Coupled Adjoint-Based Sensitivities in Large-Displacement Fluid-Structure Interaction using Algorithmic Differentiation

R. Sanchez[1], T. Albring[2], R. Palacios[1*], N. R. Gauger[2], T. D. Economon[3], J. J. Alonso[3]

[1]Department of Aeronautics, Imperial College London, SW7 2AZ, London, United Kingdom

[2]Chair for Scientific Computing, TU Kaiserslautern, 67663 Kaiserslautern, Germany.

[3]Department of Aeronautics & Astronautics, Stanford University, 94305 Stanford, CA, USA

*Corresponding author: r.palacios@imperial.ac.uk

Abstract

A methodology for the calculation of gradients with respect to design parameters in general Fluid-Structure Interaction problems is presented. It is based on fixed-point iterations on the adjoint variables of the coupled system using Algorithmic Differentiation. This removes the need for the construction of the analytic Jacobian for the coupled physical problem, which is the usual limitation for the computation of adjoints in most realistic applications. The formulation is shown to be amenable to partitioned solution methods for the adjoint equations. It also poses no restrictions to the nonlinear physics in either the fluid or structural field, other than the existence of a converged solution to the primal problem from which to compute the adjoints. We demonstrate the applicability of this procedure and the accuracy of the computed gradients on coupled problems involving viscous flows with geometrical and material non-linearities in the structural domain.

I. Introduction

Computational methods in Fluid-Structure Interaction (FSI) have reached sufficient maturity to address complex engineering problems, including those defined in the analysis stage of industrial applications. A major open challenge is still their effective incorporation into the design chain, which requires the integration of complex FSI solvers into (possibly even more complex) optimization frameworks. Due to the high computational cost of FSI analysis for large problems, gradient-based algorithms are usually the most appropriate for this task, but they rely on the efficient computation of sensitivities for the coupled problem [1]. Given that a small change in a single design variable is likely to affect the behavior of the full system, the task of assessing the influence of a large number of design parameters is a challenging task.

Adjoint methods have shown to be an efficient alternative for problems involving a very large number of design variables. They have been known for over four decades and have been extensively applied to problems in fluid dynamics [2, 3] and structural mechanics [4]. The main appeal of adjoint methods is that their cost is practically independent of the number of design variables, unlike alternative methods for computing gradients, such as finite differences, the complex-step method [5], or the direct method (see, e.g., [4,6]). In the early 2000s, adjoint method applications for the optimization of high-fidelity coupled systems began to appear. This was initiated by Maute *et al.* [7, 8] and Martins *et al.* [9, 10] for aero-structural steady-state optimization, and adjoint methods have since become a key feature in the development of high-fidelity multidisciplinary optimization.

Later works by Maute *et al.* [11], Martins *et al.* [12], Barcelos *et al.* [13], Fazzolari *et al.* [14] and Abu-Zurayk and Brezillon [15], among others, have further explored the use of adjoints for optimizations in computational aeroelasticity. Given the ability of the adjoint method to efficiently

compute sensitivities with respect to thousands of design variables, recent applications have tackled very large multiphysics problems, such as those associated with the shape optimization of full aircraft configurations [16]. Adjoint methods for FSI have also been investigated in other disciplines, such as parameter identification for medical applications [17, 18], coupled electrostatic FSI problems [19], goal-oriented error estimation [20, 21], steering applications [22], to name a few. However, the complexity of this technique often leads to simplifications in their implementation that have limited the scope of their application.

From a fluid-dynamics point of view, many applications in aero-structural optimisation have neglected viscosity (see, for example [7, 11, 16, 23]), or, if the Reynolds-averaged Navier-Stokes equations were considered, turbulence models have often been "frozen," meaning that their linearization is ignored and omitted from the adjoint formulation [24]. The implications of this assumption are discussed in a paper by Dwight and Brezillon [25]. Although linearization of viscous terms and turbulence models and their inclusion within adjoint formulations is now becoming routine, the process can be cumbersome and error-prone for non-automatic techniques of differentiation.

From a structural point of view, in most cases only linear elastic approaches have been adopted. This assumption limits the structural deflections to the small deformation regime [11, 12, 16, 23]. Although structural non-linearities have been incorporated by Lund *et al.* [26], the problem is in that case solved monolithically using an approximate Jacobian, which as pointed out later by Barcelos and Maute [6], limits its application to small "academic" problems.

These simplifications are normally related to the requirement, for an accurate computation of the adjoint, of an exact linearization of the physical problem. Obtaining the *exact* analytic Jacobian on fully-nonlinear coupled FSI problems is however an extremely complex task, both in terms of code development and memory handling. Peter and Dwight [24], working on the adjoint method for fluid applications, have recently pointed out that, given the complexity of the physical models needed for realistic applications, analytical Jacobians are becoming increasingly impractical. This complexity increases even more for coupled, multidisciplinary problems.

Algorithmic Differentiation (AD) techniques [27, 28] provide a computational alternative to the analytic evaluation of the Jacobian. The basic concept is that any computational code is built as a sequence of elementary operations that only depend on one or two variables [27], and therefore, the sensitivities may be obtained applying the chain rule recursively. AD has been applied to compute discrete adjoints in both fluid [29–31] and structural [32] problems, and also for FSI adjoint applications [16]. In these cases, AD is typically applied selectively to some routines that compute the flow residual [30] or some cross-dependencies in the FSI problem [16], which complicates the generalization of the methodology.

In this work, we propose an alternative approach that consistently applies AD to a partitioned FSI solver. It is a natural extension to the work presented by Albring [33, 34] and Zhou [35] for aerodynamic and aeroacoustic optimization, where it has been already shown that it brings minimum requirements to the characteristics of the problem or its solution methods and it does not interfere in the future development of the direct solvers. Here, we will introduce an AD-based Discrete Adjoint (ADDA) iterative methods to compute accurate gradients for FSI problems involving both viscous flows and non-linear structural behaviour. We employ Expression Templates [36] in order to avoid the large computational cost and memory requirements that result from the application of the most common AD techniques (operator overloading and source code transformation). The implementation has been done in the partitioned, strongly-coupled FSI solver available in the open-source SU2 software suite [37, 38]. SU2 consists of a set of C++ libraries developed from the ground up for the solution of complex, multiphysics Partial Differential Equation-based problems [39–41] and has already shown its viability for aerodynamic shape design in industrial-scale applications, including full aircraft configurations and wind turbine blades [41]. The software

is supported by an international consortium[1] that seeks collaborative work in the solution of complex, multidisciplinary problems. In this work, we will restrict ourselves to an computation of sensitivities for steady-state problems with compressible flows, and the gradients will be computed with respect to structural parameters only. These starting assumptions have been adopted in order to develop and test the methodology, but constitute no fundamental limitation to the proposed method.

This paper is organized as follows. Section II reviews the different methods available in the literature to compute sensitivities for gradient-based optimization problems. Section III presents the concepts behind the forward and reverse modes of AD and also covers Expression Templates. Section IV briefly summarizes the governing equations and the coupling conditions for the FSI primal solver. Section V presents both the conventional and the new methodology developed in this paper for the application of the adjoint method to structural and FSI problems. The sensitivities computed using finite differences, the conventional adjoint approach, and the AD-based methodology that we have developed in this work are shown in section VI, demonstrating the potential of this new technique. Finally, section VII summarizes the main findings and the contributions of this work.

II. Background

Given a physical system, we define an objective function J as

$$J = J(\mathbf{x}(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \tag{1}$$

which depends on a vector of design variables $\boldsymbol{\alpha}$ explicitly and/or implicitly through the state variables of the system $\mathbf{x}(\boldsymbol{\alpha})$. The implicit dependency is determined by the governing equations of the system, which we will write as

$$\mathscr{G} = \mathscr{G}(\mathbf{x}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = 0, \tag{2}$$

with the equations being satisfied for each value of $\boldsymbol{\alpha}$. We can then define the following constrained optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \; J(\mathbf{x}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &\text{subject to } \; \mathscr{G}(\mathbf{x}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = 0. \end{aligned} \tag{3}$$

Many optimization algorithms have been proposed to solve (3). They are often divided into two large groups, namely, zero-order (gradient-free) and gradient-based methods, as described, for example, by Hernández [42] and Martins *et al.* [12]. Zero-order methods, such as conjugate directions, genetic algorithms, or neural networks, require only the evaluation of the objective function J during the iterations of the optimizer. However, when dealing with a large design space and complex problems where the calculation of the objective function is expensive, the computational cost of these methods rapidly increases, rendering them prohibitively expensive.

First-order, or gradient-based, methods require the evaluation of the derivative of the objective function with respect to the design variables (and generally assume a smooth design space). They typically require a smaller number of evaluations, which is advantageous for large multiphysics problems due to the high cost of computing the objective function. Given an initial guess of the design variables $\boldsymbol{\alpha}_0$, each iteration of the optimizer updates the value of $\boldsymbol{\alpha}$ using an expression of the form [42, 43]

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k + \beta_k \mathbf{d}_J(\nabla_\alpha J(\mathbf{x}(\boldsymbol{\alpha}_k), \boldsymbol{\alpha}_k)), \tag{4}$$

[1] http://su2.stanford.edu/

where β is the step size, and $\mathbf{d}_J$ is the search direction, which is a function of the gradient of the objective function, $\nabla_\alpha J(\mathbf{x}(\boldsymbol{\alpha}_k), \boldsymbol{\alpha}_k)$. Clearly, an efficient computation of this gradient is essential for keeping the cost of gradient-based algorithms low relative to zero-order methods, and the main strategies adopted in the literature are briefly reviewed next. Further details can be found in the books of Arora [44] and Gunzburger [45].

II.1. Finite differences

The simplest, yet still extensively used, method for computing the gradient of the objective function is by finite differences. The method is closely related to the definition of a derivative: a small perturbation h in a design variable is introduced, the objective function is re-evaluated, and the resulting change is computed by subtraction. It is possible to approximate the gradient of the objective function using first-order schemes, such as forward or backward differences, or alternatively using a second-order central difference scheme, such as the one that will be employed later in this work, or

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha_j} = \frac{J(\mathbf{x}(\alpha_j + \frac{1}{2}h_j), \alpha_j + \frac{1}{2}h_j) - J(\mathbf{x}(\alpha_j - \frac{1}{2}h_j), \alpha_j - \frac{1}{2}h_j)}{h_j} + \mathscr{O}(h_j^2), \qquad \forall j \in n_\alpha. \tag{5}$$

Finite differences are extremely popular in engineering applications due to their ease of implementation, especially for black-box solvers. They do not require any source code modifications and will give reasonably accurate gradients, assuming that the increment h is adequately chosen. However, these methods suffer both from cancellation and truncation errors that result in their accuracy being highly dependent on the value of h, which generally requires a parametric study. Additionally, the computational cost of the calculation of the gradient is directly dependent on the number of design variables of the problem, n_α. For first-order schemes, it is necessary to evaluate the objective function at least $n_\alpha + 1$ times, while for second-order schemes, this number increases to $2n_\alpha + 1$.

II.2. Complex step method

The complex step method is an alternative to finite differences which effectively eliminates the cancellation errors due to the subtraction in the numerator of (5), thereby allowing one to choose a very small increment h. The method is conceptually very similar to finite differences. However, in this case, the increment is applied in the complex domain [5]. The computational cost of the gradient calculation remains directly dependent on the number of design variables of the problem, i.e., the objective function needs to be computed at least $n_\alpha + 1$ times. Moreover, it requires access to the source code in order to redefine the variables of the problem in the complex domain, which generally reduces its range of applicability to in-house codes or open-source platforms. However, due to its accuracy and lack of dependency in the value of h, the derivatives obtained using this method are commonly used to verify the accuracy of gradients obtained by other techniques [16].

II.3. Direct method

Using the chain rule, one can express the gradient of the objective function as the row vector

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \frac{\partial J}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\alpha}}. \tag{6}$$

It can also be noted that, for each value of the design variable $\boldsymbol{\alpha}$ (within certain bounds), the governing equations (2) will be satisfied, which implies that their total derivative with respect to the design variables is equal to zero, i.e.,

$$\frac{\mathrm{d}\mathscr{G}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial\mathscr{G}}{\partial\boldsymbol{\alpha}} + \frac{\partial\mathscr{G}}{\partial\mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\alpha}} = 0 \tag{7}$$

within the design space. If we now define the Jacobian of the governing equations as $\mathbf{K}_{\mathscr{G}} = \partial\mathscr{G}/\partial\mathbf{x}$, we can rewrite (7) as

$$\mathbf{K}_{\mathscr{G}}\,\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\alpha}} = -\frac{\partial\mathscr{G}}{\partial\boldsymbol{\alpha}}. \tag{8}$$

Therefore, one can obtain the sensitivities of the state variables with respect to the control parameters $\mathrm{d}\mathbf{x}/\mathrm{d}\boldsymbol{\alpha}$ by solving one linear system of the size of problem $\mathscr{G}$ per design variable and then introducing this result into (6). This is called the direct method [4, 11], and its computational cost is directly proportional to the number of design variables n_α, as this defines the number of linear systems (8) that must be solved. The cost of solving this linear system is equivalent to one implicit iteration of the solver of the primal problem. However, it must be noted that the linear system in (8) requires the exact representation of the Jacobian of the problem $\mathbf{K}_{\mathscr{G}}$. Unfortunately, depending on the solution method chosen for the primal problem, this Jacobian may not be explicitly available or may be available only in an approximate form, which is the case in many problems in fluid dynamics and also in structural analysis with complex material models.

II.4. Adjoint methods

Problem (3) can also be recast as an unconstrained optimization problem. In this case, its solution is obtained by minimizing the corresponding Lagrangian, $L = J + \bar{\mathbf{x}}^T\mathscr{G}$, i.e.,

$$\min_{\boldsymbol{\alpha},\mathbf{x},\bar{\mathbf{x}}}\; J(\mathbf{x},\boldsymbol{\alpha}) + \bar{\mathbf{x}}^T\mathscr{G}(\mathbf{x},\boldsymbol{\alpha}), \tag{9}$$

where the Lagrange multipliers $\bar{\mathbf{x}}$ are typically referred to as the adjoint, or dual, variables. The solution to this problem is sought by differentiating the Lagrangian with respect to the adjoint variables, the state variables, and the design variables, which yields, respectively, the governing equation (2), the adjoint equation

$$\mathbf{K}_{\mathscr{G}}^T\,\bar{\mathbf{x}} = -\frac{\partial J}{\partial\mathbf{x}}^T, \tag{10}$$

and the optimality condition, $\partial\mathscr{L}/\partial\boldsymbol{\alpha} = \mathbf{0}$ at the optimum. These are the well-known Karush-Kuhn-Tucker (KKT) optimality conditions [44, 45]. Substituting now the value of $\partial J/\partial\mathbf{x}$ from (10) into (6) and using (8), we can finally write

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J}{\partial\boldsymbol{\alpha}} + \bar{\mathbf{x}}^T\frac{\partial\mathscr{G}}{\partial\boldsymbol{\alpha}}. \tag{11}$$

Adjoint methods have superior performance for the computation of sensitivities for a single objective with a large number of design variables, as the adjoint equation does not depend on the number of design parameters n_α. The biggest contribution to the computational cost of the solution process is the linear system in (10), which is comparable to the cost of solving (8) for the direct method. In both cases, the number of equations and variables of the linear system is equal to the number of state variables of the system $n_{\mathbf{x}}$. However, for the direct case it is necessary to solve n_α linear systems; alternatively, this number is the number of objective functions, n_J, for the adjoint method. As in the case of the direct method, solving the adjoint equation (10) requires the Jacobian

of the problem $\mathbf{K}_{\mathscr{G}}$, which may not be readily available.

III. Algorithmic Differentiation

Of particular interest in this work is the evaluation of derivatives by means of Algorithmic Differentiation (AD). The main idea behind AD is better described with an example: Let $\mathbf{y} = \mathbf{f}(\mathbf{x})$ be a function of some input variables $\mathbf{x}$. The numerical evaluation of $\mathbf{f}$ can be represented as a sequence of l elementary functions φ_i. These functions are simple operations, such as addition, subtraction, multiplication, sine, etc. Let v_i be a set of intermediate values in the computational sequence. Following the description of Albring *et al.* [33], one can express the evaluation of this function as shown in Tab. 1 (for a function with two inputs and one output).

Table 1: *Computational evaluation of the function* $\mathbf{y} = \mathbf{f}(\mathbf{x})$, $n_x = 2$, $n_y = 1$.

v_{-1}	$= x_1$		
v_0	$= x_2$		
v_i	$= \varphi_i(v_j)_{j<i}$	i	$= 1,\ldots, l$
y	$= v_l$		

In this paper, we rely on AD for the computation of gradients. The forward mode will be used as a reference, as it is able to provide the exact gradient of any magnitude in the solution process with respect to (only one at a time) design parameter, accurate to the same level of accuracy of the primal solver. On the other hand, the reverse mode will allow for the efficient computation of gradients with respect to any number of design variables, in a similar fashion as the adjoint methods described in section II.4. Both methods will be briefly described in the next sections, and the use of Expression Templates [36] for the implementation of the reverse mode will be then outlined. For further reference, the reader can refer to the book by Grienwank [28].

III.1. Forward mode of Algorithmic Differentiation

The application of the chain rule to each elementary functions in Tab. 1 yields the so-called forward mode of AD. This method propagates the tangents of the variables x_i in the numerical solution by evaluating the derivative of each intermediate function φ_i along with the function itself [28, 33]. This is shown in Tab. 2.

Table 2: *Forward mode of AD.*

Function Evaluation		Derivative Evaluation			
v_{-1}	$= x_1$	$\dot{v}_{-1}$	$= \dot{x}_1$		
v_0	$= x_2$	$\dot{v}_0$	$= \dot{x}_2$		
v_i	$= \varphi_i(v_j)_{j<i}$	$\dot{v}_i$	$= \sum_{j<i} \frac{\partial \varphi_i(v_j)}{\partial v_j} \dot{v}_j$	i	$= 1,\ldots, l$
y	$= v_l$	$\dot{y}$	$= \dot{v}_l$		

The forward mode yields the propagation of infinitesimal changes in the input value throughout the evaluation of the function [33]. Noting that it is $\dot{\mathbf{y}} = \frac{d\mathbf{f}}{d\mathbf{x}}\dot{\mathbf{x}}$, we can now directly obtain any

matrix-vector product involving the Jacobian. The method is conceptually similar to the complex step method, as defined in section II.2. The main difference lies in the quantity that is being propagated through the computational solution process, which in this case is no longer a complex increment but rather the value of the derivative itself.

Implementation of the forward mode often leverages advanced capabilities of some programming languages, such as operator overloading in C++. Here, each variable in the original code is augmented to store the value of its derivative, and every operator computes not only the elementary function that it represents, but also its derivative.

An alternative is to use a source-code transformation tool, such as Tapenade [46]. This approach is normally more efficient in terms of memory requirements than operator overloading [28], which may make it more appealing for large problem sizes. This is especially relevant for application in the reverse mode, as explained in the next section. In both cases, access to the source code of the solver is necessary in order to implement the forward mode of AD. Similarly to the complex-step method described in section II.2 and the direct method in section II.3, the cost of the calculation of the gradients using the forward mode is proportional to the number of design variables n_α, as one simulation needs to be run per input parameter being investigated.

III.2. Reverse mode of Algorithmic Differentiation

Analogously to the direct and adjoint methods described in sections II.3 and II.4, in the context of AD, it is also possible to redefine the problem to determine the sensitivities of an output value with respect to a large number of design variables $\boldsymbol{\alpha}$. It is based on a backward application of the chain rule in the sequence of elementary functions of the numerical solution. This technique is known as the reverse mode of AD. Following the description of Grienwank *et al.* [28], we consider the function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ and let $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$ be the dual variables of the inputs and outputs in the forward problem. The overbar effectively indicates the (transpose of) derivative of the output with respect to the corresponding variable. Consequently, the equivalent relation to the adjoint equation in (10) is

$$\bar{\mathbf{x}} \quad = \left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}}\right)^T \bar{\mathbf{y}}, \tag{12}$$

which implies the relation $\bar{\mathbf{y}}^T \dot{\mathbf{y}} = \bar{\mathbf{x}}^T \dot{\mathbf{x}}$ between the primal and dual variables. The transpose in Eq. (12) implies a reverse sequence of operations starting with $\bar{\mathbf{y}}$ [47], which results in the numerical procedure of Tab. 3.

Table 3: *Reverse mode of AD with two inputs and one output.*

v_{-1}	$= x_1$		
v_0	$= x_2$		
v_i	$= \varphi_i(v_j)_{j<i}$	i	$= 1,\ldots, l$
y_1	$= v_l$		
$\bar{v}_l$	$= \bar{y}_1$		
$\bar{v}_j$	$= \bar{v}_j + \bar{v}_i \frac{\partial \varphi_i(v_j)}{\partial v_j},\ i > j$	i	$= l,\ldots, 1$
$\bar{x}_2$	$= \bar{v}_0$		
$\bar{x}_1$	$= \bar{v}_{-1}$		

The backward evaluation of the elementary functions $\varphi_i(v_j)$ requires that the intermediate variables v_j have been previously computed. The reverse mode allows for the calculation of as

many rows of $\frac{\mathrm{df}}{\mathrm{dx}}$ as needed using only one additional computation run backwards through the adjoint path [28], which corresponds to the second block in Table 3. This means that the gradients with respect to any number of input variables can be obtained performing a single run, which has a cost between one to four times that required for the primal problem [28]. However, this is only true for a single output, and the reverse method needs to be rerun as many times as the number of outputs.

The program reversal is the main challenge for an the efficient implementation of the reverse AD method. Both operator overloading and source code transformation, as described in section III.1, may be used, although source code transformation is normally preferred, as explained by Mader *et al.* [30]. The main drawbacks relate to the memory footprint, which is largely increased for both methods with respect to the forward mode due to the need to store intermediate results. Moreover, the use of source code transformation requires a very deep knowledge of the solution process and can hinder the applicability to realistic problems, which often require large, sophisticated software packages that would be difficult to transform. Source code transformation also complicates the addition of new methods to the solver, as they would not be automatically included in the differentiation and would require constant updates of the code.

III.2.1 Expression Templates.

In this work we will implement the reverse mode using Expression Templates (ET), a methodology first proposed by Veldhuizen [48]. ET is an efficient alternative to source code transformation and regular methods for operator overloading, and that can be defined as a meta-programming technique that creates a compile-time parse tree of the overloaded expressions in the code. This method has been successfully applied in the efficient solution of complex PDE systems using optimized code. Some examples of this are the application on the Stokes operator by Pflaum [49], or the application on continuous and discontinuous Galerkin FE by Di Pietro and Veneziani [50].

A very interesting feature of ET is that it allows for the computational graph of any expression to be efficiently recorded and run in reverse mode using an AD tool. An example of implementation of this methodology in C++ libraries for reverse adjoint applications was recently described by Hogan [36]. A similar philosophy has been followed in order to link the AD tool CoDiPack with the PDE solver used in this work, SU2, as discussed in previous works by Albring *et al.* [33, 34]. An example of a function such as that in Tabs. 2 and 3 is presented in Fig. 1 for the computational path of $y = v_0 \cos(v_{-1})$. When using ET, the operators in the code return a relatively small, temporary object instead of the more expensive result of the expression [33, 36]. Each one of these objects has been highlighted over a corresponding arrow in Fig. 1 for this particular example. It must be noted that *su2double* in the figure corresponds to the type that has been defined in SU2 for doubles in the current implementation of AD-based sensitivities.

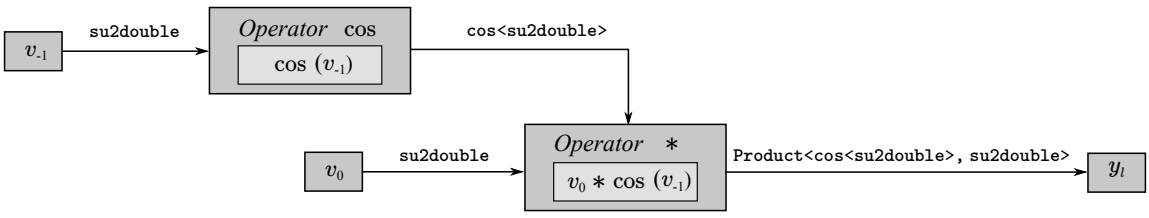

Figure 1: *Computational graph for the expression $y = v_0 \cos(v_{-1})$. Adapted from [33, 36]*

The process of the backwards run of the computational graph in Fig. 1 is presented in Fig. 2.

As shown by Hogan [36], ET is able to improve the efficiency of the reverse mode by dramatically increasing the speed of the backwards run, while also reducing the memory usage with respect to regular operator-overloading libraries. This increase in performance makes ET a very competitive alternative for the application of the reverse mode of AD in complex problems.

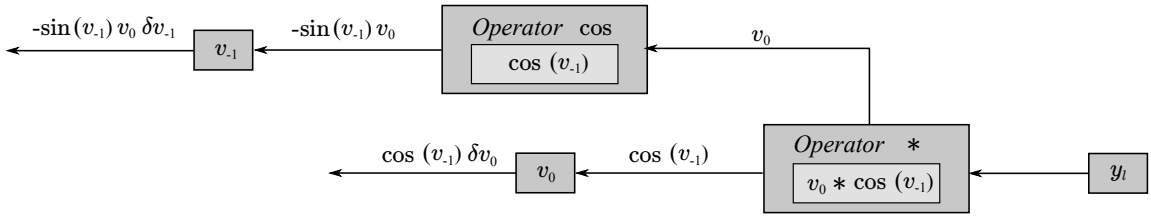

Figure 2: *Propagation of gradients in reverse mode for the expression $y = v_0 \cos(v_{-1})$. Adapted from [33, 36]*

Finally, another advantage of ET for the reverse mode of AD is that, once the framework has been set, its extension to new iterative solvers is almost automatic as only the definition of the inputs and outputs of the solver are required, and the AD tool then intrinsically deals with all the solver operations independently of its physics. This is the case of the fluid-structure interaction solver employed in this work, which is described next.

IV. Fluid-Structure Interaction Problem

The physical problem of interest is the reciprocal relationship that exists between a fluid and a structural domain, governed each by their own constitutive equations, $\mathscr{F}(\mathbf{w},\mathbf{z}) = \mathbf{0}$ and $\mathscr{S}(\mathbf{u},\mathbf{z},\mathbf{w}) = \mathbf{0}$, respectively, and affecting the behavior of each other over a common interface. Finite-volume and finite-element spatial discretizations are on the fluid and the structural domains, respectively.

We define the flow conservative variable vector $\mathbf{w} = \left(\rho_f, \rho_f \mathbf{v}, \rho_f e\right)$, where ρ_f is the fluid density, $\mathbf{v}$ is the vector of flow velocities in a three-dimensional Cartesian coordinate system, and e is the total energy of the flow per unit mass. The position of the fluid mesh nodes of the finite-volume scheme is defined as $\mathbf{z}$. Finally, let $\mathbf{u}$ be the vector of displacements of the structural nodes.

In this work, we adopt a *partitioned*, three-field formulation for the solution of the primal problem, as described in Fig. 3, which has been implemented in SU2 and thoroughly described in a previous paper by Sanchez *et al* [37]. The mesh $\mathscr{M} = \mathscr{M}(\mathbf{z},\mathbf{u})$ becomes a third field of the problem, as large displacements on the structural mesh are subject to change the fluid domain.

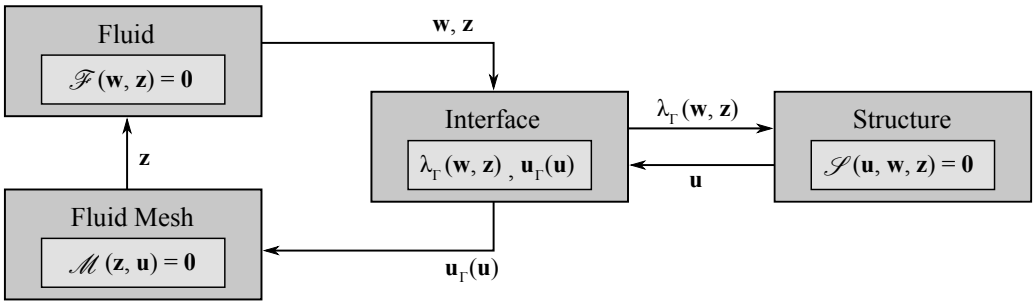

Figure 3: *Three-field formulation for Fluid-Structure Interaction problems.*

This three-field formulation was first proposed by Farhat *et al.* [51] and was further developed by Maute and co-workers [7, 8, 11, 52]. According to Maute *et al.*, [11], this formulation is suitable for problems with large structural deformations, even though it has a higher computational cost than two-field approaches used by Martins *et al.* [9, 10, 16]. The three-field formulation leads to the following governing equations,

$$\mathscr{G} = \mathscr{G}(\mathbf{u}, \mathbf{w}, \mathbf{z}) = \begin{cases} \mathscr{S} & = \mathscr{S}(\mathbf{u}, \quad \mathbf{w}, \quad \mathbf{z}) \quad = \mathbf{0}, \\ \mathscr{F} & = \mathscr{F}(\quad \mathbf{w}, \quad \mathbf{z}) \quad = \mathbf{0}, \\ \mathscr{M} & = \mathscr{M}(\mathbf{u}, \quad \quad \mathbf{z}) \quad = \mathbf{0}. \end{cases} \tag{13}$$

We use an appropriate solver for each subproblem that can treat strong nonlinearities in both the fluid and structural domains. Then, we impose the coupling conditions on the interface and solve the FSI problem using a strongly-coupled, block Gauss-Seidel (BGS) methodology. The solution methods employed for each domain and the coupling strategy will be briefly described next.

IV.1. Fluid domain

The behavior of a viscous, Newtonian flow is governed by the compressible form of the Navier-Stokes equations, [39]

$$\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathbf{F}^c(\mathbf{w}, \mathbf{z}) - \nabla \cdot \mathbf{F}^v(\mathbf{w}, \mathbf{z}) = \mathbf{0}, \tag{14}$$

where $\mathbf{F}^c(\mathbf{w}, \mathbf{z})$ is the vector of convective fluxes and $\mathbf{F}^v(\mathbf{w}, \mathbf{z})$ corresponds to the viscous fluxes. Eq. (14) will be solved using the dual-grid, edge-based discretization implemented in SU2 [39]. The semi-discrete form of the fluid problem can be written as

$$\mathscr{F}(\mathbf{w}, \mathbf{z}) = \int_{\Omega_i} \frac{\partial \mathbf{w}}{\partial t} \mathrm{d}\Omega + \sum_{j \epsilon N(i)} \left[\tilde{\mathbf{F}}^c_{ij}(\mathbf{w}, \mathbf{z}) - \tilde{\mathbf{F}}^v_{ij}(\mathbf{w}, \mathbf{z}) \right] \Delta A_{ij}(\mathbf{z}) = 0, \tag{15}$$

where the summation corresponds to the spatial residual, which is computed and stored at the nodes. Also, $\tilde{\mathbf{F}}^c_{ij}$ and $\tilde{\mathbf{F}}^v_{ij}$ are the numerical convective and viscous fluxes projected over the edge ij, respectively. $\Delta A_{ij}(\mathbf{z})$ is the area of the face associated to the edge ij, and $N(i)$ is the set of nodes in the neighborhood of i. Ω_i is the control volume associated with node i.

After discretizing the time derivative term in Eqn. (15), we recover a fully-discrete finite volume form of the governing equations. The choice of time-marching method depends on whether a steady state or a time-accurate solution is desired. In this work, we will be applying a backward Euler implicit scheme to relax the flow equations to a steady solution, where the time derivative term will vanish.

IV.2. Structural domain

We are interested in problems with large deformations and complex material behavior. For that purpose, we adopt the finite-deformation framework of Bonet and Wood [53]. Ignoring inertial effects, the point-wise equilibrium condition of the continuum is expressed as

$$\mathrm{div}\ \sigma + f = \mathbf{0}. \tag{16}$$

In order to adopt a finite-element solution strategy, a weak form of the equilibrium equation must be obtained. In particular, we will compute the equilibrium in the current configuration

using the principle of virtual work,

$$\delta W = \int_v \boldsymbol{\sigma} : \delta\mathbf{d}\; dv - \left(\int_v \mathbf{f} \cdot \delta\mathbf{v}\; dv + \int_{\partial v} \mathbf{t} \cdot \delta\mathbf{v}\; da \right) = 0, \tag{17}$$

where the Cauchy stress tensor $\boldsymbol{\sigma}$ and the virtual rate of deformation tensor $\delta\mathbf{d}$ determine the internal component of the virtual work, δW. The external component depends on the actuating forces per unit volume $\mathbf{f}$, and the applied surface tractions per unit area $\mathbf{t}$. $\delta\mathbf{v}$ is a virtual velocity field $\mathbf{v}$ on the current configuration of the body. The integrals are defined over the current volume of the body, v, and its external surface, ∂v.

Eq. (17) is discretized using isoparametric finite elements. The static equilibrium can then be written as

$$\mathscr{S}(\mathbf{u}) = \mathbf{T}(\mathbf{u}) - \mathbf{F}_b - \mathbf{F}_\Gamma(\mathbf{u}) = \mathbf{0}, \tag{18}$$

where $\mathbf{T}(\mathbf{u})$ are the internal equivalent forces, $\mathbf{F}_b$ the body forces and $\mathbf{F}_\Gamma(\mathbf{u})$ the surface forces acting over the boundary Γ. Equation (18) is solved iteratively using a Newton method,

$$\mathbf{K}^i \Delta\mathbf{u}^{i+1} = -\mathscr{S}(\mathbf{u}^i), \tag{19a}$$

$$\mathbf{u}^{i+1} = \mathbf{u}^i + \Delta\mathbf{u}^{i+1}, \tag{19b}$$

until $\mathscr{S}(\mathbf{u}) \approx 0$. The tangent matrix $\mathbf{K} = \partial\mathscr{S}(\mathbf{u})/\partial\mathbf{u}$ may be assembled as [53]

$$\mathbf{K} = \mathbf{K}_c + \mathbf{K}_\sigma - \mathbf{K}_\Gamma, \tag{20}$$

where $\mathbf{K}_c$ and $\mathbf{K}_\sigma$ are, respectively, the constitutive and initial stress terms arising from the linearization of the first term in (17). The term $\mathbf{K}_\Gamma$ results from the linearization of the vector of surface forces $\mathbf{F}_\Gamma(\mathbf{u})$ described in (18), and it is normally non-symmetric [53]. The body loads $\mathbf{F}_b$ do not contribute to the tangent matrix.

IV.3. Mesh domain

Mesh deformations are obtained by solving a pseudo-elastic linear problem [11]. If $\tilde{\mathbf{K}}_m$ is the fictitious stiffness matrix of the mesh and $\tilde{\mathbf{f}}$ is a fictitious force to enforce the boundary displacements, the resulting linear elasticity problem is written as

$$\mathscr{M}(\mathbf{u}, \mathbf{z}) = \tilde{\mathbf{K}}_m \mathbf{z} - \tilde{\mathbf{f}}(\mathbf{u}) = \mathbf{0}. \tag{21}$$

IV.4. Coupling conditions

The coupling between both domains needs to be appropriately defined over the common interface [54–56]. Continuity of displacements is imposed on the FSI interface Γ_i as

$$\mathbf{u}_\Gamma = \mathbf{z}_\Gamma \quad \text{on } \Gamma_i. \tag{22}$$

The tractions $\boldsymbol{\lambda}_f$ on the fluid side of the interface may be defined as

$$\boldsymbol{\lambda}_f = -p\mathbf{n}_f + \boldsymbol{\tau}_f \mathbf{n}_f \quad \text{on } \Gamma_{f,i}, \tag{23}$$

where $\mathbf{n}_f$ is the dimensional normal oriented outward from the surface, including area information.

Analogously for the structural domain, $\boldsymbol{\lambda}_s$ is defined as

$$\boldsymbol{\lambda}_s = \boldsymbol{\sigma}_s \mathbf{n}_s \quad \text{on } \Gamma_{s,i} \tag{24}$$

Imposing now equilibrium conditions as

$$\boldsymbol{\lambda}_f + \boldsymbol{\lambda}_s = 0, \tag{25}$$

the problem may be rewritten as a fixed-point equation using the Dirichlet-to-Neumann nonlinear operators as in [54, 56]

$$\begin{cases} \boldsymbol{\lambda}_f = \mathscr{D}_F(\mathbf{z}_\Gamma) & \text{on } \Gamma_{f,i} \\ \boldsymbol{\lambda}_s = \mathscr{D}_S(\mathbf{u}_\Gamma) & \text{on } \Gamma_{s,i}. \end{cases} \tag{26}$$

Combining (22) and (25) while defining the inverse Neumann-to-Dirichlet operator on the structural problem $\mathscr{D}_S^{-1}(\mathbf{u}_\Gamma)$, we recover the following fixed-point iteration:

$$\mathscr{D}_S^{-1}(-\mathscr{D}_F(\mathbf{u}_\Gamma)) = \mathbf{u}_\Gamma. \tag{27}$$

IV.5. Solution method

The governing equations in Eq. (13) are now linearized as

$$\begin{bmatrix} \mathscr{S} \\ \mathscr{F} \\ \mathscr{M} \end{bmatrix} + \begin{bmatrix} \frac{\partial \mathscr{S}}{\partial \mathbf{u}} & \frac{\partial \mathscr{S}}{\partial \mathbf{w}} & \frac{\partial \mathscr{S}}{\partial \mathbf{z}} \\ \mathbf{0} & \frac{\partial \mathscr{F}}{\partial \mathbf{w}} & \frac{\partial \mathscr{F}}{\partial \mathbf{z}} \\ \frac{\partial \mathscr{M}}{\partial \mathbf{u}} & \mathbf{0} & \frac{\partial \mathscr{M}}{\partial \mathbf{z}} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \mathbf{w} \\ \Delta \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{28}$$

In order to solve (28), we adopt a strongly-coupled strategy [6, 56–63]. We follow the implemementation described by Barcelos and Maute [6], who propose nonlinear BGS (NLBGS) iterations, where the upper diagonal terms that couple the subproblems are neglected. The fluid, structural and mesh subproblems are solved sequentially, and iterations are performed until convergence. Here, the convergence criterion will be that a certain tolerance ϵ is met in the residual of the displacements on the interface, as $\mathscr{R}_\mathbf{u} = \left\| \mathbf{u}_\Gamma^{n+1} - \mathbf{u}_\Gamma^n \right\| < \epsilon$.

We have included in the formulation a relaxation parameter, ω, which has been shown to help preventing divergence in BGS schemes [60], as long as it is sufficiently small. We have also introduced a polynomial coefficient that transfers the loads to the structural domain incrementally during the first N BGS subiterations, $\gamma^n = -2(n/N)^3 + 3(n/N)^2$. This coefficient prevents from a large imbalance between external and internal equivalent forces in the structural residual in the first subiterations, and thus avoids overshooting of the solution process. We have also observed that the final deformed fluid mesh, in problems with large structural displacements, maintains a

better quality if the loads are transferred incrementally. The full approach is described in Alg. 1.

$n \leftarrow 0$;

$\tilde{\mathbf{u}}_\Gamma^0 = \mathbf{0}$;

while $n < n_{max}$ **do**

  Mesh update: $\mathscr{M}(\tilde{\mathbf{u}}_\Gamma^n(\mathbf{u}), \mathbf{z}) = \mathbf{0}$

  Fluid solver: $\boldsymbol{\lambda}_{f,\Gamma}(p_\Gamma, \boldsymbol{\tau}_\Gamma) = \mathscr{D}_F(\tilde{\mathbf{u}}_\Gamma^n)$

  Structural solver: $\mathbf{u}_\Gamma^{n+1} = \mathscr{D}_S^{-1}(-\gamma^n \boldsymbol{\lambda}_{f,\Gamma})$

  Relaxation: $\tilde{\mathbf{u}}_\Gamma^{n+1} = (1-\omega)\tilde{\mathbf{u}}_\Gamma^n + \omega \mathbf{u}_\Gamma^{n+1}$

  if $\left\|\tilde{\mathbf{u}}_\Gamma^{n+1} - \tilde{\mathbf{u}}_\Gamma^n\right\| < \epsilon$ **then**

    break

  else

    $n \leftarrow n+1$

  end

end

Algorithm 1: Block Gauss-Seidel algorithm for the FSI problem.

V. Adjoint problem

With the solution method for the primal problem defined, we will now describe the proposed approach to compute sensitivities using an adjoint-based strategy with Algorithmic Differentiation. We will begin in section V.1 by describing the application of the classic direct and adjoint methods to a single-discipline structural problem. Some considerations will be made in this section that will be later extended to the coupled problem in section V.2. A new approach to the computation of gradients using AD will be introduced in section V.3, and lastly, its extension to the fully-coupled FSI problem will be developed in section V.4.

For the purpose of the discussion in this paper, we will restrict ourselves to objective functions of the form

$$J(\mathbf{u}) = (\mathbf{u} - \mathbf{u}_{ref})^T \mathbf{C} (\mathbf{u} - \mathbf{u}_{ref}), \tag{29}$$

which defines a measure of the deformation of the structure $\mathbf{u}$ with respect to a reference shape $\mathbf{u}_{ref}$. In (29), $\mathbf{C}$ is a weighting matrix of size is $n_u \times n_u$, where n_u is the number of degrees of freedom in the structural problem.

V.1. Application of the direct and adjoint method to a single-discipline structural problem

We first consider the structure problem separately, in order to outline the main features of the solution procedure in a simple setting, and will demonstrate the approach on hyperelastic structures built of a single isotropic material. Its Young's modulus E is selected as the design variable for which we want to obtain sensitivities. Recalling the definition of the non-linear structural problem $\mathscr{S}(\mathbf{u})$ from (18), we obtain its partial derivative with respect to E as

$$\frac{\partial \mathscr{S}(\mathbf{u})}{\partial E} = \frac{\partial \mathbf{T}(\mathbf{u})}{\partial E} = \int_v \frac{\partial \boldsymbol{\sigma}}{\partial E} : \delta \mathbf{d} \, dv. \tag{30}$$

We then define the Cauchy stress tensor for the Neo-Hookean material model [53] as

$$\boldsymbol{\sigma}_{NH} = \frac{\mu}{j}(\mathbf{b} - \mathbf{I}) + \frac{\lambda}{j}(\ln j)\mathbf{I}, \tag{31}$$

where j is the determinant of the deformation gradient of the structural problem, $\mathbf{b}$ is the left Cauchy-Green deformation tensor, and λ and μ are defined as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad ; \quad \mu = \frac{E}{2(1+\nu)}. \tag{32}$$

The only terms in (31) that depend explicitly on E are λ and μ, which allows us to compute $\frac{\partial \boldsymbol{\sigma}_{NH}}{\partial E}$ analytically. Given that E does not appear explicitly in the objective function, we have $\partial J/\partial E = 0$ and can solve the direct problem as

$$\frac{\mathrm{d}J}{\mathrm{d}E} = \frac{\partial J}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}E}, \tag{33}$$

where the first term can be analytically obtained from (29) to give

$$\frac{\partial J}{\partial \mathbf{u}} = 2k(\mathbf{u} - \mathbf{u}_{ref})^T\mathbf{C}. \tag{34}$$

while the second term is computed from

$$\mathbf{K}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}E} = -\int_v \frac{\partial \boldsymbol{\sigma}}{\partial E} : \delta\mathbf{d}\, dv = -\frac{\partial \mathscr{S}(\mathbf{u})}{\partial E}, \tag{35}$$

where the right-hand side corresponds to a column vector of dimension n_u.

Alternatively, the problem may be solved using the adjoint equations (10) and (11). The adjoint vector is obtained from

$$\mathbf{K}^T\,\bar{\mathbf{u}} = -2(\mathbf{u} - \mathbf{u}_{ref})^T\mathbf{C}, \tag{36}$$

and the gradient is computed as

$$\frac{\mathrm{d}J}{\mathrm{d}E} = \bar{\mathbf{u}}^T\int_v \frac{\partial \boldsymbol{\sigma}}{\partial E} : \delta\mathbf{d}\, dv = \bar{\mathbf{u}}^T\frac{\partial \mathscr{S}(\mathbf{u})}{\partial E}. \tag{37}$$

The computational cost of the direct and adjoint problems is, for this case, equivalent. However, now let $\boldsymbol{\alpha} = [E_1, E_2, ..., E_n, \nu_1, \nu_2, ..., \nu_n]$ be the vector of design variables, where E_i and ν_i are, respectively, the Young's modulus and the Poisson ratio for any given element i, $i \in n$, and n is the total number of elements in the structural domain. In this case with the direct method, we need to solve $2n$ linear systems of equations

$$\mathbf{K}\left[\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}E_1}, ..., \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}E_n}, \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\nu_1}, ..., \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\nu_n}\right] = -\left[\frac{\partial \mathscr{S}(\mathbf{u})}{\partial E_1}, ..., \frac{\partial \mathscr{S}(\mathbf{u})}{\partial E_n}, \frac{\partial \mathscr{S}(\mathbf{u})}{\partial \nu_1}, ..., \frac{\partial \mathscr{S}(\mathbf{u})}{\partial \nu_n}\right] \tag{38}$$

to compute the gradient $\mathrm{d}J/\mathrm{d}\boldsymbol{\alpha}$ using (6). If $\mathbf{K}$ has not been factorized, which is typically the case for the large problems of interest here, the cost of solving this problem would be equivalent to the cost of solving $2n$ times (35). For the adjoint method, on the other hand, it is only necessary to calculate $2n$ vector products in (37):

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = -\bar{\mathbf{u}}^T\left[\frac{\partial \mathscr{S}(\mathbf{u})}{\partial E_1}, ..., \frac{\partial \mathscr{S}(\mathbf{u})}{\partial E_n}, \frac{\partial \mathscr{S}(\mathbf{u})}{\partial \nu_1}, ..., \frac{\partial \mathscr{S}(\mathbf{u})}{\partial \nu_n}\right]. \tag{39}$$

For linear structural problems, the matrix $\mathbf{K}$ is symmetric and only retains the term $\mathbf{K}_c$ in (20). Therefore, the problem is self-adjoint. This property also holds for geometrically non-linear problems involving only the effects of dead loads, which have no contribution to the Jacobian. However, it is no longer valid for problems involving follower forces (e.g., surface pressures), which introduce a non-symmetric term in the tangent matrix as explained in section IV.2 above. In order to converge the equilibrium equations defined in (19), this contribution may be neglected, since convergence can often still be achieved using an approximate Jacobian. The implications of directly using this approximate tangent matrix for the adjoint problem are more serious and will be discussed later in section VI.2.

The calculation of the analytical tangent matrix is relatively straight-forward for structures with simple material models. However, for strongly non-linear materials with a complex definition of the Cauchy stress tensor $\boldsymbol{\sigma}$, the linearization process required to obtain the exact analytic representation of the tangent stiffness matrix is non-trivial, and in most cases it is cumbersome and error-prone (e.g., see [64, 65]). This can be overcome using Algorithm Differentiation, as will be shown in section V.3 below.

V.2. Discrete adjoint solution of Fluid-Structure Interaction problems with analytic Jacobians

The formulation of the adjoint problem for Fluid-Structure Interaction applications may be obtained by combining (10) and (28) as

$$\mathbf{K}_{\mathscr{G}}^{T}\,\bar{\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathscr{S}}{\partial \mathbf{u}}^{T} & \mathbf{0} & \frac{\partial \mathscr{M}}{\partial \mathbf{u}}^{T} \\ \frac{\partial \mathscr{S}}{\partial \mathbf{w}}^{T} & \frac{\partial \mathscr{F}}{\partial \mathbf{w}}^{T} & \mathbf{0} \\ \frac{\partial \mathscr{S}}{\partial \mathbf{z}}^{T} & \frac{\partial \mathscr{F}}{\partial \mathbf{z}}^{T} & \frac{\partial \mathscr{M}}{\partial \mathbf{z}}^{T} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{u}} \\ \bar{\mathbf{w}} \\ \bar{\mathbf{z}} \end{bmatrix} = - \begin{bmatrix} \frac{\partial J}{\partial \mathbf{u}}^{T} \\ \frac{\partial J}{\partial \mathbf{w}}^{T} \\ \frac{\partial J}{\partial \mathbf{z}}^{T} \end{bmatrix} = - \frac{\partial J}{\partial \mathbf{x}}^{T}. \tag{40}$$

Eq. (40) warrants further discussion. Starting from the terms in the diagonal, it must be noted that the structural equation now depends explicitly on the mesh and fluid variables,

$$\mathscr{S}(\mathbf{u}, \mathbf{z}, \mathbf{w}) = \mathbf{T}(\mathbf{u}) - \mathbf{F}_b - \mathbf{F}_\Gamma(\mathbf{u}, \mathbf{z}, \mathbf{w}) = \mathbf{0}, \tag{41}$$

where the surface forces $\mathbf{F}_\Gamma(\mathbf{u}, \mathbf{z}, \mathbf{w})$ in (18) are now a function of the aerodynamic forces and the position and the normals of the fluid wet surface. Kenway *et al.* [16] point out that the effect of the changes in the normals with the applied forces, or $\partial \mathbf{F}_\Gamma / \partial \mathbf{u}$, must be taken into account. This effect is equivalent to the directional derivative of the external work in the direction of an infinitesimal perturbation of the structural displacements $D\delta W_{ext}(\phi_k, \delta\mathbf{v})[\Delta\mathbf{u}]$.

With regard to the off-diagonal terms, their calculation is normally convoluted and requires special attention. In the solution of the primal problem, (28), the objective is to iteratively obtain the increments of the state variables $\Delta\mathbf{x}$ that minimize the value of the residual. Therefore, it is generally sufficient to employ an approximate tangent matrix, which is good enough to achieve convergence. However, the adjoint problem (40) is a system of linear equations in which the matrix of coefficients of the system must be computed exactly. Although it is still possible to solve the problem using an iterative BGS-like approach such as the Lagged Coupled Adjoint (LCA) method proposed by Martins *et al.* in [12], it is necessary to add the contributions of the all the cross terms in the matrix to the right-hand side of the reduced systems of equations, and then iterate until convergence.

In particular, the terms $\partial\mathscr{S}/\partial\mathbf{z}$ and $\partial\mathscr{S}/\partial\mathbf{w}$ correspond to the derivatives of the aerodynamic forces with respect to the fluid mesh positions $\partial\mathbf{F}_\Gamma/\partial\mathbf{z}$ and to the fluid variables $\partial\mathbf{F}_\Gamma/\partial\mathbf{w}$ respectively. The term $\partial\mathscr{M}/\partial\mathbf{u}$ may be obtained from (21) as $\partial\tilde{\mathbf{f}}/\partial\mathbf{u}$. Finally, the term $\partial\mathscr{F}/\partial\mathbf{z}$ accounts for the sensitivity of the fluid problem with respect to variations in the positions of the fluid mesh, both in the FSI interface and its internal nodes due to the ALE formulation.

The computation of these terms becomes one of the most important tasks in the solution of the adjoint FSI problem. If possible, they should be computed analytically. However, this is not always an option, and therefore, some alternatives arise [12]. One of the most common approaches is finite differences, due to their ease of implementation. A more accurate, yet still simple to implement, method to compute such sensitivities is the use of the complex-step derivative approximation [5]. Finally, it is also possible to use AD [27, 28] both in its forward and reverse modes. All of these options, except for the reverse mode of AD, require one computation per state variable, which may be prohibitively expensive for realistic applications [5].

Finally, it must be noted that the need for an exact analytic tangent matrix for the adjoint problem also applies to the diagonal terms in (40). The Jacobian in the adjoint equation is the matrix of the linear system (10), and thus changes in the matrix will result in change sin the solution. On the contrary, exact Jacobians are not needed while converging a non-linear problem using Newton iterations, as the objective in that case is not to solve a linear system but to minimize the residual. For the structural problem, this was discussed in section V.1, and can become an issue for complex material models. Analogously, the exact Jacobian of the flow problem, especially when viscous and turbulent effects are included, is not typically available due to memory requirements and low-order approximations commonly introduced in the numerical solution of the primal problem.

V.3. Adjoint solution of a structural problem using Algorithmic Differentiation

Optimization problem (3), when particularized for a non-linear structural problem, can be written as

$$\begin{aligned} &\min\ J(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) \\ &\text{subject to}\ \ \mathscr{S}(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) = 0, \end{aligned} \tag{42}$$

where $\mathscr{S}(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha})$ is described in (18). The linearization of (18) yields the Newton iteration in (19), where as previously explained in section V.1, the matrix $\mathbf{K}$ may not necessarily be the *exact* Jacobian $\partial\mathscr{S}/\partial\mathbf{u}$ of the problem. If convergence is achieved, the solution $\mathbf{u}^*$ will only depend on the definition of the residual and not the Jacobian. The problem may be rewritten as a fixed-point iteration [34]:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \mathbf{K}^{-1}(\mathbf{u}^n)\mathscr{S}(\mathbf{u}^n,\boldsymbol{\alpha}) =: \mathbf{S}(\mathbf{u}^n,\boldsymbol{\alpha}), \tag{43}$$

where the operator $\mathbf{S}(\mathbf{u}^n,\boldsymbol{\alpha})$ is only feasible at the solution $\mathbf{u}^*$, i.e.,

$$\mathscr{S}(\mathbf{u}^*,\boldsymbol{\alpha}) = \mathbf{0} \Leftrightarrow \mathbf{u}^* = \mathbf{S}(\mathbf{u}^*,\boldsymbol{\alpha}). \tag{44}$$

According to the Banach fixed-point theorem, the problem defined by (43) converges if $\mathbf{S}(\mathbf{u}^n\boldsymbol{\alpha})$ is contractive, which means here that, in a suitable matrix norm, it is [34]

$$\left\| \frac{\partial\mathbf{S}}{\partial\mathbf{u}} \right\| < 1. \tag{45}$$

As already mentioned, the traditional adjoint approach requires the construction of the exact Jacobian $\mathbf{K}$. This task is normally complex, and it may not even be possible in some cases. To avoid

this issue, we pursue the procedure proposed by Korivi *et al.* [66] and adopted by Albring *et al.* [34] for CFD problems that permits the use of an approximate Jacobian and efficiently evaluates the gradients using AD.

In this scenario, the optimization problem takes the form

$$\begin{aligned} &\min\ J(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha}) \\ &\text{subject to}\ \ \mathbf{u}(\boldsymbol{\alpha}) = \mathbf{S}(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha}). \end{aligned} \tag{46}$$

We can define the Lagrangian associated to this problem as

$$L(\boldsymbol{\alpha},\mathbf{u},\bar{\mathbf{u}}) = J(\mathbf{u},\boldsymbol{\alpha}) + \bar{\mathbf{u}}^T[\mathbf{S}(\mathbf{u},\boldsymbol{\alpha}) - \mathbf{u}]. \tag{47}$$

Differentiating the Lagrangian with respect to the state variables, one obtains

$$\bar{\mathbf{u}}^T = \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u},\boldsymbol{\alpha}) + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{u}}(\mathbf{u},\boldsymbol{\alpha}), \tag{48}$$

which allows us to define the following fixed-point iteration in $\bar{\mathbf{u}}$, with the derivatives of J and $\mathbf{S}$ evaluated around the previously converged (feasible) solution (44), as proposed in Ref. [34]:

$$\bar{\mathbf{u}}^{n+1} = \left[\frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}^*,\boldsymbol{\alpha})\right]^T + \left[\frac{\partial \mathbf{S}}{\partial \mathbf{u}}(\mathbf{u}^*,\boldsymbol{\alpha})\right]^T \bar{\mathbf{u}}^n = \bar{\mathbf{S}}(\bar{\mathbf{u}}^n). \tag{49}$$

It can be shown that $\bar{\mathbf{S}}$ is contractive provided that $\mathbf{S}$ is contractive [33], although, in general, this cannot be guaranteed *a priori* for the primal solver in coupled problems and so far has been checked numerically. Once the Lagrange multipliers have been obtained, the total derivative of the objective function with respect to the design parameters is, from Eq. (11),

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \boldsymbol{\alpha}}. \tag{50}$$

Note that, for the objective function defined in (29), the first term in (50) cancels out. Albring *et al.* [33,34] have further shown that the right-hand side of (49) can be easily evaluated using AD of a numerical solution process for $\mathbf{S}(\mathbf{u}(\boldsymbol{\alpha}),\boldsymbol{\alpha})$. For that purpose, we have linked the non-linear structural solver developed within the open-source SU2 suite [37] with the also open-source C++ library *Code-Differentiation Package* (CoDiPack[2]). CoDiPack provides a special datatype for use within SU2 and the ability to run the reverse mode of AD using the Expression Templates method [36]. A top-level approach to the procedure is shown in Fig. 4. For additional details on the implementation of the AD-tool in SU2, the reader is referred to the work by Albring *et al.* [33, 34] and Zhou *et al.* [35].

When solving an AD-based Discrete Adjoint (ADDA) problem in SU2, the first step is to load the converged solution of the non-linear structural problem (primal problem), $\mathbf{u}^*$. This solution must have been previously computed using the structural solver in SU2. Then, the computational graph of the primal problem is recorded using CoDiPack, which corresponds to the grey-shaded area indicated as RECORDING in Fig. 4. The state variables $\mathbf{u}$ and the design variables $\boldsymbol{\alpha}$ are registered in the AD tool as the *input* of the fixed-point solver (43). One iteration of the structural solver is executed, obtaining a new solution $\mathbf{u}^* + \Delta\mathbf{u}^* \approx \mathbf{u}^*$. If the primal problem has converged, the residual of this new solution should be practically negligible. The objective function is now evaluated using

[2] http://www.scicomp.uni-kl.de/software/codi/

this updated solution, where $J(\mathbf{u}^* + \Delta\mathbf{u}^*) \approx J(\mathbf{u}^*)$. Finally, and before stopping the recording process, the displacement variables are registered in the AD tool as the *output* of the primal iteration, as these are the variables for which the adjoint, $\bar{\mathbf{u}}$, will be computed.

The vector of the adjoint variables is then initialized as $\bar{\mathbf{u}}^0$, and the fixed-point iterations in (49) are carried out using the recorded computational graph until convergence, when $\left\|\bar{\mathbf{u}}^{n+1} - \bar{\mathbf{u}}^n\right\| < \epsilon$. The resulting adjoint $\bar{\mathbf{u}}$ is then used in (50) to obtain the sensitivities with respect to the design variables.

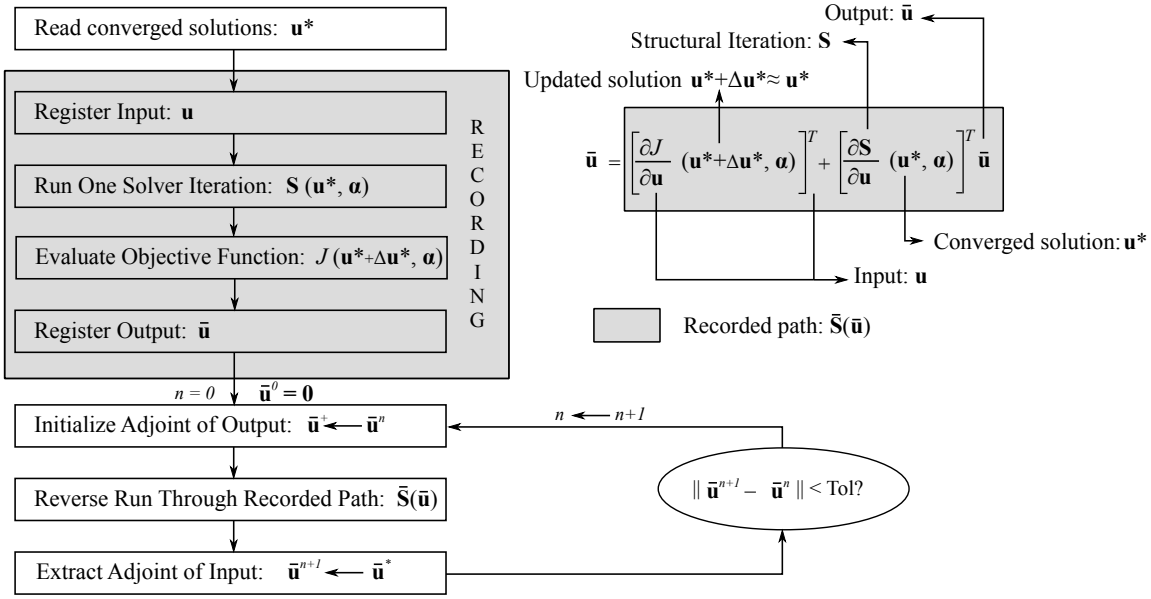

Figure 4: *Top-level approach to the structural adjoint calculation using AD*

V.4. Fully-coupled adjoint solution of a Fluid-Structure Interaction problem using Algorithmic Differentiation

The complexity associated with the solution of the fully-coupled adjoint solution of a Fluid-Structure Interaction problem has already been discussed in section V.2. In particular, the BGS Alg. 1 used to solve the primal problem in (28) never constructs the *exact* Jacobian of the coupled problem, a very difficult task in most realistic cases. This Jacobian is needed to compute the adjoint variables in (40), or alternatively, all of the off-diagonal terms have to be computed. To overcome some of these difficulties, we present a consistent methodology that extends the procedure presented in section V.3, and in previous papers by Albring *et al.* [33, 34] and Zhou *et al.* [35], to fully-coupled, Fluid-Structure Interaction problems. Each of the three sub-problems defined in (13) is now rewritten in the form of fixed-point iterations. The structural problem depends on its own state variables $\mathbf{u}$, on the flow state $\mathbf{w}$, on the positions of the flow nodes $\mathbf{z}$, and may also depend explicitly on the design variables $\boldsymbol{\alpha}$:

$$\mathscr{S}(\mathbf{u}^*, \mathbf{z}^*, \mathbf{w}^*, \boldsymbol{\alpha}) = \mathbf{0} \Leftrightarrow \mathbf{u}^* = \mathbf{S}(\mathbf{u}^*, \mathbf{z}^*, \mathbf{w}^*, \boldsymbol{\alpha}). \tag{51}$$

The fluid problem is dependent on its own conservative variables and the position of the nodes both on the FSI interface ($\mathbf{z}_\Gamma \in \mathbf{z}$) and the interior of the domain ($\mathbf{z}_\Omega \in \mathbf{z}$). For the purpose of

generality, an explicit dependency on the design variables $\boldsymbol{\alpha}$ will also be considered:

$$\mathscr{F}(\mathbf{w}^*, \mathbf{z}^*, \boldsymbol{\alpha}) = \mathbf{0} \Leftrightarrow \mathbf{w}^* = \mathbf{G}(\mathbf{w}^*, \mathbf{z}^*, \boldsymbol{\alpha}). \tag{52}$$

Finally, the mesh problem depends on the position of the interface, which is determined by the structural domain ($\mathbf{u}_\Gamma \in \mathbf{u}$). As described in (21), the problem is linear. Therefore, the dependency on the node positions $\mathbf{z}$ is explicit:

$$\mathscr{M}(\mathbf{u}^*, \mathbf{z}^*) = \tilde{\mathbf{K}}_m \mathbf{z}^* - \tilde{\mathbf{f}}(\mathbf{u}^*) = \mathbf{0} \Leftrightarrow \mathbf{z}^* = \tilde{\mathbf{K}}_m^{-1} \tilde{\mathbf{f}}(\mathbf{u}^*) \Leftrightarrow \mathbf{z}^* = \mathbf{M}(\mathbf{u}^*, \boldsymbol{\alpha}), \tag{53}$$

where a generic explicit dependency on the design variables $\boldsymbol{\alpha}$ has been included.

The overall optimization problem takes the form

$$\begin{aligned} \min \quad & J(\mathbf{w}(\boldsymbol{\alpha}), \mathbf{z}(\boldsymbol{\alpha}), \mathbf{u}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ \text{subject to} \quad & \mathbf{u}(\boldsymbol{\alpha}) = \mathbf{S}(\mathbf{u}(\boldsymbol{\alpha}), \mathbf{w}(\boldsymbol{\alpha}), \mathbf{z}(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \\ & \mathbf{w}(\boldsymbol{\alpha}) = \mathbf{G}(\mathbf{w}(\boldsymbol{\alpha}), \mathbf{z}(\boldsymbol{\alpha}), \boldsymbol{\alpha}), \\ & \mathbf{z}(\boldsymbol{\alpha}) = \mathbf{M}(\mathbf{u}(\boldsymbol{\alpha}), \boldsymbol{\alpha}). \end{aligned} \tag{54}$$

We can now define the Lagrangian of the coupled problem as

$$\begin{aligned} L(\boldsymbol{\alpha}, \mathbf{u}, \bar{\mathbf{u}}, \mathbf{w}, \bar{\mathbf{w}}, \mathbf{z}, \bar{\mathbf{z}}) = J(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{w}}^T [\mathbf{G}(\mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) - \mathbf{w}] \\ + \bar{\mathbf{u}}^T [\mathbf{S}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) - \mathbf{u}] + \bar{\mathbf{z}}^T [\mathbf{M}(\mathbf{u}, \boldsymbol{\alpha}) - \mathbf{z}], \end{aligned} \tag{55}$$

and the gradient of the objective function can be then evaluated using the adjoint variables as

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J}{\partial \boldsymbol{\alpha}} + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \boldsymbol{\alpha}} + \bar{\mathbf{w}}^T \frac{\partial \mathbf{G}}{\partial \boldsymbol{\alpha}} + \bar{\mathbf{z}}^T \frac{\partial \mathbf{M}}{\partial \boldsymbol{\alpha}}. \tag{56}$$

We evaluate the derivatives of L around the feasible solution of the FSI problem, defined by $\mathbf{u}^*$, $\mathbf{w}^*$, and $\mathbf{z}^*$, in a similar fashion as in (49). Differentiating (55) with respect to the structural variables $\mathbf{u}$, the flow conservative variables $\mathbf{w}$, and the mesh variables $\mathbf{z}$, we obtain implicit equations for their adjoints, as

$$\begin{aligned} \bar{\mathbf{u}}^T &= \frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{u}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{z}}^T \frac{\partial \mathbf{M}}{\partial \mathbf{u}}(\mathbf{u}), \\ \bar{\mathbf{w}}^T &= \frac{\partial J}{\partial \mathbf{w}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{w}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{w}}(\mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{w}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}), \\ \bar{\mathbf{z}}^T &= \frac{\partial J}{\partial \mathbf{z}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{w}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{z}}(\mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}) + \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{z}}(\mathbf{u}, \mathbf{w}, \mathbf{z}, \boldsymbol{\alpha}). \end{aligned} \tag{57}$$

Eqn. (57) is a linear system in the adjoint variables. However, the partial derivatives of $\mathbf{S}$, $\mathbf{G}$ and $\mathbf{M}$ in the equation are not directly computed, as this will result in very large memory requirements, and instead CoDiPack returns the matrix-vector products of these derivatives multiplied by the adjoint variables. As a result, the solution of (57) is sought again using an iterative procedure. For that purpose, we first redefine the cross-dependencies in (57) as

$$\bar{\mathbf{w}}_{\bar{\mathbf{u}}}^T = \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{w}} \quad ; \qquad \bar{\mathbf{u}}_{\bar{\mathbf{z}}}^T = \bar{\mathbf{z}}^T \frac{\partial \mathbf{M}}{\partial \mathbf{u}} \quad ; \qquad \bar{\mathbf{z}}_{\bar{\mathbf{w}}}^T = \bar{\mathbf{w}}^T \frac{\partial \mathbf{G}}{\partial \mathbf{z}} \quad ; \qquad \bar{\mathbf{z}}_{\bar{\mathbf{u}}}^T = \bar{\mathbf{u}}^T \frac{\partial \mathbf{S}}{\partial \mathbf{z}}, \tag{58}$$

which mathematically resemble the coupling effects in the solution of the primal problem. Next,

we approximate Eq. (57) by the fixed-point iteration

$$\bar{\mathbf{x}}^{n+1} = \bar{\mathscr{G}}(\bar{\mathbf{x}}^n) \Leftrightarrow \begin{cases} \bar{\mathbf{u}}^{n+1} & = \tilde{\mathbf{S}}(\bar{\mathbf{u}}^n, \bar{\mathbf{u}}_{\bar{\mathbf{z}}}) & = \dfrac{\partial J}{\partial \mathbf{u}}^T + \dfrac{\partial \mathbf{S}}{\partial \mathbf{u}}^T \bar{\mathbf{u}}^n + \bar{\mathbf{u}}_{\bar{\mathbf{z}}} \,, \\ \bar{\mathbf{w}}^{n+1} & = \tilde{\mathbf{G}}(\bar{\mathbf{w}}^n, \bar{\mathbf{w}}_{\bar{\mathbf{u}}}) & = \dfrac{\partial J}{\partial \mathbf{w}}^T + \dfrac{\partial \mathbf{G}}{\partial \mathbf{w}}^T \bar{\mathbf{w}}^n + \bar{\mathbf{w}}_{\bar{\mathbf{u}}} \,, \\ \bar{\mathbf{z}} & = \tilde{\mathbf{M}}(\bar{\mathbf{z}}_{\bar{\mathbf{u}}}, \bar{\mathbf{z}}_{\bar{\mathbf{w}}}) & = \dfrac{\partial J}{\partial \mathbf{z}}^T + \bar{\mathbf{z}}_{\bar{\mathbf{u}}} + \bar{\mathbf{z}}_{\bar{\mathbf{w}}} \,. \end{cases} \tag{59}$$

For the solution of (59), we adopt a block Gauss-Seidel procedure, analogous to the Lagged Coupled Adjoint method proposed by Martins *et al.* [12]. Barcelos and Maute [6] used a very similar method for the computation of FSI sensitivities using a 3-field formulation and the direct approach, which also requires incorporating coupled terms to the formulation. This method is described in Alg. 2.

$k \leftarrow 0$;
$\bar{\mathbf{x}}_0 = \left[\bar{\mathbf{u}}_0, \bar{\mathbf{w}}_0, \bar{\mathbf{z}}_0\right] = \mathbf{0}$
$\bar{\mathbf{u}}_{\bar{\mathbf{z}},0} = \mathbf{0}$
while $k < k_{max}$ **do**
  $k \leftarrow k + 1$
  Iterate structural adjoint: $\bar{\mathbf{u}}_k^{n+1} = \tilde{\mathbf{S}}(\bar{\mathbf{u}}_k^n, \bar{\mathbf{u}}_{\bar{\mathbf{z}},k-1})$
  Compute structural cross dependencies: $\bar{\mathbf{w}}_{\bar{\mathbf{u}},k}(\bar{\mathbf{u}}_k)$, $\bar{\mathbf{z}}_{\bar{\mathbf{u}},k}(\bar{\mathbf{u}}_k)$
  Iterate fluid adjoint: $\bar{\mathbf{w}}_k^{n+1} = \tilde{\mathbf{G}}(\bar{\mathbf{w}}_k^n, \bar{\mathbf{w}}_{\bar{\mathbf{u}},k})$
  Compute flow cross dependency: $\bar{\mathbf{z}}_{\bar{\mathbf{w}},k}(\bar{\mathbf{w}}_k)$
  Evaluate mesh adjoint: $\bar{\mathbf{z}}_k = \tilde{\mathbf{M}}(\bar{\mathbf{z}}_{\bar{\mathbf{u}},k}, \bar{\mathbf{z}}_{\bar{\mathbf{w}},k})$
  Compute mesh cross-dependency: $\bar{\mathbf{u}}_{\bar{\mathbf{z}},k}(\bar{\mathbf{z}}_k)$
  Construct complete adjoint vector: $\bar{\mathbf{x}}_k = \left[\bar{\mathbf{u}}_k, \bar{\mathbf{w}}_k, \bar{\mathbf{z}}_k\right]$
  if $\|\bar{\mathbf{x}}_k - \bar{\mathbf{x}}_{k-1}\| < \epsilon$ **then**
    break
  end
end

Algorithm 2: Block Gauss-Seidel algorithm for the adjoint FSI problem. The subindex k refers to outer loop iterations (BGS iterations) and the superindex n refers to inner iterations in the flow and structural adjoint subsolvers

The top-level approach in Fig. 4 is also applicable to the computation of the fixed-point iteration of the flow and structural adjoints in (59). The cross terms are evaluated using AD in a very similar fashion, shown in Fig. 5 for the term $\bar{\mathbf{z}}_{\bar{\mathbf{w}}}$. The input are the primal variables for which we want to obtain the adjoint, in this example the positions of the mesh nodes, $\mathbf{z}$. The output corresponds to the state variables of the problem being differentiated, in this case the fluid conservative variables. The adjoint of the output, $\bar{\mathbf{w}}$, is initialized with the approximate value computed in the stage "Iterate fluid adjoint" of Algorithm 2 for the outer loop BGS iteration k, or $\bar{\mathbf{w}}_k$. One single reverse

run through the path of the flow iteration $\mathbf{G}(\mathbf{w}^*, \mathbf{z}^*, \boldsymbol{\alpha})$ is enough to obtain the cross contribution $\bar{\mathbf{z}}_{\mathbf{w},k}$ as there is no implicit dependency on $\bar{\mathbf{z}}$ in the reverse path $\bar{\mathbf{G}}(\bar{\mathbf{w}})$. The value of $\bar{\mathbf{z}}_{\bar{\mathbf{w}},k}$ will then be used to obtain $\bar{\mathbf{z}}_k = \tilde{\mathbf{M}}(\bar{\mathbf{z}}_{\bar{\mathbf{u}},k}, \bar{\mathbf{z}}_{\bar{\mathbf{w}},k})$ in the step "Evaluate mesh adjoint" of Algorithm 2.

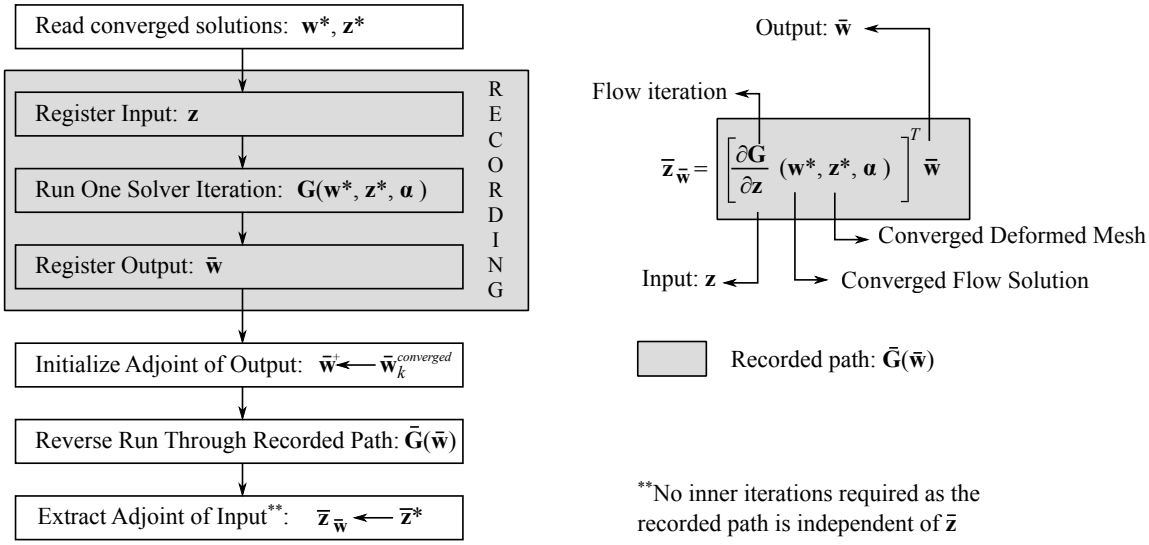

Figure 5: *Top-level approach to the cross-term calculation in the adjoint of FSI problems using AD.*

VI. Results and Discussion

The ADDA methodology introduced above will be demonstrated on a flexible, vertical cantilever beam, which is modeled using solid elements, under three different loading states: a uniform body load, a uniform follower pressure, and a fluid load for which the fully-coupled, FSI problem is solved. These test cases are defined in Fig. 6.

We will compute the gradient of the objective function defined in (29) with $\mathbf{C} = k\mathbf{I}$, $k = 10^6$ and $\mathbf{u}_{ref} = 0$, with respect to the Young's modulus E of the cantilever, which will be initially constant for the structure but it will be subsequently redefined independently in multiple regions. The Young's modulus ranges in this study from 13 – 28 kPa. This problem is consistent with the classic adjoint methodology presented in section II.4, where k is a scaling factor. The Poisson ratio is $\nu = 0.4$ for all cases, and the material model is Neo-Hookean (31). We will consider both material and geometrical non-linearities in the structural domain and adopt a plain-strain assumption for the solution of the two-dimensional problem.

VI.1. Test case (a): single-discipline, non-linear structural problem under a uniform body load

The test case (a) in Fig. 6 corresponds to a uniform body load applied to a flexible cantilever beam. We compute the objective function of the problem for a range of Young's moduli, $E = (13 - 28)$ kPa, in increments $\Delta E = 0.25$ kPa. We also compute the gradient $\mathrm{d}J/\mathrm{d}E$ using *central differences*, as defined in (5), with $h = 2\Delta E = 0.5$ kPa.

As it was discussed in section IV.2, the linearization of the problem $\mathscr{S}(\mathbf{u})$ for cases under a uniform body load results in a symmetric tangent matrix $\mathbf{K} = \mathbf{K}^T$. The only non-zero terms are the constitutive and stress terms in (20). Consequently, the non-linear structural problem becomes

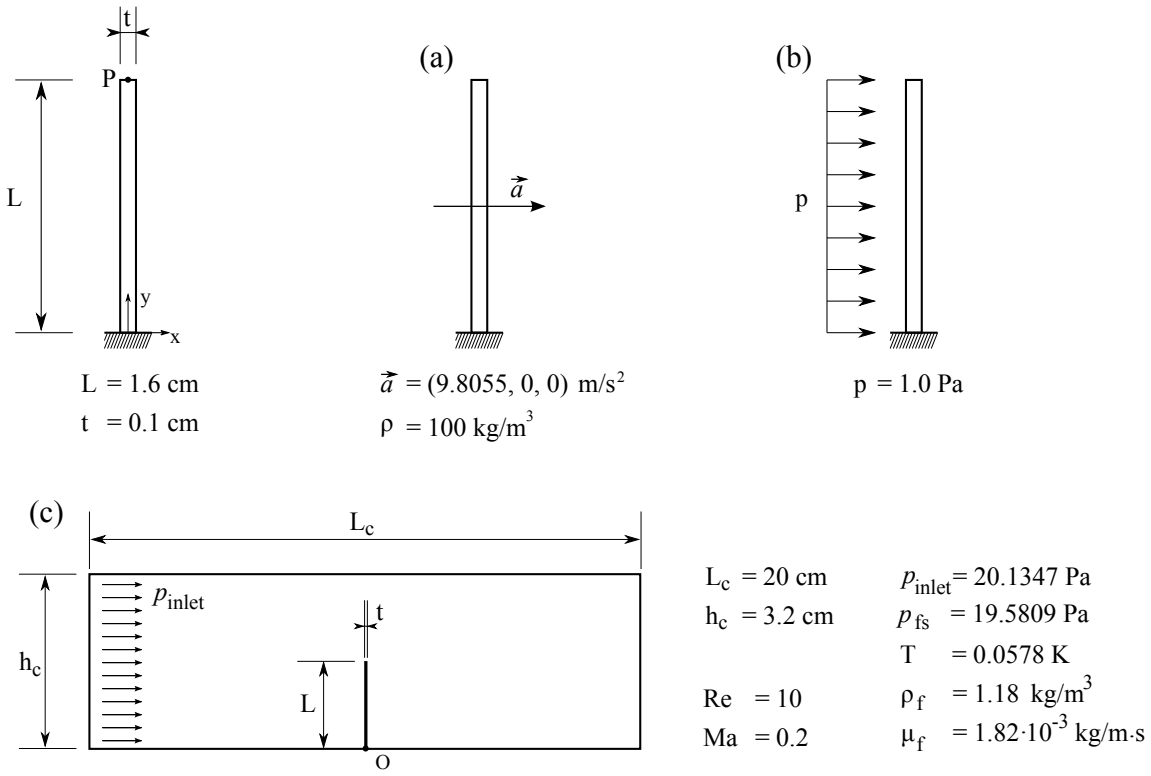

Figure 6: *Test cases studied in this work. Flexible cantilever under the effect of (a) a uniform body load, (b) a follower, uniform pressure, and (c) a fluid load in a straight channel at a low-Reynolds number condition.*

self-adjoint, and we may obtain the gradient dJ/dE using the adjoint method described in section V.1 with the same **K** as for the primal problem. The sensitivities obtained in this section using the conventional discrete method will be referred to here as the *Conventional Adjoint*. We will also compute the gradient using the *forward* mode of AD and the *AD-based Discrete Adjoint* (ADDA) method introduced in section V.3. The *forward* mode, as discussed in section III.1, consists of the propagation of the derivative of the input variable E throughout the whole solution process of the primal solver. The level of accuracy of the gradient will be exactly consistent with the accuracy of the solution scheme. Throughout this section, the gradients obtained using the *Forward* mode will be the reference value that defines the most accurate representation of the sensitivity studied. Finally, we compute dJ/dE using the (ADDA) iterative method.

In Fig. 7, we present the values of the objective function, the sensitivity computed using the *Forward* mode, and the relative error for this value when computed using the *AD-based Discrete Adjoint*, the *Conventional Adjoint* method, and *Central Differences*.

Fig. 7 shows that the gradients computed using both the *Conventional Adjoint* and the *AD-based Discrete Adjoint* have negligible errors compared to the *Forward* mode. This is further shown in Tab. 4 for E = 15 kPa and E = 25 kPa, where the relative error ranges from 1E-14 to 1E-12 for the *AD-based Discrete Adjoint* and is around 1E-10 for the *Conventional Adjoint*, demonstrating the validity of both approaches in computing nearly exact gradients. The error obtained using a *Central Differences* scheme is on the order of 0.01 - 0.04 %, an approximation likely to be acceptable for most engineering applications of the gradient.

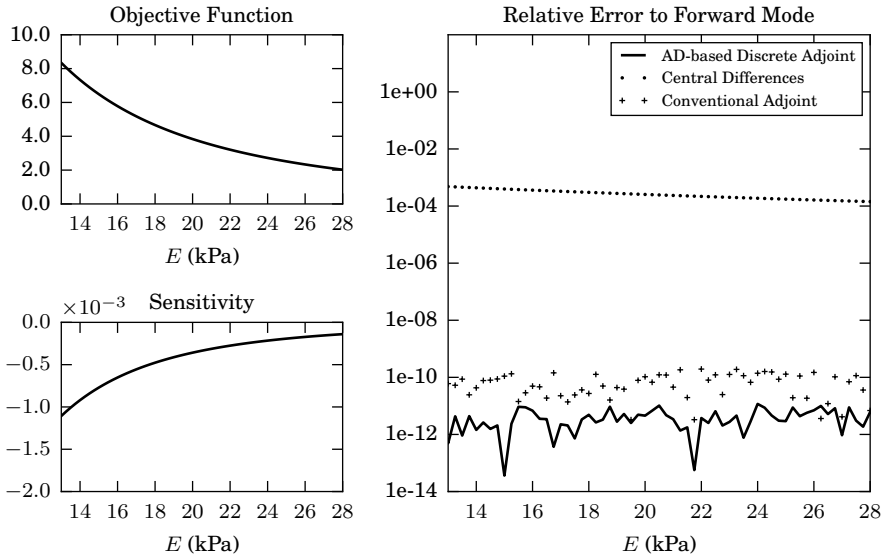

Figure 7: *Objective function, sensitivity and relative error between AD-based Discrete Adjoint, Central Differences ($h = 500$) and Conventional Adjoint with respect to Forward Mode, for test case (a).*

Table 4: *Test case (a). Value of the objective function (OF), sensitivities computed using the Forward Mode (FM), AD-based Discrete Adjoint (ADDA), Conventional Adjoint (CA) and Central Differences (CD), and relative error to FM.*

	E = 15 kPa		E = 25 kPa	
OF	6.48903888843946 E+00		2.51557271486488 E+00	
	dJ/dE	Rel. Error to FM	dJ/dE	Rel. Error to FM
FM	-7.70726301107035 E-04	-	-1.92352570834721 E-04	-
ADDA	-7.70726301107007 E-04	3.629 E-14	-1.92352570834159 E-04	2.922 E-12
CA	-7.70726301191935 E-04	1.102 E-10	-1.92352570809854 E-04	1.293 E-10
CD	-7.71031487924020 E-04	3.960 E-04	-1.92386207228861 E-04	1.749 E-04

VI.2. Test case (b): single-discipline, non-linear structural problem under a follower, uniform pressure

As discussed in section V.3, a major advantage of the *AD-based Discrete Adjoint* method is that it does not require the construction of the exact analytic Jacobian $\mathbf{K}$ of the problem. We will demonstrate this feature in this section using test case (b) in Fig. 6. Although for most structural problems this matrix may be exactly obtained, it becomes more complicated for problems in which the loading condition or the material model are difficult to linearize.

The problem involves a uniform pressure applied on the left boundary of the vertical cantilever. The pressure follows the deformation of the beam, and thus the component $\mathbf{K}_\Gamma$ in (20) will be nonzero. Moreover, this term will be non-symmetric and thus $\mathbf{K}_{exact} \neq \mathbf{K}_{exact}^T$. However, we will use a symmetric, simplified tangent matrix $\mathbf{K} = \mathbf{K}_c + \mathbf{K}_\sigma \neq \mathbf{K}_{exact}$ for the solution of the structural problem, which proves enough to converge the linearized problem while only reducing the order of the Newton scheme.

In order to quantify the error that results of using an approximate Jacobian for the solution of

the adjoint problem, we will deliberately use the same tangent matrix for the solution of (36). We will refer to this as the "Conventional Adjoint" approach. It is noted that, especially for problems in which the displacements are large, $\mathbf{K}_\Gamma$ may have a relatively large contribution to the Jacobian.

As shown in Fig. 8, the use of an approximate Jacobian yields a much poorer approximation to the gradient dJ/dE computed using the *Conventional Adjoint*. The relative error is quantified in table 5, obtaining relative errors of 6.5 % for E = 15 kPa and 2.9 % for E = 25 kPa compared to the solution of the *Forward Mode*.

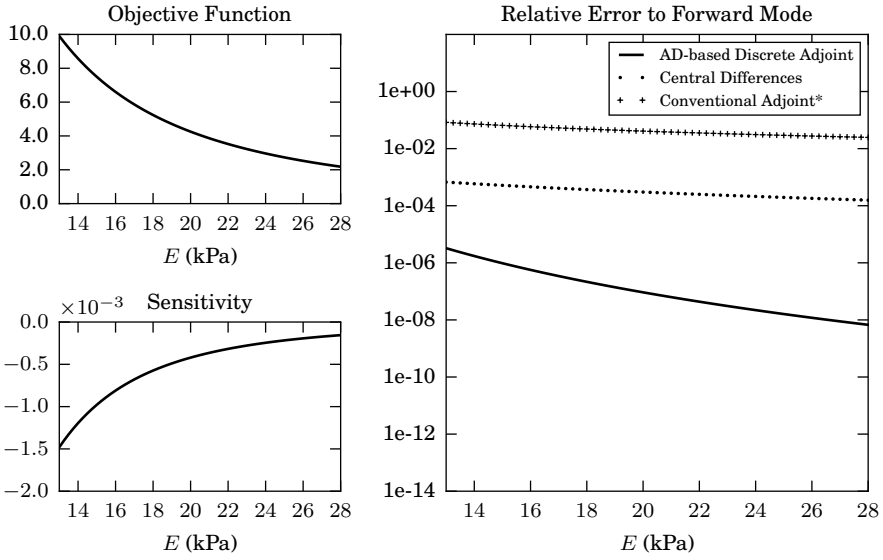

Figure 8: *Objective function, sensitivity and relative error between AD-based Discrete Adjoint, Central Differences (h = 500) and Conventional Adjoint with respect to Forward Mode, for test case (b).*

Table 5: *Test case (b). Value of the objective function (OF), sensitivities computed using the Forward Mode (FM), AD-based Discrete Adjoint (ADDA), Conventional Adjoint (CA) and Central Differences (CD), and relative error to FM.*

	E = 15 kPa		E = 25 kPa	
OF	7.49298988 E+00		2.73272689 E+00	
	dJ/dE	Rel. Error to FM	dJ/dE	Rel. Error to FM
FM	-9.77613731 E-04	-	-2.17180956 E-04	-
ADDA	-9.7761<u>4677</u> E-04	9.672 E-07	-2.1718096<u>0</u> E-04	1.607 E-08
CA	<u>-1.04131237</u> E-03	6.516 E-02	-2.<u>23525958</u> E-04	2.922 E-02
CD	-9.7<u>8119490</u> E-04	5.173 E-04	-2.17<u>223413</u> E-04	1.955 E-04

The *AD-based Discrete Adjoint* methodology again provides an excellent approximation to the gradient, even when an approximate Jacobian is being used. The relative error remains below 1E-6, this is a 0.0001 %, for most cases. This is approximately 2 orders of magnitude more accurate than *Central Differences*, which as expected give a very similar level of error as for the previous test case. This is due to the dependency of the finite-difference scheme on the size of the increment h, which remains h = 0.5 kPa.

However, we do see that the results obtained for the *AD-based Discrete Adjoint* approach are partially affected by the accuracy of the Jacobian. Although the approximation to the gradient is very good in all cases, the error is more than two orders of magnitude larger for the most flexible case as compared to the stiffest. The exact explanation for this is still under investigation, but one possible explanation for this behaviour may be related to the recording method. All test cases were run to the same convergence criteria for the residual. However, as discussed in section V.3, when the solution process is recorded, only one iteration of the primal solver is executed. We assume that the new solution is approximately equal to the converged one, or $\mathbf{u}^* + \Delta\mathbf{u}^* \approx \mathbf{u}^*$. The size of $\Delta\mathbf{u}^*$, however, is likely to be larger relative to the stiffness matrix $\mathbf{K}$ for the most flexible cases.

VI.3. Test case (c1): fully-coupled, non-linear FSI problem with a single material

We have shown in section VI.2 that the *AD-based Discrete Adjoint* methodology for the calculation of gradients is able to provide accurate sensitivities for problems in which the exact Jacobian is not available. As it was discussed in section IV.5, this is almost always the case for coupled Fluid-Structure Interaction problems, as demonstrated next. Here, the structure will be the same flexible cantilever used in sections VI.1 and VI.2. In this case, however, it is attached to the wall in a channel with a low-Reynolds number flow, which defines a new loading condition, as shown in (c) of Fig. 6. Therefore, a fully-coupled steady-state FSI problem needs to be run, as the solution in the fluid domain will be determined by the deformation of the structural domain, and conversely, the forces acting over the structure are determined by the solution of the flow field. We adopt a BGS iterative procedure for the solution of the primal problem, and we solve for the same range of values of Young's modulus as for the previous tests.

First, we note that, although the FSI problem of this study is reduced in size (i.e., numbers of degrees of freedom) to simplify our investigation and have a shorter turn-around time, we are not reducing the complexity of the physics involved. The flow is fully-resolved using the Navier-Stokes equations, which include the effects of viscosity. On the structural side, both geometrical and material non-linearities are included. As an example of this, the deformation for the two most extreme cases of the range of FSI problems solved is shown in Fig. 9. It may be observed that, for the case of $E = 13$ kPa, the amplitude of the displacement at the tip is nearly 40% of the cantilever length.

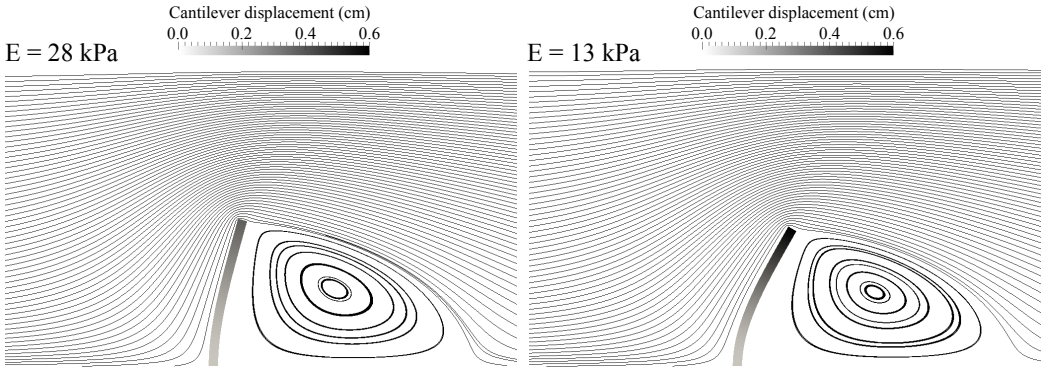

Figure 9: *Flow streamlines and cantilever deformation for test case (c1) with E = 28 and 13 kPa.*

In Fig. 10, the objective function, the sensitivity dJ/dE, and the relative error to the *Forward Mode* are shown. The relative error obtained using the *Central Differences* scheme is, consistent with the values obtained in sections VI.1 and VI.2, on the order of 1E-04 for $h = 0.5$ kPa. Since the exact

Jacobian is not available for the solution method employed in this work, a *Conventional Adjoint* solution of this problem has not been attempted. It is well known that the *Conventional Adjoint* approach is very challenging [11,12,16], and indeed, the purpose of the current exercise is to avoid it altogether.

The gradient obtained using the ADDA approach of V.4 yields results on the same order of magnitude as those obtained using the *Central Differences* scheme. The approximation to the *exact* gradient obtained using the forward mode is excellent: the gradients for a fully-coupled non-linear Fluid-Structure Interaction problem are computed with an error that is below 0.05 %. As it is explained by Albring *et al.* in [34], some approximations are made in the general reversal procedure for the fixed-point solvers in order to improve the performance of the fluid adjoint. These are very likely related to the increase in the relative error with respect to the purely structural problem in sections VI.1 and VI.2.

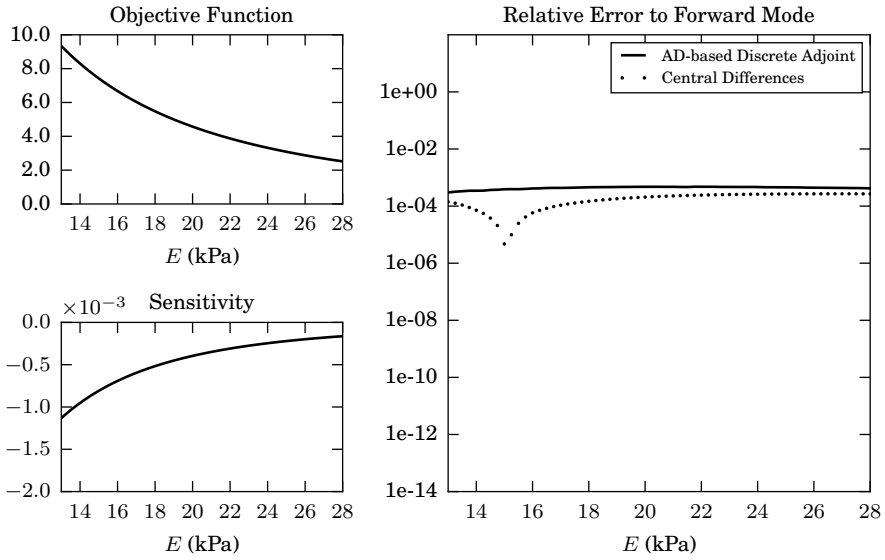

Figure 10: *Objective function, sensitivity and relative error between AD-based Discrete Adjoint and Central Differences with respect to Forward Mode, for test case (c1).*

Let $\boldsymbol{\alpha} = [E, \nu]$ be now the vector of design variables, where ν is the Poisson ratio. It was shown in section V.1 that the direct method requires one solution of the linear system (8) *per design variable*. This is also applicable to the *Forward Mode* of AD: one simulation is required for each input parameter for which we desire the sensitivity. However, for the *AD-based Discrete Adjoint*, we may obtain the sensitivities with respect to the entire design variable vector and their level of accuracy is consistent for both design parameters, as shown in Tab. 6.

Moreover, a parametric study is often required in order to determine the optimum value of h that minimizes the error of the scheme. This is shown in Fig. 11, which shows that good values of h need to be independently determined for each of the design variable.

In order to analyze the convergence properties of the BGS method for the ADDA solver, the residuals of the flow and structural adjoints are plotted in Fig. 12 against the number of iterations. A very similar pattern of convergence appears in both domains. In the first stage, the convergence is extremely fast, reducing the residuals of the flow adjoint by 3 orders of magnitude and the residuals of the structural adjoint by 5 in only a few iterations. A slower convergence pattern is

Table 6: *Test case (c1). Value of the objective function (OF), sensitivities computed using the Forward Mode (FM), AD-based Discrete Adjoint (ADDA) and Central Differences (CD), and relative error to FM.*

	E = 21 kPa, ν = 0.4			
OF	4.20038299 E+00			
	dJ/dE	Rel. Error to FM	$dJ/d\nu$	Rel. Error to FM
FM	-3.49058121 E-04	-	-8.33972786 E+00	-
ADDA	-3.48892392 E-04	4.748 E-04	-8.33583720 E+00	4.665 E-04
CD	-3.48978725 E-04	2.275 E-04	-8.33823016 E+00	1.796 E-04

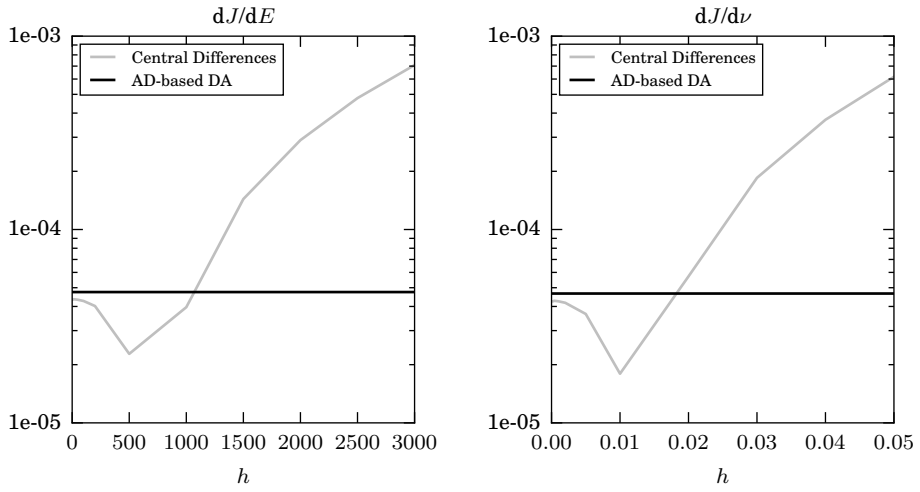

Figure 11: *Relative error for Center Differences for different values of h, compared with the relative error of the AD-Based Discrete Adjoint, with respect to the* exact *gradient computed using the Forward Mode.*

then followed until the solution is converged to the maximum level of convergence for each domain, which appears to be closely related to the convergence threshold imposed for each independent domain in the solution of the primal problem.

The convergence of the sensitivities dJ/dE and $dJ/d\nu$ is shown in Fig. 13. A total of 94 iterations were necessary for a reduction of 4 orders of magnitude in the flow adjoint residual and 6 for the structural adjoint. It must be noted that, to ensure numerical stability of the primal problem, the fluid load transfer was relaxed using a cubic function for $N = 40$ iterations as explained in section IV.5. No relaxation parameter was found to be necessary for the *AD-based Discrete Adjoint* simulation for this particular test case.

The excellent computational performance of the *AD-based Discrete Adjoint* is finally shown in Tab. 7, both in terms of computational time and memory footprint. With regards to the former, the converged solution in Fig. 13 takes 73.4% of the computational time that is required for the full convergence of the primal FSI problem. Even though the flow and structural adjoints may be further converged and thus the computational time increased, it is shown in Tab. 7 that the improvement in the calculation of the gradients is not substantial enough to justify it in this particular case. In fact, after only 40 iterations of the adjoint solver, a very good approximation of the gradient with a relative error under a 0.05 % is already obtained, employing a computational time of only 28.3% of the FSI solution. The computation of the *Forward Mode* takes a 77.8% more

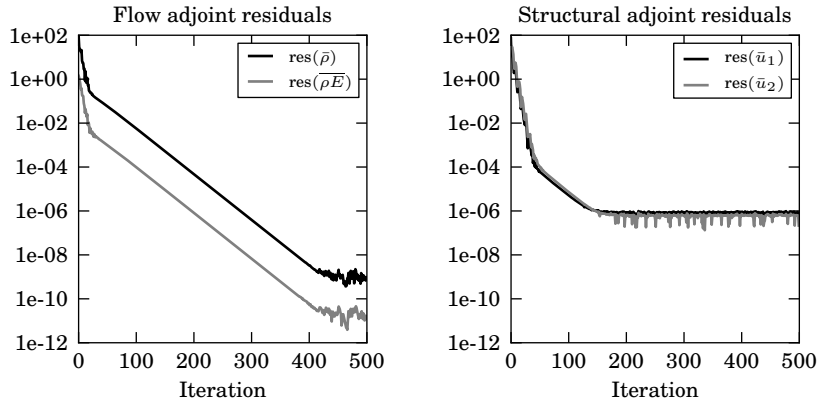

Figure 12: *Residuals of the flow and structural adjoints for 500 iterations*

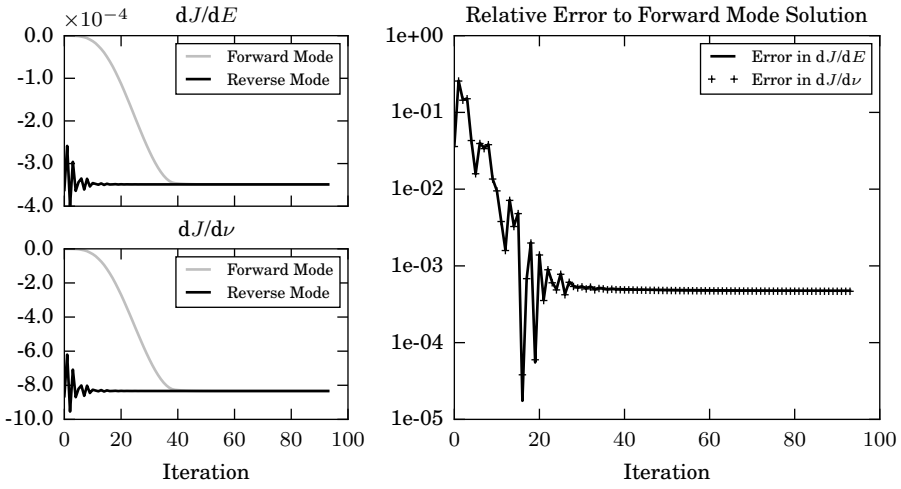

Figure 13: *Convergence of the gradients* dJ/dE *and* dJ/dν *for test case (c1) using ADDA.*

than the solution of the primal problem alone, due to the increased number of operations needed to propagate the gradients.

Regarding the memory footprint, the *AD-based Discrete Adjoint* uses approximately 6 times more memory than the primal solution of the FSI problem. This compares positively with other examples in the literature. Mader *et al.* [30] report a memory footprint of approximately 10 times for a single-discipline flow adjoint problem, where the employment of AD was selective and limited to the routine that calculated the flow residual.

VI.4. Test case (c2): fully-coupled, non-linear FSI problem with multiple materials

As discussed, the *AD-based Discrete Adjoint* method solves the adjoint problem with the major advantage that, from only one solution of the reverse problem, we can obtain the derivatives of the objective function with respect to many design variables or input parameters at a fixed cost.

Table 7: *Performance study. Memory footprint, computational time and convergence of the ADDA solver.*

	RAM (Mb)	Time (rel.)	Convergence of theAD-based Discrete Adjoint					
FSI	68	t	Relative Error to FM		$\log_{10}$ of residual reduction			
FSI - FM	99	1.778t	$\mathrm{d}J/\mathrm{d}E$	$\mathrm{d}J/\mathrm{d}\nu$	$\bar{\rho}$	$\overline{\rho E}$	$\bar{u}_1$	$\bar{u}_2$
FSI - RM	404	0.283t	4.9476 E-04	4.8625 E-04	-3.024	-2.953	-5.210	-5.298
		0.734t	4.7479 E-04	4.6660 E-04	-4.080	-4.013	-6.536	-6.854
		0.913t	4.7344 E-04	4.6527 E-04	-4.847	-4.780	-7.215	-7.599
		1.291t	4.7319 E-04	4.6502 E-04	-5.971	-5.904	-7.4475	-8.064
		1.575t	4.7317 E-04	4.6500 E-04	-6.825	-6.758	-7.416	-7.997
		1.925t	4.7317 E-04	4.6500 E-04	-7.867	-7.800	-7.443	-8.024

In order to demonstrate the potential of this technique for optimal design with a large number of design variables, we now discretize the cantilever beam into 10 regions, as shown in Fig. 14. The regions are smaller towards the clamped end, as we expect to find larger sensitivity with respect to the Young's modulus in areas where the moment generated by the actuating forces is larger. The vector of design variables is now $\boldsymbol{\alpha} = [E_1, ..., E_{10}]$.

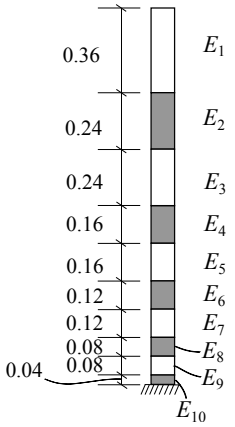

Figure 14: *Geometrical distribution of the design variables for test case (c2).*

We use an uniform value of the Young's modulus, $E_i = 21$ kPa $\forall i \in [1, 10]$ and compute the value of the sensitivities for all 10 regions. We obtain this value by running 10 FSI simulations using the forward mode, one per design variable. The results are presented in Fig. 15, together with the values of the absolute and relative error obtained using the *AD-based Discrete Adjoint* (ADDA) and Central Differences (CD).

The relative error of the *AD-based Discrete Adjoint* method is lower than the value obtained using Central Differences (and below 0.01%) for 9 out of the 10 regions. This can be explained from the absolute error, also shown in Fig. 15. The absolute error of the ADDA method with respect to the Forward Mode remains almost constant, at around 1e-09, while the central differences method gives nearly constant relative errors. This is linked to the simultaneous evaluation of all sensitivities in the ADDA, while in the central differences each sensitivity requires a simulation to convergence of the full problem. As the interest is on the design variables that have a larger value of the sensitivities and thus a larger impact on the value of the objective function, the ADDA

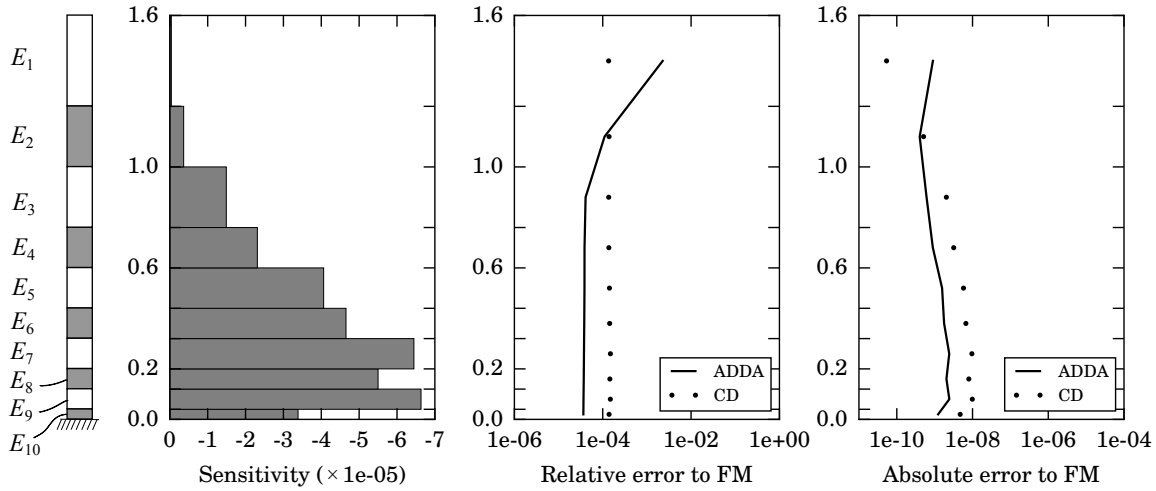

Figure 15: *Left: Sensitivities for the design vector* $\boldsymbol{\alpha} = [E_1, ..., E_{10}]$. *Center: Relative error of the sensitivities computed using* AD-based Discrete Adjoint *(ADDA) and Central Differences (CD),* $h = 500$, *with respect to the value computed using the forward mode (FM). Right: Absolute error of the same magnitude.*

naturally targets those sensitivities.

The main advantage of the ADDA however relates to its computationally efficiency for gradient-based optimal design. In a gradient-based optimization setting, we would normally need to compute the value of the objective function and the sensitivities with respect to all the design variables once per iteration. For the test case presented here, using the forward mode requires at least 10 evaluations of the solver, which would all require the FM to be enabled (increasing the cost of each simulation). 10 evaluations, and not 11, are needed given that each evaluation will implicitly compute the value of J. Using central differences requires running 20 additional FSI simulations. The ADDA method allows for an accurate computation of the gradients with only 1 extra simulation, which has a cost comparable to the evaluation of the primal solver as shown in Tab. 7. This is summarized in Tab. 8.

Table 8: *Number of evaluations per iteration of a problem with DV = 10 in an optimization framework.*

	Evaluation of J		Evaluation of $\mathrm{d}J/\mathrm{d}\boldsymbol{\alpha}$			Total
	FSI	FSI - FM	FSI	FSI - FM	FSI - RM	Evaluations
Central Differences	1	-	20	-	-	21
Forward Mode	-	1	-	9	-	10
AD-based Discrete Adjoint	1	-	-	-	1	2

VII. Conclusion

We have introduced a numerical approach for the efficient calculation of derivatives of fully-coupled, non-linear Fluid-Structure Interaction problems. The primal problem is a three-field formulation, where the fluid domain is characterized by the compressible Navier-Stokes equations discretized using finite volumes, the structure by the geometrical and materially nonlinear solid mechanics equations discretized using finite elements, and the mesh domain is solved using as a pseudo-elastic problem undergoing finite displacements. This partitioned approach allows then

naturally to develop independently solution features in either domain, including turbulence models in the fluid and nonlinear material models in the structure, with an iterative solution method based on a conventional block Gauss-Seidel method.

It has been shown that, if the coupled equations are cast in the form of fixed-point iterations, then the coupled adjoint equations can also be written as fixed-point iterations and solved as well using a block Gauss-Seidel algorithm. Moreover, the iterative solution process for the adjoint equation facilitates the evaluation of the coupled adjoint equations using reverse-mode algorithmic differentiation with a small memory footprint and including all cross-dependencies between different domains. This methodology eliminates the need for analytical linearizations of the system to obtain design sensitivities and replaces it instead by numerical approximations. Since the evaluation of the gradients is done through a reverse set of operations on the last iteration of the primal problem, the accuracy in the evaluation of the adjoint depends on the convergence criteria selected for both primal and adjoint Gauss-Seidel iterations.

We have applied this technique to a non-linear cantilever beam modeled using solid elements and subject to three different loading states: a uniform body load, a follower pressure, and a coupled problem in which a viscous flow field determines the loads acting over the structure. The proposed method performs consistently well for all three test cases, and the accuracy of the gradients computed is demonstrated with convergence studies. Moreover, in the present implementation leveraging the reverse mode of AD, we have demonstrated that the computational performance is very competitive (less that the cost of the primal in some cases) and that the memory overhead is quite modest. The main advantage of this approach for optimal design with multiple design variables have been highlighted, demonstrating that it can dramatically reduce the cost of computing gradients with respect to more extended techniques.

The results in this paper have demonstrated a computationally-efficient strategy to incorporate fluid-structure interaction capabilities into SU2. For the demonstration of the approach in this paper, the objective function has been restricted to the structural displacements, and the gradients have also been limited to structural variables only. We have limited the implementation at this stage to steady-state FSI problems and compressible flows. However, these limitations only responds to the current stage of the software development and constitute no fundamental limitation in the extension of the method. SU2 has already been shown as a viable analysis and design solution for industrial-scale optimization applications in optimal flow design. Ongoing research relates to the implementation of more complex objective functions on larger problems, and potentially to time-domain FSI.

References

[1] Sobieszczanski-Sobieski J. Sensitivity of complex, internally coupled systems. *AIAA Journal* 1990; **28**(1):153–160.

[2] Pironneau O. On optimum design in fluid mechanics. *Journal of Fluid Mechanics* 1974; **64**(1):97–110, doi:10.1017/S0022112074002023.

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

[4] Adelman HM, Haftka RT. Sensitivity Analysis of Discrete Structural Systems. *AIAA Journal* 1986; **24**(5):823–832.

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

[6] Barcelos M, Maute K. Aeroelastic design optimization for laminar and turbulent flows. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(19-20):1813–1832, doi: 10.1016/j.cma.2007.03.009.

[7] Maute K, Nikbay M, Farhat C. Analytically based sensitivity analysis and optimization of nonlinear aeroelastic systems. *8th Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, 6-8 September, 2000.

[8] Maute K, Nikbay M, Farhat C. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA Journal* 2001; **39**(11):2051–2061.

[9] Martins J, Alonso J, Reuther J. Aero-structural wing design optimization using high-fidelity sensitivity analysis. *Proceedings - CEAS Conference on Multidisciplinary Aircraft Design Optimization*, Honlinger H (ed.), Cologne, Germany, 2001; 211–226.

[10] Martins J, Alonso J, Reuther J. Complete configuration aero-structural optimization using a coupled sensitivity analysis method. *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization 2002*, American Institute of Aeronautics and Astronautics Inc.: Atlanta, GA, 4-6 September, 2002.

[11] Maute K, Nikbay M, Farhat C. Sensitivity analysis and design optimization of three-dimensional non-linear aeroelastic systems by the adjoint method. *International Journal for Numerical Methods in Engineering* 2003; **56**(6):911–933, doi:10.1002/nme.599.

[12] Martins J, Alonso J, Reuther J. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering* 2005; **6**(1):33–62, doi:10.1023/B:OPTE.0000048536.47956.62.

[13] Barcelos M, Bavestrello H, Maute K. A schur-newton-krylov solver for steady-state aeroelastic analysis and design sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(17-18):2050–2069, doi:10.1016/j.cma.2004.09.013.

[14] Fazzolari A, Gauger N, Brezillon J. Efficient aerodynamic shape optimization in MDO context. *Journal of Computational and Applied Mathematics* 2007; **203**(2 SPEC. ISS.):548–560, doi: 10.1016/j.cam.2006.04.013.

[15] Abu-Zurayk M, Brezillon J. Development of the adjoint approach for aeroelastic wing optimization. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design* 2013; **121**:59–66, doi:10.1007/978-3-642-35680-3_8.

[16] Kenway G, Kennedy G, Martins J. Scalable parallel approach for high-fidelity steady-state aeroelastic analysis and adjoint derivative computations. *AIAA Journal* 2014; **52**(5):935–951, doi:10.2514/1.J052255.

[17] Degroote J, Hojjat M, Stavropoulou E, Wüchner R, Bletzinger KU. Partitioned solution of an unsteady adjoint for strongly coupled fluid-structure interactions and application to parameter identification of a one-dimensional problem. *Structural and Multidisciplinary Optimization* 2013; **47**(1):77–94, doi:10.1007/s00158-012-0808-2.

[18] Perego M, Veneziani A, Vergara C. A variational approach for estimating the compliance of the cardiovascular tissue: An inverse fluid-structure interaction problem. *SIAM Journal on Scientific Computing* 2011; **33**(3):1181–1211, doi:10.1137/100808277.

[19] Raulli M, Maute K. Optimization of fully coupled electrostatic–fluid–structure interaction problems. *Computers & Structures* 2005; **83**(2–3):221 – 233, doi:http://dx.doi.org/10.1016/j.compstruc.2004.08.003.

[20] Fick P, van Brummelen E, van der Zee K. On the adjoint-consistent formulation of interface conditions in goal-oriented error estimation and adaptivity for fluid-structure interaction. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(49-52):3369–3385, doi:10.1016/j.cma.2010.07.009.

[21] van der Zee K, van Brummelen E, Akkerman I, de Borst R. Goal-oriented error estimation and adaptivity for fluid-structure interaction using exact linearized adjoints. *Computer Methods in Applied Mechanics and Engineering* 2011; **200**(37-40):2738–2757, doi:10.1016/j.cma.2010.12.010.

[22] Bazilevs Y, Hsu MC, Bement M. Adjoint-based control of fluid-structure interaction for computational steering applications. *13th Annual International Conference on Computational Science, ICCS*, vol. 18, Elsevier: Barcelona, 2013; 1989–1998, doi:10.1016/j.procs.2013.05.368.

[23] Martins J, Alonso J, Reuther J. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft* 2004; **41**(3):523–530.

[24] Peter J, Dwight R. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers and Fluids* 2010; **39**(3):373–391, doi:10.1016/j.compfluid.2009.09.013.

[25] Dwight R, Brezillon J. Effect of approximations of the discrete adjoint on gradient-based optimization. *AIAA Journal* 2006; **44**(12):3022–3031, doi:10.2514/1.21744.

[26] Lund E, Møller H, Jakobsen L. Shape design optimization of stationary fluid-structure interaction problems with large displacements and turbulence. *Structural and Multidisciplinary Optimization* 2003; **25**(5-6):383–392, doi:10.1007/s00158-003-0288-5.

[27] Griewank A, Corliss GF ((eds.)). *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, vol. Proceedings of the first SIAM Workshop on Automatic Differentiation, SIAM - Society for Industrial and Applied Mathematics: Breckenridge, CO, January 6-8, 1991.

[28] Griewank A, Walther A. *Evaluating Derivatives*. Second edn., Society for Industrial and Applied Mathematics, 2008, doi:10.1137/1.9780898717761.

[29] Marta A, Mader C, Martins J, Van Der Weide E, Alonso J. A methodology for the development of discrete adjoint solvers using automatic differentiation tools. *International Journal of Computational Fluid Dynamics* 2007; **21**(9-10):307–327, doi:10.1080/10618560701678647.

[30] Mader C, Martins J, Alonso J, Van Weide E. Adjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA Journal* 2008; **46**(4):863–873, doi:10.2514/1.29123.

[31] Mader C, Martins J. Derivatives for time-spectral computational fluid dynamics using an automatic differentiation adjoint. *AIAA Journal* 2012; **50**(12):2809–2819, doi:10.2514/1.J051658.

[32] Ozaki I, Kimura F, Berz M. Higher-order sensitivity analysis of finite element method by automatic differentiation. *Computational Mechanics* 1995; **16**(4):223–234, doi:10.1007/BF00369867.

[33] Albring T, Sagebaum M, Gauger N. Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework. *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 22-26 June*, Dallas, TX, 2015.

[34] Albring T, Sagebaum M, Gauger N. Efficient aerodynamic design using the discrete adjoint method in SU2. *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2016.

[35] Zhou B, Albring T, Gauger N, Economon TD, Palacios F, Alonso JJ. A discrete adjoint framework for unsteady aerodynamic and aeroacoustic optimization. *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 22-26 June*, Dallas, TX, 2015.

[36] Hogan R. Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Transactions on Mathematical Software* 2014; **40**(4), doi:10.1145/2560359.

[37] Sanchez R, Palacios R, Economon T, Kline H, Alonso J, Palacios F. Towards a Fluid-Structure Interaction solver for problems with large deformations within the open-source SU2 suite. *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech,*, San Diego, CA, 4-8 Jan, 2016.

[38] Sanchez R, Kline H, Thomas D, Variyar A, Righi M, Economon T, Alonso J, Palacios R, Dimitriadis G, Terrapon V. Assessment of the Fluid-Structure Interaction Capabilities for Aeronautical Applications of the Open-Source Solver SU2. *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2016)*, Papadrakakis M, Papadopoulos V, Stefanou G, Plevris V (eds.), Crete Island, Greece, 5-10 June, 2016.

[39] Palacios F, Colonno M, Aranake A, Campos A, Copeland S, Economon T, Lonkar A, Lukaczyk T, Taylor T, Alonso J. Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design. *AIAA 51st Aerospace Sciences Meeting, 7-10 January*, Grapevine, TX, 2013.

[40] Palacios F, Economon T, Aranake A, Copeland S, Lonkar A, Lukaczyk T, Manosalvas D, Naik K, Santiago Padrón A, Tracey B, *et al.*. Stanford university unstructured (SU2): Open-source analysis and design technology for turbulent flows. *AIAA 52nd Aerospace Sciences Meeting, SciTech, 13-17 January*, National Harbor, MD, 2014.

[41] Economon T, Palacios F, Copeland S, Lukaczyk T, Alonso J. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal* 2016; **54**(3):828–846, doi:10.2514/1.J053813.

[42] Hernández S. *Métodos de diseño óptimo de estructuras*. CICCP, 1990.

[43] Hojjat M, Stavropoulou E, Gallinger T, Israel U, Wüchner R, K-U B. *Fluid Structure Interaction II*, vol. 73, chap. Fluid-Structure Interaction in the Context of Shape Optimization and Computational Wind Engineering. Springer Berlin Heidelberg, 2010; 351–381.

[44] Arora JS. *Introduction to Design Optimization*. 3rd edn., Academic Press: Boston, 2012.

[45] Gunzburger M. *Perspectives in Flow Control and Optimization*. Society for Industrial and Applied Mathematics, 2002, doi:10.1137/1.9780898718720.

[46] Hascoet L, Pascual V. The tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software* 2013; **39**(3), doi:10.1145/2450153.2450158.

[47] Martins J, Hwang J. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal* 2013; **51**(11):2582–2599, doi:10.2514/1.J052184.

[48] Veldhuizen T. Expression templates. *C++ Report* 1995; **7(5)**:26–31.

[49] Pflaum C. Expression templates for partial differential equations. *Computing and Visualization in Science* 2001; **4**(1):1–8, doi:10.1007/s007910100051.

[50] Di Pietro D, Veneziani A. Expression templates implementation of continuous and discontinuous galerkin methods. *Computing and Visualization in Science* 2009; **12**(8):421–436, doi:10.1007/s00791-008-0117-x.

[51] Farhat C, Lesoinne M, Maman N. Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Fluids* 1995; **21**(10):807–835.

[52] Maute K, Allen M. Conceptual design of aeroelastic structures by topology optimization. *Structural and Multidisciplinary Optimization* 2004; **27**(1-2):27–42.

[53] Bonet J, Wood RD. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 2008.

[54] Deparis S, Discacciati M, Fourestey G, Quarteroni A. Fluid-structure algorithms based on Steklov-Poincaré operators. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**(41-43):5797–5812, doi:10.1016/j.cma.2005.09.029.

[55] Farhat C, Lesoinne M, LeTallec 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* 1998; **157**(1-2):95–114.

[56] Kassiotis C, Ibrahimbegovic A, Niekamp R, Matthies H. Nonlinear fluid-structure interaction problem. part I: Implicit partitioned algorithm, nonlinear stability proof and validation examples. *Computational Mechanics* 2011; **47**(3):305–323.

[57] Von Scheven M, Ramm E. Strong coupling schemes for interaction of thin-walled structures and incompressible flows. *International Journal for Numerical Methods in Engineering* 2011; **87**(1-5):214–231, doi:10.1002/nme.3033.

[58] Le Tallec P, Mouro J. Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering* 2001; **190**(24-25):3039–3067, doi:10.1016/S0045-7825(00)00381-9.

[59] Matthies HG, Steindorf J. Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction. *Computers & Structures* 2002; **80**(27–30):199–1999, doi:http://dx.doi.org/10.1016/S0045-7949(02)00259-6.

[60] Küttler U, Wall W. Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics* 2008; **43**(1):61–72.

[61] Degroote J, Bathe KJ, Vierendeels J. Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. *Computers and Structures* 2009; **87**(11-12):793–801, doi:10.1016/j.compstruc.2008.11.013.

[62] Habchi C, Russeil S, Bougeard D, Harion JL, Lemenand T, Ghanem A, Valle D, Peerhossaini H. Partitioned solver for strongly coupled fluid-structure interaction. *Computers and Fluids* 2013; **71**:306–319.

[63] van Brummelen E. Partitioned iterative solution methods for fluid–structure interaction. *International Journal For Numerical Methods In Fluids* 2011; **65**:3–27, doi:10.1002/fld.2465.

[64] Suchocki C. A finite element implementation of knowles stored-energy function: Theory, coding and applications. *Archive of Mechanical Engineering* 2011; **58**(3):319–346, doi:10.2478/v10180-011-0021-7.

[65] Buoso S, Palacios R. Electro-aeromechanical modelling of actuated membrane wings. *Journal of Fluids and Structures* 2015; **58**:188–202, doi:10.1016/j.jfluidstructs.2015.08.010.

[66] Korivi VM, Taylor A, Newman P, How G, Jones H. An incremental strategy for calculating consistent discrete CFD sensitivity derivatives. *Technical Report*, National Aeronautics and Space Administration, Langley Research Center 1992. NASA TM-104207.