ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

ASSESSMENT OF THE FLUID-STRUCTURE INTERACTION CAPABILITIES FOR AERONAUTICAL APPLICATIONS OF THE OPEN-SOURCE SOLVER SU2.

Ruben Sanchez[1*], H. L. Kline[2], David Thomas[3], Anil Variyar[2], Marcello Righi[4], Thomas D. Economon[2], Juan J. Alonso[2], Rafael Palacios[1], Grigorios Dimitriadis[3], Vincent Terrapon[3]

[1]Department of Aeronautics, Imperial College London
London, SW7 2AZ, United Kingdom
e-mail: {r.sanchez-fernandez14, r.palacios}@imperial.ac.uk

[2] Department of Aeronautics & Astronautics, Stanford University
Stanford, CA, 94305, USA
e-mail: {hlkline, anilvar, economon, jjalonso}@stanford.edu

[3] Department of Mechanical and Aerospace Engineering, University of Liege
Liege, 4000, Belgium
e-mail: {dthomas, vincent.terrapon, gdimitriadis}@ulg.ac.be

[4] School of Engineering, Zurich University of Applied Sciences
Winterthur, 8401, Switzerland
e-mail: rigm@zhaw.ch

Keywords: Fluid-Structure Interaction, Computational Aeroelasticity, Reynolds-Averaged Navier Stokes, Geometrically-Nonlinear Solid Mechanics

Abstract. *We report on an international effort to develop an open-source computational environment for high-fidelity fluid-structure interaction analysis. In particular, we will focus on verification of the implementation for application in computational aeroelasticity. The capabilities of the SU2 code for aeroelastic analysis have been further enhanced both by developing natively embedded tools for the study of largely deformable solids, and by wrapping it using Python tools for an improved communication with external solvers. Both capabilities will be demonstrated on relevant test cases, including rigid-airfoil solutions with indicial functions, the Isogai Wing Section, test cases from the AIAA 2nd Aeroelastic Prediction Workshop, and the vortex-induced vibrations of a flexible cantilever in the wake of a square cylinder. Results show very good performance both in terms of accuracy and computational efficiency. The modularity and versatility of the baseline suite allows for a flexible framework for multidisciplinary computational analysis. The software libraries have been freely shared with the community to encourage further engagement in the improvement, validation and further development of this open-source project.*

1 INTRODUCTION

The interaction between fluids and structures on complex geometries is a challenging computational problem that has attracted substantial interest due to its relevance across disciplines, including aeronautical and civil engineering, biomechanics, and turbomachinery. In this work, we will focus on aeronautical applications, and particularly on computational aeroelasticity [1–3]. Due to the complexity of the constitutive equations and the strongly non-linear coupling effects between fluid and structure, the development of accurate and efficient solvers able to be run in highly-parallel environments becomes a determining factor, specially when addressing large problems that require refined discretisations.

With this in mind, the open-source software suite SU2 [4–6] has been used as the basic infrastructure for the development of a set of tools for Fluid-Structure Interaction (FSI). Due to its modularity and versatility, it has been possible to develop natively embedded tools for non-linear structural analysis in SU2, thus allowing for a self-contained, open-source suite capable to deal with FSI problems involving largely deformable solids [7]. At the same time, the suite has been wrapped using Python tools for improved communication with external solvers, which increases the flexibility for the user when aiming to solve problems using other in-house or commercial platforms. In this paper we will detail the modular implementations we have included in SU2 in the context of FSI, and we will assess the validity of these methods.

The paper is organized as follows. Section 2 briefly describes the governing equations for fluid and structural mechanics, and establishes the interface coupling conditions. Section 3 sketches the structure of the code for the fluid and structural solver, and the architecture that permits the coupling of the independent solvers with each other or with external tools via Python. Section 4 presents the first set of results obtained using the different approaches described. Finally, section 5 summarizes the current state of the project and outlines some of the planned future developments.

2 BACKGROUND

2.1 Fluid mechanics

2.1.1 Governing equations

We are concerned with compressible, turbulent fluid flows governed by the Reynolds-averaged Navier-Stokes (RANS) equations, which can be expressed in arbitrary Lagrangian-Eulerian (ALE) [8] differential form as

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathbf{F}^c - \nabla \cdot \mathbf{F}^v - Q = 0 \qquad \text{in } \Omega \times [0, t] \tag{1}$$

where the conservative variables are given by $U = \{\rho, \rho\mathbf{v}, \rho E\}^\mathsf{T}$, and the convective fluxes, viscous fluxes, and generic source term are

$$\mathbf{F}^c = \begin{Bmatrix} \rho(\mathbf{v} - \dot{\mathbf{u}}_\Omega) \\ \rho\mathbf{v} \otimes (\mathbf{v} - \dot{\mathbf{u}}_\Omega) + \bar{\bar{I}}p \\ \rho E(\mathbf{v} - \dot{\mathbf{u}}_\Omega) + p\mathbf{v} \end{Bmatrix}, \quad \mathbf{F}^v = \begin{Bmatrix} \cdot \\ \bar{\bar{\tau}} \\ \bar{\bar{\tau}} \cdot \mathbf{v} + \mu^*_{tot} c_p \nabla T \end{Bmatrix}, \tag{2}$$

and $Q = \{q_\rho, \mathbf{q}_{\rho\mathbf{v}}, q_{\rho E}\}^\mathsf{T}$, where ρ is the fluid density, $\mathbf{v} = \{v_1, v_2, v_3\}^\mathsf{T} \in \mathbb{R}^3$ is the flow speed in a Cartesian system of reference, $\dot{\mathbf{u}}_\Omega$ is the velocity vector at a point for a domain in motion (mesh velocity after discretization), E is the total energy per unit mass, p is the static pressure,

c_p is the specific heat at constant pressure, T is the temperature, and the viscous stress tensor can be written in vector notation as

$$\bar{\bar{\tau}} = \mu_{tot}\left(\nabla\mathbf{v} + \nabla\mathbf{v}^{\mathsf{T}} - \frac{2}{3}\bar{\bar{I}}(\nabla\cdot\mathbf{v})\right). \tag{3}$$

In 3D, Eqn. 1 is a set of five coupled, nonlinear partial differential equations (PDEs) that are statements of mass (1), momentum (3), and energy (1) conservation in a fluid. Additional boundary and temporal conditions will be required and are problem dependent. Most commonly, no-slip conditions ($\mathbf{v} = \dot{\mathbf{u}}_\Omega$) are applied to solid surfaces, and far-field approximations that mimic the fluid behavior at infinity [9] are applied to outer boundaries.

Assuming a perfect gas with a ratio of specific heats γ and gas constant R, one can determine the pressure from $p = (\gamma-1)\rho\left[E - \frac{1}{2}(\mathbf{v}\cdot\mathbf{v})\right]$, the temperature is given by $T = p/(\rho R)$, and $c_p = \gamma R/(\gamma-1)$. In accord with the standard approach to turbulence modeling based upon the Boussinesq hypothesis [10], which states that the effect of turbulence can be represented as an increased viscosity, the total viscosity is divided into laminar and turbulent components, or μ_{dyn} and μ_{tur}, respectively. In order to close the system of equations, the dynamic viscosity μ_{dyn} is assumed to satisfy Sutherland's law [11] (a function of temperature alone), the turbulent viscosity μ_{tur} is computed via a turbulence model, and

$$\mu_{tot} = \mu_{dyn} + \mu_{tur}, \quad \mu^*_{tot} = \frac{\mu_{dyn}}{Pr_d} + \frac{\mu_{tur}}{Pr_t}, \tag{4}$$

where Pr_d and Pr_t are the dynamic and turbulent Prandtl numbers, respectively.

The turbulent viscosity μ_{tur} is obtained from a suitable turbulence model. The Shear Stress Transport (SST) model of Menter [12] and the Spalart-Allmaras (S-A) [13] model are two of the most common and widely used turbulence models. For treating purely laminar or inviscid problems, note that the laminar Navier-Stokes equations and the Euler equations can be recovered from Eqn. 1 by removing the contribution of the turbulence model to the viscosity and by eliminating all viscous terms, respectively, when appropriate. With the Euler equations, flow tangency boundary conditions are applied to solid surfaces in the place of a no-slip condition.

2.1.2 Numerical implementation

The governing fluid equations are spatially discretized on unstructured meshes via the Finite Volume Method (FVM) using a median-dual, vertex-based scheme with a standard edge-based structure. Instances of the state vector U are stored at the nodes of the primal mesh, and the dual mesh is constructed by connecting the primal cell centroids, face centroids, and edge midpoints surrounding a particular node.

Convective fluxes are discretized using either a centered scheme, such as the Jameson-Schmidt-Turkel (JST) [14] method, or an upwind scheme, such as the Roe [15] method. For upwind methods, second-order accuracy is easily achieved via reconstruction of variables on the cell interfaces by using a MUSCL approach with gradient limiters. The convection of the turbulence variables is discretized using an upwind scheme (typically first-order). Viscous fluxes are computed with the node-gradient-based approach due to Weiss et al. [16]. The Green-Gauss or weighted least-squares methods are available for approximating the spatial gradients of the flow variables. Source terms are approximated via piece-wise reconstruction in the finite-volume cells.

For unsteady flows, a dual time-stepping strategy [17, 18] has been implemented for obtaining a specified accuracy in time. In this method, the unsteady problem is transformed into a

series of steady problems at each physical time step that can then be solved using all of the well-known convergence acceleration techniques for steady problems. Each physical time step is relaxed in pseudo time using implicit integration.

During time-accurate calculations on dynamic meshes, the coordinates and velocities of the grid nodes must be updated at each physical time step using suitable methods for displacing the nodes of the volume mesh and computing the resulting grid node velocities. Two typical strategies involve rigid mesh transformations or dynamically deforming meshes. In the former, the grid node coordinates and velocities are often computed analytically based on a prescribed motion of the mesh as a rigid body. The dynamically deforming case, which is typically necessary for FSI problems, requires an additional technique to deform the fluid volume mesh to conform to specified changes in the positions of solid boundaries. After updating the position of all nodes in the fluid mesh, the local grid velocity at a node can be computed by storing the node coordinates at prior time instances and using a finite differencing approximation that is consistent with the chosen dual time-stepping scheme.

Finally, when computing unsteady flows on dynamic meshes with the ALE form of the equations, a Geometric Conservation Law (GCL) should be satisfied. First introduced by Thomas and Lombard [19], it has been shown mathematically and through numerical experiment [20–22] that satisfying the GCL can improve the accuracy and stability of the chosen scheme. A straightforward technique for the numerical implementation of the GCL [23, 24] has been included as part of the dual-time stepping approach.

2.2 Structural mechanics

2.2.1 Rigid body models

The following model can be considered for the constrained displacement of non-deformable bodies. The kinematics can be described by a finite set of degrees of freedom corresponding to a translation of body-attached point O and a rotation around this point. This is illustrated in Fig. 1 where both an absolute frame of reference and a relative frame of reference attached to the solid body (denoted by the ' symbol) are considered. Initially, these two referential frames are superposed. After a displacement, the absolute position $\mathbf{s}$ of any point P within the solid can thus be written as [25]:

$$\mathbf{s}_\mathrm{P} = \mathbf{s}_\mathrm{O} + \mathbf{r}' = \mathbf{s}_\mathrm{O} + \mathbf{R}\mathbf{r} \tag{5}$$

where $\mathbf{s}_\mathrm{O}$ is the translation of the reference point, $\mathbf{r}'$ is the relative position of P in the body-attached frame of reference, $\mathbf{r}$ is the position of P before the displacement and $\mathbf{R}$ is a rotation matrix.

The rotation matrix is based on the choice of the parametrization for the rotation and describes the rotation of the relative frame of reference O' in the absolute referential frame. In the case of a three-dimensional rotation, it can be separated into ordered rotations of angles θ, ϕ and ψ around the x, y and z-axis respectively, which gives the following rotation matrix:

$$\mathbf{R}(\theta,\phi,\psi) = \begin{bmatrix} \cos\phi\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\theta\sin\psi & \cos\theta\sin\phi\cos\psi + \sin\theta\sin\psi \\ \cos\phi\sin\psi & \sin\theta\sin\phi\sin\psi + \cos\theta\cos\psi & \cos\theta\sin\phi\sin\psi - \sin\theta\cos\psi \\ -\sin\phi & \sin\theta\cos\phi & \cos\theta\cos\phi \end{bmatrix}$$

In the case where the rigid body displacement is dynamically constrained by some constant stiffness or damping, Lagrange's equation [26, 27] can be used to build the equations of motion

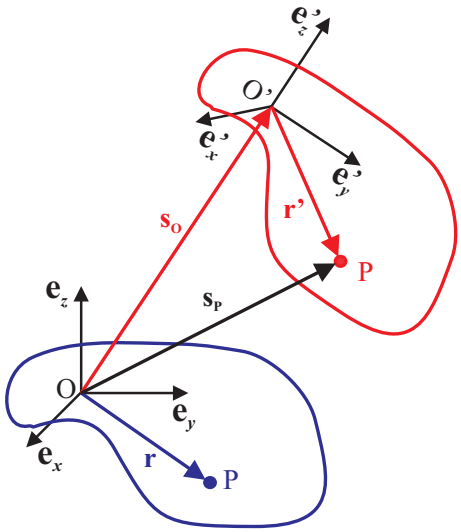


Figure 1: Vectorial description of a rigid body displacement, before (blue) and after (red) the displacement.

for an arbitrary number n (from one to six) of degrees of freedom u_j:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}_j}\right) - \left(\frac{\partial T}{\partial u_j}\right) + \left(\frac{\partial V}{\partial u_j}\right) + \left(\frac{\partial D}{\partial \dot{u}_j}\right) = F_j(t) \qquad j = 1, \cdots, n \tag{6}$$

where T is the kinetic energy related to the mass and the inertia, V the potential energy related to the stiffness, D is the dissipation function related to the damping and F_j is the generalized time-dependent force associated to the j^{th} degree of freedom. This can be expressed in matrix form as:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F}(t) \tag{7}$$

where $\mathbf{u}$ is the vector containing the u_j.

2.2.2 Solid mechanics

In the context of geometrically non-linear, solid mechanics problems, the governing equation may be written [28] as

$$\rho_s \frac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla(\mathbf{F} \cdot \mathbf{S}) + \rho_s \mathbf{f} \quad \text{in } \Omega \times [0, t] \tag{8}$$

where ρ_s is the structural density, $\mathbf{u}$ are the displacements of the solid, $\mathbf{F}$ the material deformation gradient, $\mathbf{f}$ the volume forces on the structural domain, and $\mathbf{S}$ the second Piola-Kirchhoff stress tensor, which are related to the Cauchy Stress tensor $\boldsymbol{\sigma}$ by means of a Piola transformation [29]. The mechanical response of the continuum is addressed by means of a Lagrangian (*spatial*) description, where the displacements, velocities and accelerations of the structure at the nodes are tracked by following material particles [29, 30].

The current implementation of a solid structural solver within SU2 accounts both for linear elastic problems, and also for geometrically non-linear problems with a hyperelastic, Neo-Hookean material model. In a linear setting, and assuming no structural damping, Eq. 8 may be discretized in space using the Finite Element Method (FEM), resulting in

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}_L\mathbf{u} = \mathbf{F}(t) \tag{9}$$

where $\mathbf{F}(t)$ is the vector of externally applied forces, $\mathbf{M}$ the mass matrix and $\mathbf{K}_L$ the linear stiffness matrix, which multiply, respectively, the acceleration and displacement terms of the solution [31]. In a non-linear setting, on the other hand, we linearise the principle of virtual work and reformulate the problem in an incremental, implicit Newton-Raphson way [29,31], as

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{u}}_{t+1} + \mathbf{K}_{NL}\Delta\mathbf{u}^{(k} &= -\mathbf{R}_{t+1} \\ \mathbf{u}_{t+1}^{(k} &= \mathbf{u}_{t+1}^{(k-1} + \Delta\mathbf{u}^{(k} \\ \mathbf{R}_{t+1} &= \mathbf{T}_{t+1}^{(k-1} - \mathbf{F}_{t+1} \end{aligned} \tag{10}$$

where the tangent matrix $\mathbf{K}_{NL}$ accounts for the constitutive and stress terms of the equations and $\mathbf{R}$ is the residual vector which is computed as the difference between internal equivalent nodal forces $\mathbf{T}$ and the external forces $\mathbf{F}$. In order to obtain time-accurate solutions on solid structural analysis, both the Newmark [32] and the Generalized-α [33] integration methods have been implemented in SU2.

2.3 Coupling

2.3.1 Coupling conditions

It is necessary to impose appropriate boundary conditions on the interface in order to fully define the coupled problem. First of all, continuity of displacements may be imposed over the Fluid-Structure interface Γ as

$$\mathbf{u}_{\Gamma_f} = \mathbf{u}_{\Gamma_s} = \mathbf{u}_{\Gamma}. \tag{11}$$

Second, assuming a viscous flow with no-slip boundary conditions, the tractions on the fluid interface are defined as $\mathbf{t}_f = -p\mathbf{n}_f + \bar{\bar{\boldsymbol{\tau}}}_f\mathbf{n}_f$, where p is the pressure of the fluid on the interface, $\bar{\bar{\boldsymbol{\tau}}}$ the viscous stress tensor and $\mathbf{n}_f$ the dimensional, fluid normal outwards from the surface. On the other hand, the tractions on the structural interface are defined as $\mathbf{t}_{s,n} = \bar{\bar{\boldsymbol{\sigma}}}_s\mathbf{n}_s$, where $\mathbf{n}_s$ is the dimensional, structural normal pointing away from the surface.

We can now define a Dirichlet-to-Neumann [34, 35] non-linear operator that maps the displacements on the fluid interface into the fluid tractions, $\mathbf{t}_{\Gamma_f} = \mathscr{F}(\mathbf{u}_{\Gamma_f})$ on Γ_f. Analogously for the structural side, $\mathbf{t}_{\Gamma_s} = \mathscr{S}(\mathbf{u}_{\Gamma_s})$ on Γ_s. Imposing equilibrium of tractions, and combining it with the continuity condition in Eqn. (11), we obtain a Steklov-Poincaré equation [34, 35]

$$\mathscr{F}(\mathbf{u}_{\boldsymbol{\Gamma}}) + \mathscr{S}(\mathbf{u}_{\boldsymbol{\Gamma}}) = 0. \tag{12}$$

This equation can be rewritten in the form of a fixed-point equation [35] using an inverse Neumann-to-Dirichlet operator on the structural side, $\mathbf{u}_{\Gamma_s} = \mathscr{S}^{-1}(\boldsymbol{\lambda}_{\Gamma_s})$, resulting in the interface condition

$$\mathscr{S}^{-1}(-\mathscr{F}(\mathbf{u}_{\boldsymbol{\Gamma}})) = \mathbf{u}_{\boldsymbol{\Gamma}}. \tag{13}$$

2.3.2 Time coupling

A partitioned approach has been adopted in this work, which permits the employment of an adequate solver for each subproblem [35] while maintaining the modularity of SU2. Two possible strategies are available for the integration in time of the coupled problem: an explicit, loosely-coupled method, and an implicit, strongly-coupled method. The choice of the scheme will depend on the particularities of the problem to be solved.

The explicit procedure runs sequentially one single solution of the the fluid and structure solvers and advances the solver in time without enforcing the coupling conditions at the end of

each time step. Due to its simple and modular implementation, this is widely used in computational aeroelasticity, where the coupled dynamics are often mostly determined by the vibrations characteristics of the structure [3, 36–38]. However, this scheme may be unstable or inaccurate in certain cases, due to its high dependency on the density ratio ρ_s/ρ_f and the compressibility of the flow [35, 39, 40].

An implicit scheme improves the quality of the solution for cases in which explicit algorithms are inadequate. Implicit schemes require both solvers to meet the coupling conditions defined in section 2.3.1 at the end of each time step. In particular, our implementation uses the Block Gauss-Seidel (BGS) method, which iteratively solves the fluid and structural problems within the time step until a tolerance criterion is met in the continuity condition (11).

However, low convergence and even divergence has been reported when density ratios are low, the flow is incompressible or the structural deformations are large [40–42]. One common strategy to improve the convergence of the scheme is the use of high-order displacement predictors combined with the employment of relaxation techniques [35,40,43,44]. The definition of a fixed relaxation parameter ω generally results on a large number of iterations [43]; however, the use of the Aitken's Δ^2 dynamic relaxation parameter [45], which has also been included in SU2, is a simple and efficient option [40, 43, 44] to improve the convergence of implicit schemes.

2.3.3 Spatial coupling

The computational study of a Fluid-Structure Interaction problem using a partitioned approach requires the transfer of the coupling conditions defined in 2.3.1 between the fluid and structural discretizations in an appropriate way. Due to differences in the physical behavior and the possibility of local effects on one, or both, of the domains, matching discretizations will often either over- or under-refine one of the domains. This motivates the implementation of interpolation between non-matching discretizations. It should be noted that the interpolation of fields on the non-matching interface may compromise the accuracy of the whole simulation [46], and several authors have addressed the problem of coupling two non-matching discretizations [1, 46–51]. The general criteria for choosing the appropriate interpolator relates to its efficiency, robustness, accuracy, and conservation of momentum and energy [1, 46, 49, 52].

It is a common approach in the literature to apply the principle of virtual work [1, 46, 49, 53] to define a conservative transfer scheme. Brown [52] and Farhat [1] have introduced projection methods that apply the conservation of virtual work to the problem of transferring load and deformations in FSI problems on non-matching meshes. By requiring the interpolation method to conserve virtual work across the displacements and forces on the two meshes, the resulting distributions are physically consistent and result in the same integrated forces. Following their work an expression for the virtual work performed by the forces applied to the structure $\mathbf{F}$, moved through a virtual nodal displacement $\delta\mathbf{u}_{\Gamma,s}$ is:

$$\delta W_E = \mathbf{F}^T \cdot \delta\mathbf{u}_{\Gamma,s} \tag{14}$$

The virtual work must equal the similar expression using the forces of the fluid solution and the virtual displacements of the fluid mesh:

$$\delta W_E = \int_{\Gamma,f} (-p\vec{n}_f + \bar{\bar{\boldsymbol{\tau}}}_f \vec{n}_f)^T \cdot \delta\mathbf{u}_{\Gamma,f} dS_f \tag{15}$$

where $\vec{n}_f$ is the unit normal on the fluid side of the problem and S_f the curvilinear coordinate

along the fluid interface. Equating Equations (14) and (15):

$$\mathbf{F}^T \cdot \delta\mathbf{u}_{\Gamma,s} = \int_{\Gamma,f} (-p\vec{n}_f + \bar{\bar{\boldsymbol{\tau}}}_f \vec{n}_f)^T \cdot \delta\mathbf{u}_{\Gamma,f} dS_f \tag{16}$$

In the discrete interface, the displacements on the fluid side may be mapped using the displacements on the structural side through the use of a matrix $\mathbf{H}$ which will need to be defined for each problem and method used, thus resulting in $\mathbf{u}_f = \mathbf{H}\mathbf{u}_s$. For the common case of a finer fluid mesh we have a set of distributed forces $\boldsymbol{\lambda}_f$ including both surface pressure and traction. According to the previously described principle of virtual work, the tractions at the structural interface may be mapped from the fluid domain as $\boldsymbol{\lambda}_s = \mathbf{H}^T\boldsymbol{\lambda}_f$. Therefore, it is possible to define a full space-coupling scheme that would meet the criteria of conservation of work by adequately defining a transformation matrix $\mathbf{H}$. Global conservation of forces can also be achieved by imposing that the rowsum of $\mathbf{H}$ is equal to one [49].

3 IMPLEMENTATION

The SU2 software suite was conceived as a common infrastructure for solving multi-physics PDE-based problems on unstructured meshes. The full suite is composed of compiled C++ executables and high-level Python scripts that perform a wide range of tasks related to PDE analysis, PDE-constrained optimization, or coupled multi-physics problems. Below, we will describe some of the key implementation details for the various FSI approaches for both C++ and Python. A more complete description of the suite can be found in Economon et al. [54].

3.1 Fluid solver

Much of the C++ class design in SU2 is shared by all of the core modules. In particular, this includes the geometry (i.e., grids), integration, configuration (input file), and output class structures. Only specific numerical methods for the convective, viscous, and source terms are re-implemented for different physical models where necessary. There is no fundamental limitation on the number of state variables or governing equations that can be solved simultaneously in a coupled or segregated way (other than the physical memory available on a given computer architecture), and the more complicated algorithms and numerical methods (including parallelization, multigrid, and linear solvers) have been implemented in such a way that they can be applied without special consideration during the implementation of a new physical model.

The schematic in Fig. 2 shows the structure of the fluid solver in SU2. A similar structure would arise for other single-physics solvers, as will be seen below for the structural solver. The main SU2 loop drives the iterations of the particular physics, and in this example, the iteration class for the flow problem is executed. Within that iteration, some classes that are shared among all solvers are leveraged, such as the geometry and integration classes, while others have been specialized to the flow problem, such as the solver, numerics, and variable classes. The solver, numerics, and variable classes contain the routines that are specific to a chosen physical model.

3.2 Structural solver

The structural solver has been designed based on the original fluid solver, and follows its main structure, as shown in Fig. 3. However, some modifications have been carried out in order to account for the specific characteristics of solid mechanics problems. In particular, two layers of abstraction have been included that deal, independently, with geometrical and material nonlinear effects. The Finite Element analysis is performed at the solver level and

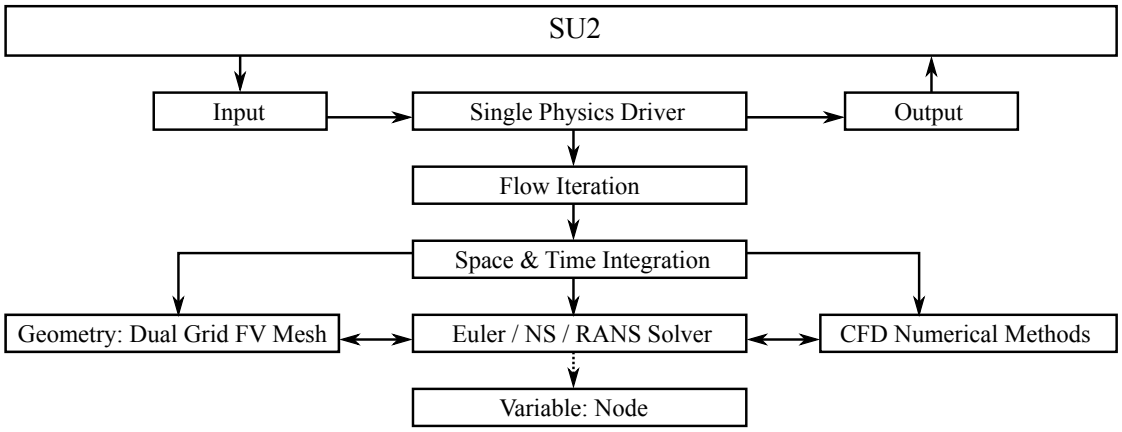


Figure 2: Structure of the fluid solver.

relies on a C++ class common for any kind of element. This structure has been designed with the philosophy of maintaining the flexibility of the code while easing and encouraging further contributions. Further details may be found in Sanchez *et al.* [7].

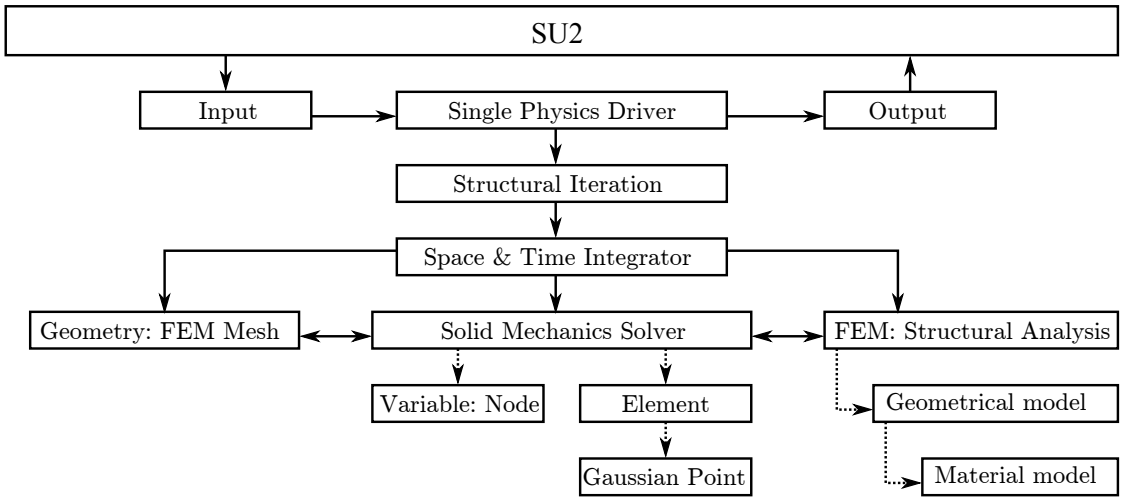


Figure 3: Structure of the Solid Mechanics solver.

3.3 Multizone solver

The solution process in SU2 has been redesigned to improve its ability to run several solvers with a single instance of the software. With this objective, a so-called *driver* structure has been introduced, in which the developer can make use of the different parts of the code as if they were black-boxes. When different solvers are used and need to interact with each other, it is necessary to introduce mechanisms to communicate information at their interface. This is a

complex task, specially when the solution process is run in multiple processors.

This situation is illustrated in Fig. 4 for an example in which a solid wall is immersed in a flow in a channel and run in two processors. A multi-core, multi-physics problem run on a distributed memory computer requires the MPI communication of information to be done not only within each independent solver, but also across the different solvers. Moreover, an optimal partition of the discretization of each zone of the physical domain, which reduces to a minimum the number of edges cut by the partitions, may result in non-compliant regions across zones. This fact forces the transfer mechanisms to be able to efficiently distribute the information from processor *i* in the *fluid* domain to processor *j* in the *structural* domain. This issue has been resolved by abstracting the physics of the information to be transferred from the MPI transfer routines, thus allowing for different and specialized teams to develop and improve each particular part of the problem independently. More specifically, and for Fluid-Structure Interaction applications, the resulting code structure is shown in Fig. 5.

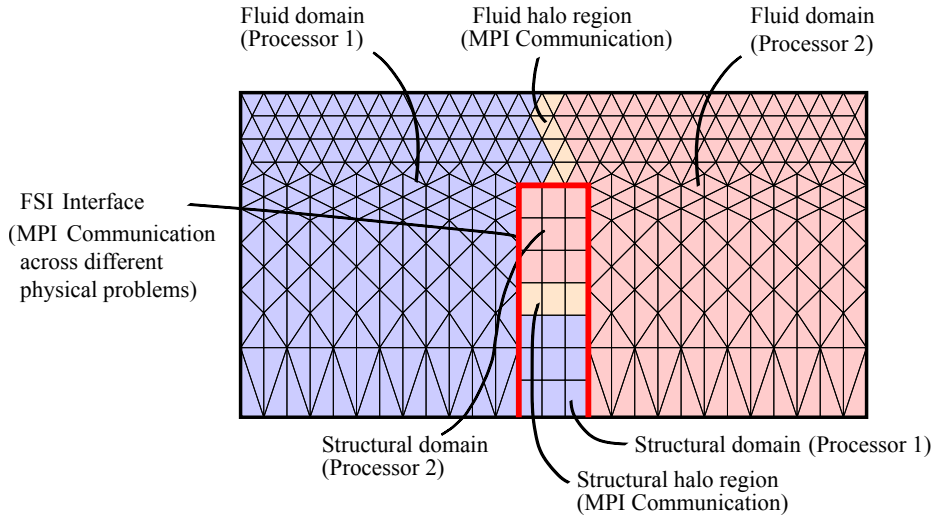


Figure 4: Sample FSI problem run in multiple cores.

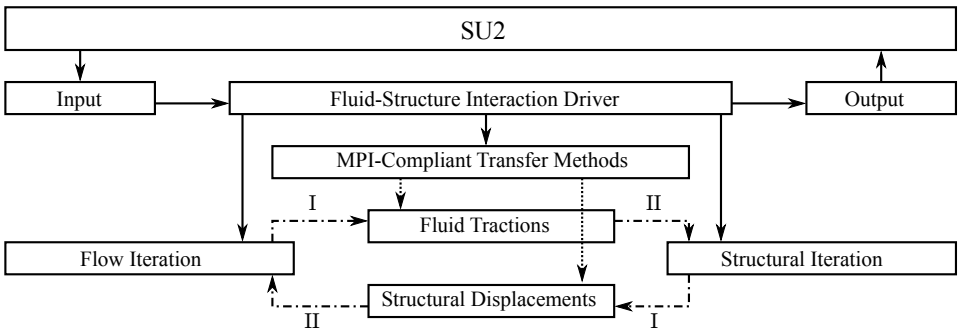


Figure 5: Transfer structure for Fluid-Structure Interaction problems.

3.3.1 Time-accurate solver

The self-contained FSI solver implemented within SU2 includes a Block Gauss-Seidel (BGS), partitioned, strongly-coupled, implicit solution method in order to advance the fluid and structural solvers in time, as described in section 2.3.1. This solver takes advantage of the modular implementation of SU2, in particular of the *driver* and *iteration* methods which allow to use each solver as a black-box, together with the transfer methods described in section 3.3. The code subiterates within each time step according to the diagram shown in Fig. 6, until the convergence criteria that has been established over the interface displacements is met.

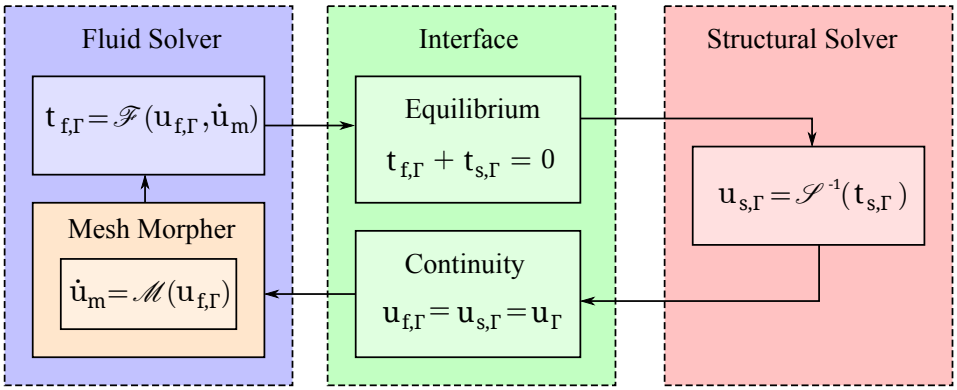


Figure 6: Block Gauss-Seidel subiterations for strong coupling.

3.3.2 Interpolation methods

It is rarely the case that the separate fluid and structure problems will require the same mesh distributions. For this reason the capability to interpolate between non-matching meshes has been implemented in SU2. The location of interpolation within the problem is shown in Fig. 7.

Interpolation is implemented as a class (CInterpolator) which locates the 'donor' nodes and calculates weighting coefficients. These values are evaluated once in a preprocessing step of the multi-zone problem, and stored such that the transfer structure will continue as described in section 3.3 with the only difference that the transferred information originates from a weighted sum of nodes rather than being restricted to a single node. The particular weighting and selection of nodes is determined by which interpolation method is chosen. Each interpolation method is implemented as a child class of CInterpolator, inheriting the data structures and methods needed by most interpolation routines.

The implemented interpolation methods include a Nearest-Neighbor approach which identifies the closest point and uses a weighting value of 1.0, and an Isoparametric approach which computes isoparametric coefficients for the nodes of the closest surface element. A 'Conservative' approach is also implemented, which re-uses the computation of isoparametric coefficients, but rather than re-computing the coefficients in either direction it computes the coefficients from the coarser mesh to the finer mesh and mirrors these values for the transfer in the opposite direction, such that the transfer matrix is the same in both directions. This is consistent with methods described in literature [52] and section 2.3.3.

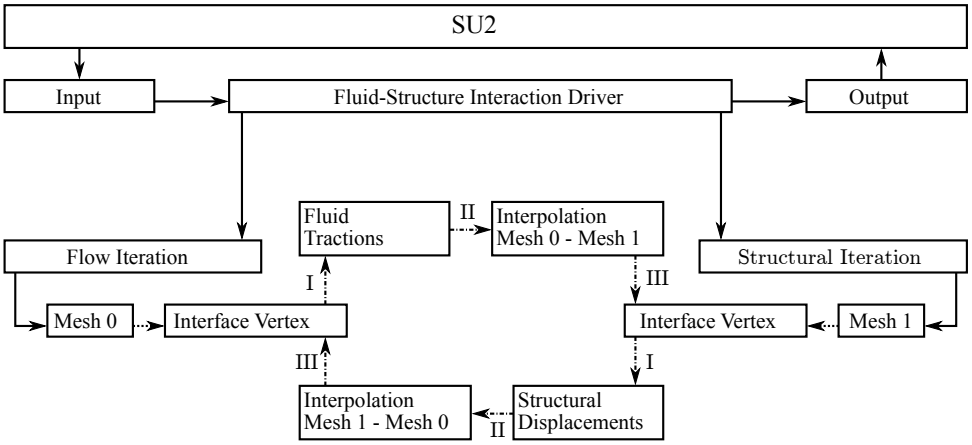


Figure 7: Interpolation structure for Fluid-Structure Interaction problems.

3.4 Python wrapper

SU2 can also be used for FSI computations when it is externally coupled with a third-party structural solver. This particular type of coupling is achieved through a Python wrapper designed to synchronize the solvers within one single Python environment. The implementation of the wrapper is detailed first, and the coupling mechanisms are illustrated after that.

3.4.1 Implementation of the wrapper

The flexibility of the Python language is used to couple SU2 with other third-party structural solvers in order to perform externally-coupled FSI simulations. Some specific functions implemented in SU2, such as those performing CFD iterations or mesh deformation, are rearranged in a new C++/API object representing the SU2 solver as the fluid part of the FSI algorithm. The SU2 code with the API object are then compiled as a dynamic library using any C++ compiler. An additional compilation using SWIG is required to translate the API into a wrapping Python module that is linked with the library and importable in any Python script where the FSI algorithm is built. In this way, the SU2 solver, through the Python-wrapped API, can be managed in an intuitive language but the critical and computationally intensive calculations are performed with the same C++ routines as in the basic source code. This wrapping methodology is illustrated in the left (black) part of Fig. 8.

3.4.2 Coupling with a third party solver

The coupling with external structural solvers is achieved within the Python script itself, providing the considered solver also has a Python-translated API that can be imported as a Python module representing the solid part of the computation. This is illustrated in the right (gray) part of Fig. 8. For modularity purposes, the methods of the API object of any structural solver to be coupled with the Python wrapper should be built on a template that covers the main steps of an FSI algorithm.

The communications between the fluid and solid modules can be directly performed through memory by explicitly exchanging memory addresses as input/output to/from the specific API functions. For example, when an update of the solid displacement is needed, the solid part

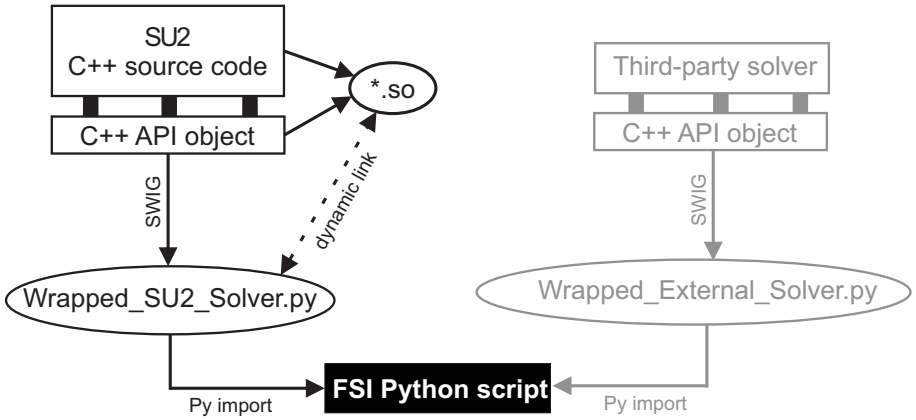


Figure 8: Python wrapping of SU2, from source code to FSI script (black). A symmetric architecture can then be applied to the structural solver (gray).

sends the memory address of the array where the new position of the fluid-solid interface is stored. These values can thus be interpolated from the solid to the fluid mesh and processed before deforming the fluid mesh. The same mechanism is used for communicating fluid loads acting on the solid. This leads to an intuitive and flexible way for synchronizing the solvers, since they are not considered as independent codes anymore but as Python objects within the FSI script. Another advantage of the wrapper is that all of the methodologies for FSI that were described in section 3.3 are also available in this approach. This implementation is also fully compatible with code parallelization using MPI thanks to appropriate MPI Python modules so that the MPI communication pattern within the solvers is kept. Third party structural solvers of various complexity can be coupled through the wrapper. In this paper, two different cases are presented. The first is one where the wrapper is used to couple SU2 to a spring-analogy solver and in the second, SU2 is coupled with an external finite element solver capable of handling 2d and 3d structural meshes for linear and non-linear structural analysis.

In the case of rigid body aeroelastic applications, the wrapper is used to couple an external in-house spring analogy solver. This solver is used to predict the dynamically constrained rigid body motion that was previously described in section 2.2.1 by integrating the equations of motion, Eq. 7, in time for a six-DOF rigid body system with the Generalized-α method. The solver itself takes the structural parameters, such as the mass, moment of inertia, stiffness or damping as inputs.

The surface tractions on each node of the interface are communicated to the spring analogy solver through the wrapper. The global aerodynamic loads, such as the lift and drag forces or the aerodynamic moments, are then computed in order to solve Eq. 7 and get the new positions of each node of the solid interface with Eq. 5. The interface displacement is then communicated from the spring analogy solver to SU2 as inputs for the dynamic mesh deformation.

For this particular case, only the fluid part is parallelized since the structural part can be almost instantaneously solved without any significant computational cost. The absence of mesh discretisation in the solid part enables the coupled simulation to get rid of the interpolation of fields at the interface. The amount of data needed to be exchanged is also small so that the communication between the fluid and the solid is performed on the master process, then the data are broadcasted/gathered to/from the other processes if needed.

For the second case, SU2 is coupled with the Toolkit for Analysis of Composite Structures (TACS) [55], a finite element based structural analysis solver. TACS has a python interface that

allows it to be coupled easily with SU2. The load transfer between the fluid and structure solvers is performed using an in-house python module developed for finite element based structural weight estimation for aircraft configurations [56]. The python based FSI framework comprising SU2, TACS and the load transfer module permits the solution of different FSI problems (steady and unsteady) in two and three dimensions. Different fluid structure coupling architectures like the Conventional Staggered or Block Gauss Seidel can be formulated without recompiling SU2 or TACS simply by rearranging the function calls to the fluid and structural modules in the Python driver scripts. A test case for the unsteady solution of an elastic beam behind a square cylinder (shown in in Section 4.2.3) is used to demonstrate the capabilities of the framework.

4 RESULTS

4.1 Fluid-Structure Interaction solver for rigid body applications

A typical aeroelastic reference model is the two-dimensional pitching-plunging airfoil [57] (2 DOF) with a chord c in a free-stream flow of velocity U_∞ (or Mach number M_∞). This model is illustrated in Fig. 9. The displacement h of the elastic axis is positive downwards and the pitch angle α is positive clockwise. The positions along the chord of the center of gravity x_{CG} and the elastic axis x_{f} are considered from the nose of the airfoil. The product between the distance $(x_{\mathrm{CG}} - x_{\mathrm{f}})$ and the airfoil mass m is the static unbalance S. The spring analogy controls the motion of the airfoil with a stiffness K_h (or K_α) and a damping C_h (or C_α) for the plunging (or pitching) mode.

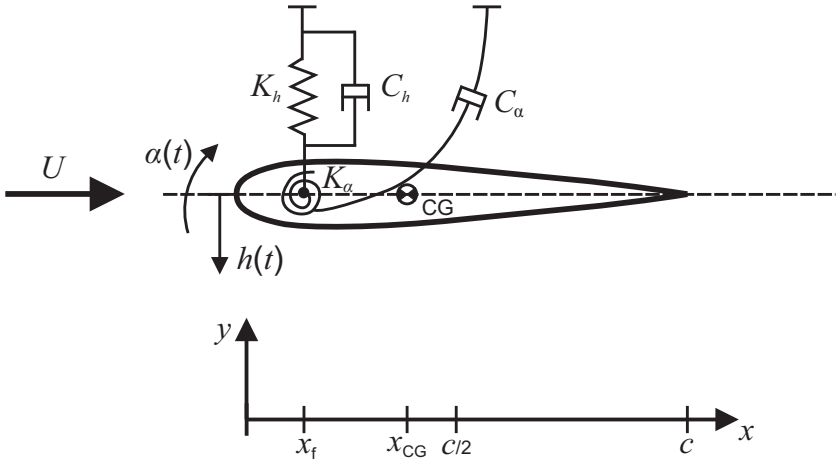


Figure 9: Schematic of a two degrees of freedom pitching-plunging airfoil aeroelastic model.

The equations of motion, Eq. 7, for this aeroelastic system can be written as:

$$\begin{aligned} m\ddot{h} + S\ddot{\alpha} + C_h\dot{h} + K_h h &= -L \\ S\ddot{h} + I_{\mathrm{f}}\ddot{\alpha} + C_\alpha\dot{\alpha} + K_\alpha\alpha &= M \end{aligned} \tag{17}$$

where I_{f} is the inertia of the airfoil around the elastic axis, L the lift and M the aerodynamic moment around the elastic axis. The lift is positive upwards and the aerodynamic moment around the elastic axis is considered positive clockwise, as the pitch angle. This model can also be described by a set of non-dimensional parameters, i.e., the normalized static unbalance χ and inertia r_α, the uncoupled natural frequency ratio $\overline{\omega}$ and the mass ratio μ:

$$\chi = \frac{S}{mb},\ r_\alpha = \frac{I_{\mathrm{f}}}{mb^2},\ \overline{\omega} = \frac{\omega_h}{\omega_\alpha},\ \mu = \frac{m}{\pi\rho_\infty b^2} \tag{18}$$

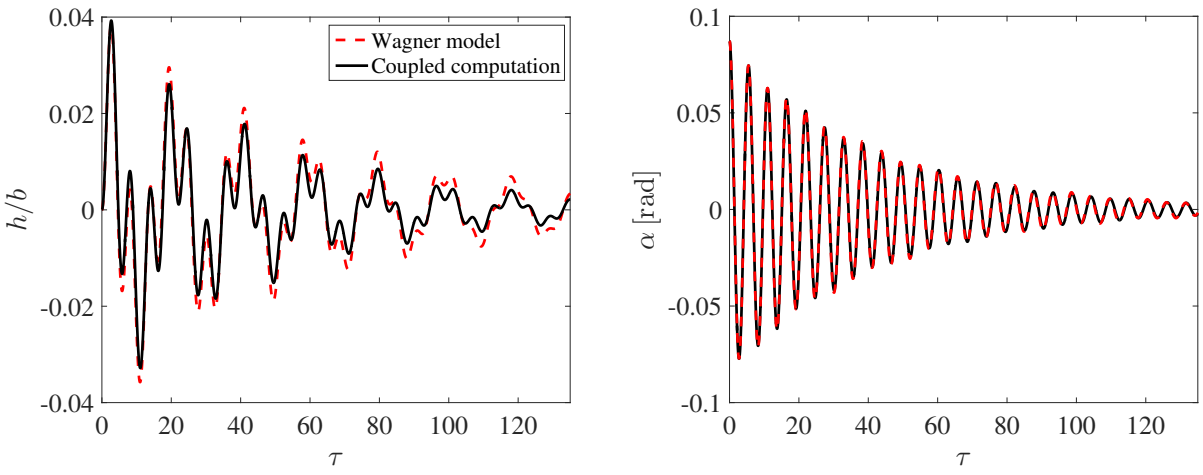
In these expressions b is the half-chord, $\omega_h = \sqrt{K_h/m}$ ($\omega_\alpha = \sqrt{K_\alpha/I_\mathrm{f}}$) is the uncoupled plunging (pitching) natural frequency and ρ_∞ is the free-stream density of the fluid.

4.1.1 Two-dimensional pitching-plunging NACA 0012 airfoil

The Python wrapper is used here to couple SU2 with the basic spring analogy integrator described in section 3.4.2 to solve the basic two-dimensional aeroelastic problem involving a pitching-plunging NACA 0012 airfoil in a free-stream flow. The unsteady RANS solver of SU2 is used with the $k-\omega$ SST turbulence model and the FSI simulation is based on a strongly-coupled scheme with a tolerance on the solid displacement of 10^{-6} (leading to about 4 FSI iterations).

The case is set with the non-dimensional parameters $\chi = 0.25$, $r_\alpha = 0.5$, $\overline{\omega} = 0.3185$ and $\mu = 100$. The elastic axis is placed at the quarter-chord point. Several free-stream Mach numbers are considered in order to capture both sub- and post-critical conditions. The chord-based Reynolds number is fixed at a value $\mathrm{Re}_c = 4 \cdot 10^6$ for a chord $c = 1$ m. The natural pitching frequency is set to $\omega_\alpha = 45$ rad/s. A pitch angle for the airfoil of $\alpha = 0.872$ rad (5°) is used as initial condition. These parameters are chosen in order to predict a subsonic flutter Mach number that allows the comparison with incompressible theoretical models (see below). The calculations are performed with 139 time steps per period of the pitch mode for accuracy purposes.

The sub-critical response of the system for $M_\infty = 0.1$ is compared with theoretical predictions using the thin airfoil theory [58] for aerodynamic load predictions and Wagner function [59] for dynamic fluid-solid coupling. Fig. 10 shows the normalized plunge displacement and pitch angle as a function of a non-dimensional time $\tau = \omega_\alpha t$. At this sub-critical Mach number the system is strongly damped. The coupled computation is in good agreement with the theoretical expectations, particularly for the pitch mode.



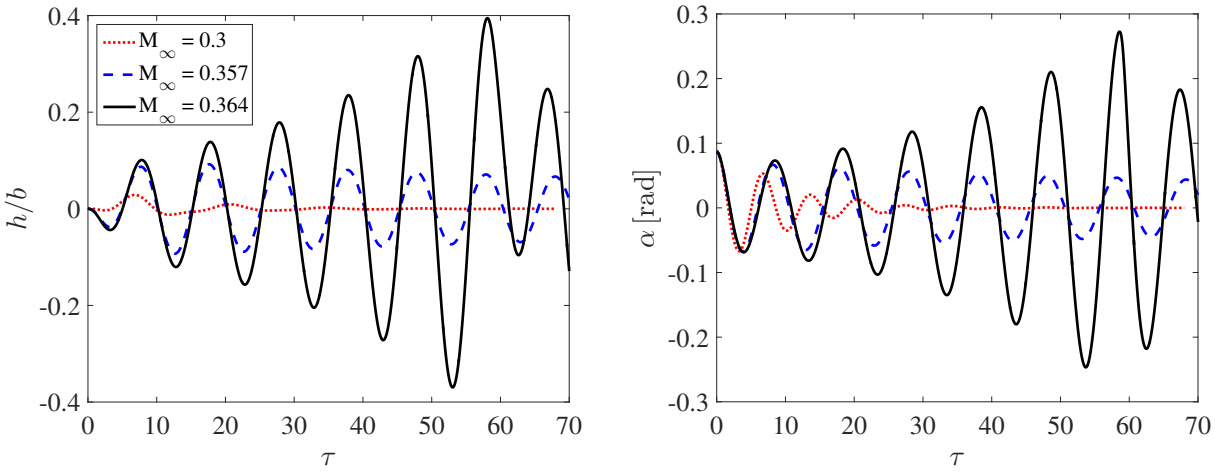
(a) Plunge displacement

(b) Pitch angle

Figure 10: Aeroelastic response as a function of time at sub-critical conditions ($M_\infty = 0.1$).

The Python wrapper is thus able to capture the response at flow conditions being quite far from critical conditions (flutter). Fig. 11 shows the computed responses of the plunging and

pitching modes at three different free-stream Mach numbers corresponding to sub-critical, critical and post-critical conditions. The flutter Mach number, $M_\infty = 0.357$, obtained from the computation differs only slightly from the theoretical prediction based on Wagner model, i.e., $M_\infty^{th} = 0.371$. As expected, the response at $M_\infty = 0.3$ is strongly damped. An increase of the Mach number reduces the damping ratio until critical conditions are reached, at which the amplitude of the response remains constant in time. Finally, at post-critical conditions (e.g., $M_\infty = 0.364$), the response becomes unstable.



(a) Plunge displacement

(b) Pitch angle

Figure 11: Aeroelastic response as a function of time at sub- and post-critical conditions.

Fig. 12 shows the computed dimensionless frequencies of the two airfoil modes as a funcion of the free-stream Mach number. The evolution of the frequencies is in good agreement with the theoretical model. In particular, the two frequencies converge with increasing M_∞ until they coincide at the flutter Mach number and beyond it.

At the post-critical conditions the amplitude of the response is strongly amplified during the first 6 cycles, but then reaches a limit-cycle oscillation (LCO). This is due to the nonlinear effects of boundary layer separation and stall that appear for large displacements and angles of attack. Fig. 13 shows the temporal response of the system over a longer time period. The red bullets correspond to instants when significant flow separation occurs alternatively on the extrados or intrados of the airfoil. A sudden drop of the amplitude is systematically observed after each separation. Subsequently, the amplitude increases again due to flutter instabilities before reaching another separation point. This mechanism allows the aeroelastic response to remain bounded in time and explains the development of a LCO mode.

A visualization of this flow dynamics is shown in Fig. 14 for times corresponding to the second set of red bullets in Fig. 13. The separation occurs when the two airfoil modes reach their local maximum values. On the suction side, the local Mach number becomes very large (close to 1.6) [Fig. 13 (a) and (b)]. Then a massive flow separation event is observed while the airfoil moves back towards its neutral position ($\alpha = h = 0$) [Fig. 13 (c) and (d)]. Finally, the flow progressively reattaches until the instantaneous displacement (both modes) of the airfoil is close to zero [Fig. 13 (e) and (f)]. A similar dynamics takes place for negative pitching angles. This highlights the capabilities of the wrapper coupling method to predict complex and nonlinear dynamics that can be observed in aeroelastic systems beyond the linear coupled mode

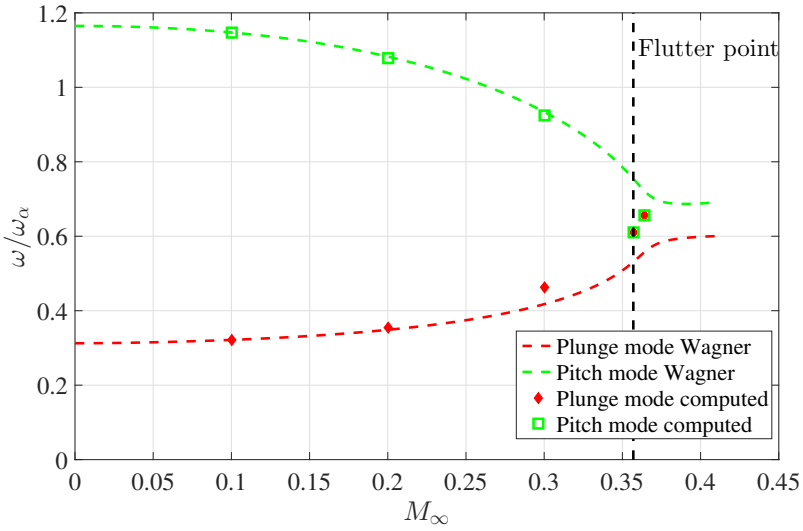


Figure 12: Dimensionless frequencies of the aeroelastic response as a function of the free-stream Mach number.

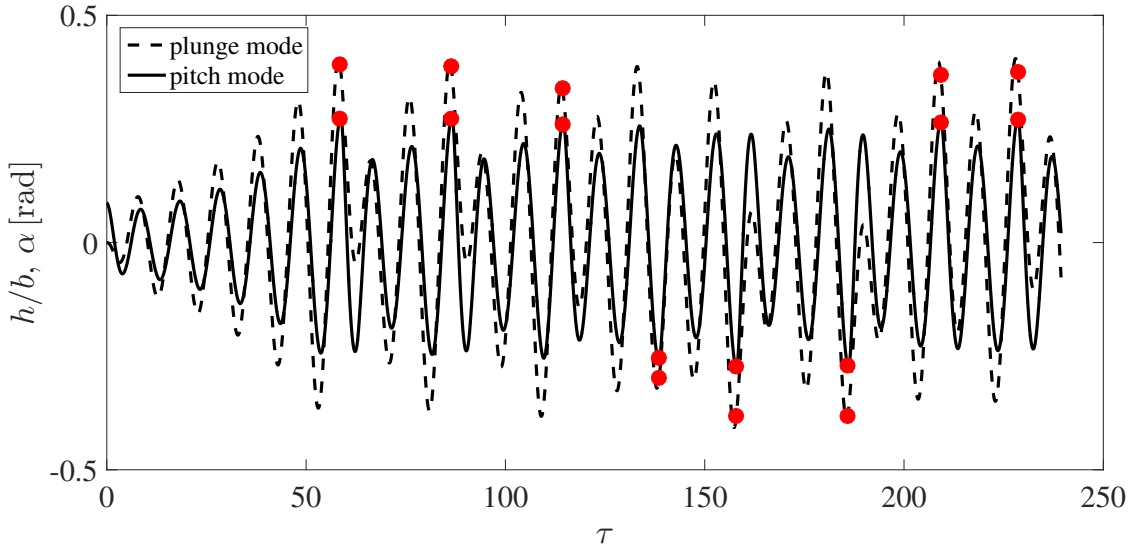


Figure 13: LCO response of the system at $M_\infty = 0.364$ under non-linear aerodynamic phenomena. The times when massive boundary layer separation occurs are marked by the red bullets.

flutter, keeping in mind the loss of accuracy for separated flows due to the RANS model.

4.1.2 Isogai wing section

The Python wrapper coupling SU2 and the spring-analogy integrator is also applied to the classical Isogai wing section aeroelastic case (case A) [60, 61]. This test case represents the dynamics of the outboard portion of a swept back wing in the transonic regime. The corresponding parameters are $\chi = 1.8$, $r_\alpha = 1.865$, $\bar{\omega} = 1$, $\mu = 60$. The elastic axis is placed in front of the airfoil at a distance $x_\mathrm{f} = -b$ from the leading edge and the natural pitching frequency is here

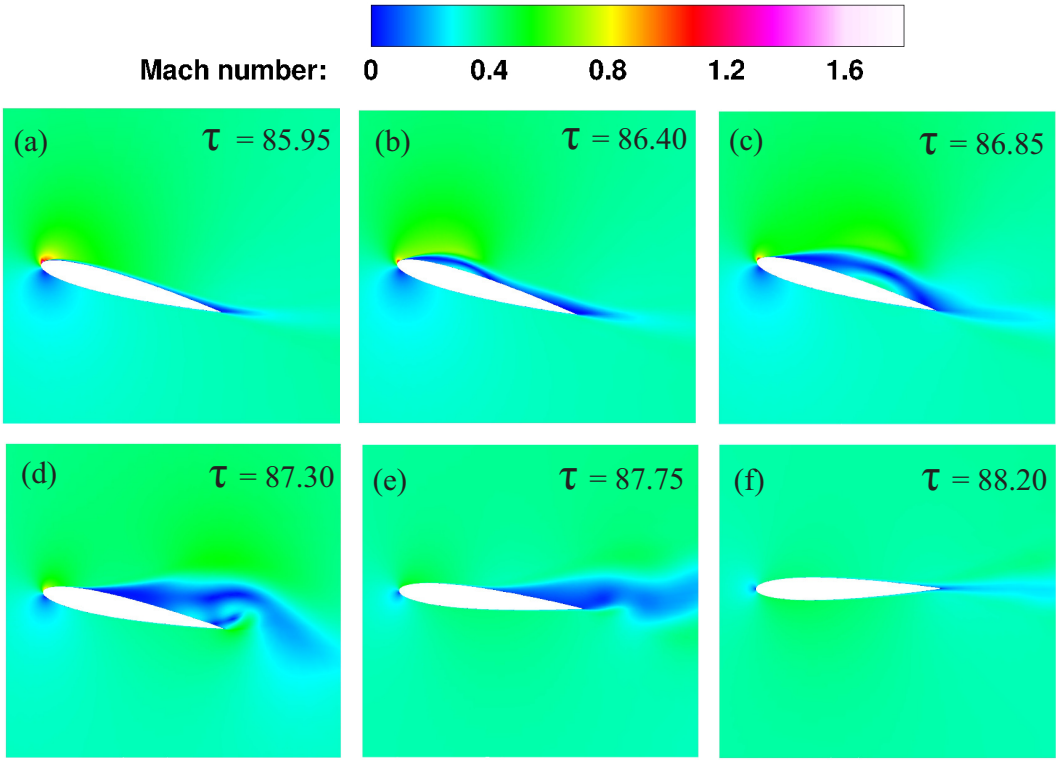


Figure 14: Visualization of the flow separation mechanism leading to a LCO response of the system beyond the flutter point.

set to $\omega_\alpha = 100$ rad/s.

The Euler equations are considered in the transonic regime ($M_\infty = 0.7 - 0.9$) with an initial pitch angle for the airfoil of $\alpha_0 = 0.0174$ rad (1°) and the FSI simulation is again performed with a strong-coupling scheme with the same tolerance than the previous test case. For this case, the number of time steps per period of the pitch mode is reduced to 39.

Several FSI simulations at different transonic Mach numbers are performed with variable speed index

$$V^* = \frac{U_\infty}{b\omega_\alpha\sqrt{\mu}} \tag{19}$$

in order to predict the flutter point. This state is reached when the damping extracted from the dynamic response of the system is close to zero. The results are illustrated in Fig. 15 that shows a comparison of the computed flutter speed indices with those found in other references [62–65]. The best approximation curve (spline) is a representation of the limit between stable and unstable regions. It can be seen that the "transonic dip" and the typical "S-shape" flutter boundary for M_∞ between 0.7 and 0.9 are both well predicted.

4.1.3 Test cases from the 2nd AIAA Aeroelastic Prediction Workshop

The second AIAA Aeroelastic Prediction Workshop [66] was officially launched in January 2015 and took place in January 2016 (both events at Scitech) with the aim of assessing the accuracy of the predictions provided by the available numerical approaches. The planned test cases include the investigation of forced and unforced wing oscillations as well as the study of wing flutter with a pitch-and-plunge kinematics. All measurements were carried out by

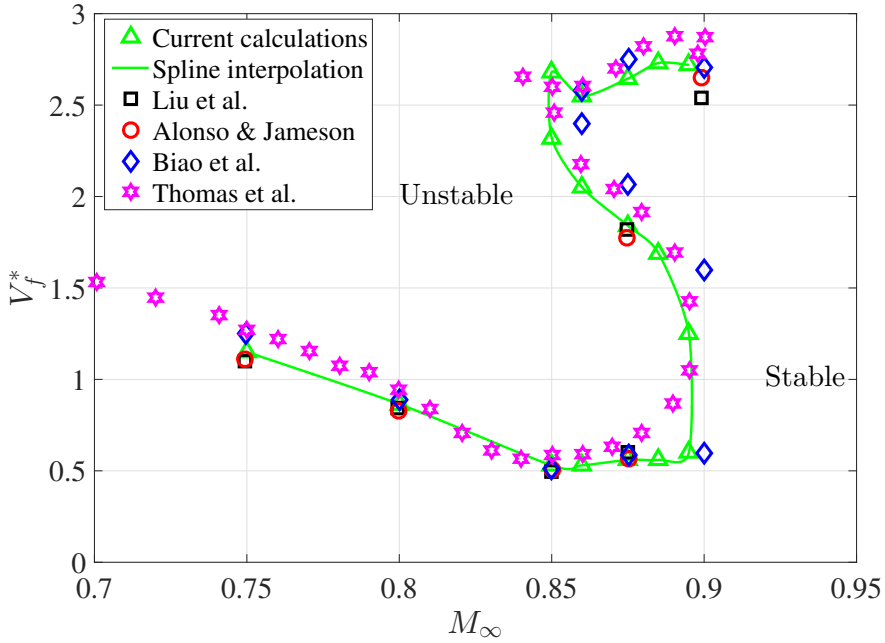


Figure 15: Flutter speed index as a function of the free-stream Mach number. Comparison between current computations and numerical results from the literature.

NASA in Langley's TDT wind tunnel. The wing model is the Benchmark Supercritical Wing (BSCW) [67–69]. All test cases are transonic and supercritical for the BSCW wing. Mach number ranges between 0.70 and 0.84, Reynolds number (based on chord) is around 5 million. The fluids used for the experiments are heavy gases (R12 and R134a depending on the test case). The angle of attack is chosen in order to position the shock wave differently in each test case. The boundary layer is tripped at 7% of the chord. The three dimensionality of the flow in the tip region – the BSCW wing has an aspect ratio of 4 – contributes to the complexity of the case.

Pre-requisites for a physically consistent simulation include the full resolution of the boundary layer, and an acceptable prediction of the steady-state shock-boundary layer interaction, hence a correct positioning of the shock on the upper and lower side of the wing.

The fluid solver in SU2 has been used to simulate test case 1, which includes forced pitch oscillations around the axis at 30% of the chord at frequency 10 Hz with an amplitude of $\pm 1^\circ$. The median angle of attack of 3° is sufficient to generate a small supersonic region around the upper side of the nose of the wing and a shock at approximately 10% of the chord. The distribution of the pressure, obtained from SU2, is shown in Fig. 16. A time-accurate simulation with rigid grid motion was conducted with the second order dual-time stepping, following a steady-state solution, also obtained with SU2. Both simulations were run with the central scheme (JST). The Spalart-Allmaras one-equation turbulence model [13] was used. For the steady-state simulation and for the subiterations in the time-accurate run, a CFL of 4.0 was used with the Euler implicit time stepping (iterations limited to 5).

Time-accurate pressure measurements - static data and first harmonic - are available in a number of points at the section at 60% of the span. The results obtained with SU2 are compared with experimental data, whenever available, and with those obtained with Edge [70], a well-known CFD package developed by the Swedish Defence Research Centre (FOI) for aeroelastic

problems.

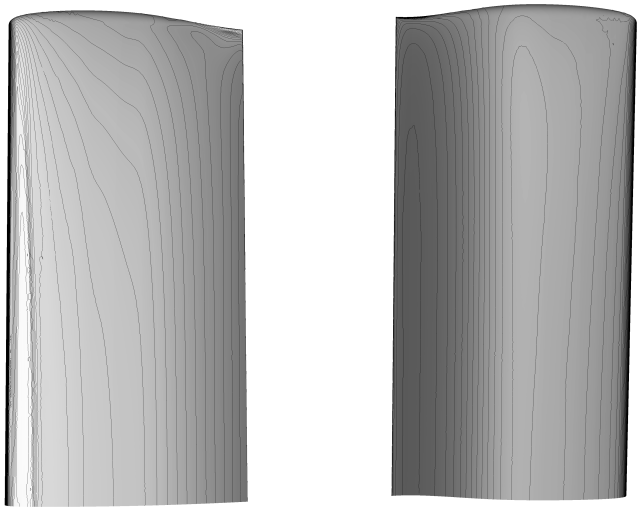


Figure 16: Steady state solution, pressure coefficient distribution over upper and lower wing sides

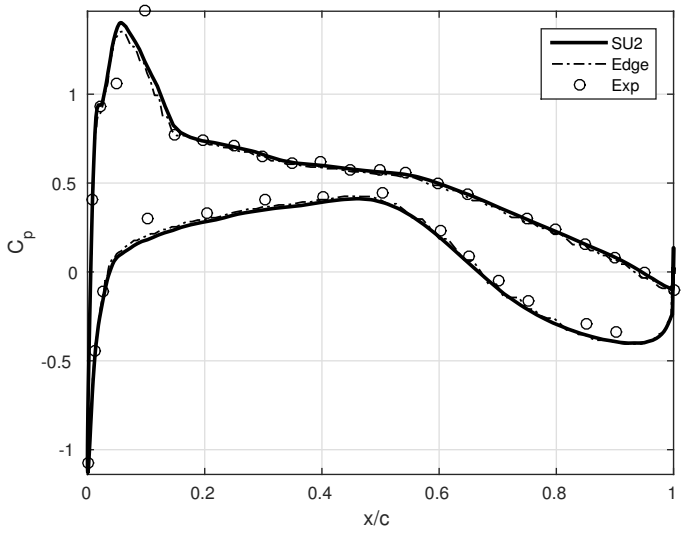


Figure 17: Pressure coefficient distribution averaged over ten periods

The agreement with the experimental data in terms of average pressure coefficient, presented in Fig. 16, may be considered as good despite a difference of a few percentage points of chord in shock position. The agreement between SU2 and Edge is excellent. The comparison of the first harmonic (10 Hz), which is presented in Fig. 18 (magnitude of oscillations) and Fig. 19 (phase delay with respect to pitch signal) shows again a very good agreement between SU2 and Edge and larger deviations, but still acceptable to engineering standards, between CFD and experiments. The most noticeable differences concern the recovery area, downstream of

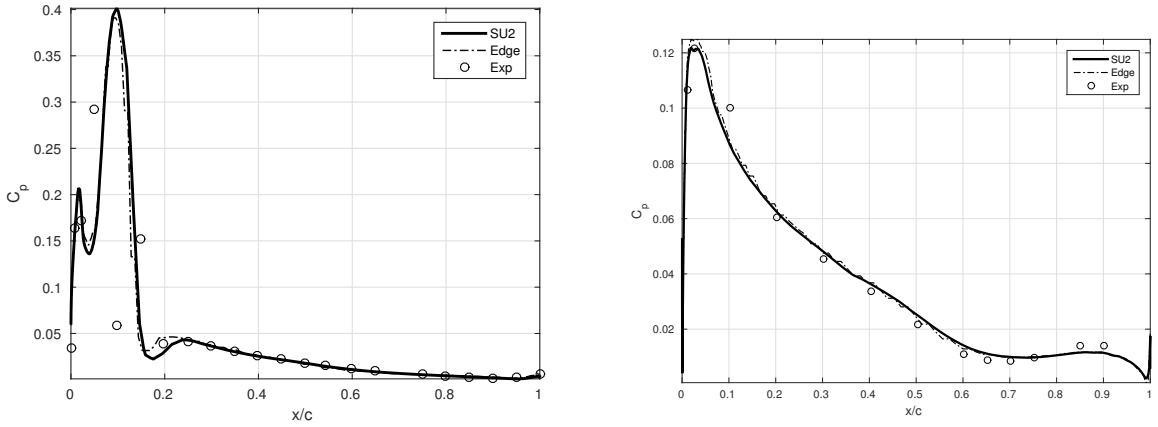


Figure 18: First harmonic of response, magnitude, pressure coefficient, upper (lhs) and lower (rhs) wing side

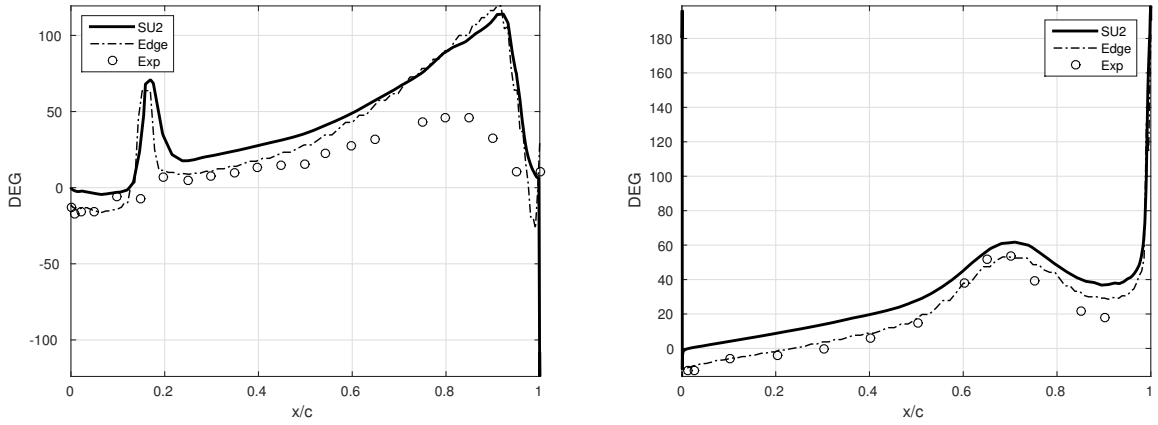


Figure 19: First harmonic of response, phase delay in degrees, pressure coefficient, upper (lhs) and lower (rhs) wing side

the shock, where the predicted phase angle is overestimated, and the shock area, where the experimental data show much lower value in correspondence of sensor 5. The most likely cause - according to the researchers and organizers of the Workshop - is some local deformation effect which is not being taken into consideration in CFD simulation. Two effects are expected to play a major role in the evolution of static pressure on the upper wing side as a function of the oscillating pitch angle; on the one hand, the motion of the stagnation point causes a first pressure peak to grow in phase with pitch angle, on the other hand, the shock wave moves up and downstream also in phase with the pitch angle. The two effects generate a "double peak", which is clearly visible in Fig. 18, in both CFD and experimental data.

The flow being strongly non-linear, the response of the aerodynamics contains several non-negligible harmonics; the Fourier expansion of the pressure coefficients has a non negligible amplitude at the reference frequency $f = 10\,Hz$ and also at $2 \times f$, $3 \times f$ and $4 \times f$. This is presented in Fig. 20. Again, the agreement between SU2 and Edge is very good. No experimental data is available for frequencies higher than 10 Hz.

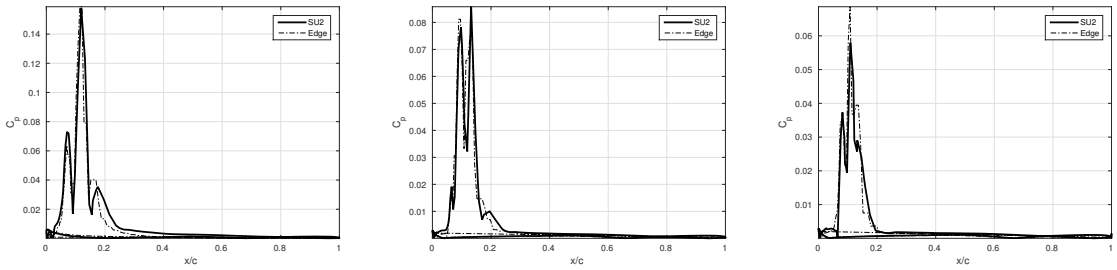


Figure 20: Second, third and fourth harmonics of response, magnitude, pressure coefficient

4.2 Fluid-Structure Interaction solver with deformable solids

4.2.1 Benchmark problem for native FSI solver in SU2

In order to test our native implementation of the FSI solver for deformable solids within SU2, we focus on a well-known benchmark test case first proposed by Wall and Ramm [71] (see, for example, [35, 44, 72–75]). They investigated the dynamics of a flexible cantilever attached to the downwind side of a square cylinder in a low-speed flow, as described in Fig. 21.

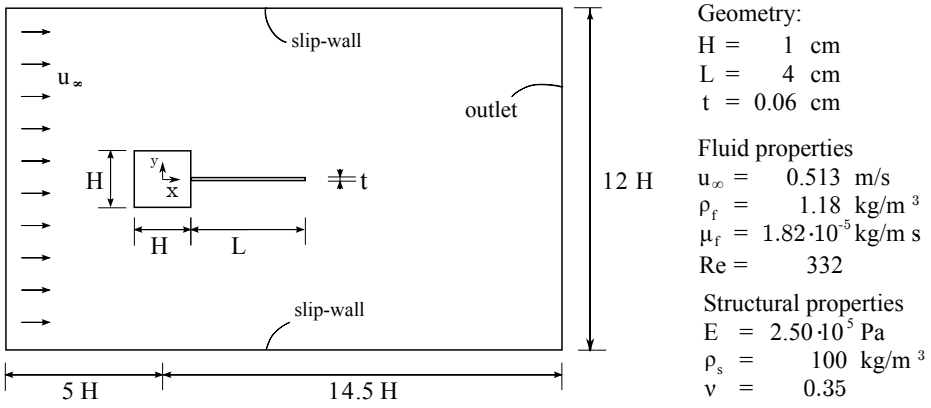


Figure 21: Benchmark for FSI solver with deformable solids.

The physics that lie behind this problem may be briefly described as follows. The square cylinder, rigid and static, generates vortical structures in the low-Reynolds number flow. These vortices generate an alternating pressure in the wake, that induces motions of the flexible cantilever attached to the downwind side of the square, as shown in Fig. 22.

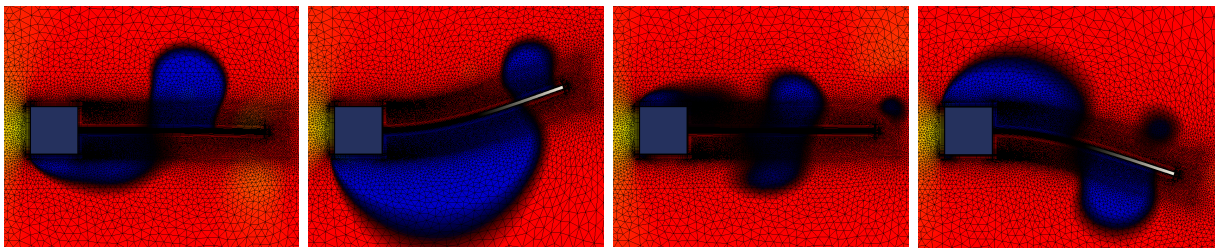


Figure 22: Pressure contours and structural displacements for the benchmark test cases for a full cycle of oscillation. From left to right, T=0, T=π/2, T= π, T=3π/2.

The two parameters that have been used in the literature to assess the validity of different FSI implementations are the frequency and amplitude of the vertical displacement at the tip of the cantilever. With regards to the frequency, published values range from 2.78 to 3.25 Hz, this is, in the surroundings of the first natural frequency of the cantilever, equal to 3.03 Hz. The maximum tip displacement has been found to be within the range 1-1.4 cm, thus a 25-35% of the cantilever length, which results in geometrically-nonlinear effects on the structure.

We have tested this problem using SU2 and several different time and space discretisations. In these tests, we have obtained values of frequency ranging from 3.05 to 3.15 Hz and maximum tip displacements in the range of 1.05 - 1.15 cm, showing a very good agreement with the literature. In a previous work, see [7], we have also shown the ability of the solver to capture some modulations in the amplitude due to the complex physics involved, and the different behaviour of the coupled problem when modifying some relevant parameters, such as the structural density and first natural frequency.

In order to reduce the computational time, we have also tested in this work the ability of the Aitken's dynamic parameter and the first order displacement predictor to reduce the number of BGS subiterations. In Fig. 23, we compare different configurations against a BGS strongly-coupled strategy in which we use a fixed relaxation parameter ω_{fixed} = 0.5. This approach converges slowly to the solution, in about 8-10 iterations per time step.

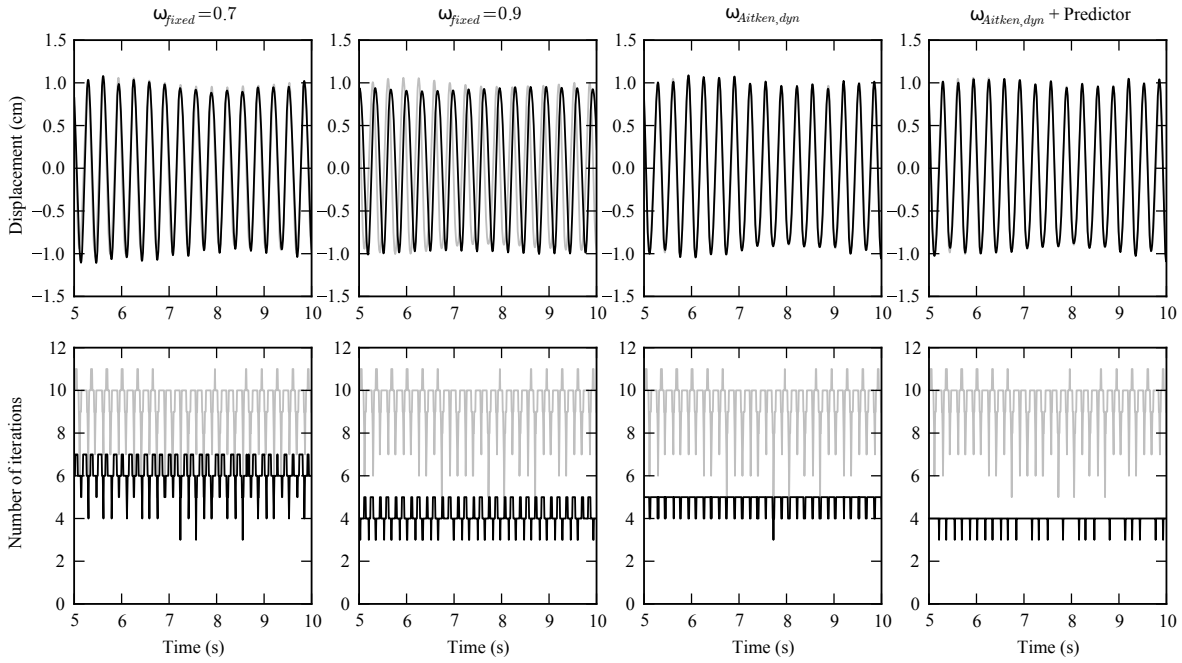


Figure 23: Time histories of the vertical tip displacements for different values of the relaxation parameter for the BGS FSI solver. The base solution (in grey) is obtained with ω_{fixed} = 0.5, and no predictor.

By increasing the fixed relaxation parameter to ω_{fixed} = 0.7, we reduce the computational time in a 33%. However, as it may be seen in Figure 23, a higher relaxation parameter introduces some deviations in the solution, that we believe are closely related to the value of ω being too large in the first subiteration, ω^0. This effect becomes more clear when we increase the fixed relaxation parameter to ω_{fixed} = 0.9, when both the frequency and the amplitude are affected

and the amplitude modulation is almost damped out.

On the other hand, the use of the Aitken's dynamic relaxation parameter ($\omega_{Aitken,dyn}$) clearly improves the convergence of the scheme. In particular, the number of BGS subiterations is reduced to 4-5 per time step, resulting in the computational time being a 38% shorter. By using a first order predictor on the first BGS subiteration, we can further reduce the number of iterations to 3-4 and the computational time by a 56% with respect to the baseline case, while obtaining effectively the same solution. It is important to note that, in these two cases, we have limited the value of the relaxation parameter in the first BGS iteration to $\omega^0 = 0.5$, then allowing it to adapt dynamically in the remaining subiterations.

4.2.2 Interpolation framework for non-matching discretizations

To test the implementation of the interpolation routines, the case of a solid wall immersed in a flow in a channel case is used. The geometry is shown in Fig. 24, and the flow conditions were set as Mach 0.15 viscous unsteady flow with adiabatic no-slip walls on the flexible vertical wall within the channel, and slip wall conditions on the upper and lower surfaces. The Reynolds number is 300 per meter. The ratio of densities between the structure and the fluid is $1 : 0.0106$.

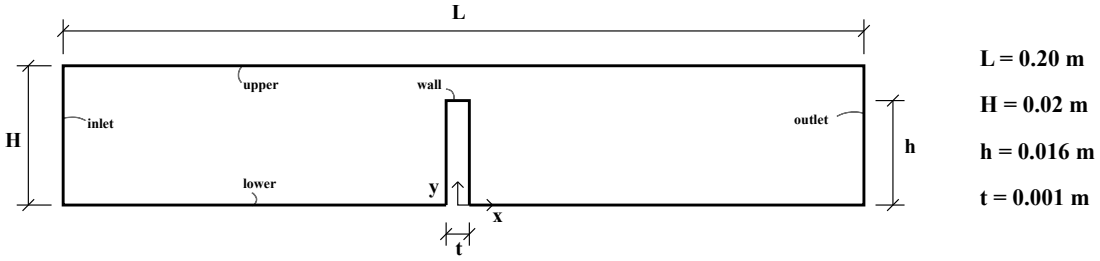


Figure 24: Test case used for interpolation routines.

Several meshes were generated with varying refinement. The results shown are from a 'fine' mesh with 100 elements on the upwind surface of the vertical wall, and a 'coarser' mesh with 90 elements on that same wall. Fig. 25 illustrates the results of simulating the coupled dynamics with either matching 'fine' discretizations, matching 'coarser' discretizations, or a non-matching discretization with the 'coarser' mesh used for the structural dynamics. This plot illustrates the effect of using these methods on the accuracy of the solution. Figure 26 illustrates the deformed geometry at time step 25. We can see from these plots that the nearest-neighbor interpolation results in discontinuous values. Using isoparametric weighting coefficients results in a smoother deformation, however when examining the accuracy of this deformation in Figure 25, we can see that a smoother interpolation has not resulted in more accurate results as compared to the nearest neighbor approach. Both of these methods use a transformation matrix $\mathbf{H}$ that has been calculated separately for the two transfer directions. The combination of this smoother interpolation and the use of a single $\mathbf{H}$ results in the most accurate interpolated results.

4.2.3 Python-wrapped FSI with interface to external solvers

In order to test/demonstrate the flexibility of the python interface of SU2, the FSI problem of a flexible cantilever attached to the downwind side of a square cylinder (described in Section 4.2.1) is solved by coupling the flow and structural solvers using the python interface. The

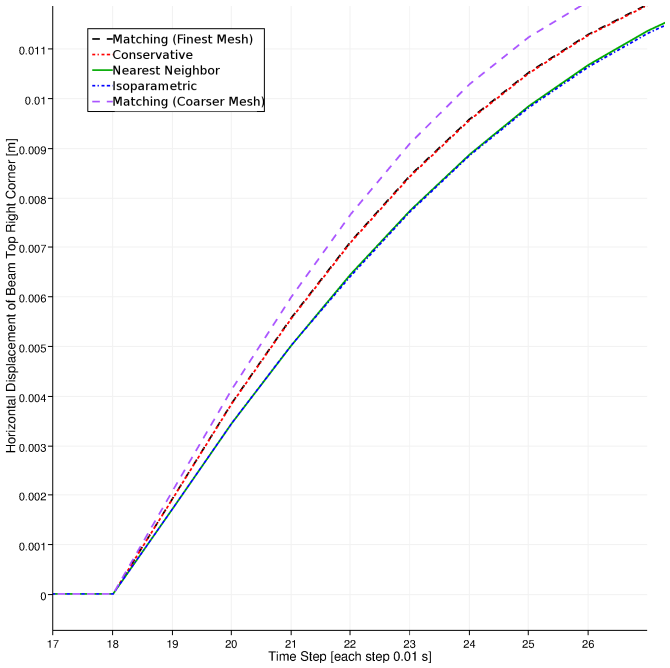
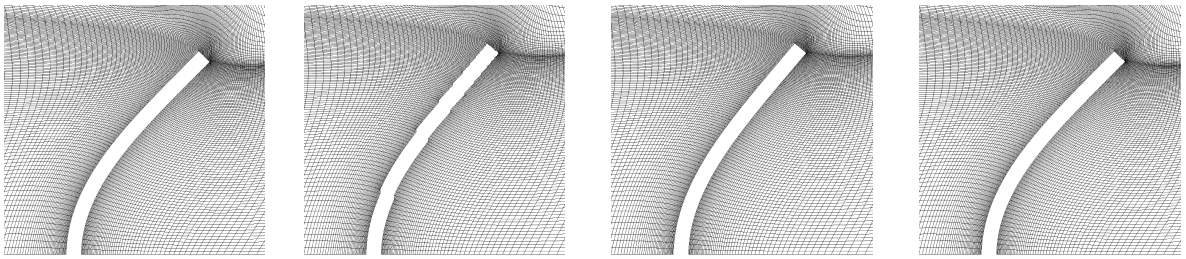


Figure 25: Time history of horizontal displacement of the upper right-hand corner of the deflecting beam. The geometry has been held static until time step 18.



(a) Matching discretization.

(b) Unmatched discretization with Nearest-Neighbor interpolation. Discontinuities can be seen in the deformed shape.

(c) Unmatched discretization with Isoparametric weights used in interpolation. The deformation is smooth, but does not agree with matched discretization results.

(d) Unmatched discretization with 'Conservative' method as in Section 2.3.3. The deformation agrees with the matched discretization, and is smooth.

Figure 26: Deformed geometry after 25 time steps

geometry of the problem is the same as Section 4.2.1 but the conditions are slightly different corresponding to those specified in Dettmer et al. [73]. These are stated again in Fig. 27. The results shown in Figs. 28, 29 were computed using a conventional staggered (CS) coupling approach. We see that the maximum tip displacement of the beam, not including the transient region, is 1.03 and the average frequency is 2.93 Hz showing a good agreement with [73] and the results in Section 4.2.1. Simulations with the Block Gauss Seidel (BGS) coupling approach give similar results but these have not been shown here.

Geometry:		Fluid properties:		Structural properties:	
H	= 1	U_∞	= 51.3	E	= $2.5 \bullet 10^6$
L	= 4	ρ_f	= $1.18 \bullet 10^{-3}$	ρ_s	= 0.1
t	= 0.06	μ_f	= $1.82 \bullet 10^{-4}$	v	= 0.35
		Re	= 333		

Figure 27: Geometry, flow and structural properties used for the 2d simulation using the python interface

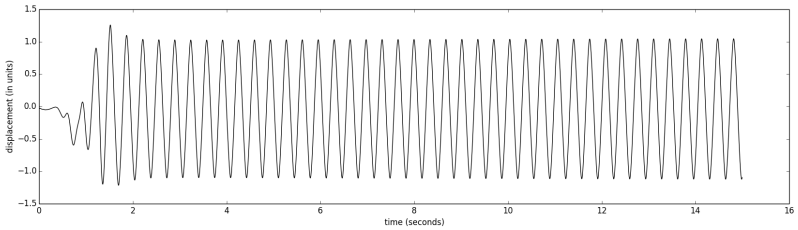


Figure 28: Time history of vertical tip displacements obtained using the python based coupling

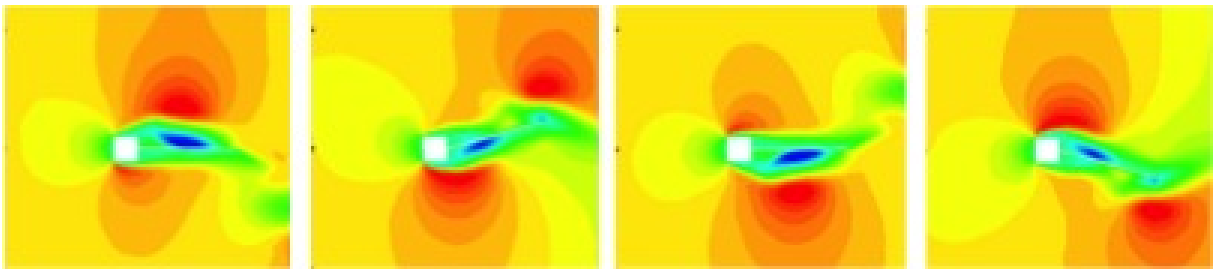


Figure 29: Convective flux contours and structural displacements for the benchmark test cases (using the python based framework) for a full cycle of oscillation. From left to right, T=0, T=π/2, T= π, T=3π/2.

5 CONCLUSION

We have introduced a new open-source Fluid-Structure Interaction solution framework tailored to high-fidelity analysis in computational aeroelasticity. It includes a natively embedded FSI solver and a code wrapper methodology that simplify the communication of the code with other external commercial and in-house solvers. We have carried out some relevant tests of the methodology and the different capabilities that have been implemented. We have compared the results obtained using this software with some relevant benchmark test cases, obtaining results that agree well with the open literature.

This infrastructure has been implemented by an international group of researchers with different backgrounds and expertise, and is publicly available in a github repository that can be accessed through the SU2 project website, http://su2.stanford.edu/. The main goal of this collaborative work is to progressively incorporate new capabilities into the platform while maintaining its modularity, thus encouraging for further developments of state-of-the-art techniques

within the different disciplines that conform the study of the interaction between fluids and structures.

ACKNOWLEDGEMENTS

H. Kline would like to acknowledge the support of a NASA Space Technology Research Fellowship.

REFERENCES

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

[2] A. Beckert. Coupling fluid (CFD) and structural (FE) models using finite interpolation elements. *Aerospace Science and Technology*, 4(1):13–22, 2000.

[3] C. Farhat, K.G. van der Zee, and P. Geuzaine. Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Computer Methods in Applied Mechanics and Engineering*, 195(17-18):1973–2001, 2006.

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

[5] F. Palacios, T.D. Economon, A.C. Aranake, S.R. Copeland, A.K. Lonkar, T.W. Lukaczyk, D.E. Manosalvas, K.R. Naik, A. Santiago Padrn, B. Tracey, A. Variyar, and J.J. Alonso. Stanford university unstructured (SU2): Open-source analysis and design technology for turbulent flows. In *AIAA 52nd Aerospace Sciences Meeting, SciTech, 13-17 January*, National Harbor, MD, 2014.

[6] T. Economon, F. Palacios, J. Alonso, G. Bansal, D. Mudigere, A. Deshpande, A. Heinecke, and A. Smelyanskiy. Towards High-Performance Optimizations of the Unstructured Open-Source SU2 suite. In *AIAA 53rd Aerospace Sciences Meeting, Scitech, 5-9 January*, Kissimmee , FL, 2015.

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

[8] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodriguez-Ferran. *Arbitrary Lagrangian-Eulerian Methods in Encyclopedia of Computational Mechanics*. John Wiley and Sons, 2004.

[9] C. Hirsch. *Numerical Computation of Internal and External Flows*. Wiley, New York, 1984.

[10] D.C. Wilcox. *Turbulence Modeling for CFD*. 2nd Ed., DCW Industries, Inc., 1998.

[11] F. M. White. *Viscous Fluid Flow*. McGraw Hill Inc., New York, 1974.

[12] F.R. Menter. Zonal two equation $k - \omega$, turbulence models for aerodynamic flows. *AIAA Paper 1993-2906*, 1993.

[13] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. Number AIAA Paper 1992-0439, 1992.

[14] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. Number AIAA Paper 1981-1259, 1981.

[15] P. L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.

[16] J. M. Weiss, J. P. Maruszewski, and A. S. Wayne. Implicit solution of the Navier-Stokes equation on unstructured meshes. *AIAA Paper 1997-2103*, 1997.

[17] A. Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. Number AIAA Paper 1991-1596, 1991.

[18] A. Jameson and S. Schenectady. An assessment of dual-time stepping, time spectral and artificial compressibility based numerical algorithms for unsteady flow with applications to flapping wings. Number AIAA Paper 2009-4273, 2009.

[19] P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17(10):1030–1037, October 1979.

[20] J. T. Batina. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, August 1990.

[21] M. Lesoinne and C. Farhat. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134(1–2):71 – 90, 1996.

[22] C. Farhat, P. Geuzaine, and C. Grandmont. The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems. *Journal of Computational Physics*, 174(2):669–694, December 2001.

[23] S. A. Morton, R. B. Melville, and M. R. Visbal. Accuracy and coupling issues of aeroelastic Navier-Stokes solutions on deforming meshes. *Journal of Aircraft*, 35(5):798–805, 1998.

[24] R. T. Biedron and J. L. Thomas. Recent enhancements to the FUN3D flow solver for moving-mesh applications. *AIAA Paper 2009-1360*, 2009.

[25] M. Geradin and A. Cardnna. *Flexible Multibody Dynamics, A Finite Element Approach*, chapter Kinematics of Finite Motion. John Wiley & Sons, LTD, 2001.

[26] M. Geradin and D.J. Rixen. *Mechanical Vibrations : Theory and Application to Structural Dynamics*, chapter Analytical Dynamics of Discrete Systems. Wiley, 2015.

[27] F. Amirouche. *Fundamentals of Multibody Dynamics, Theory and Applications*, chapter Hamilton-Lagrange and Gibbs-Appel Equations. Birkhauser, 2006.

[28] M. Hojjat, E. Stavropoulou, T. Gallinger, U. Israel, R. Wchner, and Bletzinger K.-U. *Fluid Structure Interaction II*, volume 73, chapter Fluid-Structure Interaction in the Context of Shape Optimization and Computational Wind Engineering, pages 351–381. Springer Berlin Heidelberg, 2010.

[29] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[30] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodríguez-Ferran. *Encyclopedia of Computational Mechanics*, chapter Arbitrary Lagrangian-Eulerian Methods. John Wiley & Sons, Ltd, 2004.

[31] K.-J. Bathe. *Finite Element Procedures in Engineering Analysis*. Civil engineering and engineering mechanics series. Prentice-Hall, 1982.

[32] N. M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85(3):67–94, 1959.

[33] J. Chung and G.M. Hulbert. Time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of Applied Mechanics, Transactions ASME*, 60(2):371–375, 1993.

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

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

[36] S. Piperno and C. Farhat. Partitioned procedures for the transient solution of coupled aeroelastic problems part II: energy transfer analysis and three-dimensional applications. *Computer Methods in Applied Mechanics and Engineering*, 190(2425):3147–3170, 2001. Advances in Computational Methods for Fluid-Structure Interaction.

[37] C. Farhat, M. Lesoinne, P. Stern, and S. Lantri. High performance solution of three-dimensional nonlinear aeroelastic problems via parallel partitioned algorithms: Methodology and preliminary results. *Advances in Engineering Software*, 28(1):43–61, 1997.

[38] C. Farhat and M. Lesoinne. Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Computer Methods in Applied Mechanics and Engineering*, 182(34):499–515, 2000.

[39] W.G. Dettmer and D. Perić. A new staggered scheme for fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 93(1):1–22, 2013.

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

[41] J. Degroote, K.-J. Bathe, and J. Vierendeels. Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. *Computers and Structures*, 87(11-12):793–801, 2009.

[42] P. Le Tallec and J. Mouro. Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3039–3067, 2001.

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

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

[45] B. M. Irons and R. C. Tuck. A version of the Aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1:275–277, 1969.

[46] M.J. Smith, D.H. Hodges, and C.E.S. Cesnik. Evaluation of computational algorithms suitable for fluid-structure interactions. *Journal of Aircraft*, 37(2):282–294, 2000.

[47] J.R. Cebral and R. Löhner. Conservative load projection and tracking for fluid-structure problems. *AIAA Journal*, 35(4):687–692, 1997.

[48] G.P. Guruswamy. A review of numerical fluids/structures interface methods for computations using high-fidelity equations. *Computers and Structures*, 80(1):31–41, 2002.

[49] A. de Boer, A.H. van Zuijlen, and H. Bijl. Review of coupling methods for non-matching meshes. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1515–1525, 2007.

[50] X. Jiao and M.T. Heath. Overlaying surface meshes, part I: Algorithms. *International Journal of Computational Geometry and Applications*, 14(6):379–402, 2004.

[51] R.K. Jaiman, X. Jiao, P.H. Geubelle, and E. Loth. Assessment of conservative load transfer for fluid-solid interface with non-matching meshes. *International Journal for Numerical Methods in Engineering*, 64(15):2014–2038, 2005.

[52] S A Brown. Displacement Extrapolations for CFD+CSM Aeroelastic Analysis. *AIAA Paper*, pages 291–300, 1997.

[53] A. Beckert and H. Wendland. Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerospace Science and Technology*, 5(2):125–134, 2001.

[54] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An Open-Source Suite for Multi-Physics Simulation and Design. *AIAA Journal*, (accepted), 2015.

[55] G. J. Kennedy and J.R.R.A. Martins. A parallel aerostructural optimization framework for aircraft design studies. *Structural and Multidisciplinary Optimization, Volume 50, Issue 6*, 2014.

[56] A. Variyar, T.D. Economon, and J. J. Alonso. Multifidelity conceptual design and optimization of strut-braced wing aircraft using physics based methods. *54th AIAA Aerospace Sciences Meeting*, 2016.

[57] R.L. Bisplinghoff, H. Ashley, and R. L. Halfman. *Aeroelasticity*. Dover Publications, 1996.

[58] Anderson J.D. *Fundamentals of Aerodynamics*, chapter Incompressible Flow over Airfoils. Mc Graw Hill, Inc., 2011.

[59] Y. C. Fung. *An Introduction to the theory of aeroelasticity*, chapter Fundamentals of flutter analysis. Dover Publications, 2002.

[60] K. Isogai. On the transonic-dip mechanism of flutter of a sweptback wing. *AIAA Journal*, 17(7):793–795, 1979.

[61] K. Isogai. Transonic-dip mechanism of flutter of a sweptback wing: part ii. *AIAA Journal*, 19(9):1240–1242, 1981.

[62] F. Liu, J. Cai, and Y. Zhu. Calculation of wing flutter by a coupled fluid-structure method. *Journal of Aircraft*, 38(2):334–342, 2001.

[63] J.J. Alonso and A. Jameson. Fully-implicit time marching aeroelastic solution. In *AIAA 32nd Aerospace Sciences Meeting and Exhibit, 10-13 January*, 1994.

[64] Z. Biao, Q. Zhide, and G. Chao. Transonic flutter analysis of an airfoil with approximate boundary method. In *26th international congress of the aeronautical sciences*, 2008.

[65] J.T. Thomas, K.C. Hall, and E.H. Dowell. Reduced-order aeroelastic modeling using proper-orthogonal decomposition. *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics*, 1999.

[66] J. Heeg, P. Chwalowski, D. M. Schuster, D. Raveh, A. Jirasek, and M. Dalenbring. Plans and example results for the 2nd AIAA Aeroelastic Prediction Workshop. *AIAA Paper*, 2015.

[67] B. E. Dansberry, M. H. Durham, R. M. Bennett, D. L. Turnock, E. A. Silva, and J. A. Rivera Jr. *Physical properties of the benchmark models program supercritical wing*, volume 4457. Citeseer, 1993.

[68] D. J. Piatak and C. S. Cleckner. Oscillating turntable for the measurement of unsteady aerodynamic phenomena. *Journal of aircraft*, 40(1):181–188, 2003.

[69] R. M. Bennett, C. V. Eckstrom, J. A Rivera Jr, B. E. Dansberry, M. G. Farmer, and M. H. Durham. The benchmark aeroelastic models program: Description and highlights of initial results. 1991.

[70] P. Eliasson and P. Weinerfelt. Recent Applications of the Flow Solver Edge. *Proceedings of the 7th Asian CFD Conference*, 2007.

[71] W.A. Wall and E. Ramm. Fluid-Structure interaction based upon a stabilized (ALE) finite element method. In E. Oñate S.R. Idelsohn and E.N. Dvorkin (Eds.), editors, *Computational Mechanics. New Trends and Applications*. CIMNE, Barcelona, Spain, 1998.

[72] H. G. Matthies and J. Steindorf. Partitioned strong coupling algorithms for fluidstructure interaction. *Computers & Structures*, 81(811):805–812, 2003.

[73] W. Dettmer and D. Perić. A computational framework for fluid-structure interaction: Finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5754–5779, 2006.

[74] C. Wood, A.J. Gil, O. Hassan, and J. Bonet. A partitioned coupling approach for dynamic fluid-structure interaction with applications to biological membranes. *International Journal for Numerical Methods in Fluids*, 57(5):555–581, 2008.

[75] B. Froehle and P.-O. Persson. A high-order discontinuous Galerkin method for fluid-structure interaction with efficient implicit-explicit time stepping. *Journal of Computational Physics*, 272:97–104, 2014.