International Forum on Aeroelasticity and Structural Dynamics
IFASD 2019
9-13 June 2019, Savannah, Georgia, USA

A FULL POTENTIAL STATIC AEROELASTIC SOLVER FOR PRELIMINARY AIRCRAFT DESIGN

Adrien Crovato[1], Romain Boman[1], Huseyin Guner[1], Vincent E. Terrapon[1], Grigorios Dimitriadis[1], Hugo S. Almeida[2], Alex P. Prado[3], Carlos Breviglieri[3], Pedro H. Cabral[3], Gustavo H. Silva[4]

[1]University of Liege
Liege, Belgium
{a.crovato, r.boman, h.guner, vincent.terrrapon, gdimitriadis}@uliege.be

[2]Embraer S.A. and Ph.D. student at ITA Instituto Tecnologico de Aeronautica
Sao Jose dos Campos, Brazil
hugo.almeida@embraer.com.br

[3]Embraer S.A.
Sao Jose dos Campos, Brazil
{alex.prado, carlos.breviglieri, pedro.cabral}@embraer.com.br

[4]DLR German Aerospace Center
Gottingen, Germany
gustavo.silva@dlr.de

Keywords: aeroelasticity, aerodynamics, transonic flow, full-potential, aircraft design

Abstract: There is a consensus in the aerospace research community that future aircraft will be more flexible and their wings will be more highly loaded. While this development is likely to increase aircraft efficiency, it poses several aeroelastic questions. Current aeroelastic tailoring practice for early preliminary aircraft design relies on linear aerodynamic modeling, unable to predict shocks. On the other hand, nonlinear solvers, although they provide a wide range of functionality and are reliable, often consist in monolithic code structures and cannot be efficiently coupled to external structural mechanics codes. They are therefore usually not readily usable for coupled fluid-structure interaction computations. The objective of the present work is to carry out aerodynamic and static aeroelastic computations in the context of preliminary aircraft design. To this end, an open-source, fast and reliable, unstructured finite element, Full Potential solver has been developed. Preliminary results are presented and show a significant improvement over the classical linear potential method and are in good agreement with higher fidelity nonlinear solvers.

1 INTRODUCTION

In the on-going effort to build more efficient aircraft, the minimization of the structural weight and the maximization of the aerodynamic efficiency usually lead to the design of very flexible and highly loaded composite wings. Aeroelastic analysis thus plays an increasingly important role in preliminary aircraft design. In the early stages of the design process, the computational cost of the methods is of uttermost importance and must be kept as low as possible. Engineers thus usually rely on linear aerodynamic solvers. However, these solvers are unable to predict

shockwaves, which play an important role in transonic aircraft design. In a study comparing the different levels of fidelity used in aerodynamic modeling [1], the authors found that the Full Potential equation model offers a good trade-off between accuracy and computational cost. An open-source, fast and reliable, unstructured finite element Full Potential solver has been subsequently developed, with the purpose of carrying out aerodynamic computations in the context of preliminary aircraft design. Moreover, the solver has been designed so that it can be easily coupled to external Computational Structural Mechanics codes in order to provide fast static aeroelastic solutions.
First, the different equations and solvers used in the present work are briefly reviewed. Then, the formulation and main features of the newly developed Full Potential finite element code is presented. Finally, aerodynamic and static aeroelastic computations are performed. Results are presented and future improvements are described.

2 METHODOLOGY

In the present work, three levels of fidelity, Euler, Full Potential and Linear Potential equations, are used to perform aerodynamic computations over 3D wings. Moreover, these equations are coupled to two sets of structural mechanics equations to perform aeroelastic computations. This section presents the fluid and structural dynamics equations used in the present work, as well as the different solvers used to solve these equations.

2.1 Fluid dynamics equations

The unsteady Euler equations are derived from the Navier-Stokes equations by assuming that the flow is inviscid. They can be written as,

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho\mathbf{u}\\ \rho E\end{bmatrix} + \nabla\cdot\begin{bmatrix}\rho\mathbf{u}\\ \rho\mathbf{u}\otimes\mathbf{u}\\ \rho E\mathbf{u}+p\mathbf{u}\end{bmatrix} = 0, \tag{1}$$

where ρ is the density, $\mathbf{u}$ is the velocity vector, p is the pressure and E is the total energy per unit mass. The system of equations 1 needs to be closed with state equations,

$$\begin{aligned} E &= c_v T + \frac{1}{2}\rho|\mathbf{u}|^2,\\ p &= \rho R T, \end{aligned} \tag{2}$$

where c_v is the specific heat capacity at constant volume, T is the temperature and R is the ideal gas constant. In the present work, Equation 1 are solved using the open-source finite-volume solver `SU2` [2,3]. In `SU2`, steady state is reached through time marching, *i.e.* the time dependent terms are discretized and the solution is iterated until it does not change in time anymore.

The steady Full Potential equation assumes that the fluid is inviscid and the flow is steady, irrotational and isentropic. The velocity thus derives from a potential ϕ : $\mathbf{u} = \nabla\phi$. The conservation of momentum is automatically satisfied and only the conservation of mass remains,

$$\nabla\cdot(\rho\nabla\phi) = 0, \tag{3}$$

where the fluid density ρ is given by the isentropic flow relationship,

$$\rho = \rho_\infty\left[1+\frac{\gamma-1}{2}M_\infty^2\left(1-|\nabla\phi|^2\right)\right]^{\frac{1}{\gamma-1}}, \tag{4}$$

where ρ_∞ is the freestream fluid density, γ is the heat capacity ratio and M_∞ is the freestream Mach number. Since the flow is irrotational, it cannot generate aerodynamic loads. In order to allow lift and drag prediction, the Kutta condition must be enforced. This supplementary condition is based on the physical observation that a fluid must leave a sharp trailing edge smoothly. It can be enforced mathematically by imposing that the magnitude of the velocities on the upper and lower sides of the trailing edge of a wing are equal. In the present work, Equation 3 is solved by `Tranair` [4], a finite element solver commonly developed by NASA and Boeing, and by `Flow` [5], a new in-house finite element code that will be presented in the next section.

The Full Potential equation can be linearized to yield the Linear Potential equation by assuming that the density remains constant. The linear equation can then be transformed into an integral equation by using Green's third identity. As for the Full Potential, the Kutta condition must also be enforced. In the present work, the integral Linear Potential equation is solved either in its classical form by the doublet source panel method implemented in `Panair` [6], or in its acoustic form by the doublet lattice method implemented in `NASTRAN` [7].

2.2 Structural dynamics equations

Neglecting internal damping, the equilibrium equations of a solid are obtained by balancing the inertial and elastic forces in the solid with the external forces applied onto it. The equations can be written as,

$$\rho_\mathrm{s} \frac{d^2\mathbf{x}}{dt^2} - \nabla \cdot \boldsymbol{\sigma} = \mathbf{f}, \tag{5}$$

where ρ_s is the solid density, $\boldsymbol{\sigma}$ is the stress tensor, $\mathbf{f}$ are the external forces and $\mathbf{x}$ are the displacements. In the present work, Equation 5 is solved either by `Metafor` [8], an in-house nonlinear finite element code, or by the linear finite element method implemented in `NASTRAN`. In order to reach a steady state in the static aeroelastic computations, the time dependent terms are integrated using a quasi-static time integration procedure.

The displacements of the solid can be expressed in the modal space as,

$$\mathbf{x} = \boldsymbol{\Phi}\mathbf{q}, \tag{6}$$

where $\mathbf{q}$ are the modal coordinates of the solid and $\boldsymbol{\Phi}$ is the mode matrix, containing the mode shapes of the solid. By neglecting the time dependent terms, equation 5 can be further discretized into,

$$\mathbf{K}_\mathrm{q}\mathbf{q} = -\mathbf{f}_\mathrm{q}, \tag{7}$$

where $\mathbf{K}_\mathrm{q}$ is the modal stiffness matrix and $\mathbf{f}_\mathrm{q}$ is the vector of modal forces, obtained by multiplying the physical terms by the mode matrix. In the present work, the modal equation is solved by an in-house simple modal solver [9].

3 FLOW SOLVER IMPLEMENTATION

This sections presents `Flow`, a new in-house finite element Full Potential code. `Flow` is designed to obtain fast transonic results for aerodynamic and static aeroelastic computations in the context of preliminary aircraft design.

3.1 Basic formulation

The weak formulation of the Full Potential equation is obtained by multiplying Equation 3 by a test function ψ and integrating by parts. This yields,

$$F = \int_\Omega \rho \nabla\phi \cdot \nabla\psi \, dV - \int_\Gamma \overline{\rho\nabla\phi} \cdot \mathbf{n}\psi \, dS = 0, \quad \forall\psi, \tag{8}$$

where Γ is the boundary of the volume Ω, $\mathbf{n}$ is the normal to Γ pointing inwards and $\overline{\rho\nabla\phi}$ is the known mass-flux through the boundary.
The domain Ω is discretized into finite elements and Equation 8 is expressed on each element. The potential is discretized on each element by interpolating the values at the nodes using linear shape functions,

$$\phi = \sum_i N_i \phi_i, \tag{9}$$

where N_i is the shape function and ϕ_i is the potential associated to node i of the element. The contribution of each element is then assembled at the nodes (formed by the vertices of the elements), and the resulting nonlinear system of equations is solved with the Newton-Raphson method as,

$$F = 0 \Rightarrow \frac{\partial F}{\partial \phi}\Delta\phi + F + O\left((\Delta\phi)^2\right) = 0, \tag{10}$$

where F is the function defined by Equation 8. Each step of the Newton algorithm produces a linear system of equations solved for $\Delta\phi$ using the MUMPS [11] direct solver. The amplitude of $\Delta\phi$ is then adapted thanks to a quadratic line search [10], in order to improve the robustness and convergence characteristics of the method.

3.2 Implementation of the Kutta condition

The Kutta condition has been implemented in order to predict flows over lifting configurations. The implementation is mainly based on the methodology derived by Nishida [12] and Galbraith et al. [13].
As illustrated on Figure 1, a flat wake extending from the trailing edge of any lifting configuration and aligned with its bisector is created. The unknown potential value attached to each node on this wake is then duplicated, except at the trailing edge and at the free edge of the wake, located downstream of the wingtip. In order to enforce continuity in the derivative of the potential, *i.e.* the velocity, two supplementary boundary conditions must be enforced. This procedure is similar to the implementation of periodic boundary conditions.

The first condition is the equality of the mass-flux on the upper and lower sides of the wake,

$$\int_{\Gamma_\mathrm{w}} \rho_\mathrm{u} \nabla\phi_\mathrm{u} \cdot \mathbf{n}_\mathrm{u} \, dS = -\int_{\Gamma_\mathrm{w}} \rho_\mathrm{l} \nabla\phi_\mathrm{l} \cdot \mathbf{n}_\mathrm{l} \, dS, \tag{11}$$

where subscripts u and l refer to the upper and lower sides of the wake, respectively. Substituting Equation 8 in Equation 11 allows to replace the surface terms by volume terms. As a consequence, mass-flux continuity can be imposed by adding a (lower) volume term to the (upper) elements sharing a node on the wake, that is,

$$\int_{\Omega_\mathrm{l}} \rho \nabla\phi \cdot \nabla\psi \, dV + \int_{\Omega_\mathrm{u}} \rho \nabla\phi \cdot \nabla\psi \, dV = 0. \tag{12}$$

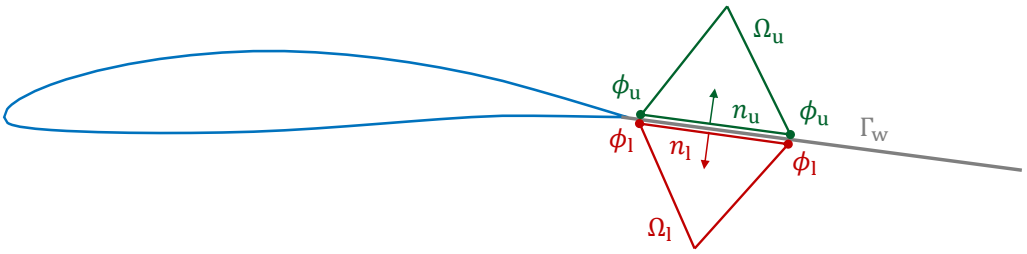


Figure 1: Illustration of the trailing wake configuration used in `Flow`.

Numerically, this can be implemented by adding upper wake rows to lower wake rows in the Jacobian matrix. The upper wake rows are then reset to implement the second condition: the zero-pressure jump, which simplifies to continuity in the velocity magnitude in the case of potential flows. The condition can be written as,

$$\int_\Gamma (\psi + \Psi) [\![|\nabla \phi|^2]\!] \, dS = 0, \tag{13}$$

where the double square bracket indicates a jump between the quantities on the upper and lower sides of the wake. In the case of 3D flows, the implementation has to be stabilized for example with a Petrov-Galerkin formulation. The test functions are then complemented by,

$$\Psi = \frac{1}{2} \frac{h}{u_\infty} \left(\mathbf{u}_\infty \cdot \nabla \phi \right), \tag{14}$$

where $\mathbf{u}_\infty$ is the freestream velocity vector, u_∞ is its norm and h is a characteristic length, here taken to be the square root of the element area. For 2D flows, Ψ can be set to zero.

3.3 Treatment of supersonic flow

The Full Potential equation is elliptic for subsonic flows and becomes hyperbolic when the flow becomes supersonic. This change in the mathematical nature of the equation must be reflected in the numerical scheme in order to prevent unphysical expansion shocks. In the present work, transonic flow computations are stabilized with a density upwinding procedure originally proposed by Hafez et al. [14] and Eberle [15]. The switching function μ is defined as,

$$\mu = \mu_C \max \left(0, 1 - \frac{M_C^2}{M^2} \right), \tag{15}$$

where μ_C and M_C are parameters that control the dissipation. The physical density is then replaced by,

$$\tilde{\rho} = \rho - \mu \overleftarrow{\delta_s} \rho \Delta s, \tag{16}$$

where s is the local direction of the flow and Δs is the local cell size. The streamline derivative of the density multiplied by a measure of the cell length $\overleftarrow{\delta_s} \rho \Delta s$ is approximated by $\rho - \rho_U$, with ρ_U being the density in the associated upwind element. For each element, an upwind element is selected by identifying the adjacent element closest to the reverse streamline direction. In practice, the simulation is started with a large initial value of μ_C and a small value of M_C to produce

large amount of dissipation that roughly locate the shock location. During the course of the simulation, these parameters are adjusted to reduce dissipation so as to predict a sharp discontinuity while ensuring stability. This procedure is similar to that implemented in `Tranair`.

The vortex generated at the wingtip trailing edge of 3D lifting configurations induces an infinite velocity. For high-speed flows, the singularity induces large Mach numbers which may cause the method to diverge. To alleviate this issue, the density associated with large velocity magnitudes is limited according to the following Padé approximation [13],

$$\rho = \frac{\rho_{\text{crit}}}{1 + \frac{M_\infty^2 \frac{u_{\text{crit}}}{u_\infty}}{1+\frac{\gamma-1}{2}M_\infty^2\left(1-\frac{u_{\text{crit}}^2}{u_\infty^2}\right)}\left(\frac{u}{u_{\text{crit}}}-1\right)}, \quad u > u_{\text{crit}}, \tag{17}$$

where u_{crit} is the velocity corresponding to a user-specified critical Mach number, usually $\sim \sqrt{5}$.

3.4 Integration

The Finite Element procedure described in the previous section is implemented in a C++ code wrapped in Python through `SWIG` [16]. The wrapping allows to take advantage of both the high computing efficiency of the C++ language and the flexibility of the Python language. More specifically, the geometry is parametrized in a Python script, meshed with `gmsh` [17] and loaded in the solver's data structure. An external process can then be used to drive the computation for coupled physics simulations or optimization. To this end, the solver has been interfaced with `CUPyDO` [18, 19] and coupled to several structural mechanics solvers, such as `Metafor` and a modal solver.

4 AERODYNAMIC COMPUTATIONS

This section presents aerodynamic results on two benchmark cases: the Onera M6 wing and the Embraer Benchmark Wing. The results obtained by `Flow` are compared to `Tranair` in order to validate the new solver. Both results are then compared, on one hand to results obtained by solving the Euler equations using `SU2` and, on the other hand, to the `Panair` solution of the linear potential equation, that is routinely used in industrial preliminary aircraft design.

4.1 Onera M6

The Onera M6 wing is a low aspect ratio, swept and tapered wing. The wing was tested at transonic conditions ($\alpha = 3.06°$, Mach 0.839) and is now widely used as a standard validation case. A surface grid of 1000 rectangular surface panels is used for discretizing the `Panair` model. The `Tranair` model is enclosed in a box whose boundaries are placed 2 chord lengths away from the wing in the chordwise and normal directions, and a half-span length from the wingtip in the spanwise direction. The final grid, built automatically by a solution-based adaptive procedure, consists of $500,000$ hexahedra with a minimum cell size of $1/200$ of the chord at the shock and leading edge. The `Flow` model is enclosed in a box whose boundary faces are placed 3.5 chord lengths away from the wing in the chordwise and normal directions, and 1 span length away from the wingtip in the spanwise direction. The unstructured grid is built using `gmsh` and counts $590,000$ cells, with a characteristic size of $1/200$ and $1/100$ of the local chord at the leading edge and at the trailing edge, respectively. The `SU2` model is also built with `gmsh` based on an unstructured O-grid topology extending 50 root chords away from the wing. The mesh has a characteristic cell size of $1/200$ and $1/100$ of the local chord at the leading edge and at the trailing edge, respectively, for a total count of $510,000$ cells. A convergence study

was performed on each grid. In each case, the selected mesh is the one for which the results did not change significantly when the number of cells was increased.

Figure 2 shows the pressure distribution along the mean aerodynamic chord of the wing at an angle of attack of $3.06°$ and a Mach number of 0.839. The results obtained with `Flow` are in good agreement with the results obtained with `Tranair`. However, `Flow` predicts a shock located upstream and a slightly different pressure recovery near the pressure peak at the leading edge. `Flow` and `Tranair` Full Potential solutions compare well to `SU2` Euler solution and are able to correctly predict the transonic flow physics, contrary to `Panair`'s Linear Potential solution.

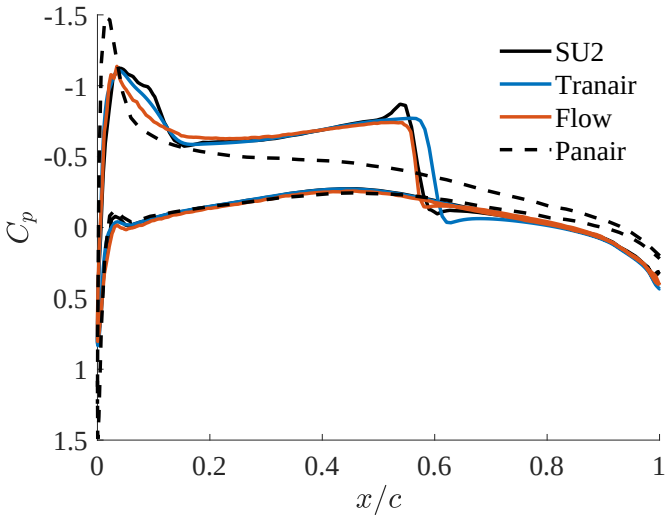


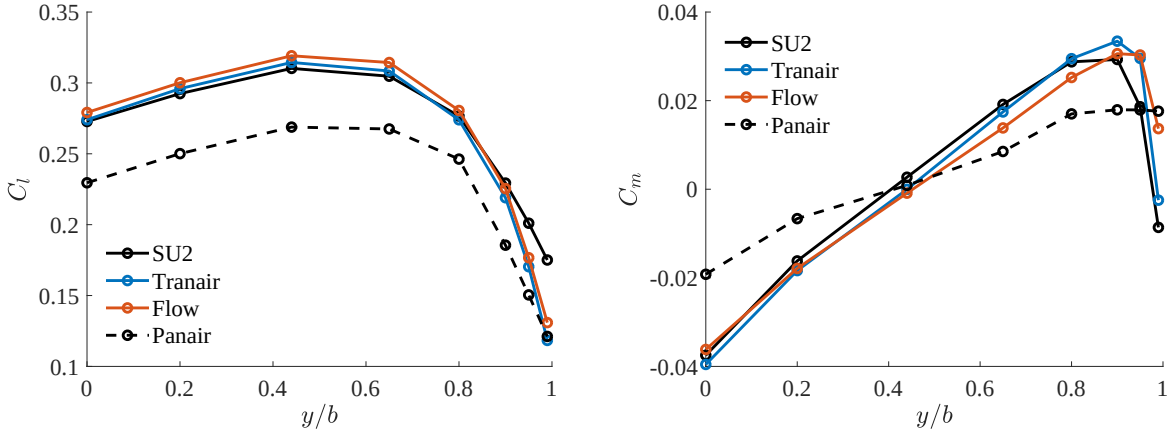
Figure 2: Pressure distribution along the mean aerodynamic chord of the Onera M6 at $\alpha = 3°$, Mach 0.84, obtained from `Flow` and compared to `SU2`, `Tranair` and `Panair`.

Figure 3(a) depicts the lift distribution along the span and shows an excellent agreement between `SU2`, `Tranair` and `Flow`. The lift predicted by `Panair`'s linear solution is lower compared to the lift obtained from the nonlinear solvers, but the distribution is similar. Figure 3(b) shows the quarter-chord moment coefficient along the span. The results obtained from `Flow` are in good agreement with those obtained from `Tranair` and `Euler`, and show a significant improvement compared to `Panair`. The difference in moment magnitude between the different solutions is due to the difference in shock location and pressure peak shape.

The mesh size and the computational time required to run the simulations are given in Table 1. The serial runs were performed on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz, 8 threads) while the parallel runs were performed on a cluster equipped with Intel Xeon X5650 processors (2.7 GHz, 12 threads). `Flow` requires twice the amount of time taken by `Tranair`. However, `Flow` has not been optimized yet. Several improvements such as, inner solver optimization, solution based grid adaptation and compiler optimization will be investigated in the future. Both Full Potential solvers are faster than `SU2` by more than one order of magnitude and slower than `Panair` by two orders of magnitude.

4.2 Embraer Benchmark Wing

The Embraer Benchmark Wing (EBW2) is a generic benchmark wing model representative of an airliner wing. It has a double planform, large aspect ratio, and is a swept, twisted and tapered wing. The wing has been simulated in established maneuver condition at Mach number 0.78,



(a) Sectional lift coefficient

(b) Sectional moment coefficient

Figure 3: Sectional aerodynamic coefficients distribution along the span of the Onera M6 at $\alpha = 3°$, Mach 0.84, obtained from `Flow` and compared to `SU2`, `Tranair` and `Panair`.

Table 1: Mesh size and computational time required by `SU2`, `Tranair`, `Flow` and `Panair` for the Onera M6 benchmark case.

Solver	n. cells	n. threads	time
`SU2`	$510,000$	12	1400 s
`Tranair`	$500,000$	1	800 s
`Flow`	$590,000$	1	1600 s
`Panair`	$1,000$	1	10 s

altitude of 21000 ft. For each calculation, the angle of attack is set such that the resulting lift coefficient is $C_L = 0.53$. The models for the different solvers are built in the same way as in the Onera M6 case, but the grid sizes are different. The final grid used by `Tranair` consists of $500,000$ hexahedra. The mesh used by `Flow` counts 1 million tetrahedra, with a characteristic size of $1/100$ of the local chord at the leading and trailing edges. Finally, the `SU2`'s grid counts 1.1 million tetrahedra. As in the case of the Onera M6, these grid sizes were obtained by performing a convergence study.

Figure 4 shows the pressure distribution along the mean aerodynamic chord of the wing. There is good agreement between the nonlinear solvers, although the shocks predicted by `Flow` and `SU2` are weaker compared to those predicted by `Tranair`, as opposed to observations made for the Onera M6 case. Despite the difference in shock prediction, the solution is improved compared to `Panair` linear solution.

Figure 5 shows the lift and the quarter-chord moment coefficient distribution along the span, respectively. `Flow` closely follows `Tranair` results and both solvers predict load distributions similar to `SU2`. Contrary to the Onera M6 case, the wing lift is fixed and the shocks are weak. The load distribution predicted by `Panair` is then comparable to those obtained using the nonlinear solvers, except near the kink of the wing. The angles of attack predicted by the different solvers are similar: `SU2` and `Tranair`: $-1.4°$, `Flow`: $-1.3°$, and `Panair`: $-1.1°$.

The mesh size and the computational time required to run the simulations are given in Table 2. The serial runs were performed on a laptop fitted with an Intel i7-7700HQ processor (2.8 GHz, 8 threads) while the parallel runs were performed on a desktop station equipped with an Intel

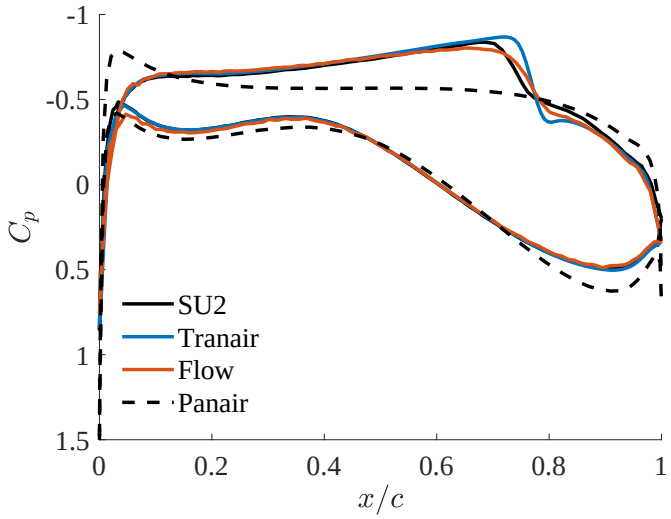
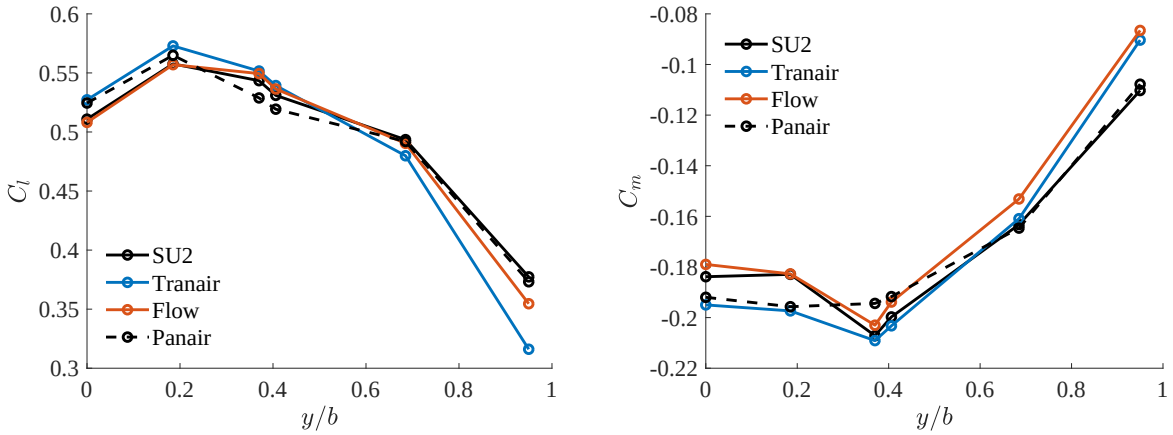


Figure 4: Pressure distribution along the mean aerodynamic chord of the EBW2 at $C_L = 0.53$, Mach 0.78, obtained from `Flow` and compared to `SU2`, `Tranair` and `Panair`.



(a) Sectional lift coefficient

(b) Sectional moment coefficient

Figure 5: Sectional aerodynamic coefficients distribution along the span of the EBW2 at $C_L = 0.53$, Mach 0.78, obtained from `Flow` and compared to `SU2`, `Tranair` and `Panair`.

i7-4930K processor (3.4 GHz, 12 threads). Compared to the Onera M6 case, `Flow` requires a mesh that is twice as large, and is more than 2 orders of magnitude slower than `Panair` and about 4 times slower than `Tranair`. However, it remains more than an order of magnitude faster than `SU2`. Note that `Flow` also implements a shared memory parallelization. For this test case, the simulation took 1000 s to complete on 4 threads.

Table 2: Mesh size and computational time required by `SU2`, `Tranair`, `Flow` and `Panair` for the EBW2 benchmark case.

Solver	n. cells	n. threads	time
`SU2`	$1,100,000$	6	9000 s
`Tranair`	$500,000$	1	800 s
`Flow`	$1,000,000$	1	3400 s
`Panair`	1400	1	10 s

4.3 Discussion

The results obtained on the Onera M6 and on the EBW2 test cases show that `Flow` is able to provide results close to those predicted by `Tranair`. `Flow` however lacks the efficiency of `Tranair` and requires larger meshes for actual aircraft wings, leading to an increased computational cost. Different techniques are currently being investigated to optimize the code, such as reducing the mesh size and the linear solver time. This section also illustrates that linear solvers, such as `Panair`, are not able to capture the physics of transonic flows and lead to inaccurate predictions of the load distributions, except when the shocks are weak. Although strong shocks are not desirable in aircraft design, designing a wing that minimizes shockwaves requires to be able to predict them, which can only be achieved with nonlinear solvers.

5 AEROELASTIC COMPUTATIONS

This section presents static aeroelastic simulations on two cases: The Agard 445 wing and the Embraer Benchmark Wing 2. The coupling of the fluid and structural solvers is achieved using `CUPyDO`. The results obtained from `Flow` are compared to those obtained by `SU2` and either to those reported in the literature or to those obtained by `NASTRAN`.

5.1 Agard 445

The Agard 445 wing is a low aspect ratio, swept and tapered wing. The wing is widely used as a standard validation case for transonic flutter calculations. In the present work, the wing is simulated at an angle of attack $\alpha = 1^\circ$ and a Mach number of 0.80. Under such conditions, the Freon-12 gas has a density of 0.094 kg/m^3 and a velocity of 247 m/s [20, 21].
The numerical model used by `Flow` is built in the same way as before and the unstructured grid counts $250,000$ tetrahedra, with a characteristic size of $1/200$ and $1/100$ of the local chord at the leading and trailing edges, respectively. The `SU2` model is built with `gmsh` in a multiblock structured O-grid topology extending 25 root chords away from the wing. The mesh has 50, 20 and 30 hexahedra in the chordwise, normal and spanwise directions respectively, for a total count of $250,000$ hexahedra. The structural model is built in `Metafor` and is based on the weakened model 3 of the wing [20]. The mesh is built with `gmsh` and consists of 31, 2 and 17 hexahedral cells in the chordwise, normal and spanwise directions respectively.
The coupling is performed with the Block Gauss Seidel algorithm available in `CUPyDO`. Fluid and solid variables are interpolated between fluid and structural nodes with Radial Basis Functions, also implemented in `CUPyDO`. The tolerance for the FSI simulation is set to 10^{-3} mm, which is 10^{-4} times the expected maximum displacement.

Table 3 gives the lift coefficient of the deformed shape of the wing as well as the vertical deflections at the wingtip's leading and trailing edges of the Agard wing. The results are also compared to those obtained by Goura [21]. There is good overall agreement between the different solvers and results previously reported in the literature. The lift coefficient predicted by `Flow` differs by less than one lift count compared to the one predicted by `SU2`. `Flow` tends to predict slightly smaller displacements, but a similar wingtip's rotation, than Euler solvers.

The mesh size and the computational time required to run the simulations are given in Table 4. Both computations were performed in serial on a desktop fitted with an Intel i7-4930K processor (3.4 GHz, 12 threads) and converged in 7 FSI iterations. In this case, `Flow` is about one order of magnitude faster than `SU2`.

Table 3: Lift coefficient and wingtip vertical displacements at the leading and trailing edges of the Agard wing at $\alpha = 1°$, Mach 0.8, obtained with `SU2` and `Flow` and compared to Goura's results [21].

Solver	C_L	z_{LE} (mm)	z_{TE} (mm)
`Euler` [21]	-	11.2	12.7
`SU2`	0.0537	11.6	13.1
`Flow`	0.0544	10.6	12.0

Table 4: Mesh size and computational time required by `SU2`, and `Flow` for the Agard benchmark case.

Solver	n. cells	time
`SU2`	$250,000$	8750 s
`Flow`	$250,000$	970 s

5.2 Embraer Benchmark Wing

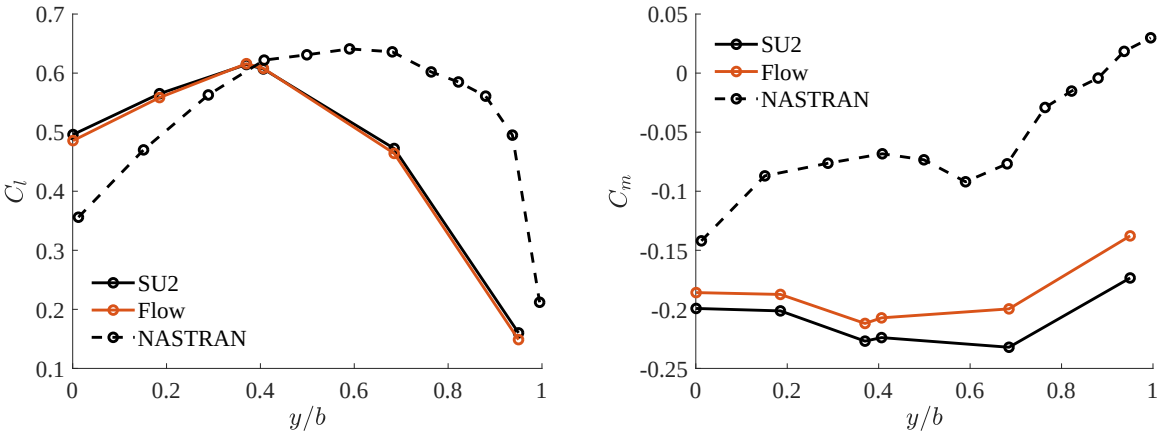
The Embraer Benchmark Wing maneuver case described in section 4.2 is analyzed in the context of an FSI simulation. The objective is to predict the deformed shape of the wing subjected to the maneuver described in section 4.2 and to recover the new angle of attack needed to sustain the maneuver, as well as the new load distributions along the span. Note that the wing is clamped at its root to represent its attachment to the fuselage. Such a boundary condition is not realistic as the fuselage is not rigid. However, this setup is only used to compare the different solvers. The mesh used by the DLM in `NASTRAN` consists of 1474 quadrilateral surface panels placed on the mean chord surface of the wing. The associated structural model is also discretized in `NASTRAN` and consists of $50,000$ shell elements.
The meshes used by `Flow` and `Euler` are those described in section 4.2. The associated structural model is based on a modal representation of the structure obtained by a modal analysis performed in `NASTRAN`. The mesh used by the modal solver consists of 2134 points distributed on the surface of the wing. Note that these points are only used to create the mode matrix to transfer quantities between the physical and modal spaces. The coupling between `SU2` or `Flow` and the modal solver is performed with the Block Gauss Seidel algorithm available in `CUPyDO`. Fluid and solid variables are interpolated between fluid and structural nodes with Radial Basis Functions, also implemented in `CUPyDO`. The tolerance for the FSI simulation is set to 10^{-4} times the expected maximum displacement.

Table 5 gives the new angle of attack α of the deformed shape of the wing as well as the vertical deflections at the wingtip leading and trailing edges of the EBW2. Note that the displacements are normalized with respect to the half-span of the wing. There is an excellent agreement between `Flow` and `SU2`. Both solvers predict roughly the same angle of attack and the difference in the wingtip's displacement are less than 3 percentage points. On the other hand, `NASTRAN` predicts a higher angle of attack, overestimates the wingtip's displacements and underestimates the wingtip's rotation. These differences are due to the difference in physics modeling and to the fact that `NASTRAN` neglects the wing's camber and thickness.

Table 5: New angle of attack and wingtip vertical displacements at the leading and trailing edges of the EBW2 at $C_L = 0.53$, Mach 0.78, obtained with `SU2`, `Flow` and `NASTRAN`.

Solver	α (°)	z_{LE} (%)	z_{TE} (%)
`SU2`	-0.4	7.23	8.13
`Flow`	-0.3	7.06	7.83
`NASTRAN`	5.6	9.10	9.30

Figure 6 shows the lift and the quarter-chord moment coefficient distribution along the span, respectively. `Flow` closely follows `SU2`'s predictions. The offset between the two curves in the sectional moment coefficient distribution is mostly due to the difference in shock strength. Linear results obtained using `NASTRAN` largely differ from results predicted by nonlinear solvers. The different shape and magnitude of the sectional lift and moment coefficients predicted by `NASTRAN` can be explained by two main factors. First, the camber is ignored and causes the angle of attack to be higher and the center of pressure to move upstream. Second, `NASTRAN` is not able to predict shockwaves, which modify the shape of the pressure distribution. These differences in loads distributions compared to nonlinear solvers result in a different deformed state.



(a) Sectional lift coefficient

(b) Sectional moment coefficient

Figure 6: Sectional aerodynamic coefficients distribution along the span of the deformed EBW2 at $C_L = 0.53$, Mach 0.78, obtained from `Flow` and compared to `SU2`, and `NASTRAN`.

The mesh size and the computational time required to run the simulations are given in Table 6. Computations were performed in serial on a desktop fitted with an Intel i7-4930K processor (3.4 GHz, 12 threads). Using the automated angle of attack adjustment strategy implemented in `SU2` allowed the FSI process to converge twice as fast as when compared to `Flow`. Despite the better convergence characteristics displayed when using `SU2`, the `Flow` computation was still more than 5 times faster. On the other hand, `NASTRAN` computations are carried out on a much smaller grid and are linear, making them more than two orders of magnitude faster.

Table 6: Mesh size and computational time required by `SU2`, and `Flow`, and `NASTRAN` for the EBW2 benchmark case.

Solver	n. cells	time
`SU2`	$1,100,000$	19 h
`Flow`	$1,000,000$	2.5 h
`NASTRAN`	1474	25 s

5.3 Discussion

The results obtained on the Agard 445 wing with `Flow` closely match `SU2`'s predictions. Moreover in the case of the Embraer Benchmark Wing, `Flow`'s results are comparable to `SU2`'s results and show a significant improvement compared to `NASTRAN`'s linear results. In order to improve the linear results, adding the camber effect in `NASTRAN` will be investigated. The higher-fidelity results come at the price of an increased runtime. Aside from the various tech-

niques proposed in section 4.3 to reduce the computational time required by `Flow`, several FSI strategies are also investigated. Such strategies include the use of a linear potential model frequently corrected by the nonlinear `Flow` solver to reach higher-fidelity results while minimizing the computational requirements.

6 CONCLUSION

As new aircraft structural weight is minimized to reduce their fuel consumption, they become more and more subjected to aeroelastic effects. As a result, aeroelasticity is taken into account sooner in the preliminary design stage, where obtaining fast solutions is crucial since the number of configurations and load cases is large. In this context, a finite element Full Potential code was developed to obtain fast and reliable transonic flow solutions for aerodynamic and aeroelastic computations. The solver was validated against several state of the art solvers. Although the present solver still lacks robustness and efficiency, preliminary results show an overall good agreement with similar and higher fidelity solvers, and a significant improvement over lower fidelity solvers.

7 ACKNOWLEDGMENT

This research is funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) through a FRIA grant fellowship. The authors would like to gratefully acknowledge the aerospace company Embraer S.A., who provided the wing model and the results obtained with `NASTRAN`, and the Nonlinear Mechanics research unit from the University of Liege, for providing access to `Metafor` as well as to the computational resources required for carrying out computations presented in sections 4 and 5.

8 REFERENCES

[1] Crovato, A., Dimitriadis, G., and Terrapon, V. (2018). Higher fidelity transonic aerodynamic modeling for preliminary aircraft design. In *31st Congress of the International Council of the Aeronautical Sciences*. International Council of the Aeronautical Sciences.

[2] Palacios, F., Colonno, M., Aranake, A., et al. (2013). Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design. *AIAA Paper*, 287, 2013.

[3] Economon, T., Palacios, F., Copeland, S., et al. (2016). Stanford University Unstructured (SU2): An open-source suite for multi-physics simulation and design. *AIAA Journal*.

[4] Johnson, F., Samant, S., Bieterman, M., et al. (1992). Tranair: A full-potential, solution-adaptative, rectangular grid-code for predicting subsonic, transonic, and supersonic flows about arbitrary configurations. Tech. rep., NASA.

[5] (2019). Waves. https://github.com/ulgltas/waves.

[6] Carmichael, R. and Erickson, L. (1981). Panair: A higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations. *AIAA journal*.

[7] Albano, E. and Rodden, W. (1969). A doublet-lattice method for calulation lift distributions on oscillatiing surfaces in subsonic flows. *AIAA journal*.

[8] (2019). METAFOR, A nonlinear finite element code, University of Liege. http://metafor.ltas.ulg.ac.be/.

[9] (2019). Modal solver for FSI computations. https://github.com/ulgltas/ModalSolver.

[10] Fleury, C. (1994-1995). Optimisation des structures. Theory course, University of Liege.

[11] (2019). MUMPS: MUltifrontal Massively Parallel sparse direct solver. http://mumps.enseeiht.fr/.

[12] Nishida, B. (1996). *Fully Simultaneous Coupling of the Full Potential Equation and the Integral Boundary Layer Equations in Three Dimensions*. Ph.D. thesis, Massachussets Institute of Technology.

[13] Galbraith, M., Allmaras, S., and Haimes, R. (2017). Full potential revisited: A medium fidelty aerodynamic analysis tool. In *55th AIAA Aerospace Sciences Meeting, SciTech Forum*. AIAA.

[14] Hafez, M., Murman, E., and South, J. J. (1979). Artificial compressibility methods for numerical solution of transonic full potential equation. *AIAA journal*.

[15] Eberle, A. (1978). *A finite volume method for calculating transonic potential flow around wings from the pressure minimum integral*. NASA.

[16] Beazley, D. M. (1996). Swig: An easy to use tool for integrating scripting languages with c and c++. In *Proceedings of the 4th conference on USENIX Tcl/Tk Workshop*, vol. 4. USENIX Association, p. 15.

[17] Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79, 1309–1331.

[18] Thomas, D., Cerquaglia, M., Boman, R., et al. (2019). Cupydo: An integrated python environement for coupled fluid-structure problems. *Advances in Engineering Software*.

[19] Cerquaglia, M., Thomas, D., Boman, R., et al. (In press, 2019). A fully partitioned lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects. *Computer Methods in Applied Mechanics and Engineering*.

[20] Thomas, D., Variyar, A., Boman, R., et al. (2017). Staggered strong coupling between existing fluid and solid solvers through a python interface for fluid-structure interaction problems. In *VII International Conference on Computational Methods for Coupled Problems in Science and Engineering*. pp. 645–660.

[21] Goura, G. (2001). *Time marching analysis of flutter using computational fluid dynamics*. Ph.D. thesis, University of Glasgow.

COPYRIGHT STATEMENT