# A modular implementation of the time spectral method for aeroelastic analysis and optimization on structured meshes

Christophe Blondeau, Cedric Liauzun

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

# A MODULAR IMPLEMENTATION OF THE TIME SPECTRAL METHOD FOR AEROELASTIC ANALYSIS AND OPTIMIZATION ON STRUCTURED MESHES

**Christophe Blondeau**[1] **and Cédric Liauzun**[1]

[1] ONERA, Université Paris Saclay
F-92322 Châtillon - France
Christophe.Blondeau@onera.fr
Cedric.Liauzun@onera.fr

**Keywords:** Time Spectral Method, Adjoint, Optimization

**Abstract:** This paper presents a modular implementation of a time spectral (TS) solver as well as an adjoint TS sensitivity analysis module. This modular framework is implemented in Python and linked to the kernel of a customized version of the elsA CFD software which has been recently equipped with the capability of exposing the flux Jacobian matrices and explicit flux residuals. This organization is very flexible as it allows a rapid prototyping of promising numerical strategies in a Python environment while still achieving a good level of performance. A main advantage is to relieve the current limitation of the existing TSM kernel solver which duplicates in core memory the fluid blocks as many times as the number of selected time instants in the period. Instead, several instances of the CFD steady-state solver run concurrently (one instance of elsA for each time instant in the period) and are synchronized through the external Python layer. The new TS solver is compared to the kernel one on the NACA64A10 oscillating airfoil in transonic regime. Following the same lines, a prototype adjoint TS sensitivity analysis module is developed. A ten design parameter shape optimization of the NACA64A10 profile is performed, aiming to minimize the pressure drag coefficient under a lift coefficient constraint. As a demonstration, the optimal shape produced by the optimization of the pressure drag coefficient of the fixed airfoil at steady mean flow conditions is compared to the optimal shape which minimizes the average pressure drag coefficient of the oscillating airfoil over a time period.

## 1 INTRODUCTION

Steady aerodynamics and aeroelasticy have reached a high level of maturity thanks to acknowledged numerical techniques like implicit pseudo-time time-marching with local time stepping and multigrid acceleration associated to the Newton's method, see [1] for a review. At each fictitious time step a linear system is approximately solved using an advanced Krylov solver like Flexible Generalized Minimum Residual (FGMRES) [2] or more conventional relaxation techniques like Lower-Upper Symmetric Successive Over-Relaxation (LU-SSOR) [3]. Depending on the availability of the exact or approximate flux Jacobian operator in standard matrix form or through a matrix-vector product, different classes of preconditioners can be applied. Similar techniques are used to solve the discrete steady linearized or adjoint equations for optimization purpose. On the other

hand, unsteady flows still present a severe challenge to computational Fluid Dynamics (CFD). One of the most used techniques is the Dual Time Stepping (DTS) Backward Difference Formula (BDF) proposed by Jameson [4]. In the DTS method, one marches from one time level to the next time accurately, using pseudo time to drive the residual of the time-accurate equations to zero. This way, all the solution techniques mentioned above for the steady problem can be re-used. Also, for optimization of unsteady systems, the computation of the direct or adjoint solution is still considered as being unaffordable for realistic configurations. Indeed, a time-dependent adjoint requires the full forward solution of the unsteady problem, storing the flow states for each time step along the way, followed by a reverse sweep of the solution process to find the adjoint solution. While this process is more efficient than computing a full unsteady solution for every design variable in the problem, it is expensive. Nevertheless, the simulation of time periodic solutions for the internal flows of turbomachinery, the external flow fields of helicopter blades or propellers, and for certain aeroelastic stability analyses is a particular class of unsteady CFD that is of prime interest to designers. In this case, only the periodic steady state is of concern to the user and the majority of the computational time is spent in resolving the transient response. In the present study we are interested in forced responses to harmonic excitations with prescribed frequency. Spectral-methods exploit the time periodicity nature of the flow by expressing the conservation variables as Fourier series in time with spatially varying coefficients. Spectral computations can be performed in either the time domain, the frequency domain, or a combination of the two. Several formulations have been proposed in the literature.

In the Linear Frequency Domain (LFD) method [5, 6, 7] it is assumed that perturbations to the flow are small and harmonic. An efficient alternative method to the full unsteady simulation can be derived by linearization of the flow equations around a steady state. With this approach the unsteady simulation reduces to a single nonlinear steady computation followed by a single linear simulation in the frequency-domain. This approach is widely used in the construction of state-space models, in frequency and time domain, dedicated to aeroelastic stability. Despite its limitation to small perturbations, it is very attractive when many forced harmonic responses for a wide range of operating conditions are required.

Unlike the LFD method, the following approaches keep the full nonlinear nature of the unsteady flow solution. The NonLinear Frequency Domain (NLFD) method developed by McMullen et al. [8, 9] adopts a solution process which primarily takes place in the frequency domain. The states of the system are stored as frequency-domain Fourier coefficients, and the solution steps are generated from the frequency-domain residual and spectral operator. To simplify the implementation, the residual is evaluated in the time domain, where the states are transformed from the frequency domain to the time domain. Then, the residual is transformed from the time domain to the frequency domain using Fast Fourier Transform (FFT) techniques.

Early work on time nonlinear spectral solution techniques was conducted by Hall et al. [10], who derived a spectral formulation for the two-dimensional Navier–Stokes equations for turbomachinery application. In the Harmonic Balance Technique (HBT), the spectral time derivative is much more accurate than the finite difference operators, which are used in the dual time-step approach. Therefore, many fewer physical time instances are required using the present method. This derivation was conducted in the frequency domain, but to facilitate the computation they transformed the flow equations back to the time domain. This allowed a typical time-domain residual formulation to be used

for the computation of the solution in each of the spectral instances. The residual was augmented by a time spectral derivative term that coupled the various solution instances. The residuals were computed in the time domain, but both the spectral operator and the boundary conditions were applied in the frequency domain, yielding a mixed time-domain/frequency-domain approach.

The time-spectral method introduced by Gopinath and Jameson [11, 12] is similar to the harmonic balance method of Hall et al. [10]. However, the time-spectral method is derived completely in the time domain. This yields a purely real spectral operator, and allows for the use of the time-domain residual operator in its original form, including the boundary conditions. In the present work we follow the implementation proposed by Sicot [13].

We are interested in the development of a modular Python framework enabling rapid prototyping of new solution algorithms for the Time Spectral Method for the computation of forced harmonic responses of aerodynamic and aeroelastic problems. We are also interested in the computation of derivatives of harmonic responses (typically an average of a drag, lift or moment coefficient over a time period) with respect to shape design parameters. Although completely based on open source scientific Python packages, this framework will be able to offer a good level of scalability thanks to an MPI architecture while offering a high level of modularity enabling rapid prototyping in a research context. These new developments will take advantage of the recent capability of the elsA[1] CFD software [14] to expose the fluid Jacobian matrix out of the core memory. The idea is to replace the standard implicit phase LU-SGS approximate solution of the time-marching to a steady state algorithm with advanced iterative solver like FGMRES combined with efficient preconditioners. Thanks to these new solvers, we will demonstrate the accelerated convergence for a nonlinear steady state analysis and for the TSM analysis compared to the existing internal modules of elsA.

Our test case is the well known pitching NACA64A010 airfoil. An Euler fluid model is adopted and the mean flow conditions are transonic with a Mach number $M = 0.796$, an incidence of $\alpha = -0.21°$. The corner stone of a TSM solver for a periodic steady-state computation is a nonlinear steady-state solver. This constitutes the first contribution of this work. We will set up an external solver linked with the kernel of the elsA CFD software. At each iteration, the exact $2^{\text{nd}}$ order Jacobian matrix and the explicit residual for each fluid block will be extracted. The solution increment will then be computed by the Python framework and sent back into elsA for the parallel treatment of the boundary conditions. Based on this elementary problem, the TSM solver will be set up. Several instances of the steady-state solver will run concurently (one instance of elsA for each time instant in the period). Once again, the time stepping will be performed externally at each iteration and the TSM source term will be computed in the Python framework in order to get the proper right-hand side for the coupled TSM system. The following step will demonstrate the computation of sensitivities of time-periodic responses with respect to shape design parameters by re-using the software bricks previously developed. Finally, an optimization will be conducted on the pitching NACA64A010 airfoil with 10 shape design parameters, with the aim of minimizing the pressure drag coefficient under a constraint on the lift coefficient.

[1] elsA is the joint property of Airbus, Safran and Onera

## 2 TIME SPECTRAL COMPUTATIONAL FLUID DYNAMICS

The goal of the time-spectral approach, as for other spectral approaches, is to find a way to solve directly for the periodic steady-state solution of a given problem. The derivation of the time-spectral equations from the general unsteady form of the equations is described below.

### 2.1 Governing equations for a viscous fluid around a moving body

We first describe the general ALE (Arbitrary Lagrangian-Eulerian) formulation of the RANS (Reynolds Averaged Navier-Stokes) equations in integral form, adapted to account for an arbitrary grid motion, covering both rigid and deformable body movement. We consider the general problem of a laminar or turbulent compressible flow in an entrained and possibly deformable control volume $\mathcal{V}(t)$ of boundary $\partial\mathcal{V}(t)$. The oriented elementary surface normal vector $\mathbf{n}d\Sigma$ also depends on time. In integral conservation law form, the system of three-dimensional RANS equations in terms of absolute velocity $\mathbf{U}$ and with respect to an absolute galilean frame of reference is expressed in the following form:

$$\frac{d}{dt}\int_{\mathcal{V}(t)} \mathbf{W} d\mathcal{V} + \oint_{\partial\mathcal{V}(t)} \mathbf{F}_c(\mathbf{W},\mathbf{s})\cdot\mathbf{n}d\Sigma + \oint_{\partial\mathcal{V}(t)} \mathbf{F}_d(\mathbf{W},\nabla\mathbf{W})\cdot\mathbf{n}d\Sigma = \mathbf{0} \tag{1}$$

where the vector of conservative variables is $\mathbf{W} = (\rho, \rho\mathbf{U}, \rho E^*)^T$. The components of $\mathbf{W}$ are density (or Favre) averaged quantities and the total energy reads $E^* = E + k$ with $k$ being the turbulent kinetic energy. The column hypervector of convective fluxes in ALE form can be written as:

$$\mathbf{F}_c(\mathbf{W},\mathbf{s}) = \begin{bmatrix} \rho(\mathbf{U}-\mathbf{s}) \\ \rho\mathbf{U}\otimes(\mathbf{U}-\mathbf{s}) + p^*\mathbf{I} \\ \rho E^*(\mathbf{U}-\mathbf{s}) + p^*\mathbf{U} \end{bmatrix} \tag{2}$$

and the column hypervector of diffusive fluxes take the form:

$$\mathbf{F}_d(\mathbf{W},\nabla\mathbf{W}) = \begin{bmatrix} 0 \\ -\boldsymbol{\tau}^* \\ -(\boldsymbol{\tau}^*\cdot\mathbf{U} - \mathbf{q}^*) \end{bmatrix} \tag{3}$$

The velocity of the control volume boundary face breaks up into an entrainement speed $\mathbf{s}_E$ and a deformation velocity $\mathbf{s}_D$ such that $\mathbf{s} = \mathbf{s}_E + \mathbf{s}_D$. In equations (2) and (3) we have introduced the following *starred* quantities which is a convenient notation for keeping the same formalism as the instantaneous Navier-Stokes equations:

$$\begin{cases} p^* = p + \dfrac{2}{3}k \\ \boldsymbol{\tau}^* = (1 + \dfrac{\mu_t}{\mu})\boldsymbol{\tau} \\ \mathbf{q}^* = \mathbf{q} + \mathbf{q}_t = (1 + \dfrac{\mu_t}{\mu}\dfrac{P_r}{P_{rt}})\mathbf{q} \end{cases} \tag{4}$$

where the heat flux vector is given by the Fourier law

$$\mathbf{q} = -K_T\nabla T \tag{5}$$

and the stress tensor by the relation

$$\boldsymbol{\tau} = -2/3\mu(\nabla \cdot \mathbf{U})\mathbf{I} + \mu(\nabla\mathbf{U} + \nabla^T\mathbf{U}) \tag{6}$$

The ALE formalism introduces an additional referential configuration which corresponds to the deformed fluid grid at the current time [15]. For the specific case of a moving rigid body, typically a rotating blade or a wing oscillating in pitch, it is preferable to derive the momentum equations in an entrained reference frame [16]. The absolute or relative velocity, expressed in the moving reference frame, can be chosen as unknown quantities. In this work we choose the formulation with the absolute velocity components. This is mainly justified by the better numerical accuracy of the absolute velocity formulation for open flows modeled with the finite volume approach.

Let $\mathcal{R}_A(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ and $\mathcal{R}_R(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ denote an absolute galilean reference frame and a relative moving reference frame respectively. The transformation giving $\mathbf{b}_i$ from $\mathbf{e}_j$ is a rotation associated to the orthogonal matrix $\mathbf{R} = \mathbf{b}_k \otimes \mathbf{e}_k$. From now on, we consider the specific case of a steady rotation $\boldsymbol{\Omega}(\mathcal{R}_R/\mathcal{R}_A)$ imposed on $\mathcal{R}_R$. This choice is directly linked to the TSM derivation which only requires steady flow computations at discrete time instances. The angular velocity is then seen as as constant by the steady solver. The moving grid formulation also implies that the control volume $\mathcal{V}$ now keeps a constant geometry during the motion and also that the boundary normal vector $\mathbf{n}_\mathcal{R} = \mathbf{R}^T\mathbf{n}$ is independent of time in $\mathcal{R}_R$. The components of the absolute velocity expressed in the rotating frame $\mathcal{R}_R$ read $\mathbf{U}_\mathcal{R} = \mathbf{R}^T\mathbf{U}$. The corresponding entrainment velocity is $\mathbf{s}_E = \boldsymbol{\Omega} \times \mathbf{r}$, $\mathbf{r}$ being the local position vector. The governing equations (1) are now projected in $\mathcal{R}_R$ to give

$$\frac{d}{dt}\int_\mathcal{V} \mathbf{W}_\mathcal{R} d\mathcal{V} + \oint_{\partial\mathcal{V}} \mathbf{F}_c(\mathbf{W}, \mathbf{s})_\mathcal{R} \cdot \mathbf{n}_\mathcal{R} d\Sigma + \oint_{\partial\mathcal{V}} \mathbf{F}_d(\mathbf{W}, \nabla\mathbf{W})_\mathcal{R} \cdot \mathbf{n}_\mathcal{R} d\Sigma = \int_\mathcal{V} \mathbf{T}(\mathbf{W}_\mathcal{R}) d\mathcal{V} \tag{7}$$

The subscript indicates that the quantities are expressed relative to $\mathcal{R}_R$. The vector of conservative variables is $\mathbf{W}_\mathcal{R} = (\rho, \rho\mathbf{U}_\mathcal{R}, \rho E^*)^T$, with the total energy being $E^* = E + k = e + \mathbf{U}_\mathcal{R}^2/2 + \mathrm{k}$. The convective fluxes expression is directly obtained by replacing $\mathbf{U}$ with $\mathbf{U}_\mathcal{R}$ and $\mathbf{s}$ with $\mathbf{s}_E = \boldsymbol{\Omega} \times \mathbf{r}$ in (2). Introducing the expression $\nabla_\mathcal{R}$ of the gradient operator with respect to coordinates in $\mathcal{R}_R$ in (5) and (6) and replacing $\mathbf{U}$ with $\mathbf{U}_\mathcal{R}$ in (3) we obtain the expression for the diffusive fluxes. A new volumic source term corresponding to the Coriolis forces acting on the fluid appears on the right-hand side of (7):

$$\int_\mathcal{V} \mathbf{T}(\mathbf{W}_\mathcal{R}) d\mathcal{V} = \begin{bmatrix} 0 \\ -\boldsymbol{\Omega} \times \int_\mathcal{V} \rho\mathbf{U}_\mathcal{R} d\mathcal{V} \\ 0 \end{bmatrix} \tag{8}$$

It is worth mentioning that if no additional time-dependent external forces appear on the right-side of (7) and recalling that the control volume and the entrainment speed are fixed, then the problem expressed in the relative frame of reference is basically a steady one which means that the leading time-derivative term vanishes.

### 2.2 Base steady solver

In the present study we are interested in the finite volume method applied to multi-domain block-structured grids. The computational domain is discretized with elementary

hexahedral cells. The unknowns are the average values at the cell centers. The current cell is materialized by the control volume $\mathcal{V}(t)$. Equation (1) is now cast in the compact form

$$\frac{d}{dt}\int_{\mathcal{V}(t)} \mathbf{W} d\mathcal{V} + \oint_{\partial\mathcal{V}(t)} \mathbf{F}(\mathbf{W})\cdot\mathbf{n} d\Sigma = \int_{\mathcal{V}} \mathbf{T}(\mathbf{W}) d\mathcal{V} \tag{9}$$

We then define the average value $\overline{\mathbf{W}}_{\mathcal{V}}$ of the vector of conservative variables in the control volume as

$$\overline{\mathbf{W}}_{\mathcal{V}} = \frac{1}{|\mathcal{V}|}\int_{\mathcal{V}} \mathbf{W}\, d\mathcal{V}, \quad |\mathcal{V}| = \int_{\mathcal{V}} d\mathcal{V} \tag{10}$$

The cell boundary is defined as $\partial\mathcal{V} = \cup_{i=1}^{6}\Sigma_i$, where $\Sigma_i$ represents the i$^{\text{th}}$ face of the hexahedral cell. Introducing further the average value of the flux vector through $\Sigma_i$

$$\overline{\mathbf{F}}_{\Sigma_i} = \int_{\Sigma_i} \mathbf{F}(\mathbf{W})\cdot\mathbf{n} d\Sigma \tag{11}$$

and the average value of the source term vector in $\mathcal{V}$

$$\overline{\mathbf{T}}_{\mathcal{V}} = \frac{1}{|\mathcal{V}|}\int_{\mathcal{V}} \mathbf{T}(\mathbf{W}) d\mathcal{V} \tag{12}$$

system (9) can be written in the form of a system of ordinary differential equations:

$$\frac{d}{dt}(|\mathcal{V}|\overline{\mathbf{W}}_{\mathcal{V}}) = -\left[\sum_{i=1}^{6}\overline{\mathbf{F}}_{\Sigma_i} - |\mathcal{V}|\overline{\mathbf{T}}_{\mathcal{V}}\right] \tag{13}$$

Finally one obtains the space semi-discrete form of the governing equations by replacing the exact average flux $\overline{\mathbf{F}}_{\Sigma_i}$ with its numerical approximation $\mathbf{F}(\mathbf{W}_{\mathcal{V}}, \mathbf{W}_{\mathcal{V}_i})$

$$\frac{d}{dt}(|\mathcal{V}|\mathbf{W}_{\mathcal{V}}) = -\left[\sum_{i=1}^{6}\mathbf{F}(\mathbf{W}_{\mathcal{V}}, \mathbf{W}_{\mathcal{V}_i})\cdot\mathbf{N}_{\Sigma_i} - |\mathcal{V}|\mathbf{T}_{\mathcal{V}}\right] = -\mathbf{R}_{\mathcal{V}} \tag{14}$$

where $\mathbf{W}_{\mathcal{V}}$ is a numerical approximation of the average value $\overline{\mathbf{W}}_{\mathcal{V}}$, $\mathbf{N}_{\Sigma_i} = \int_{\Sigma_i}\mathbf{n} d\Sigma$ is the external normal to $\Sigma_i$ and $\mathbf{T}_{\mathcal{V}}$ is an approximation of the source term $\overline{\mathbf{T}}_{\mathcal{V}}$.

For the discrete steady problem the physical time derivative $t$ in (14) vanishes and a pseudo-time $\tau$ is introduced in order to apply a time-marching strategy to the steady state solution. The cell volume is now constant and a first order implicit time discretization is considered, which leads to:

$$|\mathcal{V}|\frac{\Delta\mathbf{W}}{\Delta\tau} = \frac{|\mathcal{V}|}{\Delta\tau}(\mathbf{W}^{n+1} - \mathbf{W}^{n}) = -\mathbf{R}_{\mathcal{V}}^{n+1} \tag{15}$$

A first order time linearization of the discrete residual is introduced and the standard un-factored form of the Backward-Euler algorithm finally appears in the following form

$$\left[\frac{|\mathcal{V}|}{\Delta\tau}\mathbf{I} + \mathbf{J}\right]\Delta\mathbf{W} = -\mathbf{R}_{\mathcal{V}}^{n} \tag{16}$$

where the Jacobian matrix is formally written as $\mathbf{J} = \partial\mathbf{R}/\partial\mathbf{W}$.

The standard solution process considers a simplification of the implicit operator using a first order upwind approximation to the fluxes differential [3]. The following diagonally dominant system is obtained

$$\left[\frac{|\mathcal{V}|}{\Delta\tau}\mathbf{I} + \widetilde{\mathbf{J}}_{\mathrm{O1}}\right]\Delta\mathbf{W} = \left[\mathcal{L} + \mathcal{D} + \mathcal{U}\right]\Delta\mathbf{W} = -\mathbf{R}_{\mathcal{V}}^{n} \tag{17}$$

where $\widetilde{\mathbf{J}}_{\mathrm{O1}}$ stands for *approximate first-order Jacobian* and the matrix $\mathcal{D}$ can be chosen scalar or block diagonal. The solution of the linear system (17) is then computed using a two-stage scalar or block LU-SSOR relaxation algorithm.

Recently, the ONERA elsA CFD software has been equipped with the capability of exposing the Jacobian-vector product or the full Jacobian matrix outside of the software kernel. This adds the great flexibility of linking the CFD kernel with an external numerical solver. This also offers potentiality for alternative solution strategies and rapid prototyping in a Python-based environment. This will be discussed later in section 3 However, it is now possible to substitute for the second-order exact Jacobian matrix in (18):

$$\left[\frac{|\mathcal{V}|}{\Delta\tau}\mathbf{I} + \mathbf{J}_{\mathrm{O2}}\right]\Delta\mathbf{W} = -\mathbf{R}_{\mathcal{V}}^{n} \tag{18}$$

The second-order Jacobian matrix is numerically much more difficult to invert and standard relaxation techniques usually fail. A more efficient pre-conditioned Krylov-based iterative solver like the limited memory FGMRES-DR (Flexible GMRES with Deflated Restarting) is typically required [2, 17, 18]. Also an efficient pre-conditioning technique has to be applied. Most acknowledged ones are point or block versions of the ILU(k) (Incomple Lower Upper factorization with level of fill control) or ILUTP (Incomplete Lower Upper factorization with Threshold and Pivoting) decompositions [19].

### 2.3 Time Spectral Method

We start from the semi-discrete form (14) of the governing equations written for a constant cell volume:

$$|\mathcal{V}|\frac{\partial\mathbf{W}_{\mathcal{V}}}{\partial t} + \mathbf{R}_{\mathcal{V}} = \mathbf{0} \tag{19}$$

We then adopt the following one-sided definition for the Discrete Fourier transform of an $M$-periodic discrete function $z$ sampled at an odd number $M = 2N + 1$ points:

$$DFT(\mathbf{z})_k = \hat{z}_k = \frac{1}{M}\sum_{n=0}^{M-1} z_n e^{-i2\pi kn/M}, \quad k \in [-N, N] \tag{20}$$

The Inverse Discrete Fourier Transform is taken in its centered definition as follows

$$IDFT(\hat{\mathbf{z}})_n = z_n = \sum_{k=-\frac{M-1}{2}}^{\frac{M-1}{2}} \hat{z}_k e^{i2\pi kn/M} \tag{21}$$

Under the assumption of a time periodicity $T = M\Delta t$ with circular frequency $\omega = 2\pi/T$ for $\mathbf{W}$ and $\mathbf{R}$, we may compute the truncated centered discrete Fourier series of (19) to get:

$$\sum_{k=-N}^{N} (ik\omega|\mathcal{V}|\hat{\mathbf{W}}_k + \hat{\mathbf{R}}_k)e^{ik\omega t} = \mathbf{0} \tag{22}$$

where $\hat{\mathbf{W}}_k$ and $\hat{\mathbf{R}}_k$ represent the Fourier coefficients of $\mathbf{W}_\mathcal{V}$ and $\mathbf{R}_\mathcal{V}$. The orthogonal property of the exponential Fourier basis functions ensures that individual mode contributions are zero:

$$ik\omega|\mathcal{V}|\hat{\mathbf{W}}_k + \hat{\mathbf{R}}_k = \mathbf{0}, \quad k \in [-N, N] \tag{23}$$

which leads to a single steady equation for each mode. Computing the Inverse Discrete Fourier Transform of the $2N+1$ equations (23) we get back into the time domain with the following relation:

$$\mathbf{R}_{\mathcal{V}n} + |\mathcal{V}|\mathbf{T}_{\mathcal{V}n} = \mathbf{0}, \quad 0 \leq n < 2N+1 \tag{24}$$

where

$$\mathbf{T}_{\mathcal{V}n} = \frac{\partial \mathbf{W}_{\mathcal{V}n}}{\partial t} = \sum_{j=0}^{M-1} d_{nj}\mathbf{W}_{\mathcal{V}j} = D_t(\mathbf{W}_\mathcal{V}) \tag{25}$$

and the coefficient $d_{nj}$ given by:

$$d_{nj} = \begin{cases} \dfrac{2\pi}{T}\dfrac{1}{2}(-1)^{n-j}\csc\left[\dfrac{\pi}{M}(n-j)\right], & n \neq j \\ 0 \qquad\qquad\qquad\qquad\quad , & n = j \end{cases} \tag{26}$$

The source term $\mathbf{T}_{\mathcal{V}n}$ (25) couples the $2N+1$ steady flow solutions defined at $2N+1$ equally spaced instants in the period. Looking at (26) the skew-symmetric nature of the matrix $\mathbf{D}_t = (d_{nj})_{0\leq n,j<M}$ is obvious. We may interpret (24) as a high-order discretisation scheme in time associated to the time discretization operator $D_t$. Finally we add a pseudo-time step term in order to time-march toward the $2N+1$ coupled steady solutions:

$$|\mathcal{V}|\frac{\partial \mathbf{W}_{\mathcal{V}n}}{\partial \tau_n} + \mathbf{R}_{\mathcal{V}n} + |\mathcal{V}|\mathbf{T}_{\mathcal{V}n} = \mathbf{0}, \quad 0 \leq n < 2N+1 \tag{27}$$

In order to reveal the algebraic properties of the TSM source term operator, it is useful to rewrite equations (23) - (25) in matrix form. First recall that the one-sided DFT defines a linear transformation between a vector $\mathbf{z} \in \mathbb{R}^M$ of $M = 2N+1$ time instances of a $M$-periodic scalar function and a vector $\hat{\mathbf{z}} \in \mathbb{R}^M$ of $M$ frequencies such that

$$\hat{z}_k = \frac{1}{M}\sum_{n=0}^{2N} z_n F_M^{nk}, \quad 0 \leq k < 2N+1 \tag{28}$$

where $F_M = e^{-i2\pi/M}$ is the $M^{\text{th}}$ root of unity. It directly follows the matrix relation $\hat{\mathbf{z}} = \dfrac{1}{M}\mathbf{F}_M\mathbf{z}$. This $M \times M$ complex matrix is called the *Vandermonde-Fourier* matrix. It is symmetric, invertible, and each line or column is given by a geometric series of a power of $F_M = e^{-i2\pi/M}$. More specifically, we have

$$\mathbf{F}_M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & F_M & F_M^2 & \dots & F_M^{M-1} \\ 1 & F_M^2 & F_M^4 & \dots & F_M^{2(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & F_M^{M-1} & F_M^{2(M-1)} & \dots & F_M^{(M-1)(M-1)} \end{pmatrix} \tag{29}$$

Using the same consideration, the one-sided IDFT linear transformation is associated to the matrix $\mathbf{F}_M^{-1} = \mathbf{F}_M^*$, and $\mathbf{z} = \mathbf{F}_M^{-1}\hat{\mathbf{z}}$. Without loss of generality we now consider the special case of a single scalar unknown to be resolved in the current fluid cell. Equation(23) is now scalar and concatenating all state and residual Fourier coefficients in column vectors $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1, \cdots, \hat{w}_{M-1})^T$ and $\hat{\mathbf{r}} = (\hat{r}_0, \hat{r}_1, \cdots, \hat{r}_{M-1})^T$ we have

$$i\omega|\mathcal{V}|\boldsymbol{\mathcal{D}}\hat{\mathbf{w}} + \hat{\mathbf{r}} = \mathbf{0}, \quad \boldsymbol{\mathcal{D}} = diag(0, 1, \cdots, M-1) \tag{30}$$

Introducing the linear operators $\mathbf{F}_M$ and $\mathbf{F}_M^{-1}$ associated to the DFT and IDFT respectively gives

$$i\omega|\mathcal{V}|\mathbf{F}_M^{-1}\boldsymbol{\mathcal{D}}\mathbf{F}_M\mathbf{w} + \mathbf{r} = \mathbf{0} \tag{31}$$

The diagonal matrix $\boldsymbol{\mathcal{D}}$ can be interpreted as the eigenvalues of the matrix $\mathbf{A} = \mathbf{F}_M^{-1}\boldsymbol{\mathcal{D}}\mathbf{F}_M$ which is known to be *circulant*, i.e. $a_{m+1,n+1} = a_{m,n}$. This property is easily verified for the elements of the time-spectral derivative matrix $\mathbf{D}_t$ as $d_{n+1,j+1} = d_{nj}$ in (25).

Turning on to the full system of discretized equations on the whole structured fluid domain, we define the global vectors $\mathbf{W}$ and $\mathbf{R}$ concatenating the conservative variables $\mathbf{W}_\mathcal{V}$ and associated residuals $\mathbf{R}_\mathcal{V}$ for all the elementary fluid cells. The corresponding semi-discrete form of the governing equations is written as

$$|\boldsymbol{\mathcal{V}}|\frac{\partial \mathbf{W}_n}{\partial \tau_n} + \mathbf{R}_n + |\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n) = \mathbf{0}, \quad 0 \leq n < M \tag{32}$$

where $D_t(\mathbf{W}_n) = \sum_{j=0}^{M-1} d_{nj}\mathbf{W}_j$ and $|\boldsymbol{\mathcal{V}}|$ is now a block-diagonal matrix containing the cell volumes.

### 2.4 Fully implicit TSM solver

The time derivative in (32) is discretized by a first-order scheme

$$|\boldsymbol{\mathcal{V}}|\frac{\Delta \mathbf{W}_n}{\Delta \tau_n} + \mathbf{R}_n + |\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n) = \mathbf{0}, \quad 0 \leq n < M \tag{33}$$

where $\Delta\mathbf{W}_n = \mathbf{W}_n^{q+1} - \mathbf{W}_n^q$ is the solution increment between iterations $q$ and $q+1$. The backward-Euler implicit scheme is derived by linearizing the residual and the source term at $\mathbf{W}_n^{q+1}$. Let $\mathbf{J}_n = \partial\mathbf{R}_n/\partial\mathbf{W}_n$ be the Jacobian matrix of the residual vector, we have

$$\mathbf{R}_n^{q+1} \approx \mathbf{R}_n^q + \mathbf{J}_n\Delta\mathbf{W}_n \tag{34}$$

The time spectral derivative operator $D_t$ being linear it follows directly that

$$D_t(\mathbf{W}_n^{q+1}) = D_t(\mathbf{W}_n^q) + D_t(\Delta\mathbf{W}_n) \tag{35}$$

Substituting (34) and (35) in (33) we obtain the fully implicit TSM formulation

$$\left(\frac{|\boldsymbol{\mathcal{V}}|}{\Delta\tau_n} + \mathbf{J}_n + |\boldsymbol{\mathcal{V}}|D_t(.)\right)\Delta\mathbf{W}_n = -\mathbf{R}_n^q - |\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n^q) = -\mathbf{R}_{TSM}(\mathbf{W}_n^q) \tag{36}$$

Then the fully coupled system with all flow time instances gathered in vector $\mathbf{W} = (\mathbf{W}_0, \mathbf{W}_1, \cdots, \mathbf{W}_{M-1})^T$ can be written as

$$[\mathbf{A}]\Delta\mathbf{W} = -\mathbf{R}_{TSM}(\mathbf{W}^q) \tag{37}$$

with the resulting Jacobian matrix given by

$$[\mathbf{A}] = \begin{pmatrix} \frac{|\boldsymbol{\mathcal{V}}^0|}{\Delta\tau_0} + \mathbf{J}_0 & |\boldsymbol{\mathcal{V}}^1|\mathbf{d}_0^1 & \dots & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_0^{M-1} \\ |\boldsymbol{\mathcal{V}}^1|\mathbf{d}_1^0 & \frac{|\boldsymbol{\mathcal{V}}^1|}{\Delta\tau_1} + \mathbf{J}_1 & \dots & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_1^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_{M-1}^0 & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_{M-1}^1 & \dots & \frac{|\boldsymbol{\mathcal{V}}^{M-1}|}{\Delta\tau_{M-1}} + \mathbf{J}_{M-1} \end{pmatrix} \tag{38}$$

where $\mathbf{d}_n^j = diag(d_{nj}) = d_{nj}\mathbf{I}$. In the expression above we have emphasized that the volume matrix $|\boldsymbol{\mathcal{V}}^n|$ may depend on the time instance for the case of an ALE formulation with a deforming mesh.

As for the steady non-linear Newton algorithm in section 2.2 the Jacobian matrix can be chosen as the first-order approximate one $\mathbf{J}_{O1}$ or as the exact second-order one $\mathbf{J}_{O2}$. The solution to (37) can be obtained by inverting the large coupled matrix $[\mathbf{A}]$ but even considering the first-order Jacobian matrix in the left-hand side may become untractable for large numbers of time instances and large mesh sizes. In this work we rather adopt the block Jacobi formulation formerly proposed by [13] which allows a decoupling of the diagonal blocks in the implicit stage. This yields $2N + 1$ independent linear systems like

$$\left(\frac{|\boldsymbol{\mathcal{V}}^n|}{\Delta\tau_n} + \mathbf{J}_n\right)\Delta\mathbf{W}_n^{l+1} = -D_t(|\boldsymbol{\mathcal{V}}^n|\Delta\mathbf{W}_n^l) - \mathbf{R}_{TSM}(\mathbf{W}_n^q), \quad 0 \leq n < 2N+1 \tag{39}$$

where $l$ denotes the block Jacobi iteration and $\Delta\mathbf{W}_n^0 = \mathbf{0}$. As pointed out by Mundis and Mavriplis in [20], with an increasing number of harmonics and/or higher reduced frequencies the full Jacobian matrix in (38) proceeds farther and farther from diagonal dominance. This could become problematic for stationary relaxation methods like block-Jacobi or block-Gauss-Seidel. In order to circumvent this limitation, they apply a Flexible GMRES to the full matrix and consider various preconditioning strategies [21].

## 3 A PYTHON-BASED MODULAR SOLUTION FRAMEWORK

We have set up a Python-based external solver linked with the kernel of the elsA CFD software. At each Newton iteration, the residual Jacobian matrix and the explicit residual for each fluid block are extracted. The solution increment is computed by the Python framework and sent back into elsA for the parallel treatment of the boundary conditions and the solution increment exchange required by the additive Schwarz domain decomposition method. Here the term Jacobian matrix is somewhat generic as it may cover the exact second order Jacobian matrix $\mathbf{J}_{O2}$ or an exact first-order Jacobian matrix $\mathbf{J}_{O1}$ obtained by a first order linearization around the second order flow solution. A number of matrix-vector products are also available and serve as useful bricks for setting up external iterative Krylov solvers. More specifically, given an input vector $\mathbf{v}$ a vector $\mathbf{w}$ is returned according to the following situations:

- $\left[\frac{|\mathcal{V}|}{\Delta\tau} + \widetilde{\mathbf{J}}_{\mathrm{O1}}\right]\mathbf{w} = \mathbf{v}$ — $\mathbf{w}$ is the solution of the first-order approximate linear system
- $\left[\frac{|\mathcal{V}|}{\Delta\tau} + \widetilde{\mathbf{J}}_{\mathrm{O1}}\right]^T\mathbf{w} = \mathbf{v}$ — $\mathbf{w}$ is the solution of the adjoint first-order approximate linear system
- $\mathbf{w} = \mathbf{J}_{\mathrm{O2}}\mathbf{v}$ — $\mathbf{w}$ is the product of the exact Jacobian times $\mathbf{v}$
- $\mathbf{w} = \mathbf{J}^T_{\mathrm{O2}}\mathbf{v}$ — $\mathbf{w}$ is the product of the transpose of the exact Jacobian times $\mathbf{v}$

First or second order Jacobian matrices are available for each fluid domain as Numpy sparse matrices in local block addressing or in global addressing. This gives the added flexibility of re-assembling the full domain operator and let a third-party domain decomposition based solver to tackle the linear system solution process. Alternatively, one solver may be attached to each fluid block and the source term of the neighbor blocks, i.e. the contribution of fictitious cells, is obtained from the solver kernel.

Based on these elementary bricks, the TSM solver is then set up. We follow the implicit Block-Jacobi solution strategy proposed by Sicot [13] and detailed in (39). However, unlike the kernel implementation of the TSM solver where each fluid block is duplicated in memory according to the number of time instants, the external framework will spawn several instances of the steady-state solver that will run concurrently (one instance of elsA for each time instant in the period). The time stepping is performed externally at each iteration and the TSM source term is computed in the Python framework in order to get the proper right-hand side for the coupled TSM system. An inter-process communication strategy through pipes in shared memory is used to exchange flow solutions between the solver instances. This organization is similar to a $N^2$ dependency diagram used in complex systems analysis. It actually emulates a round-robin communicator.

## 4 NUMERICAL EXPERIMENTS

The selected test case to validate the implementation of the TSM solver is a symmetric *NACA64A010* airfoil under a prescribed harmonic pitch motion. The motion is sinusoidal with an angle of attack defined by the function

$$\alpha(t) = \alpha_0 + \widehat{\alpha}\sin(\omega t) \tag{40}$$

where $\alpha_0$ is the mean angle of attack and $\widehat{\alpha}$ is the amplitude of the pitch motion.

Experimental results are reported by the AGARD group in [22]. We have selected the *dynamic index 55* case corresponding to a transonic Mach number of 0.796 and an excitation frequency of 34.4Hz. The fluid domain is discretized with a structured 2D C-mesh of 257x33 cells. The inviscid Euler equations are considered for the fluid model. An upwind second order Roe spatial discretization scheme associated to a MUSCL reconstruction and a Van Albada limiter for the convective fluxes is used. The flow conditions are reported in Table 1.

Table 1: Flow conditions for dynamic index 55.

| | | |
|---|---|---|
| $M$ | 0.796 | Mach number |
| $p_\infty$ | 133912 Pa | free stream pressure |
| $p_t$ | 203321 Pa | total pressure |
| $q$ | 59395 Pa | dynamic pressure |
| $f$ | 34.4 Hz | pitching frequency |
| $\alpha_0$ | $-0.21^\circ$ | mean incidence |
| $\widehat{\alpha}$ | $1.02^\circ$ | pitching amplitude |

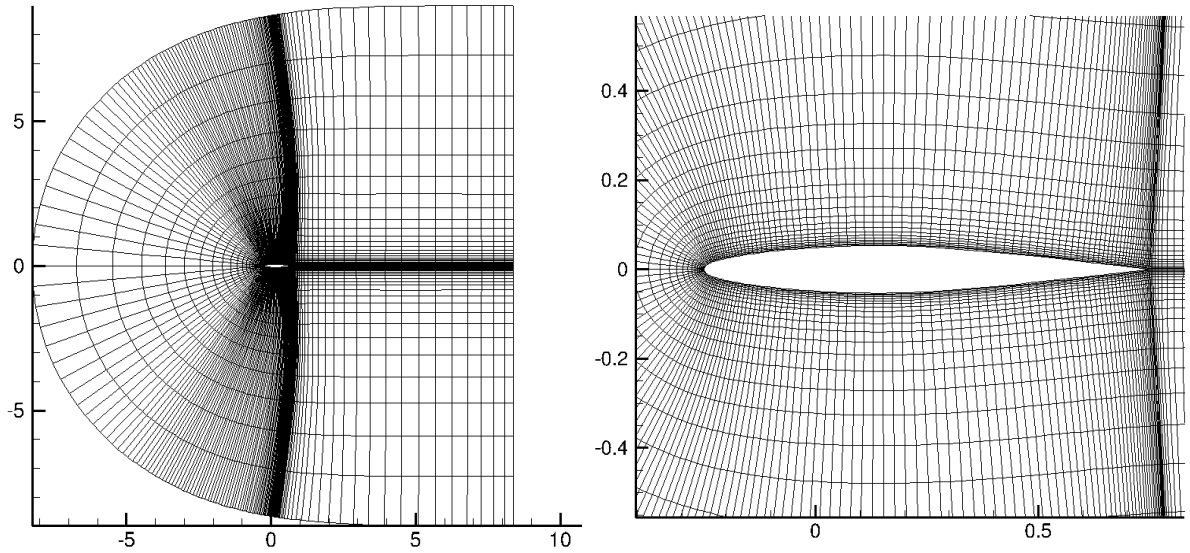


Figure 1: Structured 2D C-mesh of the *NACA64A010* airfoil.

### 4.1 Steady flow solution

The steady flow solution is computed for the mean angle of attack $\alpha_0 = -0.21^\circ$. The flow pattern is almost symmetric and exhibits strong shocks at the upper and lower skin. The pressure coefficient distributions are plotted in the left-hand side of Figure 3 while the rigth-hand side shows the density residual rate of convergence. On both plots the results obtained with the elsA kernel solver and the external Python-based solution framework are compared. First it is observed that $C_p$ distributions match perfectly. Second, the use of the $2^{\text{nd}}$ order exact Jacobian matrix by the external solver dramatically reduces the number of iterations. For this computation, we have used interchangeably a direct sparse LU solver or a Flexible GMRES solver pre-conditioned by an ILUTP factorization. Both perform equally well on the full Euler Jacobian matrix. Of course this observation would certainly be mitigated for a stiffer 3D problem and a viscous RANS fluid model.

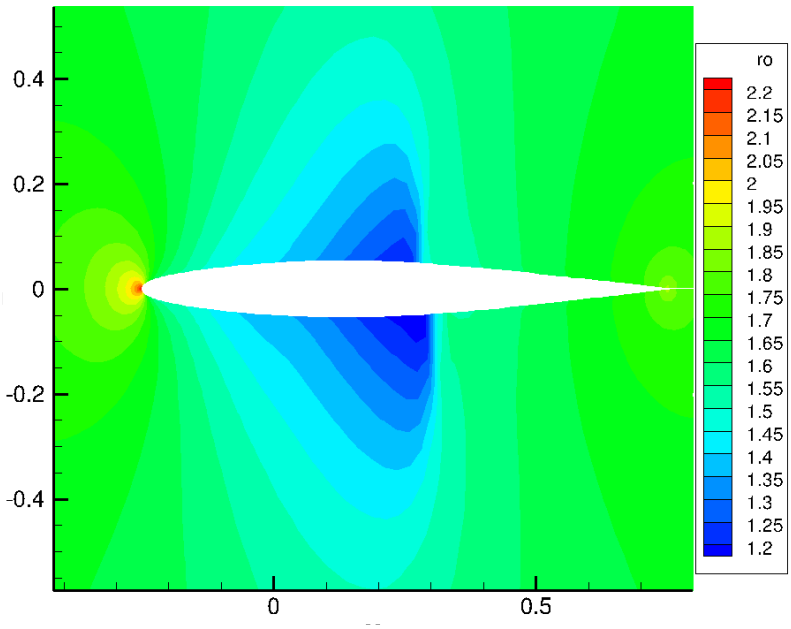


Figure 2: Steady flow density solution for the transonic Mach number test case (M=0.796).

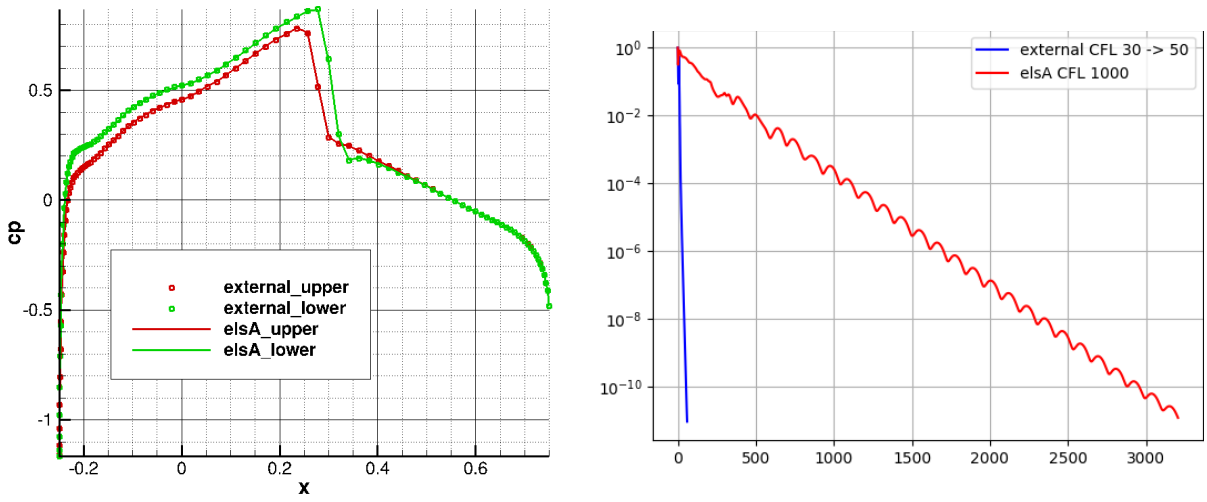


Figure 3: Comparison of Cp distribution and density residual rate of convergence between the kernel and external solver.

### 4.2 Time-Spectral flow solution

In this section the *NACA64A010* airfoil oscillating in a transonic free stream is simulated. The mean flow conditions correspond to the steady flow conditions of the previous subsection. The airfoil rotates around the $Oy$ transverse axis, $O$ being the 1/4 chord point. The pitching angle function is given in (40). Recalling that the governing equations follow an absolute velocity formulation in a moving reference frame, we set up one steady simulation with a constant angular entrainment velocity for each time instant in the period. The instantaneous angular speed vector $\mathbf{\Omega}(t) = (0, \Omega_y(t), 0)$ has an amplitude equal to the time derivative of the unsteady angle of attack: $\Omega_y(t) = \dot{\alpha}(t) = \omega\widehat{\alpha}\cos(\omega t)$. This is illustrated in Figure 4 for a 5 instant discretization where the instantaneous attitude of

the airfoil is drawn. The ALE fluxes balance and the source term in Equations (7) and (8) are then computed by the flow solver and the corresponding Jacobian matrix and explicit residual extracted towards the external Python framework for the assembly and solution of the TSM coupled system. In this process all the solver instances are independent to each other and run concurrently with their own distributed resources. Compared to the kernel implementation of the TSM solver where the bottleneck lies in the allocatable core memory, the limit now appears in terms of the available number of cores.

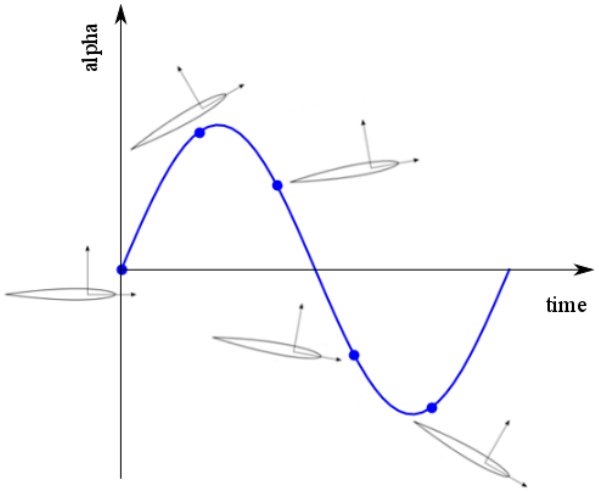


Figure 4: Illustration of the varying angle of attack and associated angular velocity over a time period.

The reconstructed time histories for the lift and pressure drag coefficients are presented in Figure 5 for a 3, 5 and 7 instant discretization of the period. At the outcome of the TSM analysis, the flow solution is known at each time instant and consequently the associated lift and drag coefficients. Using the following real Fourier series decomposition at time $t_n = n\Delta t = n/(2N+1)$, with $n = 0, \cdots, 2N$

$$f(t_n) = a_0 + \sum_{k=1}^{N} \left[ a_k \cos(\omega t_n) + b_k \sin(\omega t_n) \right] \tag{41}$$

a small linear system has to be solved to get the Fourier coefficients.

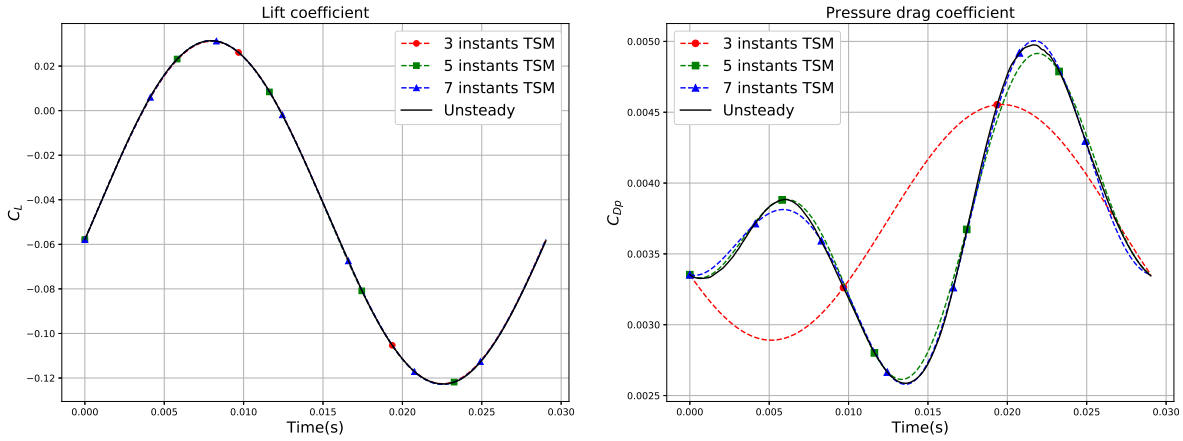


Figure 5: Accuracy of the time-spectral reconstruction for lift and drag coefficients for 3, 5 and 7 time instants.

It appears that the lift coefficient response is linear and only a single harmonic is sufficient for an accurate reconstruction. However, this is obviously not satisfactory for the reconstruction of the pressure drag nonlinear response which requires at least 3 harmonics to achieve a satisfactory accuracy. In Table 2 below are presented the average values and standard deviations of the lift and pressure drag coefficients for an increasing number of harmonics. The steady values at mean flow conditions are also reported for comparison. As expected the average values are well predicted even with only one harmonic thanks to the mathematical property of the Fourier decomposition. The standard deviation of lift coefficient is also well predicted thanks to its almost perfect linear behavior. However, the standard deviation of the nonlinear drag coefficient requires 7 instants to stabilize.

Table 2: Average values and standard deviations of aerodynamic coefficients.

| $C_{Dp}$ steady | $C_{Dp}$ avg. 3 inst. | $C_{Dp}$ avg. 5 inst. | $C_{Dp}$ avg. 7 inst. | $C_{Dp}$ std. 3 inst. | $C_{Dp}$ std. 5 inst. | $C_{Dp}$ std. 7 inst. |
|---|---|---|---|---|---|---|
| $2.508\,10^{-3}$ | $3.410\,10^{-3}$ | $3.414\,10^{-3}$ | $3.396\,10^{-3}$ | $2.222\,10^{-3}$ | $6.562\,10^{-4}$ | $6.555\,10^{-4}$ |
| $C_L$ steady | $C_L$ avg. 3 inst. | $C_L$ avg. 5 inst. | $C_L$ avg. 7 inst. | $C_L$ std. 3 inst. | $C_L$ std. 5 inst. | $C_L$ std. 7 inst. |
| $-4.655\,10^{-2}$ | $-4.680\,10^{-2}$ | $-4.660\,10^{-2}$ | $-4.672\,10^{-2}$ | $5.336\,10^{-2}$ | $5.347\,10^{-2}$ | $5.347\,10^{-2}$ |

The phase portrait of the aerodynamic coefficients as functions of the angle of attack considering the whole time history of the reference unsteady analysis are plotted in Figure 6 below. The plain bold curves correspond to the established periodic response in the $8^{\text{th}}$ time period, meaning that the computational time for the 7 preceding transient periods has been spent for nothing. The colored markers correspond to the instantaneous values obtained for a varying number of harmonics.

In Figure 7 the convergences for the density residual norm are plotted for the external Python solver and for the elsA kernel implementation. We recall that both solvers adopt a block-Jacobi strategy. The external solver is based on the exact second order Jacobian matrix whereas the internal solver is based on the cheap upwind first order decomposition developed in [3]. The same level of convergence is attained in roughly 2 times less iterations for the external solution strategy. The CFL number corresponding to the higher number of harmonics has been used even if in practice the CFL could be increased with a smaller number of harmonics. As a consequence, it is noticed that increasing the number of harmonics in the TSM approximation only slightly affects the number of iterations. The kernel solver exhibits a rapid residual norm decrease up to iteration 400 and then adopts a lower slope than the external solver. This suggests an hybrid strategy where the external solver is activated after several hundreds of iterations based on the upwind first order Jacobian approximation.

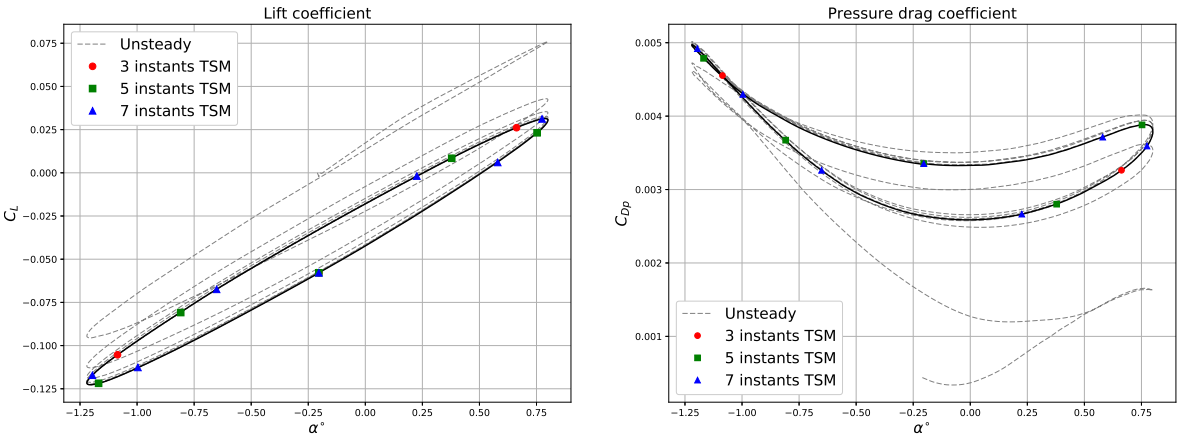


Figure 6: Aerodynamic coefficients versus angle of attack plot. The instantaneous values for a varying number of harmonics are superimposed to the reference unsteady time history.

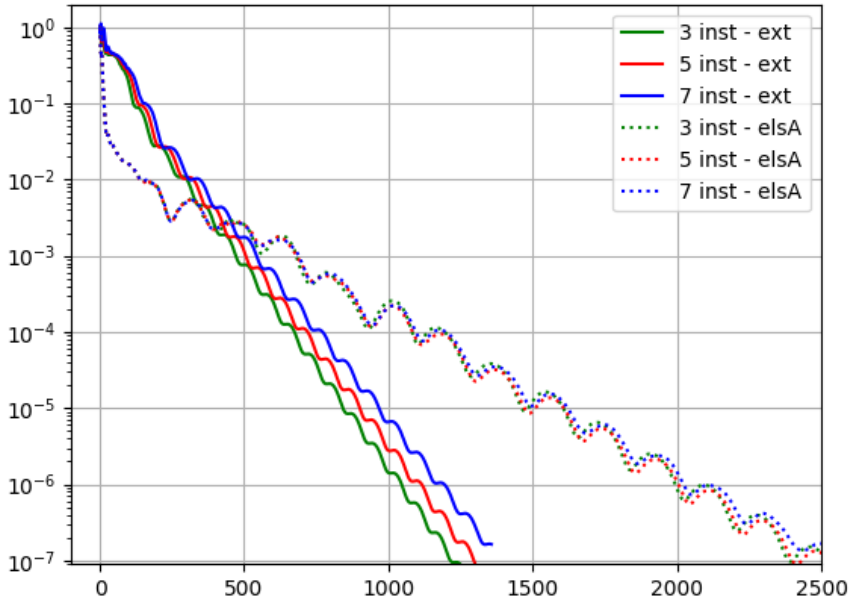


Figure 7: Convergence rate of density residual norm for an increasing number of harmonics. Comparison between kernel and external solver.

## 5 ADJOINT-BASED TSM SOLVER FOR SENSITIVITY ANALYSIS

Let $\mathbf{p}$ be a set of $n_p$ shape design parameters, $\mathbf{X}(\mathbf{p})$ the corresponding parameterized fluid mesh and $\mathcal{J}(\mathbf{p}) = J(\mathbf{W}_n(\mathbf{p}), \mathbf{X}(\mathbf{p}))$ any scalar aerodynamic function of interest like drag, lift or moment coefficient. Note that the flow variables at the $2N+1$ instants in the period appear in function $\mathcal{J}$. The flow field $\mathbf{W}_n$ and the volume mesh $\mathbf{X}$ are linked by the discrete residual equations

$$\mathbf{R}_{TSM}(\mathbf{W}_n, \mathbf{X}) = \mathbf{R}_n(\mathbf{W}_n, \mathbf{X}) + |\boldsymbol{\mathcal{V}}| D_t(\mathbf{W}_n) = \mathbf{0} \tag{42}$$

In this paper we only discuss applications of the ALE formulation for a rigid entrainment velocity field. We then consider a constant mesh geometry and the associated volume matrix $|\boldsymbol{\mathcal{V}}|$ is therefore taken constant in this section. Anyway, the extension to

a general ALE formulation is straightforward. Direct differentiation of (42) with respect to a design parameter $p$ gives

$$\frac{d\mathbf{R}_{TSM}}{dp} = \sum_{j=0}^{M-1} \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{W}_j} \frac{d\mathbf{W}_j}{dp} + \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} \tag{43}$$

A strong requirement here is that the total variation of the residual must vanish for any design parameter change, i.e. $d\mathbf{R}_{TSM}/dp = \mathbf{0}$. Inserting the definition of $\mathbf{R}_{TSM}$ in (43) leads to the following linear system:

$$\mathbf{J}_n \frac{d\mathbf{W}_n}{dp} + |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{nj} \frac{d\mathbf{W}_j}{dp} = -\frac{\partial \mathbf{R}_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} - \delta_p |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{nj} \mathbf{W}_j \tag{44}$$

where $\delta_p|\boldsymbol{\mathcal{V}}| = \dfrac{\partial |\boldsymbol{\mathcal{V}}|}{\partial \mathbf{X}} \dfrac{d\mathbf{X}}{dp}$ represents the matrix of analytic sensitivities of cell volumes to a change in $p$. On the left-hand side we recognize the full Jacobian matrix $[\mathbf{A}]$ in (38).

The general form of the objective function for a periodic response is $J = \dfrac{1}{M} \sum\limits_{n=0}^{M-1} J_n(\mathbf{W}_n, \mathbf{X})$. The total derivative of the objective function is then written as

$$\frac{d\mathcal{J}}{dp} = \frac{1}{M} \sum_n \frac{\partial J_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} + \frac{1}{M} \sum_n \frac{\partial J_n}{\partial \mathbf{W}_n} \frac{d\mathbf{W}_n}{dp} \tag{45}$$

We also have the following equality that holds $\forall\, \boldsymbol{\lambda}_n \in \mathbb{R}^{n_a}$

$$\frac{1}{M} \boldsymbol{\lambda}_n^T \frac{d\mathbf{R}_{TSM}}{dp} = 0, \quad 0 \leq n < M \tag{46}$$

and combining with (43) and (45) leads to

$$\begin{aligned} \frac{d\mathcal{J}}{dp} = & \frac{1}{M} \sum_n \frac{\partial J_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} + \frac{1}{M} \sum_n \boldsymbol{\lambda}_n^T \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} \\ & + \frac{1}{M} \sum_n \left( \frac{\partial J_n}{\partial \mathbf{W}_n} + \boldsymbol{\lambda}_n^T \mathbf{J}_n \right) \frac{d\mathbf{W}_n}{dp} + \frac{1}{M} \sum_n \boldsymbol{\lambda}_n^T \left( |\boldsymbol{\mathcal{V}}| \sum_j d_{nj} \frac{d\mathbf{W}_j}{dp} \right) \end{aligned} \tag{47}$$

The adjoint system is obtained by canceling the factor of $d\mathbf{W}_n/dp$ which defines the adjoint vectors $\boldsymbol{\lambda}_n$ as the solutions of the following coupled linear systems

$$\mathbf{J}_n^T \boldsymbol{\lambda}_n + |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{jn} \boldsymbol{\lambda}_j = -\left( \frac{\partial J_n}{\partial \mathbf{W}_n} \right)^T, \quad 0 \leq n < M \tag{48}$$

The transpose of the system matrix (38) of the TSM solver naturally appears in the adjoint linear system. Using the skew-symmetric property of the time-spectral derivative matrix $(d_{nj})_{0 \leq n,j < M}$ we have $d_{jn} = -d_{nj}$. Finally, the total derivative of the objective function turns out to be

$$\frac{d\mathcal{J}}{dp} = \frac{1}{M} \sum_n \left( \frac{\partial J_n}{\partial \mathbf{X}} + \boldsymbol{\lambda}_n^T \frac{\partial \mathbf{R}_n}{\partial \mathbf{X}} \right) \frac{d\mathbf{X}}{dp} + \frac{1}{M} \sum_n \boldsymbol{\lambda}_n^T \left( \delta_p |\boldsymbol{\mathcal{V}}|\, D_t(\mathbf{W}_n) \right) \tag{49}$$

# 6 SHAPE OPTIMIZATION OF THE NACA64A010 AIRFOIL

## 6.1 Shape parameterization

For the geometrical parameterization of the *NACA64A010* airfoil we adopt the simple but flexible Kulfan's Class and Shape Function Transformation formulation (CST) which is able to accurately approximate almost any aerofoil [23]. It consists in a universal approximation formulation for upper and lower aerofoil surfaces of the form

$$z(x, \nu_r, \nu_{LE}, z_{TE}) = \sqrt{x}(1-x)\sum_{r=0}^{n} \nu_r B_r^n(x) + z_{TE}\, x + \nu_{LE}\, x(1-x)^{n+\frac{1}{2}} \tag{50}$$

where the heart of (50) lies in a linear combination of Bernstein basis functions:

$$B_r^n(x) = \binom{n}{r} x^r (1-x)^{n-r} \quad \forall\, n \geq 0,\ 0 \leq r \leq n \text{ and } x \in [0,1] \tag{51}$$

The middle term in the right-hand side of (50) takes into account the elevation or thickness of the trailing edge in the form of a linear wedge function from the leading edge point ($x = 0$) to the trailing edge point ($x = 1$). The last term is the supplementary leading edge shaping term which brings flexibility in the control of the airfoil nose. Finally the first term is a product of the linear combination of basis functions and the so-called airfoil class function $\sqrt{x}(1-x)$. Figure 8 shows the CST approximation of the *NACA64A010* airfoil using Bernstein polynomials of order 3 for upper and lower surfaces. Considering that the trailing edge position is fixed, the 3$^{\text{rd}}$ order approximation has a total of 10 design parameters. The inset close-up views confirm the accurate reconstruction of the leading and trailing areas.

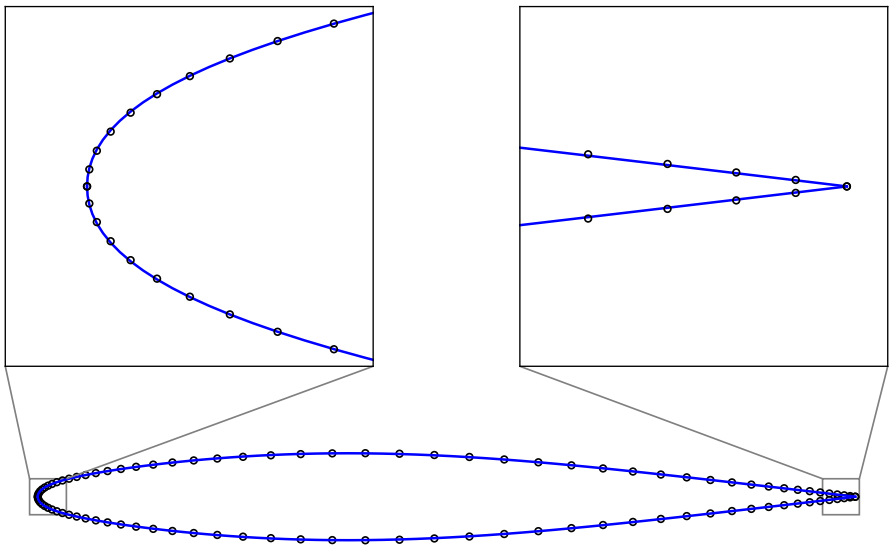

Figure 8: Kulfan's CST reconstruction of the *NACA64A010* airfoil. Circle marks correspond to CFD wall grids extracted from the 2D C-mesh in Figure 1.

From an optimization point of view it is desirable to have a parsimonious parameterization of our airfoil, that is low order polynomials in (50). Of course it must be kept in mind that few design parameters also means a limited exploration capability for promising shapes.

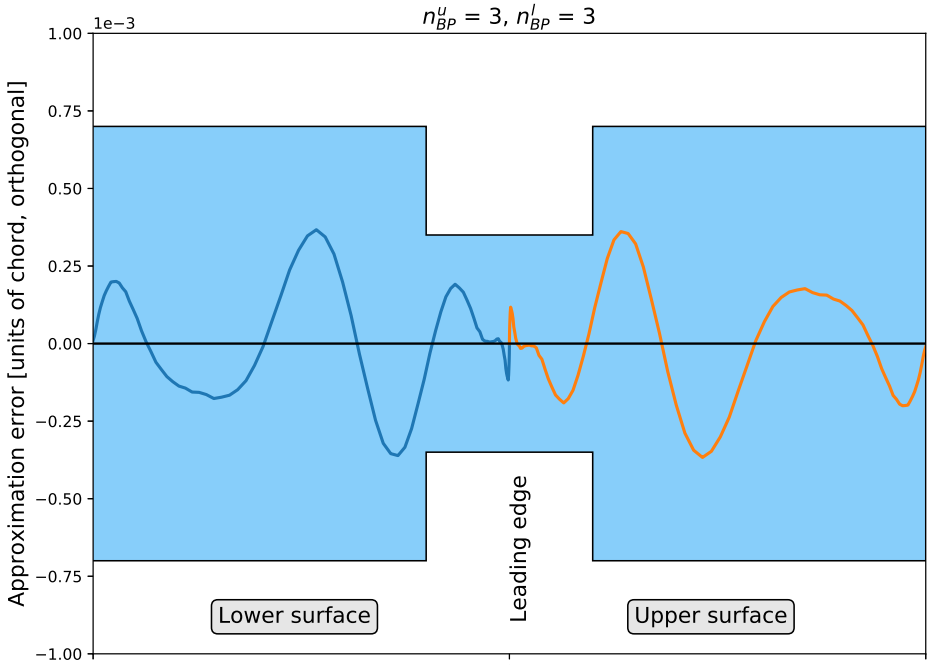


Figure 9: Kulfan's CST reconstruction error for the *NACA64A010* airfoil.

One way of assessing the accuracy of a CST reconstruction is to plot the error distance between the reference and the reconstructed airfoil along the surface. Kulfan [24] sets an approximation accuracy target based on the manufacturing accuracy of a typical wind tunnel model – the shaded regions on the plots in Figure 9 represent the corresponding error band. The Python routines for the CST approximation and the error plots follow the lines proposed by Sóbester and Forrester in [25]. As shown in Figure 9, a set of Bernstein polynomials of order 3 allows an accurate reconstruction of the *NACA64A010* airfoil.

### 6.2 Shape design at mean flow conditions

We now perform a shape optimization of the *NACA64A010* airfoil using the 10 parameter CST formulation presented in the previous section. The flow conditions are reported in Table 1. The initial shape corresponds to the CST reconstruction obtained in section 6.1 with the associated deviation error plotted in Figure 9. The optimization problem consists in minimizing the pressure drag coefficient $C_{Dp}$ while keeping the lift coefficient $C_L$ unchanged. The mean angle of attack is also fixed at $\alpha = -0.21°$. The optimizer is the SLSQP algorithm of the Scipy package [26]. Gradients of objective and constraint functions are obtained using the so-called mesh adjoint solver of the elsA software which solves the linear adjoint system for adjoint vectors $\boldsymbol{\lambda}_{C_{Dp}}$ and $\boldsymbol{\lambda}_{C_L}$ and outputs the following mesh sensitivity:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{X}} = \frac{\partial J}{\partial \mathbf{X}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{R}}{\partial \mathbf{X}} \tag{52}$$

Given the grid sensitivities $\partial \mathbf{X}/\partial p$, $p = 1 \cdots n_p$, the total gradients are post-processed from the product

$$\frac{d\mathcal{J}}{d\mathbf{p}} = \frac{\partial \mathcal{J}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \mathbf{p}} \tag{53}$$

The term $\partial \mathbf{X}/\partial p$ is obtained using a simple mesh deformation of the analytic sensitivities at the wall provided by the linear CST formulation. The same procedure is applied for the construction of the current CFD mesh $\mathbf{X}(\mathbf{p})$ which is merely a deformation of the initial mesh taking as input the difference between the current and initial wall grid locations. Initial and optimized airfoil geometries are compared in Figure 10.

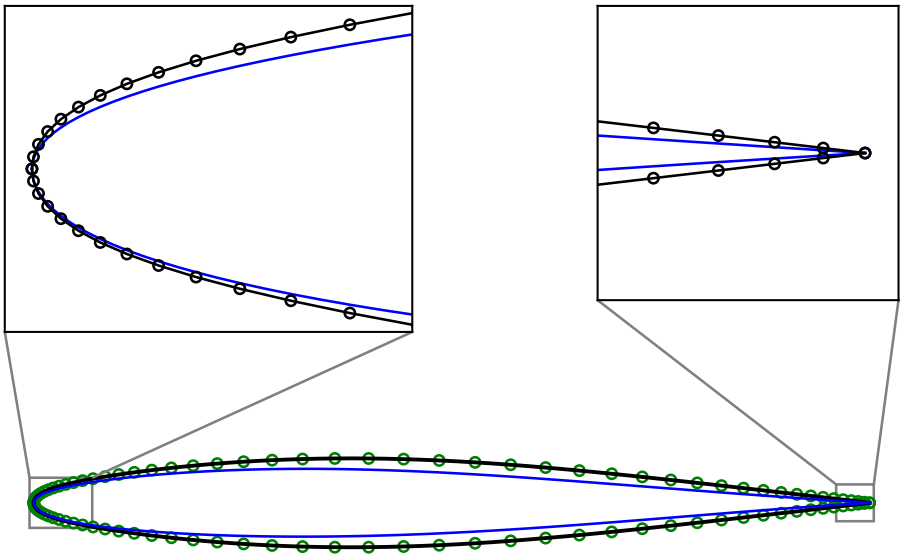

Figure 10: Initial (black line) vs. optimized (blue line) airfoil geometry.

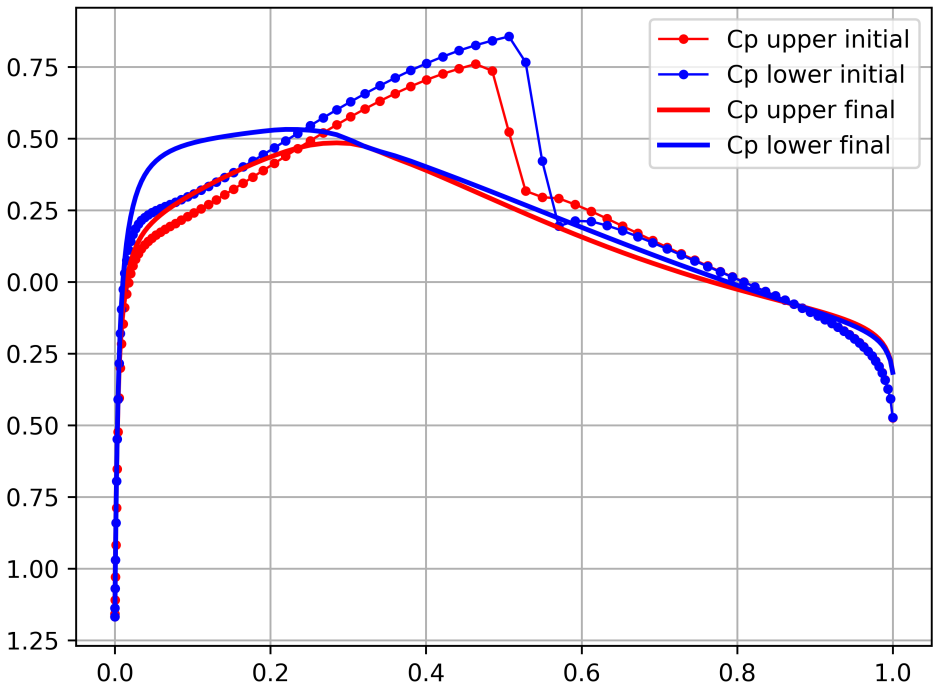


Figure 11: Initial vs. optimized $C_p$ distribution.

The final shape is not symmetrical anymore showing a slightly thinner leading edge and a profile shifted downward. The center part of the airfoil is thinner with a slightly more pronounced flattening at the upper edge. Corresponding initial and final pressure

coefficient distributions are plotted in Figure 11. The shocks have been suppressed and the major part of the lift is now produced in the leading edge region. After the 1/4 chord point we notice that the profile shape is quasi-symmetrical with similar pressure coefficient distributions.

### 6.3 TSM-based shape design over a time period

Our objective is now to minimize the average pressure drag coefficient over a time period. Given a time discretization of $M$ instants, the objective function is formulated as

$$\overline{C}_{Dp} = \frac{1}{M} \sum_{n=0}^{M-1} C_{Dp}(\mathbf{W}_n, \mathbf{X}) \tag{54}$$

Similarly to the steady case, the average lift coefficient is constrained at its initial value. In Figure 5 it is observed that the lift coefficient response remains linear and it is then expected that the average value should be close to the steady state value at an incidence $\alpha = -0.21°$. The average values of the aerodynamic coefficients for the initial airfoil geometry are reported in Table 2. The optimization has been performed with a 5 and 7 instant TS approximation. Even if average values are correctly predicted using 3 harmonics only, this was thought as an exercise to test the robustness of the TS adjoint solver. On the other hand, the minimization of the standard deviation would require at least 5 instants. The reconstructed time histories for the initial and optimal lift and pressure drag coefficients are presented in Figure 12. As expected, the lift coefficient has kept the same average value even if the peaks have been slightly increased during the optimization process. The average pressure drag coefficient has been divided by a factor 3. Interestingly, while not originally specified in the optimization problem, the standard deviation was also decreased.

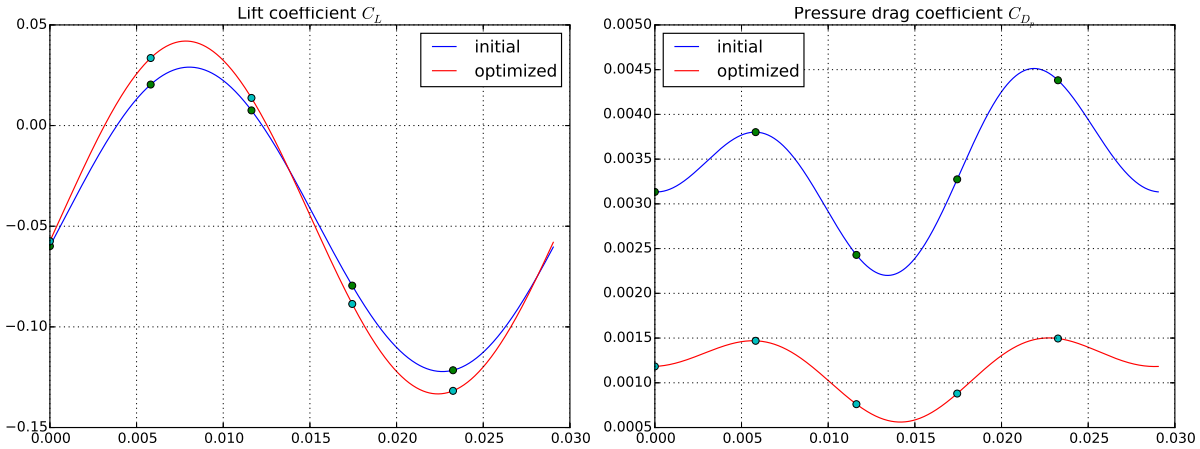


Figure 12: Unsteady aerodynamic coefficients for the initial and the optimal airfoil geometry.

The initial and final shapes are plotted in Figure 13. For better insight, the camber line has also been added. The whole profile is thinner with a large flattened area at the lower curve resulting in a positive camber in the first half of the profile. Corresponding initial and final pressure coefficient distributions obtained from a steady simulation of the optimized airfoil at mean flow conditions are plotted in Figure 14. As before, the shocks have been suppressed by the optimization process. The $C_p$ distribution on the upper skin

is quite regular while the lower $C_p$ distribution exhibits a pressure peak right after the leading edge.

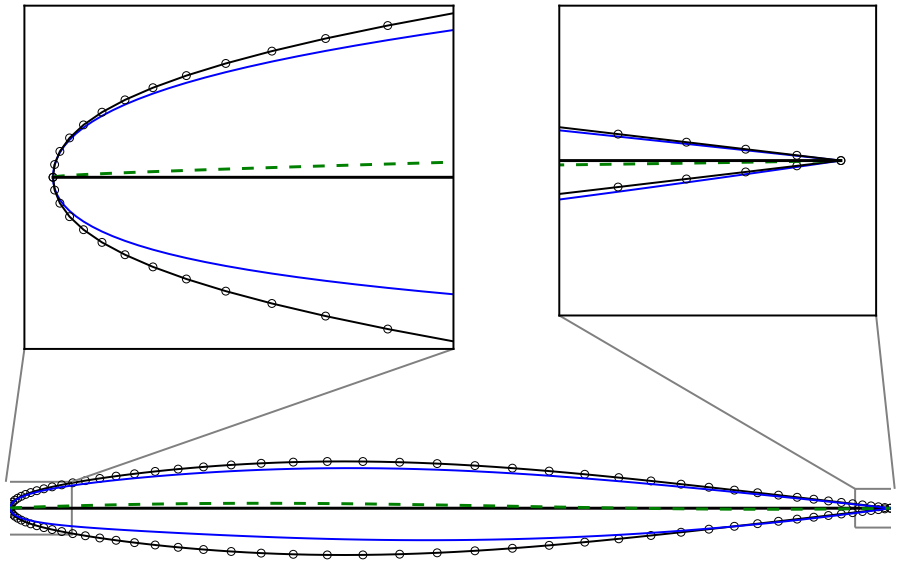

Figure 13: Initial (black line) vs. optimized (blue line) airfoil geometry.

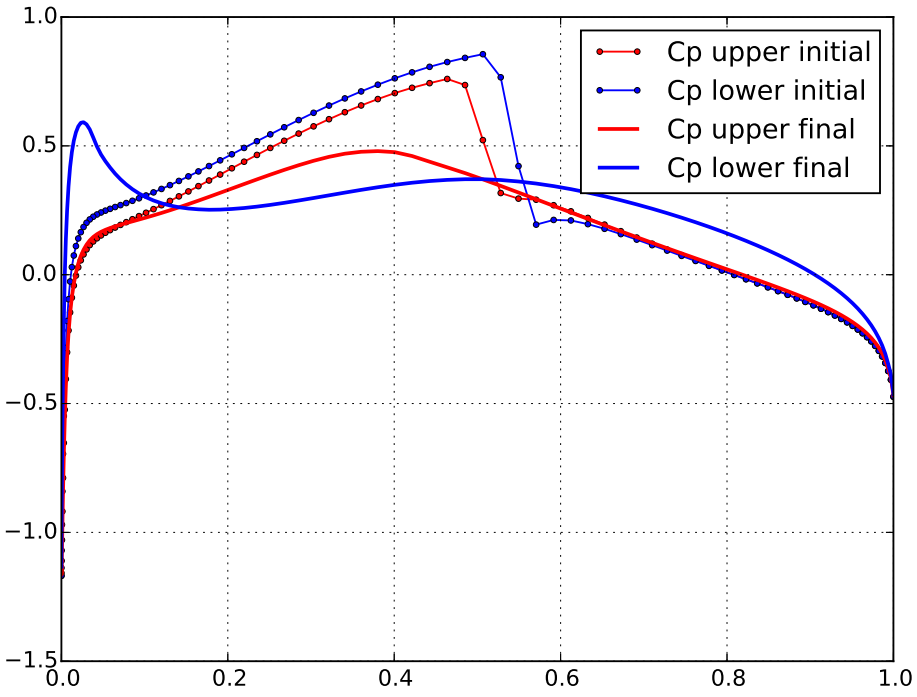


Figure 14: Initial vs. optimized $C_p$ distribution.

## 7 CONCLUSION

In this paper we presented the benefit of modularization of the elsA CFD code in terms of added flexibility and rapid prototyping of alternative and promising solution strategies for the steady, time-spectral and time-spectral adjoint problems. The externalization of

the exact and approximate flux-Jacobian matrices along with the explicit cell residuals offers a wide range of combinations for efficient linear algebra algorithms in terms of implicitation and preconditioners for iterative solvers. We have demonstrated that using standard Python scientific packages and parallelization features was able to leverage good practices for further more intensive numerical implementations. More specifically the new parallelization scheme for the TSM solver consisting of spawning several instances of the steady solver instead of duplicating the steady problem in the CPU core memory was implemented in a very short time. In the same way, the TS adjoint system was assembled easily and efficiently solved by reusing the same software bricks as for the TSM solver. As pointed out by Jameson the time-spectral method offers optimal efficiency for periodic problems and while it should make it possible to achieve spectral accuracy, numerical tests have shown that it can give the accuracy required for practical applications with very small numbers of modes. However, it remains that the CFL limitation associated to a higher number of time instances or higher excitation frequencies requires additional developments to ensure competitiveness with respect to the reference unsteady simulation. Some promising implicitation schemes decoupling spatial and time contributions have been proposed by Mundis and Mavriplis and we will investigate these in the near future. However, this was not the primary goal of this exploratory study. Concerning the optimization of a periodic function in terms of the average pressure drag over a time period, it has been shown that the associated optimal airfoil shape was quite different from the optimal shape resulting from a steady optimization conducted at the mean flow conditions only. Interestingly, the former airfoil shape still provides an improved drag performance in steady flow. Finally we will further consolidate these preliminary results to more advanced RANS fluid model and 3D multiblock configurations. In the near future, the TSM ALE formulation for deformable fluid blocks will be implemented thus allowing efficient computation of generalized aerodynamic force sensitivities and consequently optimization studies with high-fidelity models for dynamic aeroelasticity.

## 8 ACKNOWLEDGMENTS

The authors wish to thank their colleague Pascal Raud for providing a customized version of the elsA CFD software capable of exposing the flux Jacobian matrices. Without his personal involvement this exploratory study would not have been possible.

## 9 REFERENCES

[1] Jameson, A. (2003). “Time integration methods in computational aerodynamics”. Providence, RI.

[2] Saad, Y. (1993). “A flexible inner-outer preconditioned GMRES algorithm”. *SIAM Journal on Scientific Computing* 14(2), pp. 461–469.

[3] Yoon, S. and Jameson, A. (1987). “An LU-SSOR scheme for the Euler and Navier-Stokes equations”. AIAA 87-0600.

[4] Jameson, A. (1991). “Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings”. *10th Computational Fluid Dynamics Conference*.

[5] Hall, K. C. and Clark, W. S. (1993). “Linearized Euler predictions of unsteady aerodynamic loads in cascades”. *AIAA Journal* 31(3), pp. 540–550.

[6] Mortchéléwicz, G. D. (1997). "Application des équations d'Euler linéarsées à la prévision du flottement". *85th AGARD SMP Meeting, Technical Report AGARD-R-822*. Aalborg, Danemark.

[7] Widhalm, M. et al. (2010). "Efficient computation of dynamic stability data with a linearized frequency domain solver". *European Conference on Computational Fluid Dynamics*.

[8] McMullen, M., Jameson, A., and Alonso, J. (2001). "Acceleration of convergence to a periodic steady state in turbomachinery flows". *AIAA 39th Aerospace Sciences Meeting & Exhibit*. AIAA 01-0152. Reno, NV.

[9] McMullen, M. S. and Jameson, A. (2006). "The computational efficiency of non-linear frequency domain methods". *Journal of Computational Physics* 212(2), pp. 637–661.

[10] Hall, K. C., Thomas, J. P., and Clark, W. S. (2002). "Computation of unsteady nonlinear flows in cascades using a harmonic balance technique". *AIAA journal* 40(5), pp. 879–886.

[11] Gopinath, A. and Jameson, A. (2005). "Time spectral method for periodic unsteady computations over two-and three-dimensional bodies". *43rd AIAA aerospace sciences meeting and exhibit*. AIAA 2005-1220. Reno, Nevada.

[12] Gopinath, A. and Jameson, A. (2006). "Application of the time spectral method to periodic unsteady vortex shedding". *44th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA 2006-0449. Reno, Nevada.

[13] Sicot, F., Puigt, G., and Montagnac, M. (2008). "Block-Jacobi implicit algorithms for the time spectral method". *AIAA Journal* 46(12), pp. 3080–3089.

[14] Cambier, L., Heib, S., and Plot, S. (2013). "The Onera elsA CFD software: input from research and feedback from industry". *Mechanics & Industry* 14(3), pp. 159–174.

[15] Donea, J. et al. (2004). "Arbitrary Lagrangian Eulerian Methods". *Encyclopedia of Computational Mechanics*. Vol. 1. E. Stein, R. de Borst and T.J.R. Hughes, Wiley & sons. Chap. 14.

[16] Dumont, A. et al. (2011). "Aerodynamic shape optimization of hovering rotors using a discrete adjoint of the Reynolds-Averaged Navier–Stokes Equations". *Journal of the American Helicopter Society* 56(3), pp. 1–11.

[17] Morgan, R. B. (2002). "GMRES with deflated restarting". *SIAM Journal on Scientific Computing* 24(1), pp. 20–37.

[18] Pinel, X. and Montagnac, M. (2013). "Block Krylov methods to solve adjoint problems in aerodynamic design optimization". *AIAA Journal* 51(9), pp. 2183–2191.

[19] Saad, Y. (2003). *Iterative methods for sparse linear systems*. Vol. 82. SIAM.

[20] Mundis, N. L. and Mavriplis, D. J. (2013). "GMRES applied to the time spectral and quasi-periodic time spectral methods". *21st AIAA Computational Fluid Dynamics Conference*. AIAA 2013-3084.

[21] Mundis, N. L. and Mavriplis, D. J. (2017). "Toward an optimal solver for time-spectral fluid-dynamic and aeroelastic solutions on unstructured meshes". *Journal of Computational Physics* 345, pp. 132–161.

[22] Davis, S. S. (1982). "NACA 64A010 (NASA Ames model) oscillatory pitching". *AGARD report* 702.

[23] Kulfan, B. and Bussoletti, J. (2006). "Fundamental Parameteric Geometry Representations for Aircraft Component Shapes". *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*. AIAA 2006-6948.

[24] Kulfan Brenda, M. (2008). "Universal parametric geometry representation method". *Journal of Aircraft* 45(1), pp. 142–158.

[25] Sóbester, A. and Forrester, A. I. (2014). *Aircraft aerodynamic design: geometry and optimization*. John Wiley & Sons.
[26] Kraft, D. (1994). "Algorithm 733: TOMP–Fortran modules for optimal control calculations". *ACM Transactions on Mathematical Software (TOMS)* 20(3), pp. 262–281.

## COPYRIGHT STATEMENT