over the physical aeroelastic coupling mechanisms in section II, followed in section III by a brief overview of relevant

[*]Ph.D. Student, Department of Mechanical and Aerospace Engineering, AIAA student member, svanschie@ucsd.edu
[†]Ph.D. Student, Department of Mechanical and Aerospace Engineering
[‡]Ph.D. Student, Department of Mechanical and Aerospace Engineering, AIAA student member
[§]Ph.D. Student, Department of Mechanical and Aerospace Engineering
[¶]Assistant Professor, Department of Mechanical and Aerospace Engineering, AIAA member
[‖]Assistant Professor, Department of Mechanical and Aerospace Engineering

computational modeling aspects. This is followed by the main technical content of this work. We establish in section IV a solver-independent framework for aeroelastic coupling, after which we describe in section V the fluid and solid solvers that are used to demonstrate this framework. These demonstrations are then covered in section VI: Solver-specific implementation details are provided and results are given of coupled aeroelastic simulations of a wing of an electric Vertical Take-Off and Landing (VTOL) vehicle, with various numerical models. Lastly conclusions are drawn and future steps are identified in section VII.

II. Physical coupling mechanisms

The two-way physical interaction between the solid and fluid subdomains in any Fluid-Structure Interaction (FSI) problem is local to the fluid-solid interface. These interactions manifest itself through the location and displacement of this interface and the local traction vector $\boldsymbol{T}$ that acts upon said interface. Section II.A briefly covers the displacement coupling, after which section II.B goes over the fluid and solid surface tractions, how these are related to one another and their functional forms.

A. Displacements

Applying external loads to the solid (sub)domain results in a deformation field over said subdomain. This deformation can be non-zero on the boundary of the solid subdomain, leading to displacement of the fluid-solid interface. While the fluid solver does not have any dependence on the deformation of the interior of the solid subdomain, the displacement of the fluid-solid interface results in a change of the shape of the fluid subdomain, which in turn affects the solution fields on said domain. This change in the solution fields over the fluid subdomain can produce a change in the external load that is being applied to the boundary of the solid subdomain. For coupled time-dependent problems the displacement rate of the fluid-solid interface plays a role as well: Flow-tangency conditions imply that no fluid can pass through the fluid-solid interface, and thus the interface's displacement rate prescribes an interface-normal fluid velocity.

B. Surface tractions

Consistent with the available literature on FSI we take a general continuum mechanics approach to define the surface traction interactions, based on textbooks such as Bonet & Wood [3] and Chung [4]. This is also touched upon in several of the works cited in section III. Denote the Cauchy stress tensor in some fluid parcel with $\boldsymbol{\sigma}^{(f)}$, and the outward-pointing normal vector of the fluid domain with $\boldsymbol{n}^{(f)}$. Conversely, denote the Cauchy stress tensor and outward-pointing normal vector of the solid domain with $\boldsymbol{\sigma}^{(s)}$ and $\boldsymbol{n}^{(s)}$ respectively. Newton's third law (action equals reaction) implies compatibility of the tractions $\boldsymbol{T}^{(f)}$ and $\boldsymbol{T}^{(s)}$ that act upon the fluid and solid at the fluid-solid interface:

$$\boldsymbol{T}^{(f)} = \boldsymbol{\sigma}^{(f)} \cdot \boldsymbol{n}^{(f)} = -\boldsymbol{\sigma}^{(s)} \cdot \boldsymbol{n}^{(s)} = \boldsymbol{T}^{(s)} \tag{1}$$

Where we note that the outward-pointing normal vectors of the fluid and solid subdomains are collinear and opposite, i.e. $\boldsymbol{n}^{(f)} = -\boldsymbol{n}^{(s)}$. The choice of a specific approach for using Eq. (1) depends on whether the subdomain meshes are conforming or non-conforming at their interface, and whether there is exact correspondence between the Degrees of Freedom (DoFs) of each subdomain on their interface. Only when this exact correspondence exists can Eq. (1) be used directly. With non-conforming meshes a projection is required in order to map the traction components from one subdomain to the other. This projection often entails relaxing pointwise traction compatibility, especially when different discretization approaches are used for both subdomains (and their DoFs thus have different interpretations). The details of these projections vary depending on the choice of fluid and solid model. We give a brief general overview of the implications of Eq. (1) for partitioned FSI models and how these impact the fluid and solid subdomains. Both subdomains are covered separately.

1. Fluid traction

Boundary conditions are the standard method of modeling the effects of solid boundaries on the fluid domain. The fluid's velocity components normal and tangential to the fluid-solid interfaces are set to predefined values. Whereas for standard solid boundaries these velocity components are both equal to zero, applications with nonzero wall-normal or nonzero wall-tangential velocities exist. One example of the former is active flow control, where boundary layer suction and blowing is used in an attempt to improve lift or drag performance [5]. Examples of the latter are external aerodynamics simulations in the automotive industry: Instead of modeling cars as moving over a static floor, the car is defined as being static whereas the floor is given a tangential velocity boundary condition. Doing this greatly reduces the computational complexity of such simulations, since modeling the car as being static removes mesh motions in Eulerian discretizations of the fluid flow.

The role of the geometry of the fluid-solid interface is not immediately obvious in Eq. (1). Solving the solid model entails computing the deformation of the solid domain under an applied load, and thus involves movement of the fluid-solid interface. Any consistent numerical solution of the solid and fluid subdomains must thus take into account the effects that this deformation imposes upon the fluid.

Lastly we go over the stresses that these conditions impose upon a fluid domain. Throughout literature the Cauchy stress tensor components $\boldsymbol{\sigma}_{ij}^{(f)}$ in fluid mechanics are usually written as:

$$\boldsymbol{\sigma}_{ij}^{(f)} = -p\delta_{ij} + \boldsymbol{\tau}_{ij}(\mu) \tag{2}$$

Where p is the local hydrodynamic pressure, δ_{ij} is the Kronecker delta and $\boldsymbol{\tau}$ is the viscous (shear) stress tensor. The dependence of $\boldsymbol{\tau}$ on the fluid's dynamic viscosity μ is explicitly shown. Inviscid flow models do not take the shearing effects of viscosity into account, which causes $\boldsymbol{\tau}$ to drop out of Eq. (2). Viscous flow models often take the viscous (shear) stress imposed by solid (wall) boundaries into account by using some type of wall model. The universality of near-wall (boundary layer) velocity profiles is well-known in the mechanics community, and is treated extensively in any textbook that covers viscosity or turbulence effects in fluid flows. Some examples are the books of Landau & Lifshitz [6] and White [7].

2. Solid traction

Boundary conditions can also be involved in modeling the effects of the fluid-solid interface on the solid domain. The traction that acts on the solid domain's boundary can be taken from Eq. (1) as $\boldsymbol{T}^{(s)}$. With the fluid's Cauchy stress as given in Eq. (2) this means that:

$$\boldsymbol{T}^{(s)} = \underbrace{\left(p\delta_{ij} - \boldsymbol{\tau}_{ij}(\mu)\right)}_{=-\boldsymbol{\sigma}^{(f)}} \cdot \boldsymbol{n}^{(s)} = \boldsymbol{\sigma}^{(s)} \cdot \boldsymbol{n}^{(s)} \tag{3}$$

This traction is often taken into account in solid models in one of two ways. The first of these is as a traction boundary condition:

$$\boldsymbol{\sigma}^{(s)} \cdot \boldsymbol{n}^{(s)} = \boldsymbol{T}^{(s)} \text{ on } \partial\hat{\Omega}^{(s)} \tag{4}$$

Where $\partial\hat{\Omega}^{(s)}$ is the part of the solid domain boundary where $\boldsymbol{T}^{(s)}$ is known. Alternatively some discretization methods include $\boldsymbol{T}^{(s)}$ as external forcing term instead of as explicit boundary condition.

As was the case for the fluid domain, the formulation of $\boldsymbol{\sigma}^{(s)}$ is dependent on whether a viscous or inviscid fluid model is used. In the latter case the fluid's shear stress $\boldsymbol{\tau} = 0$, leaving hydrodynamic pressure p as the only source of traction.

III. Computational coupling

Several approaches exist for resolving computational FSI problems. We highlight some relevant works from literature, with a focus on establishing the desirable properties of any aeroelastic coupling method.

As mentioned by Hou et al. in [8] one way in which aeroelastic coupling approaches can be classified is according to the degree to which the numerical fluid and solid models are intertwined. On one end of this classification spectrum are the monolithic approaches, which simultaneously solve the numerical models that are posed on the fluid and solid subdomains with a single solver. Whereas this leads to fully consistent numerical solutions on both subdomains it is also an intrusive approach, since the coupling between the subdomains is explicitly encoded in the numerical problem definition. It furthermore requires one to solve linear systems that contain both the fluid and solid subproblems, plus possibly parameters that couple both subproblems. The size of this combined system can lead to excessive computational costs. In the current work a partitioned solution approach is used, in which the fluid and solid subproblems are solved in an alternating way until a predefined convergence tolerance is satisfied. The use of such a partitioned scheme is not without reason: Various fluid and solid solvers are used in the current multi-fidelity MDO framework, with different discretization approaches. With a partitioned solution approach the fluid and solid solvers can be changed without requiring significant modifications to the MDO workflow.

Another FSI solution procedure classification mentioned in [8] has to do with mesh conformity. This classification is concerned with whether both meshes on the fluid-solid interface share all of their vertices, edges and faces. Whereas using conforming meshes simplifies the transfer of loads and displacements between both subdomains due to the one-to-one correspondence of mesh elements on the fluid-solid interface, it requires both meshes to be constructed according to the most restrictive fidelity requirements of either domain. Mesh conformity furthermore also requires conformity of the non-discretized computational subdomains. Next to mesh conformity we consider in this work conformity of the non-discretized fluid and solid subdomains. Domain conformity is dependent on whether both fluid and solid method use a reduced domain representation approach. Methods that use such representations are often used in large-scale design optimization problems due to their reduced computational cost. These include (for fluids) potential flow methods such as the Vortex Lattice Method (VLM) [9, Sec. 12.3] and (for solids) beam methods [10]. Having mesh conformity implies domain conformity; non-conforming meshes on conforming domains are an example of domain conformity without mesh conformity.

A significant amount of published work on aeroelastic coupling for FSI problems already exists, owing to the ubiquity of coupled fluid-structure models. We highlight here some works from the relevant literature. In the aeroelastic coupling methods presented by Farhat et al. in [11] domain conformity is used to construct pointwise maps that directly associate the Finite Element Analysis (FEA) quadrature points of the solid subdomain to points on the boundary of the fluid subdomain. They identify consistency and conservation as important properties of any projection that maps loads from the fluid to the solid domain. Consistent methods are surface interpolation methods that take the shape functions used in the fluid simulation to compute the nodal surface traction terms in the quadrature points of the solid method. This results in “consistency", i.e. conservation of the total aerodynamic forces when projecting to the solid subdomain. Conservative methods are those that conserve the work done by aerodynamic forces in the coupled simulation. Conservation involves both the local forces and local deformation of the fluid-solid interface, whereas consistency depends solely on how the aerodynamic forces are transferred to the solid subdomain. As Farhat et al. point out, consistency is easier to attain for fluid and solid domains with different discretization methods than conservation. Furthermore, they find consistency to be more important for accuracy than exact conservation in their primary applications in compressible sub- and supersonic flow.

The properties of consistency and conservation are also highlighted in the work of Brown [12], which is in turn used by Kennedy and Martins in [13]. Kennedy and Martins couple an inviscid panel method to an FEA method to do aerostructural optimization and solve the coupled system with a monolithic/fully-coupled approach, wherein the fluid and solid solutions are computed simultaneously. For mapping aerodynamic pressure loads to the structural model they employ an expression for the virtual work of the pressure load. Posing this expression on the fluid boundary allows to insert the definition of the fluid’s surface displacement in terms of the solid displacement and rotation. This gives a computation scheme that maps the pressure load from the fluid to the solid domain boundary in a consistent and conservative way.

Rendall and Allen in [14] focus on using radial basis functions for the displacement transfer from solid to fluid. This makes the load transfer map meshfree and (nearly) independent from the discretization approaches of both subdomains. Through the principle of virtual work the (conservative) load transfer map follows from the displacement map. Rendall and Allen point to two other desirable property for an aeroelastic coupling approach:

1) When mapping displacements between both subdomains, points in both domains that are initially coincident should remain so during the coupled simulation.
2) Any load transfer approach should reduce to the identity map in the limit of the fluid and solid meshes and discretization methods matching exactly, such that the load is transferred in a pointwise-exact way.

A significant difference between the aforementioned sources and this work is the inclusion of the coupling approach within an MDO framework that employs multiple fluid and solid solution methods. We thus aim to first establish a general coupling framework that is solver-independent. The solver-specific details are filled in afterwards, and will (naturally) vary depending on which fluid and solid solvers are used.

IV. Solver-independent aeroelastic coupling framework

The coupling approach presented here is to be included in an MDO environment that includes solvers for various disciplines, each with its own input and output requirements. A solver-independent state representation is defined in order to facilitate the propagation of state information between coupled solvers. Defining such a state representation is outside of the scope of this work. Nonetheless we take into account the existence of a solver-independent state representation as an intermediate step when mapping quantities between the fluid and solid solvers. This has the added benefit of decoupling the choice for specific fluid and solid discretization methods: Solution information of each discipline-specific solver is passed to the solver-independent state representation, and thus separate solution/state maps can be defined for each solver. Figure 1 contains a visual representation of such a framework for coupled aeroelastic simulations.

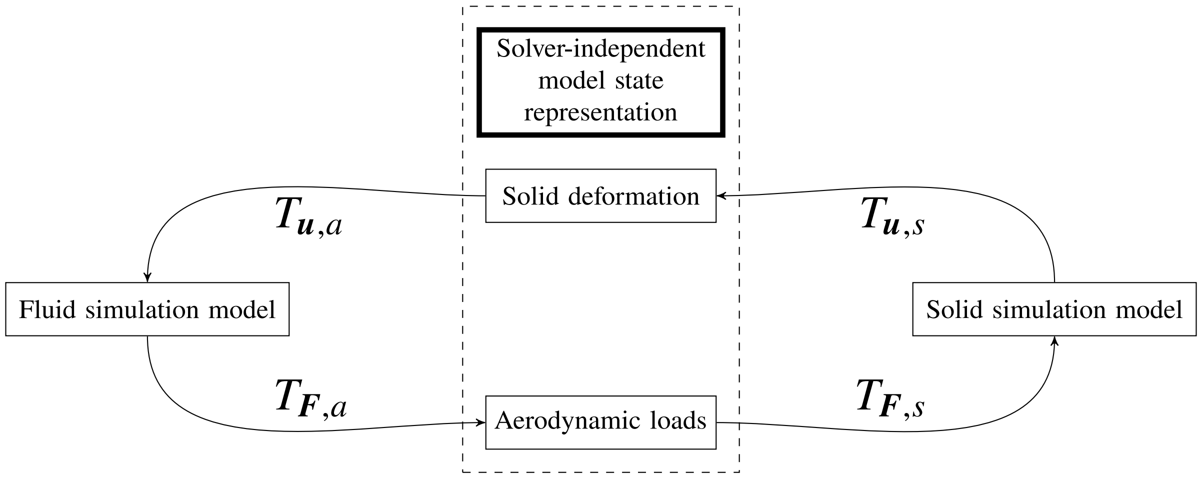


Fig. 1 General solver-independent coupling framework

Four maps are shown in Fig. 1. $T_{\boldsymbol{u},s}$ and $T_{\boldsymbol{u},a}$ map the discretized displacement vector $\boldsymbol{u}$ from the solid simulation to the solver-independent state representation ($T_{\boldsymbol{u},s}$) and from there to the fluid simulation ($T_{\boldsymbol{u},a}$). Furthermore $T_{\boldsymbol{F},a}$ and $T_{\boldsymbol{F},s}$ map the (discretized) aerodynamic force vector $\boldsymbol{F}$ from the fluid simulation to the solver-independent state ($T_{\boldsymbol{F},a}$) and on to the solid simulation model ($T_{\boldsymbol{F},s}$). We use the following subscripts to denote:

- s for quantities in the solid simulation model;
- a for quantities in the fluid ("aerodynamic") simulation model;
- r for quantities in the solver-independent ("reference") configuration.

Together with the four maps shown in Fig. 1 we thus know that:

$$\boldsymbol{u}_a = T_{\boldsymbol{u},a}\boldsymbol{u}_r = T_{\boldsymbol{u},a}T_{\boldsymbol{u},s}\boldsymbol{u}_s \tag{5}$$

$$\boldsymbol{F}_s = T_{\boldsymbol{F},s}\boldsymbol{F}_r = T_{\boldsymbol{F},s}T_{\boldsymbol{F},a}\boldsymbol{F}_a \tag{6}$$

The rest of this section covers how the four maps shown in Fig. 1 can be constructed to obtain a load- and energy-conservative aeroelastic coupling approach. We derive relations between the individual maps and constraints on their formulation, from which energy and load conservation follows automatically. Section IV.A will first cover conservation of energy, after which load conservation is discussed in section IV.B.

A. Conservation of energy

To illustrate the conservative coupling approach we first focus on the coupling between the fluid model and the solver-independent state representation. With discretized displacement and force vector variables in hand we can define the work W that is being performed by the force vector under the aforementioned displacement:

- In the fluid model:

$$W_a = \boldsymbol{u}_a^T A \boldsymbol{F}_a \tag{7}$$

- In the solver-independent state representation:

$$W_r = \boldsymbol{u}_r^T R \boldsymbol{F}_r \tag{8}$$

A and R are appropriately-sized matrices (not necessarily square) that make the multiplication with the displacement and force vectors result in a scalar value. Their structure and entries depend on how the displacement and force vectors are represented in their respective models. Two examples:

1) In fluid models that are discretized through the Finite Difference Method (FDM) or VLM the displacement $\boldsymbol{u}_a$ and force $\boldsymbol{F}_a$ represent point values. As long as the entries of these vectors at the same index refer to the same point, the matrix A will be the identity matrix and $\boldsymbol{u}_a^T A \boldsymbol{F}_a$ reduces to a summation of pointwise displacement-force products: $W_a = \boldsymbol{u}_a^T \boldsymbol{F}_a$.
2) In models that use basis function expansions to express the displacement and force variables, such as FEA-type methods or as is the case in the solver-independent state representation, matrix R is instead the mass matrix corresponding to the function bases being used.

For given pairs of displacements and force vectors we want to have conservation of energy between the fluid model and the solver-independent state representation. In other words, $W_a = W_r$. From Eq. (7) and (8) it then follows that:

$$\boldsymbol{u}_a^T A \boldsymbol{F}_a = \boldsymbol{u}_r^T R \boldsymbol{F}_r \tag{9}$$

We apply some of the relations from Eq. (5) and (6) to express $\boldsymbol{u}_a$ in terms of $\boldsymbol{u}_r$ and $\boldsymbol{F}_r$ in terms of $\boldsymbol{F}_a$. It then follows that:

$$\boldsymbol{u}_r^T (T_{\boldsymbol{u},a})^T A \boldsymbol{F}_a = \boldsymbol{u}_r^T R T_{\boldsymbol{F},a} \boldsymbol{F}_a \tag{10}$$

Which holds for all $\boldsymbol{u}_r$, $\boldsymbol{F}_a$ when:

$$(T_{\boldsymbol{u},a})^T A = R T_{\boldsymbol{F},a} \tag{11}$$

Conservation of energy thus implies that one of the pair of mappings $(T_{\boldsymbol{u},a}, T_{\boldsymbol{F},a})$ can be specified, after which the other mapping of the pair follows from Eq. (11). The resulting mapping pair will automatically conserve the work that is performed by aerodynamic forces $\boldsymbol{F}_a$ under the displacements $\boldsymbol{u}_r$. If the type of methods covered in example (1) above are used both A and R are identity matrices, and the energy-conserving mapping pair will be equal to each others transpose: $(T_{\boldsymbol{u},a})^T = T_{\boldsymbol{F},a}$.

While this illustration focuses on coupling the fluid model and the solver-independent state representation, the same relations can be derived for the mapping pair $(T_{\boldsymbol{u},s}, T_{\boldsymbol{F},s})$ that couples the solid model with the solver-independent state. We define the work W_r in the solid model, in line with the work formulations given in Eq. (7) and (8):

$$W_s = \boldsymbol{u}_s^T S \boldsymbol{F}_s \tag{12}$$

We repeat the derivation that was carried out above for the fluid model and the solver-independent state representation, by first equating W_s to the work W_r and applying Eq. (5) and (6). The result of these steps is:

$$\boldsymbol{u}_s^T (T_{\boldsymbol{u},s})^T R \boldsymbol{F}_r = \boldsymbol{u}_s^T S T_{\boldsymbol{F},s} \boldsymbol{F}_r \tag{13}$$

Since this holds for all $\boldsymbol{u}_s$, $\boldsymbol{F}_r$, we find that the energy-conserving mapping pair satisfies:

$$(T_{\boldsymbol{u},s})^T R = S T_{\boldsymbol{F},s} \tag{14}$$

We note here the similarity of Eq. (14) and (11).

So far we've established a way of constructing energy-conserving mapping pairs that relate the displacement and force vectors of the solver-independent state representation to a fluid or solid solver, i.e. the left- and right halves of Fig. 1. Quantities in the solid and fluid models can be directly related to one another through a composition of these mappings. We equate W_s and W_a:

$$W_a = \boldsymbol{u}_a^T A \boldsymbol{F}_a = \boldsymbol{u}_s^T S \boldsymbol{F}_s = W_s \tag{15}$$

Applying Eq. (5) and (6) allows us to insert the composition of mapping pairs:

$$\boldsymbol{u}_a^T A \boldsymbol{F}_a = \boldsymbol{u}_s^T \left(\boldsymbol{T}_{\boldsymbol{u},a}\boldsymbol{T}_{\boldsymbol{u},s}\right)^T A \boldsymbol{F}_a = \boldsymbol{u}_s^T S \left(\boldsymbol{T}_{\boldsymbol{F},s}\boldsymbol{T}_{\boldsymbol{F},a}\right) \boldsymbol{F}_a = \boldsymbol{u}_s^T S \boldsymbol{F}_s \tag{16}$$

Which has to hold for each $\boldsymbol{u}_s$, $\boldsymbol{F}_a$, such that:

$$\left(\boldsymbol{T}_{\boldsymbol{u},s}\right)^T \left(\boldsymbol{T}_{\boldsymbol{u},a}\right)^T A = S \boldsymbol{T}_{\boldsymbol{F},s}\boldsymbol{T}_{\boldsymbol{F},a} \tag{17}$$

Applying the transpose relation for the mapping pair $(\boldsymbol{T}_{\boldsymbol{u},a}, \boldsymbol{T}_{\boldsymbol{F},a})$ that relates the fluid model to the solver-independent state representation (given in Eq. (11)) results in:

$$\left(\boldsymbol{T}_{\boldsymbol{u},s}\right)^T R \boldsymbol{T}_{\boldsymbol{F},a} = S \boldsymbol{T}_{\boldsymbol{F},s}\boldsymbol{T}_{\boldsymbol{F},a} \tag{18}$$

In which we recognize the transpose relation for the mapping pair $(\boldsymbol{T}_{\boldsymbol{u},s}, \boldsymbol{T}_{\boldsymbol{F},s})$ that relates the solid model to the solver-independent state representation, as given in Eq. (14). In other words, forces and displacements can be mapped between the solid and fluid models in an energy-conserving way as long as each solver's mapping pair with the solver-independent state representation conserves energy. This adds to the flexibility of the proposed aeroelastic coupling approach: Individual solvers can be substituted for alternatives, and as long as a new energy-conserving mapping pair is defined for each new solver the aeroelastic coupling approach will conserve energy as well.

B. Load conservation

When applying the mapping operators shown in Fig. 1 we want to conserve the total loads in each principal direction in addition to conserving the work that is carried out by the aerodynamic loads. While energy conservation involves both the displacement $\boldsymbol{u}$ and force $\boldsymbol{F}$ and thus couples their respective mapping operators, load conservation involves only the latter quantity. This provides constraints for the force mapping operators $\boldsymbol{T}_{\boldsymbol{F},a}$ and $\boldsymbol{T}_{\boldsymbol{F},s}$. The transpose mapping properties derived in section IV.A then imply that the displacement mapping operators $\boldsymbol{T}_{\boldsymbol{u},a}$ and $\boldsymbol{T}_{\boldsymbol{u},s}$ should be affected as well if energy conservation is to be achieved.

Suppose for now that the aerodynamic force term $\boldsymbol{F}$ consists of three columns, one corresponding to each coordinate direction. We can pose a natural condition on (parts of) $\boldsymbol{T}_{\boldsymbol{F},a}$ and $\boldsymbol{T}_{\boldsymbol{F},s}$ for load conservation in each direction: The column sums of $\boldsymbol{T}_{\boldsymbol{F},a}$ and $\boldsymbol{T}_{\boldsymbol{F},s}$ should be equal to the measure (integral) of their associated basis function. The "measure" of a point value in this context is exactly 1. In other words: If $\boldsymbol{F}_a$ and $\boldsymbol{F}_r$ both consist of nodal values each column sum of the matrices making up $\boldsymbol{T}_{\boldsymbol{F},a}$ should be equal to 1. If $\boldsymbol{F}_r$ consists of FEA DoFs the column sums of $\boldsymbol{T}_{\boldsymbol{F},s}$ should be equal to one over the integral of the corresponding basis functions. These conditions for load conservation are algebraic in nature and can be encoded in the definition of suitable mapping operators $\boldsymbol{T}_{\boldsymbol{F},a}$ and $\boldsymbol{T}_{\boldsymbol{F},s}$.

V. Models used in implementation examples

Several demonstration cases have been constructed to demonstrate the versatility and solver-independent nature of the coupling approach. We include here (very) brief overviews of the fluid and solid models that are used in these examples. Only those aspects are covered here that are directly relevant in establishing solver-dependant details of the aeroelastic coupling framework introduced in section IV. Further sources are provided for interested readers. Section V.A covers the fluid models, after which the solid models are treated in section V.B.

A. Fluid models

Two fluid models are used in this work and a companion paper. First we cover the Vortex Lattice Method in section V.A.1, followed by the Vortex Particle Method in section V.A.2.

1. Vortex Lattice Method

In the Vortex Lattice Method (VLM) three-dimensional wings are reduced to their camber plane, which is then regarded as a lifting surface placed in an (inviscid) potential flow. A discrete geometry representation is produced by constructing a Cartesian mesh of quadrilateral panels on said camber plane. Each panel contains a single bound vortex ring that lags 25% of the panel's chordwise length behind its boundary, and a control point at 75% of the panel's chordwise length. This is also shown in Fig. 2. All vortex tubes induce a flow velocity vector in each control point, owing to their circulation strengths. These induced velocities can be computed with the Biot-Savart law. By requiring the flow velocity in each panel's control point to be tangential to said panel a unique solution vector of vortex circulation strengths can be determined. A more elaborate treatment of this low-fidelity potential flow model can be found in the reference work by Katz and Plotkin [9, Sec. 12.3]. Said treatment also covers how Kelvin's theorem leads to the shedding of vortex rings in unsteady VLM simulations.

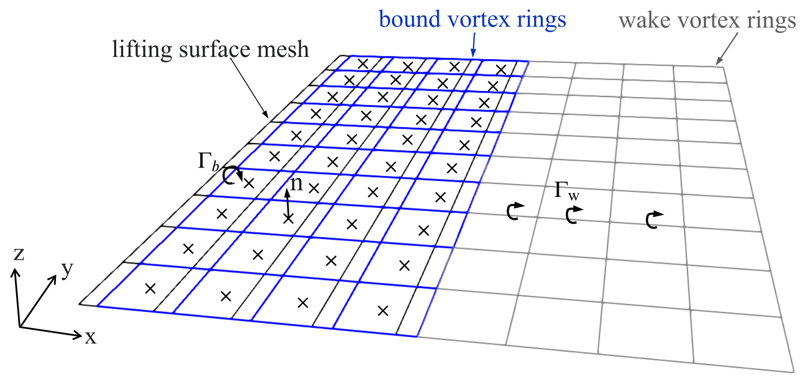


Fig. 2 VLM panel geometry

We note the following relevant aspects for establishing an aeroelastic coupling implementation:

- The vertices of the camber plane's surface mesh define the discrete VLM geometry. Thus the displacements of a panel's vertices define the motion of all points on said panel, including the control point.
- The vortex circulation strengths can be used to determine analytically the (constant) aerodynamic forces on each panel. Each panel's resultant force vector acts in its respective control point.

Both the aerodynamic displacement vector $\boldsymbol{u}_a$ and force vector $\boldsymbol{F}_a$ are thus defined in sets of points: The displacements in the vertices of the surface mesh and the force vectors in the mesh's control points. Since the displacement of a panel's vertices defines the movement of that panel's control point, we have a straightforward way of constructing a matrix A as in Eq. (7), that maps the displacements $\boldsymbol{u}_a$ of the surface mesh vertices to the corresponding displacements of the control points. This allows us to interpret W_a as the summed work resulting from the motion of the point forces acting in all control points.

Figure 3 shows an example arrangement of surface mesh vertices and panels, with associated numbering. We focus on the panel with index (i, j). Recall that the control points lie in the middle of each panel at 75% of its chordwise length. Using two-dimensional linear interpolation over the surface of panel (i, j) results in the influence coefficients shown in Tab. 1. These influence coefficients are used to populate the rows of matrix A, taking into account the indices of the vertices that border each panel. Matrix A will thus be sparse, with up to 4 nonzero entries in each of its columns and exactly 4 nonzero entries per row. The entries in each row of A sum to 1.

Table 1 Influence coefficients for mapping vertex displacements to the control point of panel (i, j) shown in Fig. 3

Vertex index	Influence coefficient
(i, j)	0.125
$(i, j+1)$	0.125
$(i+1, j)$	0.375
$(i+1, j+1)$	0.375

We finish our treatment of VLM by briefly highlighting a difference between the physical aeroelastic coupling mechanisms that were covered in section II and the actual outputs of VLM. As is mentioned above the VLM solution

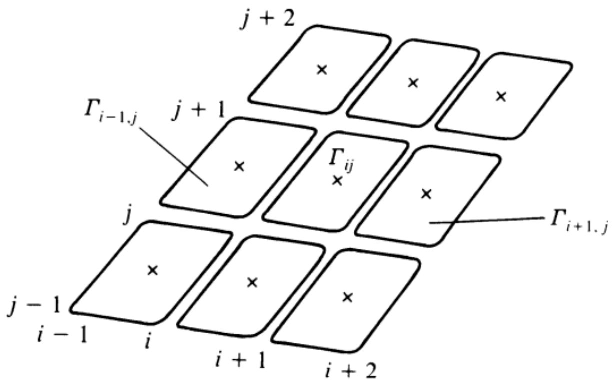


Fig. 3 Example numbering of surface mesh vertices and panels, reproduced from [9, p.342]

consists of vortex circulation strengths, which can be converted analytically into a set of force vectors, one per panel. This is markedly different from having a pressure, traction or stress field as output, which would be closer to the physical load transfer mechanism that was mentioned in section II.B. These two approaches describe the effects of the same physical interaction mechanism in different ways. These differences in the specific approaches arise due to properties of the discretization techniques that are used to obtain a computational model. Having a single force vector per surface panel is a central feature of VLM, and thus using a finite set of force vectors to model the aerodynamic loads is a straightforward way of including this physical effect. Other fluid simulation methods can use other descriptions; the mappings that couple such fluid methods to other state descriptions or tools are modified accordingly.

2. Vortex Particle Method

The Vortex Particle Method (VPM) is similar in nature to VLM. In the same way as is done in VLM three-dimensional wings are reduced to their camber plane. The VPM implementation covered here uses a single chordwise panel at all spanwise stations. Thus the interpretation of the displacement and force vectors is identical to those of VLM, with the only difference being the number of chordwise panels. The wake of the wing is modeled by releasing vortex particles at the trailing edge of each panel. Each particle induces a velocity field and is in turn advected by the combined effects of the wing-bound vortices and the other vortex particles that exist in the wing's wake. A big advantage of using VPM is its ability to accurately and efficiently resolve rotor wakes. Having such a tool is valuable within the context of this work and its applications. One major difference exists between the VLM and VPM implementations that are covered here: Whereas our VLM solver is used to generate steady-state solutions, the VPM solver is inherently transient due to the vortex particle nature of its wing wake model. Interested readers are referred to the related work by Anderson et al. [15], which goes into more detail and uses the coupling approach presented in this work.

B. Solid models

We use FEniCSx [16] and its predecessor (legacy) FEniCS [17] for the solid models used in this work. Both of these platforms allow us to leverage code generation capabilities that greatly reduce the time and effort it takes to implement FEA methods. We can define the weak forms that we want to use in precise mathematical language, after which FEniCS(x) takes care of discretizing the weak form with user-defined function bases and carrying out all the numerical integration that is necessary to set up the linear(ized) algebraic systems that are then solved.

1. FEA with Reissner-Mindlin shells in FEniCSx

The first of the solid models used in this work is a "classical" FEA implementation of the Reissner-Mindlin shell model with FEniCSx. For more information on this shell model the reader is referred to [15, 18]. A central feature of FEA-type discretizations is its precise definition of the bases in which variables are expressed. We model the solid displacements $\boldsymbol{u}_s$ in the Reissner-Mindlin model with quadratic basis functions and use a linear basis for the rotations. We also use a linear basis for the aerodynamic forces $\boldsymbol{F}_s$ that act on the solid domain $\Omega^{(s)}$. Since both the displacements and forces are defined as functions over $\Omega^{(s)}$ the work W_s that the aerodynamic loads carry out in the solid domain should be the result of an integral over said domain:

$$W_s = \int_{\Omega^{(s)}} \boldsymbol{u}_s^T S \boldsymbol{F}_s \, d\Omega^{(s)} \tag{19}$$

As $\boldsymbol{u}_s$ and $\boldsymbol{F}_s$ contain the values of their respective DoFs, the entries of S should thus contain the integrals of products of the quadratic and linear basis functions of the displacements and forces respectively. Instead of constructing S as a rectangular matrix in this way, we elect instead to make S a composition of two matrices. One matrix, which we designate with Q, will interpolate the quadratic basis functions to a linear basis on the same mesh. The product $Q\boldsymbol{u}_s$ is then a vector of the solid displacements projected to the linear basis. The entries Q_{ij} are equal to the value of the j^{th} quadratic basis function in the i^{th} solid mesh vertex. The linear basis has a direct one-to-one correspondence with the vertices of the solid mesh, which provides a straightforward association of the displacement DoFs with the mesh vertices. The second matrix that makes up S is then taken as the mass matrix M of the linear basis. With one eye on Eq. (19) it follows that $S = Q^T M$ gives an appropriate way to compute W_s for FEA-type methods.

2. IGA with Kirchhoff-Love shells in PENGoLINS

We use Isogeometric Analysis (IGA) as an alternative for "classical" FEA. IGA is an FEA-type method that was first introduced by Hughes et al. in 2005 [19]. Instead of using piecewise polynomial functions for its function bases, IGA uses Non-Uniform Rational B-Splines (NURBS). These splines are commonly used in Computer-Aided Design (CAD) software for modeling geometric shapes. IGA allows one to directly use CAD-native geometry descriptions for use in numerical simulations, without having to approximate the shape of said geometry in some way. Readers that are interested in IGA are referred to the foundational book by Cottrell et al. [20]. In this work we make use of tIGAr [21], a toolbox built on top of (legacy) FEniCS that uses the process of Bézier extraction to relate NURBS function bases to piecewise polynomial bases. This allows us to leverage the utilities and capabilities of FEniCS for IGA implementations.

Kiendl et al. were the first to use IGA for simulating the Kirchhoff-Love (KL) shell model in [22]. IGA is an attractive option for this due to the higher-order inter-element continuity of its basis functions. The KL model requires a C^1-continuous basis (i.e. continuity of functions and their first derivatives), which is possible with quadratic or higher-degree IGA bases. We use the PENGoLINS framework [23] to simulate assemblies of multiple KL shells. This allows us to simulate the behavior of aerospace structures under various loading conditions.

With PENGoLINS we do not construct a single mesh that envelops all KL shells of a given geometry, instead a penalty approach is used to glue together shell surfaces along their intersection curves. An advantage of this for aeroelastic coupling is that we natively have a separation of the computational domain into its different surface patches and (if necessary) the parts of surface patches on either side of an intersection curve. This allows us to isolate the direct effects of aeroelasticity to those (parts of) surface patches that are directly exposed to aerodynamic influences.

Since IGA is an FEA-type method the remarks that were made in section V.B.1 regarding Eq. (19) hold here as well. This time we model the solid displacements $\boldsymbol{u}_s$ as cubic NURBS functions; interpolation matrix Q thus maps from this cubic basis to the linear NURBS basis in which $\boldsymbol{F}_s$ is expressed. The linear NURBS basis has a similar one-to-one association to mesh vertices as the standard piecewise linear basis, with the main difference being that standard IGA uses structured Cartesian grids instead of triangular surface grids. Mass matrix M is now computed with the linear NURBS basis. This thus defines the product $S = Q^T M$ in a similar way to its definition in section V.B.1.

VI. Fidelity-dependent coupling examples

We work out several implementation examples to show how the solver-independent coupling framework applies to different solid and fluid models. The first of these examples is a coupling between FEniCSx and VLM in section VI.A, followed by a coupling between PENGoLINS and VLM in section VI.B. Lastly, in section VI.C we briefly cover a coupling between FEniCSx and VPM.

A. FEniCSx - Vortex Lattice Method

We couple the Vortex Lattice Method to the Reissner-Mindlin shell implementation in FEniCSx, without the use of a solver-independent state representation between them. In line with section IV we need to define the mapping pair

$(T_{\boldsymbol{u},s}, T_{\boldsymbol{F},a})$ such that:

$$(T_{\boldsymbol{u},s})^T A = S T_{\boldsymbol{F},a} \tag{20}$$

Where the matrix operators A and $S = Q^T M$ were covered in sections V.A.1 and V.B.1 respectively. We elect to define the displacement map $T_{\boldsymbol{u},s}$, from which the force map $T_{\boldsymbol{F},a}$ follows.

Recall that the solid displacements are defined in a basis of quadratic basis functions. We choose to first project the displacement from this quadratic basis to a basis of linear functions. For this we use the same matrix Q that was introduced in section V.B.1. At the end of the current derivation this results in $T_{\boldsymbol{F},a}$ being independent of Q. Note that the product $Q\boldsymbol{u}_s$ is a vector of the solid displacements projected to the linear basis.

The second step of establishing $T_{\boldsymbol{u},s}$ consists of constructing the matrix G that relates movements of the vertices of the solid mesh to the vertices of the VLM mesh. We use an approach that is similar to how matrix A was specified to relate the displacements of the vertices of the VLM mesh to its control points: For each VLM mesh vertex we take a weighted combination of the displacements of the solid mesh vertices. The first step of constructing G is to use compute the distance $r_{ij} = ||\boldsymbol{x}_{a,i} - \boldsymbol{x}_{s,j}||$ between VLM mesh vertex $\boldsymbol{x}_{a,i}$ and solid mesh vertex $\boldsymbol{x}_{s,j}$. This distance is used as input to a Radial Basis Function (RBF). Various RBFs can be used. In this work we opt to use the bump function, as it has a compact support. The bump function is defined as:

$$f_\epsilon(r_{ij}) = \begin{cases} \exp\left(-\frac{1}{1-(\epsilon r_{ij})^2}\right) & \text{if } r_{ij} < \frac{1}{\epsilon} \\ 0 & \text{else} \end{cases} \tag{21}$$

Here ϵ is a parameter that controls the size of the RBF support and the rate at which the RBF value drops off for increasing r. Generally speaking $\epsilon \in [0.1, 2]$ was found to work best during development; the specific values of ϵ are given for each presented test case. Next to computing $f_\epsilon(r_{ij})$ for all i and j, we compute the size of the support of the (linear) solid basis functions $\phi_{s,j}$, which we will denote with $||\Omega_{s,j}||$. The entries G_{ij} of displacement mapping matrix G are then defined as:

$$G_{ij} = \frac{f_\epsilon\left(||\boldsymbol{x}_{a,i} - \boldsymbol{x}_{s,j}||\right)\ ||\Omega_{s,j}||}{\sum_j f_\epsilon\left(||\boldsymbol{x}_{a,i} - \boldsymbol{x}_{s,j}||\right)\ ||\Omega_{s,j}||} \tag{22}$$

This defines G. We note that the entries in each row of G sum to 1; G thus indeed contributes to a consistent and conservative aeroelastic coupling method. The displacement map $T_{\boldsymbol{u},s}$ becomes:

$$T_{\boldsymbol{u},s} = GQ \tag{23}$$

And thus $\boldsymbol{u}_a = GQ\boldsymbol{u}_s$. Conversely, with Eq. (20) we know that $T_{\boldsymbol{F},a}$ should be defined as:

$$T_{\boldsymbol{F},a} = M^{-1} G^T A \tag{24}$$

Note that this definition is independent of Q. Letting $\boldsymbol{F}_s$ and $\boldsymbol{F}_a$ again denote the aerodynamic forces on the solid and fluid domains, we can see that inverting M can be avoided. Instead we can solve a linear system for $\boldsymbol{F}_s$:

$$\boldsymbol{F}_s = T_{\boldsymbol{F},a}\boldsymbol{F}_a = M^{-1}G^T A\boldsymbol{F}_a \Longrightarrow M\boldsymbol{F}_s = G^T A\boldsymbol{F}_a \tag{25}$$

Both formulations of $T_{\boldsymbol{F},a}$ are used interchangeably in this work.

This defines a coupled aeroelastic system where FEniCSx is used for the solid model and VLM as fluid model. We solve the coupled system iteratively until a predefined convergence tolerance is satisfied, using the following order of operations:

1) The fluid model computes $\boldsymbol{F}_a$.
2) The aerodynamic forces in the solid model are calculated with $\boldsymbol{F}_s = T_{\boldsymbol{F},a}\boldsymbol{F}_a = M^{-1}G^T A\boldsymbol{F}_a$.
3) The solid model outputs $\boldsymbol{u}_s$.
4) The displacements of the fluid model follow from $\boldsymbol{u}_a = T_{\boldsymbol{u},s}\boldsymbol{u}_s = GQ\boldsymbol{u}_s$.

Such an iterative approach can be used for steady-state and transient coupled simulations. For the examples presented in this work the coupled iterative approach is considered to have converged when:

$$\epsilon_{\text{conv}} \geq \max_i ||\boldsymbol{u}_{s,i}^{(k)} - \boldsymbol{u}_{s,i}^{(k-1)}|| \tag{26}$$

Where i is the index of the DoFs and k is the number of the current iteration. Expressed in other words: The coupled simulation is deemed to have converged when the maximum element-wise difference between solid displacement DoFs of successive iterations is lower than the convergence tolerance ϵ_{conv}.

We apply this coupling method to the discretized solid geometry of a forward-swept starboard-side wing with internal shell structure that is shown in Fig. 4. Roughly $67,000$ vertices and $137,000$ elements define the shell mesh, which thus consists of patches of (shell) surface elements. The shell mesh has been constructed with the methods outlined by Liu et al. in [18]. We apply clamping boundary conditions along the surface patch at the wing's root. A VLM grid of 80 spanwise panels (half on the starboard-side wing, half on the port-side wing) and 8 chordwise panels is used. This mesh size was chosen due to hardware constraints. We note that this VLM mesh leads to under-resolved results. We also note that this does not affect the accuracy or robustness of the aeroelastic coupling approach in any way. For clarity we repeat that the solid simulation consists of a single starboard-side wing, while the fluid simulation models both sides of the wing. As we focus here on symmetrical inflow conditions we can mirror the displacements of the starboard-side wing to apply these to the port-side fluid mesh. We lastly note that a convergence tolerance of $\epsilon_{\text{conv}} = 10^{-6}$ is used, together with RBF parameter $\epsilon = 1$.

This discretized wing is placed in an oncoming airflow of $V_\infty = 50\ m/s$ at angles of attack between $-2°$ and $14°$. Air density is kept at $1\ kg/m^3$. Since VLM is a potential flow method we do not expect any flow separation or stall effects to show up in the results. Moreover, the lack of viscosity in potential flow methods makes the lift-induced drag component the sole source of drag. All shell surfaces that make up the solid model have a thickness of $3 \cdot 10^{-3}\ m$, a Young's modulus of $E = 6.8 \cdot 10^{10}\ Pa$ and a Poisson's ratio of $\nu = 0.35$.

Figure 5 shows the drag, lift and lateral forces over the range of angles of attack mentioned above for the starboard-side wing. We see in Fig. 5a and 5c that the linear increase in lift leads to the well-known quadratic increase in lift-induced drag. Figure 5d shows the relative differences of the force vector 2-norm and amount of work performed by the aeroelastic forces between the fluid and solid simulations. Both quantities are normalized by the 2-norm of the fluid simulation's force vector and work respectively. As can be seen these relative differences are close to machine precision over the entire range of angles of attack, which demonstrates that the aeroelastic coupling approach presented here is indeed both consistent and conservative. The vertical wing tip displacement is shown in Fig. 6 and shows a similar linear trend as the total lift force.

Next we look at the aerodynamic force distributions over the wing surface. These are shown in Fig. 7 (upper wing surface) and 8 (lower wing surface) for angles of attack of $2°$, $7°$ and $12°$. As can be seen the peak force on both the upper and lower surface is present near the wing root, with the force gradually tapering off towards the wing tip. The local force magnitudes increase as the angle of attack is increased, owing to the increase in force components that is also reflected Fig. 5. Moreover, the location of the peak force moves towards the leading edge between angles of attack of $2°$ and $7°$ and stays there. Lastly we note that there is no discernible difference between the force distributions over the lower and upper surfaces. This is a consequence from the fact that VLM models the aerodynamic loads directly as force vectors instead of as surface tractions.

B. PENGoLINS - Vortex Lattice Method

Next we couple the Vortex Lattice Method with PENGoLINS and simulate the coupled system using the geometry shown in Fig. 4a. The coupling formulation that is specified in section VI.A is used here as well; since PENGoLINS is an FEA-type method much of the derivation carries over directly. The interpretation of the various terms can be adjusted accordingly:

- G now links the vertices of the VLM surface mesh to the vertices of the Cartesian IGA mesh on each KL shell patch.
- Q now maps between the linear NURBS basis and the cubic basis used for solid displacement $\boldsymbol{u}_s$.
- M is now the mass matrix of the linear NURBS basis.

We highlight one other relevant change for the construction of G: We change the RBF that we use. Instead of the bump function we use a Gaussian RBF:

$$f_\epsilon(r) = \exp\left(-(\epsilon r)^2\right) \tag{27}$$

This change was made during initial testing for reasons of robustness: Since the bump function has a compact support

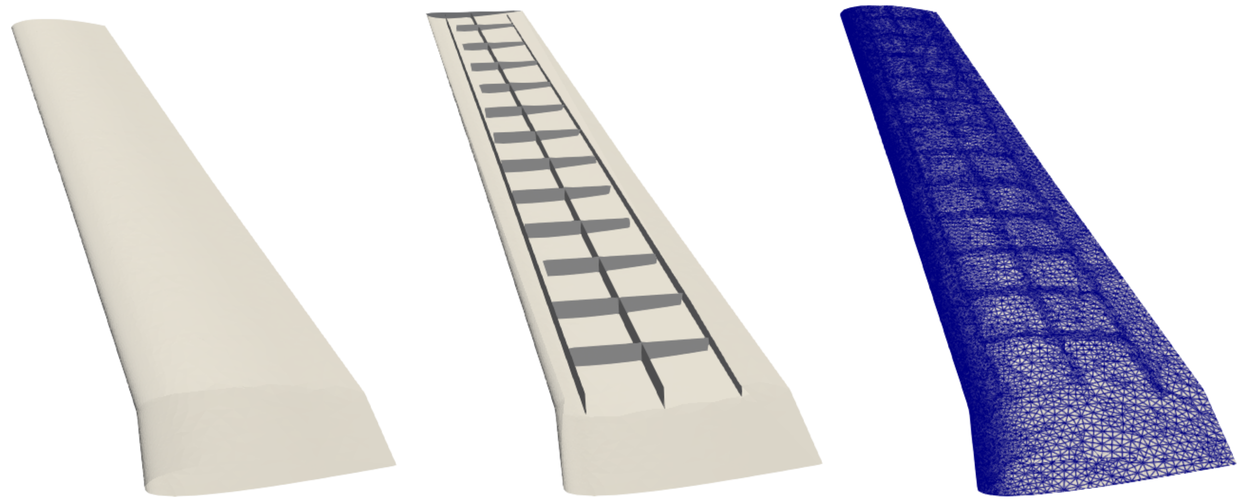

(a) Geometry **(b) Internal geometry** **(c) Meshed geometry**

Fig. 4 Views of the wing geometry and triangular shell mesh on said geometry

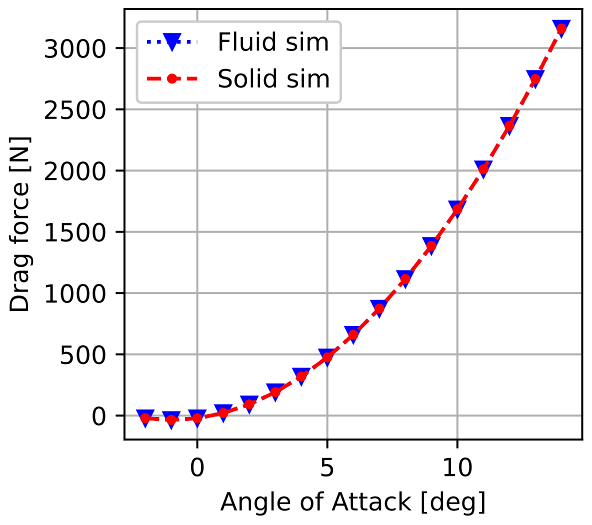
and the aeroelastic coupling implementation in PENGoLINS is local to the wing skin it is possible to pick values of ϵ that lead to rows of G with row sums equal to 0 (as the RBF centered around certain VLM mesh vertices does not have any solid mesh vertices in its support). The Gaussian RBF is plugged into Eq. (22) and the rest of the assembly procedure is identical to what was covered in section VI.A. Matrix A is unchanged as well, as is the iterative solution approach.

We run the test case that was covered in section VI.A for this solver pair. The same material parameters, inflow variables and numerical parameters are used, apart from $\epsilon = 0.05$ for the RBF. This value of ϵ leads to smeared out force fields, i.e. it limits local force variations. Another change is the mesh that is used by PENGoLINS: Quadrilateral elements are used to form a Cartesian grid on each surface patch. A total of roughly $6,000$ DoFs are used here. A nearly identical version of this geometry was used in [23], where results of a mesh independence study are given which show that this is an adequate number of DoFs. Figure 9 shows the three force components and relative work and force vector conservation errors between the solid and fluid simulations. We draw the same conclusion as for the previous example: Both work and force are conserved up to machine precision over the whole range of angles of attack that were simulated. We also note that the force components for this example are nearly identical to those of the previous example. Figure 10 shows the obtained tip displacements; these are within a couple percent of those predicted with FEniCSx. This is consistent with the results that were presented in [23].

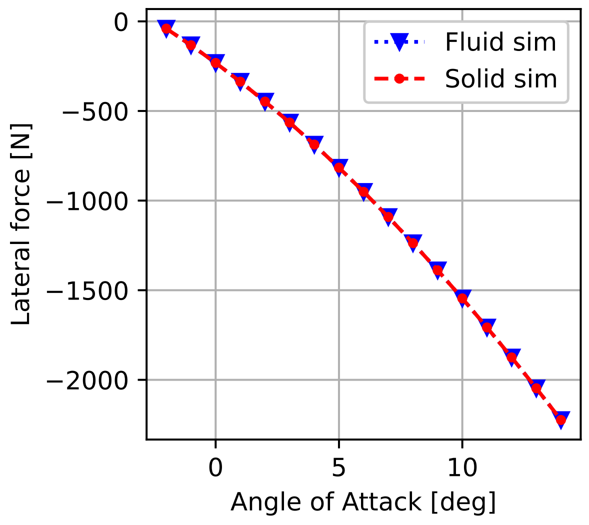
As mentioned before, using $\epsilon = 0.05$ leads to the forces being smeared out. This is reflect in Fig. 11 and 12: For all angles of attack shown the distributed load over the wing surface is nearly constant. A minor drop in force magnitude is visible when moving towards the wing root and (especially) tip, but no discernible chordwise variation is visible. Despite this the tip deflections obtained with these force distributions are similar to those obtained with FEniCSx, as alluded to above.

C. FEniCSx - Vortex Particle Method

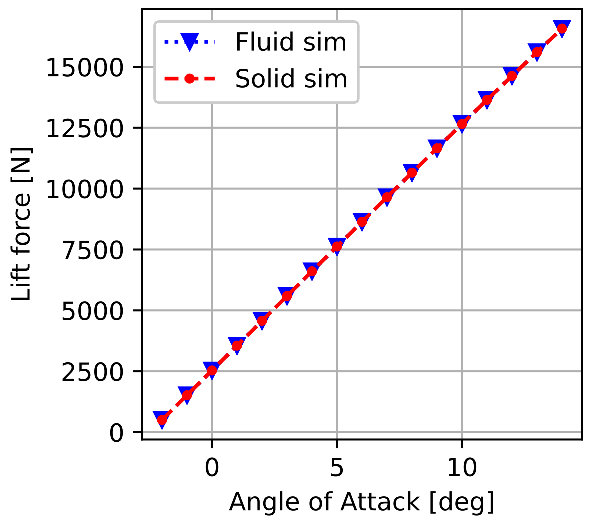
The last example mentioned in this work is that of a coupling implementation between FEniCSx as solid solver and VPM as fluid solver, applied to the wing geometry of Fig. 4a. As mentioned in section V.A.2 the VPM implementation used is nearly identical to our VLM implementation for the purpose of aeroelastic coupling. Because of this we can directly use the coupling approach that is covered in section VI.A. We do not give results for this implementation in this work. Instead we would like to refer the interested reader to the companion paper by Anderson et al. [15], where a more elaborate treatment of the specifics of the solid and fluid solvers and this example are given.



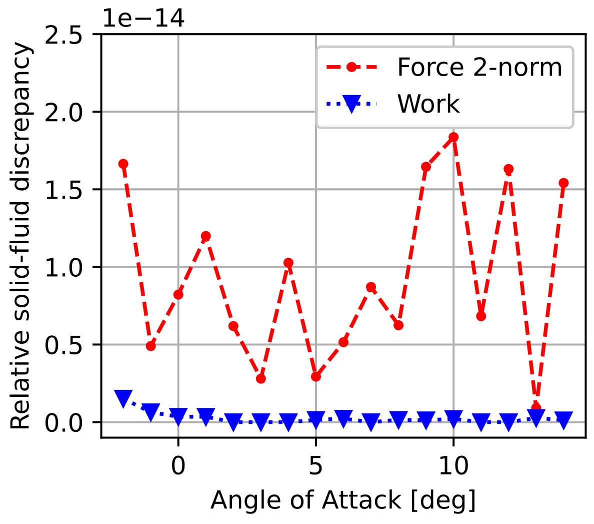
(a) Drag force



(b) Lateral force



(c) Lift force



(d) Relative conservation errors between both simulations

Fig. 5 Force components and relative conservation errors between the fluid (VLM) and solid (FEniCSx) simulations for the wing geometry of Fig. 4

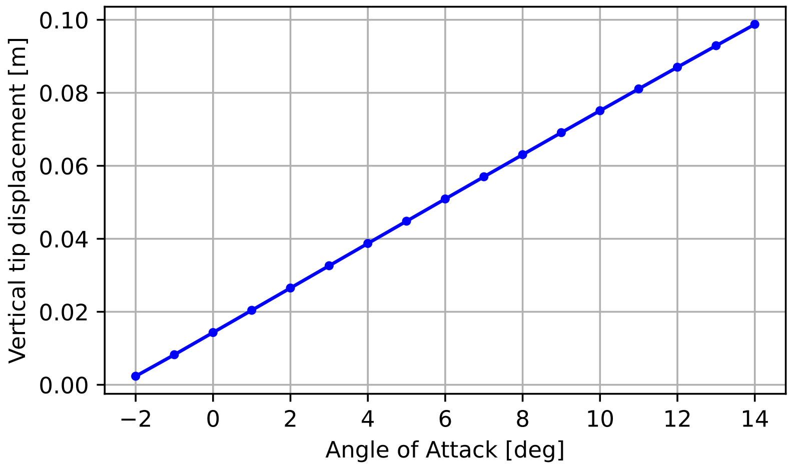


Fig. 6 Vertical tip displacement versus angle of attack for the wing geometry of Fig. 4 simulated with FEniCSx as solid solver

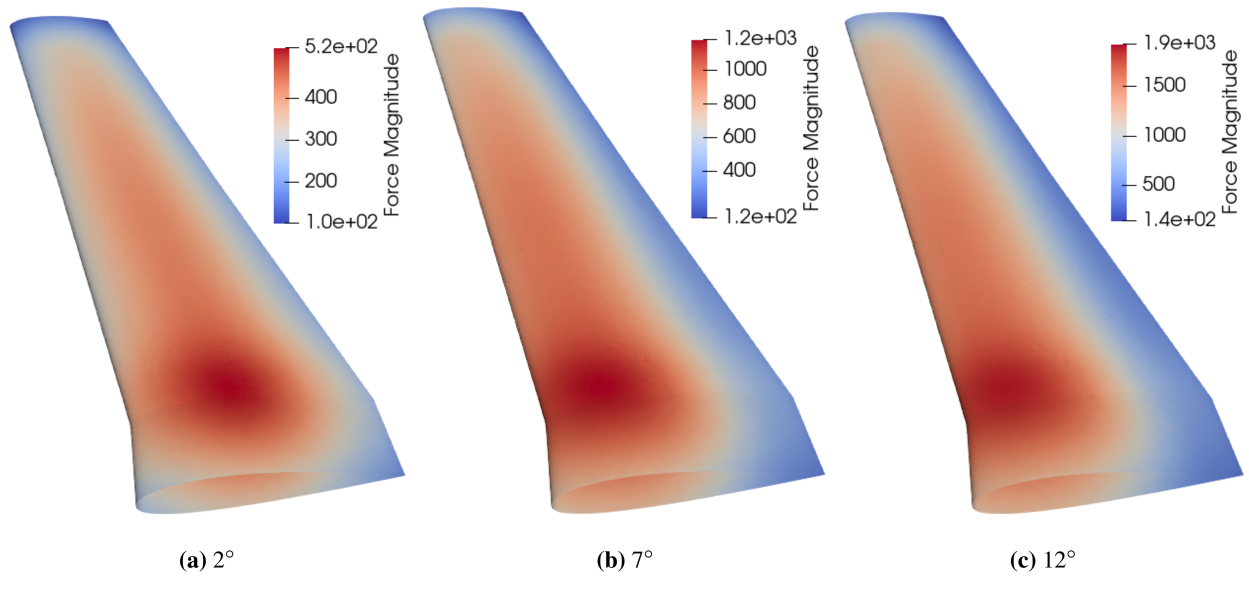


Fig. 7 Force magnitude distributions over the upper surface of the wing shown in Fig. 4 for different indicated angles of attack as obtained with FEniCSx as solid solver; leading edges on the left

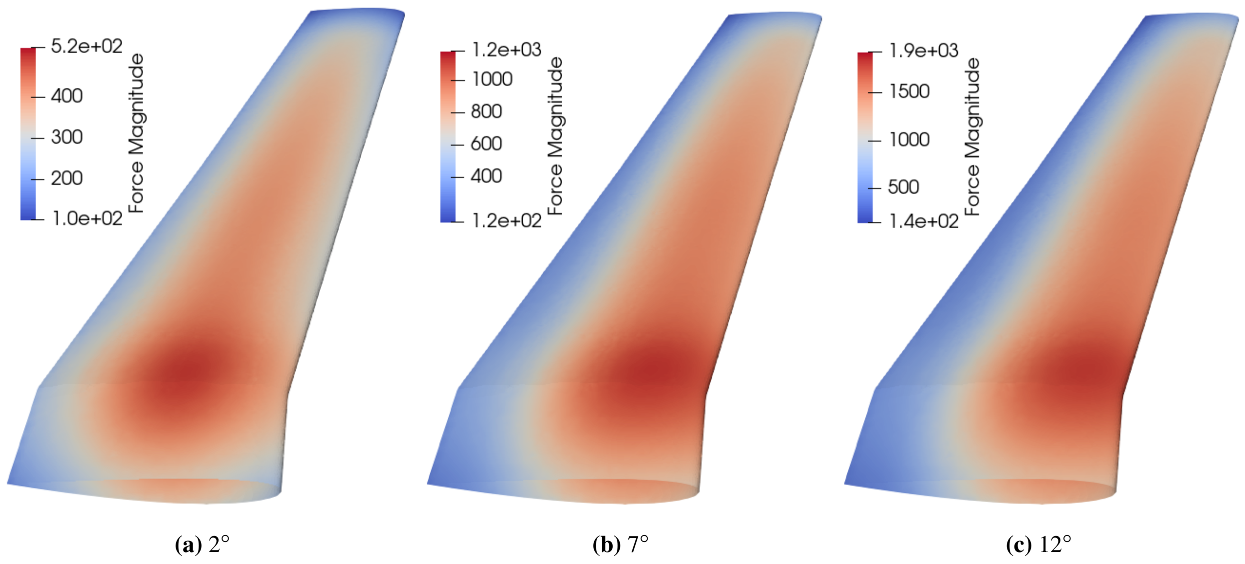


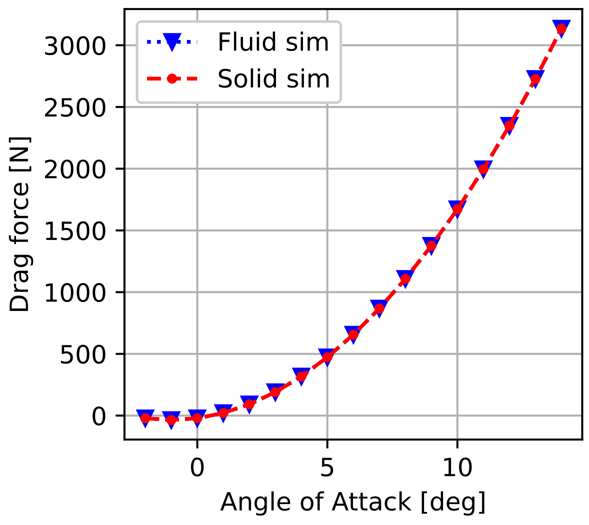
Fig. 8 Force magnitude distributions over the lower surface of the wing shown in Fig. 4 for different indicated angles of attack as obtained with FEniCSx as solid solver; leading edges on the right

VII. Conclusion

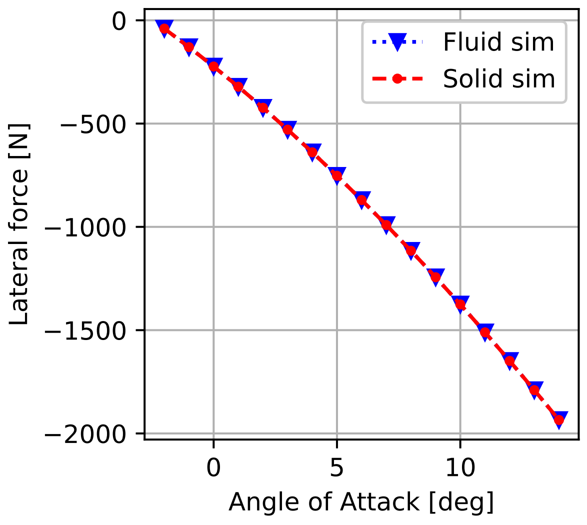
In this work we have put forth an aeroelastic coupling framework that is solver-independent and is suitable for use in large-scale Multidisciplinary Design Optimization environments. We started off by introducing the physical sources of aeroelastic coupling: Deformations of the boundary of the solid subdomain and fluid & solid surface tractions on the fluid-solid interface. We introduced some desirable properties of any computational coupling approach that are mentioned throughout the relevant literature. Chief among these are conservation of force components and the work that is performed by aerodynamic forces (or surface tractions) under the corresponding deformations induced by the solid domain. A solver-independent aeroelastic coupling framework was introduced. This framework contains a solver-independent state representation as an intermediate state that couples to both solid and fluid simulation methods. We formulated the use of pairs of maps which relate displacements and forces of the different models and states to one another, and showed that conservation of energy implies a transpose relation between the mappings that make up each pair. These transpose relations extend to compositions of mappings, when mapping through intermediate state representations.

We introduced two solid and two fluid simulation methods to show that this framework indeed leads to coupling approaches that conserve both force components and aeroelastic work. First a brief overview of the relevant solver aspects was given, after which the solver-agnostic details of each coupling example were defined. A wing shell geometry was used as representative example for the types of applications that we intend to use this coupling framework for. All implementation examples covered in this work were shown to indeed conserve both force components and aeroelastic work when mapping between various solid and fluid simulation models, which supports our notion that the current approach is applicable to a range of computational methods.

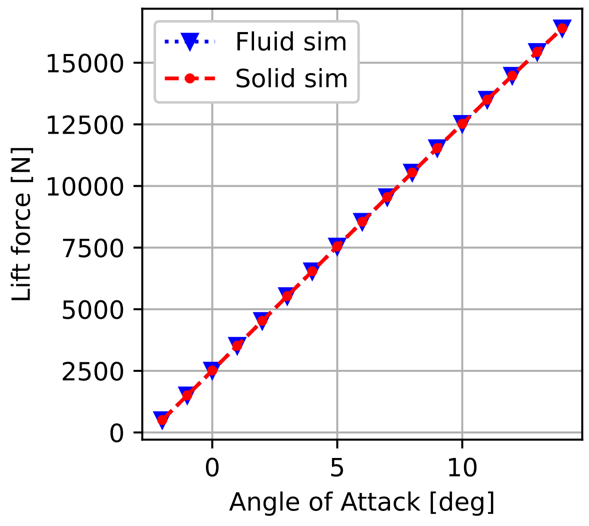
Several directions have been identified as avenues for future work. The solver-independent state representation that is present in the general coupling framework is currently not present in the examples given here; it is our plan to define such a solver-independent state as intermediate state for future aeroelastic coupling methods. We furthermore plan to look into whether it is possible to define force maps from which the displacement map follows through the aforementioned mapping pair transpose relations, instead of the current approach of doing this the other way around. Moreover, the choice for a specific Radial Basis Function formulation and its tuning parameter can affect the outputs of the aeroelastic coupling method. More work and testing is needed with different Radial Basis Functions to quantify or otherwise bound this influence. Lastly we note that we can improve upon the wing models that are used as examples in this work. Instead of using fully-enclosed airfoils we intend to limit the solid structure in future applications to just the wing box. This more accurately reflects the intended applications of this work.



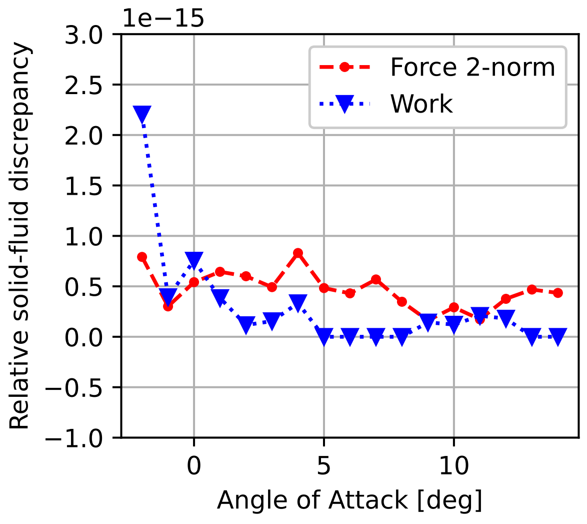
(a) Drag force



(b) Lateral force



(c) Lift force



(d) Relative conservation errors between both simulations

Fig. 9 Force components and relative conservation errors between the fluid (VLM) and solid (PENGoLINS) simulations for the wing geometry of Fig. 4a

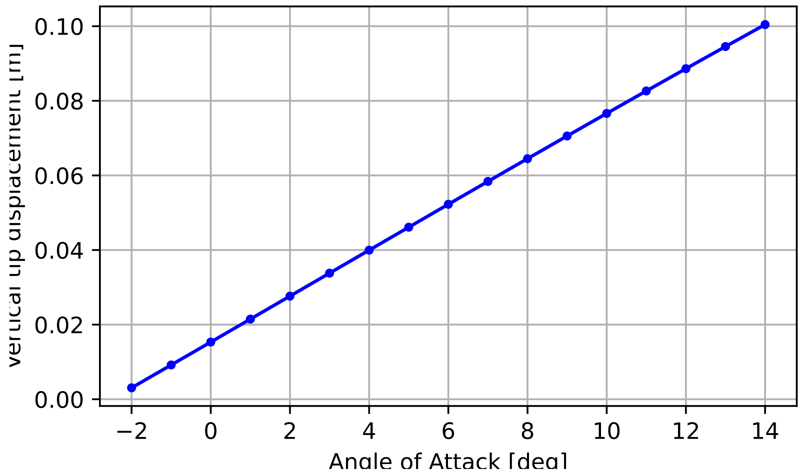
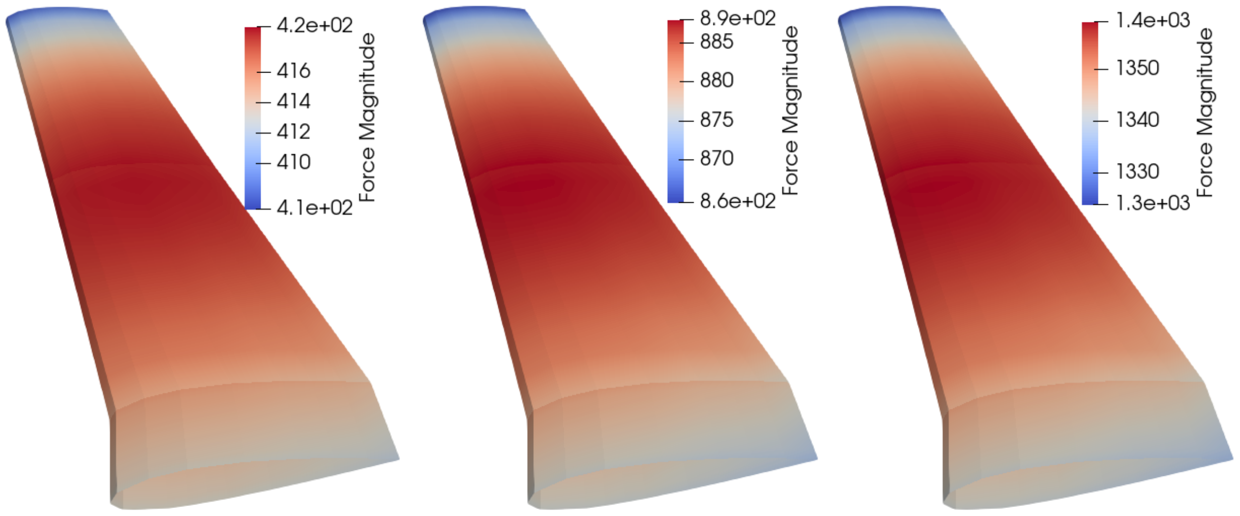
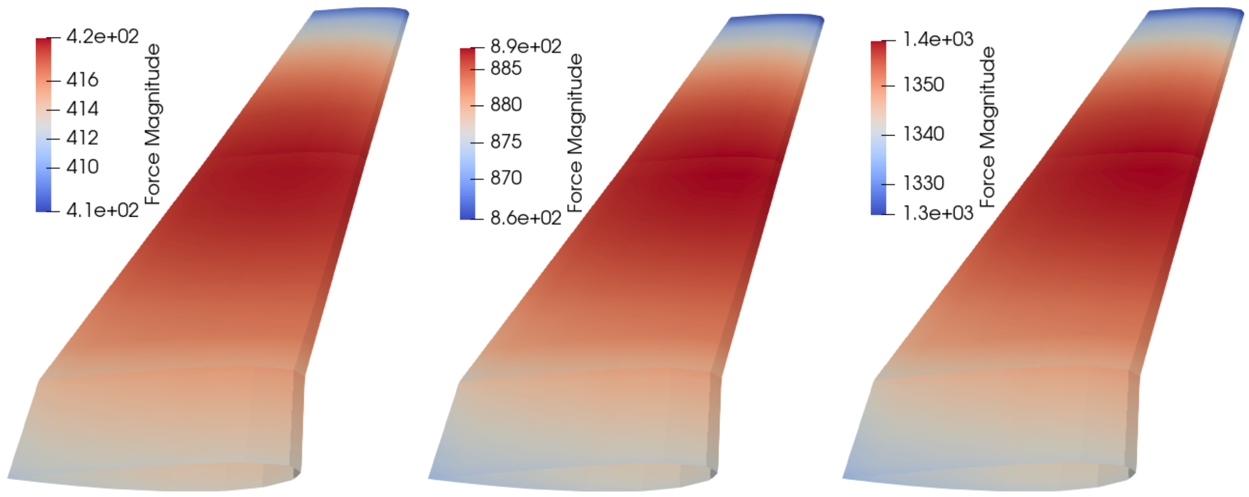


Fig. 10 Vertical tip displacement versus angle of attack for the wing geometry of Fig. 4a simulated with PENGoLINS as solid solver



(a) 2° **(b)** 7° **(c)** 12°

Fig. 11 Force magnitude distributions over the upper surface of the wing shown in Fig. 4 for different indicated angles of attack as obtained with PENGoLINS as solid solver; leading edges on the left



(a) 2° **(b)** 7° **(c)** 12°

Fig. 12 Force magnitude distributions over the lower surface of the wing shown in Fig. 4 for different indicated angles of attack as obtained with PENGoLINS as solid solver; leading edges on the right

Acknowledgments

S. P. C. van Schie was funded through fellowship funds provided by University of California San Diego and through NASA grant 80NSSC21M0070. H. Zhao, J. Yan and R. Xiang were supported through this same NASA grant.

References

[1] 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.

[2] 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.

[3] Bonet, J., and Wood, R. D., *Nonlinear Continuum Mechanics for Finite Element Analysis*, 2nd ed., Cambridge University Press, 2008.

[4] Chung, T. J., *General Continuum Mechanics*, 2nd ed., Cambridge University Press, 2007.

[5] Fahland, G., Stroh, A., Frohnapfel, B., Atzori, M., Vinuesa, R., Schlatter, P., and Gatti, D., "Investigation of Blowing and Suction for Turbulent Flow Control on Airfoils," *AIAA Journal*, Vol. 59, No. 11, 2021, pp. 4422–4436.

[6] Landau, L. D., and Lifshitz, E. M., *Fluid Mechanics*, 2nd ed., Course of Theoretical Physics, Vol. 6, Butterworth-Heinemann, 1987.

[7] White, F. M., *Viscous Fluid Flow*, 3rd ed., McGraw-Hill Series in Mechanical Engineering, McGraw-Hill Education, 2006.

[8] Hou, G., Wang, J., and Layton, A., "Numerical Methods for Fluid-Structure Interaction — A Review," *Communications in Computational Physics*, Vol. 12, No. 2, 2012, p. 337–377.

[9] Katz, J., and Plotkin, A., *Low-Speed Aerodynamics*, 2nd ed., Cambridge Aerospace Series, Cambridge University Press, 2001.

[10] Meier, C., Popp, A., and Wall, W., "Geometrically Exact Finite Element Formulations for Slender Beams: Kirchhoff–Love Theory Versus Simo–Reissner Theory," *Archives of Computational Methods in Engineering*, Vol. 26, No. 1, 2019, pp. 163–243.

[11] Farhat, C., Lesoinne, M., and Le Tallec, P., "Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 157, No. 1, 1998, pp. 95–114.

[12] Brown, S., “Displacement extrapolations for CFD+CSM aeroelastic analysis,” *38th Structures, Structural Dynamics, and Materials Conference*, 1997. AIAA Paper 97-1090.

[13] Kennedy, G., and Martins, J. R. R. A., “Parallel Solution Methods for Aerostructural Analysis and Design Optimization,” *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010. AIAA Paper 2010-9308.

[14] Rendall, T. C. S., and Allen, C. B., “Unified fluid–structure interpolation and mesh motion using radial basis functions,” *International Journal for Numerical Methods in Engineering*, Vol. 74, No. 10, 2008, pp. 1519–1559.

[15] Anderson, R. M., Ning, A., Xiang, R., van Schie, S. P. C., Sperry, M., Sarojini, D., Kamensky, D., and Hwang, J. T., “Aerostructural Predictions Combining FEniCS and a Viscous Vortex Particle Method,” *SciTech*, 2023. Submitted for publication.

[16] Alnaes, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N., “Unified Form Language: A domain-specific language for weak formulations of partial differential equations,” *ACM Transactions on Mathematical Software*, Vol. 40, 2014.

[17] Logg, A., Mardal, K.-A., Wells, G. N., et al., *Automated Solution of Differential Equations by the Finite Element Method*, Springer, 2012.

[18] Liu, X., Xiang, R., Pasetto, M., Li, N., Kamensky, D., and Hwang, J., “Shell-element mesh generation for multidisciplinary optimization of aircraft configurations,” , 2023. In review.

[19] Hughes, T., Cottrell, J., and Bazilevs, Y., “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 39, 2005, pp. 4135–4195.

[20] Cottrell, J., Hughes, T., and Bazilevs, Y., *Isogeometric Analysis: Toward integration of CAD and FEA*, John Wiley & Sons, Ltd., 2009.

[21] Kamensky, D., and Bazilevs, Y., “tIGAr: Automating isogeometric analysis with FEniCS,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 344, 2019, pp. 477–498.

[22] Kiendl, J., Bletzinger, K.-U., Linhard, J., and Wüchner, R., “Isogeometric shell analysis with Kirchhoff–Love elements,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 198, No. 49, 2009, pp. 3902–3914.

[23] Zhao, H., Liu, X., Fletcher, A. H., Xiang, R., Hwang, J. T., and Kamensky, D., “An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures,” *Computers & Mathematics with Applications*, Vol. 111, 2022, pp. 109–123.