A Time-Spectral Adjoint Approach for Aerodynamic Shape Optimization Under Periodic Wakes

Ping He, Alton J. Luder, Charles A. Mader, Joaquim R. R. A. Martins, Kevin J. Maki
University of Michigan, Ann Arbor, MI, USA 48109

Design problems in aerospace engineering often involve periodic unsteady flow, and understanding its characteristics provides useful insights for the design. Aerodynamic shape optimization has been widely used to reduce the design period for steady-flow problems; however, its application for unsteady flow remains challenging due to its high computational cost and memory usage. In this paper, an approach is proposed to reduce the cost of aerodynamic shape optimization for unsteady flow problems. The proposed approach uses the time-spectral method to simulate flow and compute derivatives. Specifically, it uses a fully-segregated, block Gauss–Seidel (FSBGS) method to solve the time-spectral flow equations. The FSBGS formulation requires minimal memory overhead and code modification effort, compared with a fully-coupled time-spectral formulation. A discrete adjoint approach is then used to compute derivatives for the time-spectral flow solver. The diagonal blocks in the time-spectral Jacobian are assembled by reusing the steady-state Jacobians from each time instance, while the off-diagonal blocks are computed analytically. The NACA0012 airfoil is selected as the baseline geometry, and an oscillating source term is added in the momentum equation to mimic periodic wakes. The time-spectral flow solver is verified by comparing the simulated time-series of drag and lift with those simulated by the time-accurate method. The performance of the time-spectral flow and adjoint solvers are evaluated in terms of speed, memory usage, and accuracy. Finally, the implemented solvers are incorporated into a gradient-based optimization framework to perform aerodynamic shape optimization under periodic wakes. The averaged drag reduces by 10.5% and the lift constraint is satisfied. The proposed method has the potential to reduce the computational cost for analyzing and designing engineering systems that involve periodic wakes, e.g., propeller-wing interaction for electric aircraft and stator-rotor interaction for turbomachinery.

I. Introduction

Design problems in aerospace engineering often require considering periodic unsteady flow, e.g, the wakes generated by propellers and rotors. Understanding the unsteady flow characteristics is critical because it significantly impacts the performance of a design. For example, the distributed propulsion design used in electric vertical take-off and landing (eVTOL) aircraft requires an understanding of the aerodynamic interaction between multiple propeller wakes and the wing [1, 2]. In addition, multi-stage turbomachinery design needs to consider the wake interaction between rotors and stators [3, 4]. Fortunately, with the advance in computing power, we can use high-fidelity numerical simulations (i.e., computational fluid dynamics; CFD) to predict the unsteady flow behavior. This eventually provides useful insight to guide the design.

Traditionally, unsteady numerical simulations are performed in the time domain. This is done by marching the solution in time and repeating the marching process by multiple flow-over times to reach the statistically steady state; this approach is also known as the time-accurate method. To ensure numerical stability and accuracy, the maximal marching time step ($\Delta t_{\max}$) of the time-accurate method is limited by the Courant–Friedrichs–Lewy (CFL) condition. Although the time-accurate method is robust and applicable for any complex flow conditions (e.g., large flow separation and complex vortices structures), its computational cost is typically one or two orders of magnitudes higher than steady-state simulations, due to the limited $\Delta t_{\max}$.

To alleviate the high cost of the time-accurate method, we can use the time-spectral method [5], also known as the harmonic-balance method [6], to simulate unsteady flow. In contrast to the time-accurate method, the time-spectral method assumes the unsteady flow is periodic and uses the Fourier series to represent the unsteady flow variables based on only a handful of solutions within one time-period. The benefit of using the time-spectral method is that it effectively reduces the computational cost because the simulation is no longer bounded by $\Delta t_{\max}$ [5, 6]. Moreover, the time-derivative term computed by the Fourier method is much more accurate than the finite-difference method (e.g., second order Euler method) used in the time-accurate method [5, 6]. One drawback of using the time-spectral method is that the dominant frequency of the flow needs to be prescribed a priori. This is challenging for self-induced unsteady flow such as the von Kármán vortices because the dominant frequency is usually not known beforehand [5, 7, 8]. Fortunately, many design problems in aerospace engineering use prescribed periodic motions (e.g., rotating rotors and propellers and oscillating wings), in which the dominant frequency is known. The time-spectral and harmonic balance methods have been widely used for design problems in aerospace engineering [4–6, 9–17].

To further improve the design efficiency, we can couple an optimization algorithm with the time-spectral simulations, such that the optimization automatically navigates the design space to find the best design. The optimization algorithm significantly reduces the design period, compared with the human supervised approach [18]; therefore it has received increasing interests in recent years.

There are two categories of optimization algorithms: gradient-free and gradient-based. The gradient-free methods require only the function values to perform optimization. Therefore, they can treat the CFD solver as a black box, and are relatively easy to implement. Some gradient-free algorithms use global search in the design space; therefore, they have a better chance to find a point that is closer to the global minimum, compared with gradient-based methods. However, one drawback of the gradient-free methods is that they reply on using CFD samples for design improvement, and the computational cost to generate these samples scales exponentially with respect to the number of design variables [19, 20].

On the other hand, gradient-based methods do not need to generate a large number of CFD samples. Instead, they typically perform local searches in the design space by using the gradient information to find the most promising direction for design improvement. To efficiently compute the gradients, one can use the adjoint method [21] whose computational cost is independent of the number of design variables. This feature is attractive for design problems in aerospace engineering, in which we need to use a large number of design variables to parameterize complex design surfaces such as aircraft wings. Therefore, we opt to use the coupled gradient-based optimization and adjoint derivative computation approach in this paper.

The adjoint method has been used in steady-state design problems involves aerodynamics [22–31], structures [32, 33], heat transfer [34, 35], as well as multidisciplinary design problems [36] such as aerostructural [37–40], aerothermal [41, 42], aeropropulsive [43, 44], and hydrostructural [45, 46] optimizations. To handle unsteady flow problems, one can apply the adjoint method to a time-accurate CFD solver [47–50]. Because the objective function (e.g., drag, lift, and moment) depends on time, one needs to assemble the unsteady adjoint formulation using state variables (e.g., velocity, density, pressure) from all the time instances. This requires saving all the intermediate state variables and repeatedly solving the adjoint equations for each time instance, starting from the last time instance and moving backward. Practically, saving all the intermediate variables in memory is not feasible for large problems. Although advanced techniques such as checkpointing can be used to trade speed for memory [51, 52], the computational cost for time-accurate unsteady adjoint solvers are high [53].

Alternatively, one can apply the adjoint method to a time-spectral CFD solver for unsteady flow. The benefit of using the time-spectral adjoint approach is that it needs to store only a handful of time instances that construct the Fourier series. Therefore, the computational and memory cost for the time-spectral adjoint is much lower than that for the time-accurate adjoint. The time-spectral adjoint approach has been reported in a handful of studies [11, 15, 16].

In this study, we develop a time-spectral solver for periodic unsteady flow problems, based on an open-source CFD package OpenFOAM. The proposed time-spectral method uses a fully-segregated solution strategy that requires minimal modification to the original time-accurate solver and incurs low memory overhead. In addition, we apply the discrete adjoint method to the time-spectral CFD solver to efficiently compute derivatives, based on an open-source, object-oriented adjoint framework DAFoam [*]. Finally, we incorporate the time-spectral flow and adjoint solvers into an open-source, high-fidelity gradient-based optimization framework MACH [†]. We optimize the aerodynamic performance of the NACA0012 airfoil under periodic wakes originated in the upstream. We mimic the unsteady wakes by adding an oscillating source term in the momentum equation.

The rest of the paper is organized as follows. In Section II, we introduce the DAFoam framework and detail the object-oriented adjoint development. The design optimization results are presented and discussed in Section III and we summarize our findings in Section IV.

[*]https://github.com/mdolab/dafoam

[†]https://github.com/mdolab/mach-aero

II. Method

A. Flow simulation

The time-spectral solver is based on the OpenFOAM's time-accurate flow solver `pisoFoam`. `pisoFoam` solves three-dimensional, unsteady turbulent flow, governed by the incompressible Navier–Stokes equations:

$$\nabla \cdot \boldsymbol{U} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot (\boldsymbol{U}\boldsymbol{U}) + \frac{1}{\rho}\nabla p - \nu_{\text{eff}}\nabla \cdot (\nabla \boldsymbol{U} + \nabla \boldsymbol{U}^T) = 0, \tag{2}$$

where t is the time, p is the pressure, ρ is the density, $\boldsymbol{U}$ is the velocity vector $\boldsymbol{U} = [u, v, w]$, $\nu_{\text{eff}} = \nu + \nu_t$ with ν and ν_t being the kinematic and turbulent eddy viscosity, respectively.

To connect the turbulent viscosity to the mean flow variables, the Spalart–Allmaras (SA) model is used:

$$\frac{\partial \tilde{\nu}}{\partial t} + \nabla \cdot (\boldsymbol{U}\tilde{\nu}) - \frac{1}{\sigma}\{\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] + C_{b2}|\nabla\tilde{\nu}|^2\} - C_{b1}\tilde{S}\tilde{\nu} + C_{w1}f_w\left(\frac{\tilde{\nu}}{d}\right)^2 = 0. \tag{3}$$

The turbulent eddy viscosity ν_t is computed from $\tilde{\nu}$ via:

$$\nu_t = \tilde{\nu}\frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}. \tag{4}$$

Spalart and Allmaras [54] provide a detailed description of the terms and parameters in this model.

To solve the continuity and momentum equations (1) and (2) in time domain, the pressure implicit with splitting of operators (PISO) algorithm [55] is used, following Jasak [56]. First, the momentum equation is discretized, and an intermediate velocity field is solved using the pressure field obtained from the previous iteration ($p^{t-\Delta t}$).

$$a_P\boldsymbol{U}_P^t = -\sum_N a_N\boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} - \nabla p^{t-\Delta t} = \boldsymbol{H}(\boldsymbol{U}) - \nabla p^{t-\Delta t}, \tag{5}$$

where a is the coefficient resulting from the finite-volume discretization, subscripts P and N denote the control volume cell and all of its neighboring cells, respectively, and

$$\boldsymbol{H}(\boldsymbol{U}) = -\sum_N a_N\boldsymbol{U}_N^t + \frac{\boldsymbol{U}_P^{t-\Delta t}}{\Delta t} \tag{6}$$

represents the influence of velocity from all the neighboring cells and from the previous iteration. A new variable ϕ (face flux) is introduced to linearize the convective term:

$$\int_S \boldsymbol{U}\boldsymbol{U} \cdot \mathrm{d}\boldsymbol{S} = \sum_f \boldsymbol{U}_f\boldsymbol{U}_f \cdot \boldsymbol{S}_f = \sum_f \phi\boldsymbol{U}_f, \tag{7}$$

where the subscript f denotes the cell face. ϕ can be obtained from the previous iteration or from an initial guess. Solving the momentum equation (5), we obtain an intermediate velocity field that does not yet satisfy the continuity equation.

Next, the continuity equation is coupled with the momentum equation to construct a pressure Poisson equation, and a new pressure field is computed. The discretized form of the continuity equation is

$$\int_S \boldsymbol{U} \cdot \mathrm{d}\boldsymbol{S} = \sum_f \boldsymbol{U}_f \cdot \boldsymbol{S}_f = 0. \tag{8}$$

Instead of using a simple linear interpolation, $\boldsymbol{U}_f$ in this equation is computed by interpolating the cell-centered velocity $\boldsymbol{U}_P$—obtained from the discretized momentum equation (5)—onto the cell face as follows:

$$\boldsymbol{U}_f = \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f. \tag{9}$$

Algorithm 1 PISO algorithm to solve the unsteady incompressible NS equations

```
Input: $w_{\text{initial}}, \Delta t$    ▷ Initial state variables, time step
Output: $w_{\text{converged}}$    ▷ Converged state variables
 1: $w \leftarrow [U^0, p^0, \phi^0, \tilde{\nu}^0]$    ▷ Assign initial state variables
 2: for $t \in \{\Delta t, 2\Delta t, \dots, \infty\}$ do    ▷ Main loop with time interval $\Delta t$
 3:     $U^t \leftarrow a_P U_P^t + \sum a_N U_N^t = U_P^{t-\Delta t}/\Delta t - \nabla p^{t-\Delta t}$    ▷ Solve momentum equation
 4:     for $m \in \{1, 2, \dots, M\}$ do    ▷ $M$: number of corrector loops
 5:         $H(U) \leftarrow -\sum a_N U_N^t + U_P^{t-\Delta t}/\Delta t$    ▷ Compute the $H(U)$ term
 6:         $p^t \leftarrow \nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{H(U)}{a_P}\right)$    ▷ Solve pressure Poisson equation
 7:         $\phi^t \leftarrow \left[\left(\frac{H(U)}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p^t)_f\right] \cdot S_f$    ▷ Update face flux $\phi$
 8:         $U^t \leftarrow \frac{1}{a_P}[H(U) - \nabla p^t]$    ▷ Correct velocity
 9:     $\tilde{\nu}^t \leftarrow$ turbulence model    ▷ Solve turbulence equations
10:     if $t > t_{\max}$ then
11:         break    ▷ Maximal runtime is reach
```

This idea of momentum interpolation was initially proposed by Rhie and Chow [57] and is effective in mitigating the oscillating pressure (checkerboard) issue resulting from the collocated mesh configuration. Substituting Eq. (9) into Eq. (8), we obtain the pressure Poisson equation:

$$\nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right). \tag{10}$$

Solving Eq. (10), we obtain an updated pressure field p^t. Then, the new pressure field p^t is used to correct the face flux

$$\phi^t = \boldsymbol{U}_f \cdot \boldsymbol{S}_f = \left[\left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p^t)_f\right] \cdot \boldsymbol{S}_f, \tag{11}$$

and velocity field

$$\boldsymbol{U}^t = \frac{1}{a_P}[\boldsymbol{H}(\boldsymbol{U}) - \nabla p^t]. \tag{12}$$

Note that the $\boldsymbol{H}(\boldsymbol{U})$ term depends on $\boldsymbol{U}$ but has not been updated so far. To account for this, one needs to repeatedly solve the Eq. 6 and Eqs. 10 to 12 (corrector loop), such that the velocity and pressure fields fully satisfy the continuity and momentum equations at each time step. In this paper, we perform two iterations for the corrector loop.

Once done, we solve for the turbulence equation (3) to obtain an updated turbulence viscosity (ν_t). The above process is repeated for several flow-over times until a statistically steady state is reached. The PISO algorithm is summarized in Algorithm 1.

B. Time spectral method

As mentioned in the introduction, we use the time-spectral method to solve the unsteady NS equations. The time-spectral method assumes that the unsteady flow is periodic, such that any variables can be represented by the discrete Fourier series. Taking the velocity as an example:

$$\boldsymbol{U}(n\Delta t) \approx \boldsymbol{U}^n = \sum_{k=\frac{N-1}{2}}^{\frac{N-1}{2}} \tilde{\boldsymbol{U}}_k e^{ik\frac{2\pi}{T}n\Delta t}, \tag{13}$$

where $T = N\Delta t$ is the time period with Δt being the time interval and N being the total number of time instances, n is the nth discrete time instance, and $\tilde{\boldsymbol{U}}_k$ is the Fourier coefficient. The corresponding inverse Fourier transform is:

$$\tilde{\boldsymbol{U}}_k = \frac{1}{N}\sum_{n=0}^{N-1} \boldsymbol{U}^n e^{-ik\frac{2\pi}{T}n\Delta t}. \tag{14}$$

Using Eq. 13, we can compute the time-derivative as:

$$\frac{\partial \boldsymbol{U}^n}{\partial t} = \sum_{k=\frac{N-1}{2}}^{\frac{N-1}{2}} ik\frac{2\pi}{T}\tilde{\boldsymbol{U}}_k e^{ik\frac{2\pi}{T}n\Delta t}. \tag{15}$$

Substituting the inverse Fourier transform equation (14) into the time derivative equation (15) and simplify the expression using a power-sum method, we obtain:

$$\frac{\partial \boldsymbol{U}^n}{\partial t} = \frac{\pi}{T}\sum_{m=0}^{N-1} \boldsymbol{U}^m (-1)^{n-m} \csc\left(\frac{\pi(n-m)}{N}\right). \tag{16}$$

Substituting the above equation into the momentum equation (2) and assuming 3 time instances ($N = 3$), we obtain:

$$-\frac{\pi}{T}\csc(\pi/3)\boldsymbol{U}^{n-1} + \frac{\pi}{T}\csc(\pi/3)\boldsymbol{U}^{n+1} + \overline{\boldsymbol{R}}_{\boldsymbol{U}}^{n} = 0, \tag{17}$$

where $\overline{\boldsymbol{R}}_{\boldsymbol{U}}^{n}$ is the residual equation for the steady-state problem, i.e., the discretized momentum equation (5) without the $\boldsymbol{U}_p/\Delta t$ term. Similarly, we can compute the time-derivative term for the turbulence equation (3) as:

$$-\frac{\pi}{T}\csc(\pi/3)\tilde{\nu}^{n-1} + \frac{\pi}{T}\csc(\pi/3)\tilde{\nu}^{n+1} + \overline{\boldsymbol{R}}_{\tilde{\nu}}^{n} = 0. \tag{18}$$

Note that for a three-time-instance formulation, the current states depend on the states in the previous ($n-1$) and next ($n+1$) time instances, as well as the steady-state residual $\overline{\boldsymbol{R}}$ from the current time instance. In contrast, for time-accurate methods that use a backward differentiation scheme (e.g., the Euler method), the current states depend on only the states in the previous ($n-1$) time instance. Because the states for all the time instances are coupled, the time-spectral method can use only a handful of time instances and choose a large time interval of $\Delta t = T/N$.

The block-matrix representation of time-spectral formulation for the PISO algorithm is as follows, assuming three time instances:

$$\begin{bmatrix} \overline{\boldsymbol{A}}^{n-1} & \boldsymbol{D}^+ & \boldsymbol{D}^- \\ \boldsymbol{D}^- & \overline{\boldsymbol{A}}^{n} & \boldsymbol{D}^+ \\ \boldsymbol{D}^+ & \boldsymbol{D}^- & \overline{\boldsymbol{A}}^{n+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{n-1} \\ \boldsymbol{w}^{n} \\ \boldsymbol{w}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}^{n-1} \\ \boldsymbol{b}^{n} \\ \boldsymbol{b}^{n+1} \end{bmatrix}, \tag{19}$$

where $\boldsymbol{w} = [\boldsymbol{U}^T, p, \phi, \tilde{\nu}]^T \in \mathbb{R}^{n_w}$ is the state variable vector with n_w being the number of discrete state variables for one instance, and $\boldsymbol{b} \in \mathbb{R}^{n_w}$ is the source terms from the discrete governing equations, $\overline{\boldsymbol{A}}^{n} \in \mathbb{R}^{n_w \times n_w}$ denotes the steady-state finite-volume discretization matrix for the nth time instance:

$$\overline{\boldsymbol{A}}^{n} = \begin{bmatrix} \overline{\boldsymbol{A}}_{\boldsymbol{U}}^{n} \\ \overline{\boldsymbol{A}}_{p}^{n} \\ \overline{\boldsymbol{A}}_{\phi}^{n} \\ \overline{\boldsymbol{A}}_{\tilde{\nu}}^{n} \end{bmatrix}, \tag{20}$$

where $\overline{\boldsymbol{A}}_{\boldsymbol{U}}^{n} \in \mathbb{R}^{n_u \times n_w}$ is steady-state finite-volume discretization matrix for $\boldsymbol{U}$ with n_u being the number of discrete $\boldsymbol{U}$, similarly for other block matrices ($n_w = n_u + n_p + n_\phi + n_{\tilde{\nu}}$). $\boldsymbol{D}^+ \in \mathbb{R}^{n_w \times n_w}$ and $\boldsymbol{D}^- \in \mathbb{R}^{n_w \times n_w}$ denote the time-spectral time-derivative matrices for the next and previous time instances, respectively. For example,

$$\boldsymbol{D}^+ = \begin{bmatrix} \boldsymbol{D}_{\boldsymbol{U}}^+ & & & \\ & \boldsymbol{D}_p^+ & & \\ & & \boldsymbol{D}_\phi^+ & \\ & & & \boldsymbol{D}_{\tilde{\nu}}^+ \end{bmatrix}, \tag{21}$$

in which

$$\boldsymbol{D}_{\boldsymbol{U}}^{+} = -\boldsymbol{D}_{\boldsymbol{U}}^{-} = \begin{bmatrix} \tilde{d} & & \\ & \ddots & \\ & & \tilde{d} \end{bmatrix} \in \mathbb{R}^{n_u \times n_u} \tag{22}$$

are block matrices containing the contribution of time-spectral derivatives for $\boldsymbol{U}$, where $\tilde{d} = \frac{\pi}{T}\csc(\frac{\pi}{3})$ is the coefficient for the time derivatives. $\boldsymbol{D}_{\tilde{\nu}}$ has a similar pattern as $\boldsymbol{D}_{\boldsymbol{U}}$.

The $\boldsymbol{D}$ matrices for $\boldsymbol{U}$ and $\tilde{\nu}$ have only diagonal elements because the time-spectral time-derivative contains the contribution from other time instances at the residual cell (see Eqs. 17 and 18). However, for p and ϕ, their residuals depend on variables from other time instances at the residual cell, as well as at its surrounding cells. This is because the Rhie–Chow interpolation is used in the p and ϕ equations (10) and (11), such that the velocity at the cell face ($\boldsymbol{U}_f$) is interpolated from the momentum predictor (9). Therefore, $\boldsymbol{D}_p$ and $\boldsymbol{D}_\phi$ matrices contain diagonal and off-diagonal elements. In contrast to $\boldsymbol{D}_{\boldsymbol{U}}$ and $\boldsymbol{D}_{\tilde{\nu}}$, the elements in $\boldsymbol{D}_p$ and $\boldsymbol{D}_\phi$ are not constants (for given T and N) because they depend on mesh topology.

To solve for Eq. 19, we can assemble the whole left-hand side matrix and use a nonlinear equation solution strategy (e.g, the Newton method). However, this strategy requires significant modification for the original time-accurate code structure, and its implementation can be time-consuming.

In this study, we use a fully-segregated, block Gauss–Seidel (FSBGS) method to solve the time-spectral equations. This approach was originally developed in Luder [12] to simulate unsteady flow for marine propellers, and here we extend its application for aerospace engineering design problems.

To enable the FSBGS method, we rearrange Eq. 19 as:

$$\begin{bmatrix} \overline{\boldsymbol{A}}^{n-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{D}^{-} & \overline{\boldsymbol{A}}^{n} & \boldsymbol{0} \\ \boldsymbol{D}^{+} & \boldsymbol{D}^{-} & \overline{\boldsymbol{A}}^{n+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{n-1} \\ \boldsymbol{w}^{n} \\ \boldsymbol{w}^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}^{n-1} \\ \boldsymbol{b}^{n} \\ \boldsymbol{b}^{n+1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{0} & \boldsymbol{D}^{+} & \boldsymbol{D}^{-} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{D}^{+} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^{n-1} \\ \boldsymbol{w}^{n} \\ \boldsymbol{w}^{n+1} \end{bmatrix}. \tag{23}$$

With this treatment, we can solve for state variables at each time instance, while using the most up-to-date state variables for all other time instances. In addition, at each time instance, the $\boldsymbol{U}$, p, ϕ, and $\tilde{\nu}$ equations are solved in a segregated manner, following a similar PISO solution process detailed in Algorithm 1. The FSBGS algorithm is summarized in Algorithm 2. Compared with the original PISO algorithm, the FSBGS method has the following major differences:

1) The momentum predictor step uses time-derivative computed based on the time-spectral method, as opposed to the backward differentiation scheme. As a result, the time-derivative terms differ, as can be seen by comparing line 5 in Algorithm 1 and line 4 in Algorithm 2. Similarly for the $\boldsymbol{H}(\boldsymbol{U})$ term.
2) Because we no longer require the velocity and pressure fully satisfy the continuity and momentum equations at each time step, the corrector loop in the original PISO algorithm (line 4, Algorithm 1) is performed only once, instead of M times.
3) We require the residuals for the solution of $\boldsymbol{U}$, p, and $\tilde{\nu}$ linear equations to drop only one order of magnitude (relative tolerance to be 10^{-1}) at each time step, as opposed to a tight absolute tolerance (10^{-8}) used in the original PISO algorithm.
4) We use implicit and explicit relaxations ($\alpha = 0.05$, $\beta = 0.15$) to ensure the numerical stability of the FSBGS method (see lines 4, 7, and 10 in Algorithm 2).

In summary, Algorithms 1 and 2 use the same solution strategy. To apply the time-spectral method for the original PISO algorithm, the major modification is to add extra terms and under-relaxations, while keeping the original code structure unchanged. This feature greatly facilitates the implementation of the time-spectral method for the original CFD solver. Using this strategy, we develop a new flow solver called pisoTSDAFoam that uses the time-spectral method to compute incompressible unsteady flow with OpenFOAM. This solver can also compute derivatives for gradient-based optimization, as elaborated on in the next subsection.

One drawback of the FSBGS approach is that we solve for the state variables at each time instance in a segregated manner, which converges slower than a fully coupled approach. To accelerate the convergence, we can use a fully-coupled strategy such as the Newton and approximated Newton methods as elaborated on in [58].

Algorithm 2 Fully-segregated, block Gauss–Seidel algorithm to solve time-spectral incompressible NS equations

```
Input: $T$, $N$, $\boldsymbol{w}^n_{\text{initial}}$ for $n \in \{1, 2, \ldots, N\}$    ▹ Time period, number of time instances, and initial state variables
Output: $\boldsymbol{w}^n_{\text{converged}}$ for $n \in \{1, 2, \ldots, N\}$    ▹ Converged state variables
 1: $\boldsymbol{w}^n \leftarrow [\boldsymbol{U}_0, p_0, \phi_0, \tilde{\nu}_0]$ for each $n \in \{1, 2, \ldots, N\}$    ▹ Assign initial state variables for all time instances
 2: $\Delta t = T/N$    ▹ Compute time interval
 3: for $t \in \{\Delta t, 2\Delta t, \ldots, \infty\}$ do    ▹ Main loop with time interval $\Delta t$
 4:     $\boldsymbol{U}^t \leftarrow \frac{a_P}{\alpha}\boldsymbol{U}^t_P + \sum a_N \boldsymbol{U}^t_N = \frac{1-\alpha}{\alpha} a_P \boldsymbol{U}^{t-\Delta t}_P + \boldsymbol{U}^{t+\Delta t}_P \tilde{d} - \boldsymbol{U}^{t-\Delta t}_P \tilde{d} - \nabla p^{t-\Delta t}$    ▹ Solve momentum equation (relaxed)
 5:     $H(\boldsymbol{U}) \leftarrow \frac{1-\alpha}{\alpha} a_P \boldsymbol{U}^{t-\Delta t}_P + \boldsymbol{U}^{t+\Delta t}_P \tilde{d} - \boldsymbol{U}^{t-\Delta t}_P \tilde{d} - \sum a_N \boldsymbol{U}^t_N$    ▹ Compute the $H(\boldsymbol{U})$ term
 6:     $p^t \leftarrow \nabla \cdot \left(\frac{1}{a_P}\nabla p\right) = \nabla \cdot \left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right).$    ▹ Solve pressure Poisson equation
 7:     $p^t \leftarrow p^t + \beta(p^t - p^{t-\Delta t})$    ▹ Explicitly relaxing $p$
 8:     $\phi^t \leftarrow \left[\left(\frac{\boldsymbol{H}(\boldsymbol{U})}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p^t)_f\right] \cdot \boldsymbol{S}_f$    ▹ Update face flux $\phi$
 9:     $\boldsymbol{U}^t \leftarrow \frac{1}{a_P}[\boldsymbol{H}(\boldsymbol{U}) - \nabla p^t]$    ▹ Correct velocity
10:     $\tilde{\nu}^t \leftarrow$ turbulence model    ▹ Solve turbulence equations (relaxed)
11:     if $t > t_{\max}$ then
12:         break    ▹ Maximal runtime is reach
```

C. Discrete adjoint method for a time-spectral solver

In this subsection, we derive the formulations for the time-spectral discrete adjoint approach. In the discrete adjoint, we assume that a discretized form of the governing equations is available through the flow solver, and that the design vector $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and the flow state variable vector $\boldsymbol{w} \in \mathbb{R}^{N_w}$ satisfy the discrete residual equations:

$$\boldsymbol{R}(\boldsymbol{x}, \boldsymbol{w}) = 0, \tag{24}$$

where n_x is the number of design variables, $\boldsymbol{R} \in \mathbb{R}^{N_w}$ is the residual vector with $N_w = N \cdot n_w$ being the total number of state variables for all time instances. Note that the state variable and residual vectors contain all the discrete variables at all time instances. For example:

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}^{n=1} \\ \boldsymbol{w}^{n=2} \\ \ldots \\ \boldsymbol{w}^{n=N} \end{bmatrix}, \tag{25}$$

where $\boldsymbol{w}^{n=1} = [\boldsymbol{U}^T_1, p_1, \phi_1, \tilde{\nu}_1, \boldsymbol{U}^T_2, p_2, \phi_2, \tilde{\nu}_2, ..., \boldsymbol{U}^T_{n_c}, p_{n_c}, \phi_{n_c}, \tilde{\nu}_{n_c}]^T_{n=1}$ is the state variable at the 1st time instance. The subscript denotes the cell number with n_c being the total number of cells. The size of $\boldsymbol{w}^{n=1}$ is n_w.

The functions of interest are then functions of both the design variables and the flow variables, that is,

$$f = f(\boldsymbol{x}, \boldsymbol{w}). \tag{26}$$

f typically depends on the state variables at all time instances (e.g., the averaged drag), therefore, here $\boldsymbol{w}$ also contains all variables at all time instances.

In general, we have multiple functions of interest (the objective and multiple design constraints), but in the following derivations, we consider f to be a scalar without loss of generality. As we will see later, each additional function requires the solution of another adjoint system.

To obtain the total derivative $\mathrm{d}f/\,\mathrm{d}\boldsymbol{x}$, we apply the chain rule as follows:

$$\underbrace{\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}}}_{1\times n_x} = \underbrace{\frac{\partial f}{\partial \boldsymbol{x}}}_{1\times n_x} + \underbrace{\frac{\partial f}{\partial \boldsymbol{w}}}_{1\times N_w} \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{N_w\times n_x}, \tag{27}$$

where the partial derivatives $\partial f/\partial \boldsymbol{x}$ and $\partial f/\partial \boldsymbol{w}$ are relatively cheap to evaluate because they only involve explicit computations. The total derivative $\mathrm{d}\boldsymbol{w}/\,\mathrm{d}\boldsymbol{x}$ matrix, on the other hand, is expensive, because $\boldsymbol{w}$ and $\boldsymbol{x}$ are implicitly determined by the residual equations $\boldsymbol{R}(\boldsymbol{w}, \boldsymbol{x}) = 0$.

To solve for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$, we can apply the chain rule for $\boldsymbol{R}$. We then use the fact that the governing equations should always hold, independent of the values of design variables $\boldsymbol{x}$. Therefore, the total derivative $\mathrm{d}\boldsymbol{R}/\mathrm{d}\boldsymbol{x}$ must be zero:

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\boldsymbol{x}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}} = 0. \tag{28}$$

Rearranging the above equation, we get the linear system

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}}_{N_w\times N_w}\underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{x}}}_{N_w\times n_x} = -\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{N_w\times n_x}. \tag{29}$$

We can then substitute the solution for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{x}$ from Eq. (29) into Eq. (27) to get

$$\underbrace{\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}}}_{1\times n_x} = \underbrace{\frac{\partial f}{\partial \boldsymbol{x}}}_{1\times n_x} - \overbrace{\underbrace{\frac{\partial f}{\partial \boldsymbol{w}}}_{1\times N_w}\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^{-1}}_{N_w\times N_w}}^{\boldsymbol{\psi}^T}\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}}_{N_w\times n_x}. \tag{30}$$

Now we can transpose the Jacobian and solve with $[\partial f/\partial \boldsymbol{w}]^T$ as the right-hand side, which yields the *adjoint equations*,

$$\underbrace{\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{w}}^T}_{N_w\times N_w}\underbrace{\boldsymbol{\psi}}_{N_w\times 1} = \underbrace{\frac{\partial f}{\partial \boldsymbol{w}}^T}_{N_w\times 1}, \tag{31}$$

where $\boldsymbol{\psi}$ is the *adjoint vector*. Then, we can compute the total derivative by substituting the adjoint vector into Eq. (30):

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{x}} - \boldsymbol{\psi}^T\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}. \tag{32}$$

For each function of interest, we need to solve the adjoint equations only once, because the design variable is not explicitly present in Eq. (31). Therefore, its computational cost is independent of the number of design variables, but proportional to the number of objective functions. This approach is also known as the *adjoint method* and is advantageous for aerodynamic design, because we typically have only a few functions of interest but may use several hundred design variables.

The block matrix form of the adjoint linear equations (31) for a three-time-instance system is as follows:

$$\begin{bmatrix} [\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{n=1} & [\boldsymbol{D}^-]^T & [\boldsymbol{D}^+]^T \\ [\boldsymbol{D}^+]^T & [\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{n=2} & [\boldsymbol{D}^-]^T \\ [\boldsymbol{D}^-]^T & [\boldsymbol{D}^+]^T & [\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{n=3} \end{bmatrix}\begin{bmatrix}\boldsymbol{\psi}_{n=1}\\ \boldsymbol{\psi}_{n=2}\\ \boldsymbol{\psi}_{n=3}\end{bmatrix} = \begin{bmatrix}[\partial f/\boldsymbol{w}]^T_{n=1}\\ [\partial f/\boldsymbol{w}]^T_{n=2}\\ [\partial f/\boldsymbol{w}]^T_{n=3}\end{bmatrix}, \tag{33}$$

where $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{n=1} \in \mathbb{R}^{n_w\times n_w}$ denotes the transpose of state Jacobian for time instance $n = 1$, and similarly for the other terms.

To summarize, a discrete adjoint consists of the following four major steps:

1) Compute $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T$ and $[\partial f/\partial \boldsymbol{w}]^T$.
2) Solve the adjoint equation $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T\boldsymbol{\psi} = [\partial f/\partial \boldsymbol{w}]^T$ to obtain $\boldsymbol{\psi}$.
3) Compute $\partial \boldsymbol{R}/\partial \boldsymbol{x}$ and $\partial f/\partial \boldsymbol{x}$.
4) Compute the total derivative $\mathrm{d}f/\mathrm{d}\boldsymbol{x} = \partial f/\partial \boldsymbol{x} - \boldsymbol{\psi}^T[\partial \boldsymbol{R}/\partial \boldsymbol{x}]$.

In this study, we build the time-spectral adjoint solver pisoTSDAFoam based on an open-source, object-oriented adjoint framework DAFoam [‡]. DAFoam provides high-level interfaces that allow developers to implement the discrete adjoint method for any steady-state primal solvers in OpenFOAM [59, 60]. In this work, we extend DAFoam's capability to implement the discrete adjoint method for time-spectral flow solvers.

According to Eq. 33, the left-hand side matrix consists of the steady-state Jacobians for the diagonal blocks and the time-spectral derivative matrices for the off-diagonal blocks. In light of this observation, we reuse the DAFoam's existing partial derivative computational functions to compute $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T$ and $[\partial f/\partial \boldsymbol{w}]^T$ for each time instance. This

[‡]`https://github.com/mdolab/dafoam`

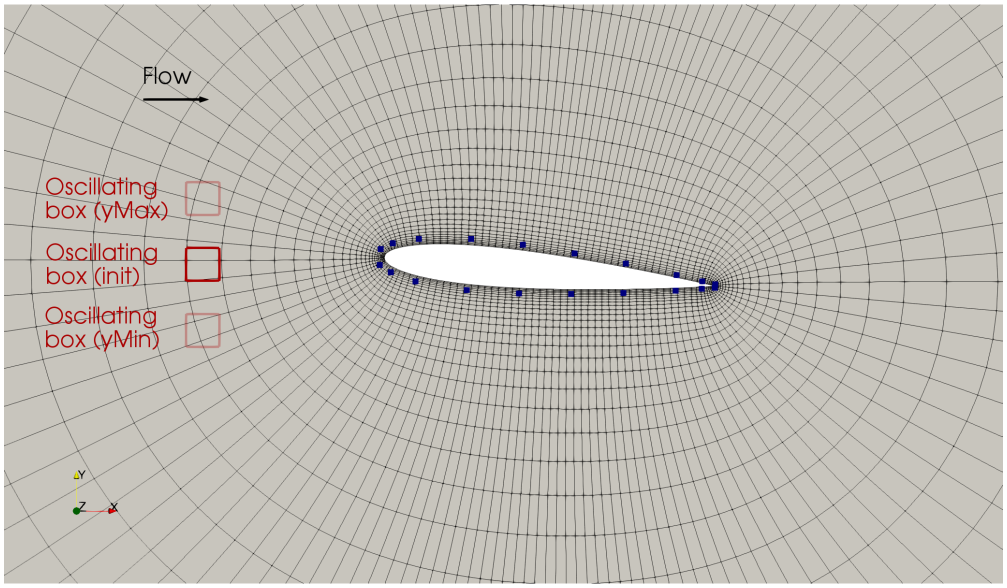

Fig. 1 Structural mesh for the NACA0012 airfoil. The blue squares are the FFD control points to morph the airfoil shape. The red hollow squares are the oscillating box to generate wakes and the solid red square denotes the initial position and the transparent squares represent the extreme positions of its oscillation.

is done through the coloring-accelerated finite difference Jacobian (FD Jacobian) approach detailed in [53, 59, 60]. For the off-diagonal $\boldsymbol{D}$ matrices, we manually assign the values for $\boldsymbol{D}_{\boldsymbol{U}}$ and $\boldsymbol{D}_{\tilde{\nu}}$ (Eq. 22). To facilitate the adjoint implementation, we ignore the $\boldsymbol{D}_p$ and $\boldsymbol{D}_\phi$ blocks in this paper.

Finally, we use the PETSc [61] library to solve the time-spectral adjoint equations (33). We use the generalized minimal residual (GMRES) iterative linear equation solver. We use a nested preconditioning strategy with the additive Schwartz method as the global preconditioner and the incomplete lower and upper (ILU) factorization approach with zero fill-in for the local preconditioning. To improve convergence, we construct the preconditioner matrix $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{\text{PC}}$ by approximating the residuals and their linearizations [53, 59]. $[\partial \boldsymbol{R}/\partial \boldsymbol{w}]^T_{\text{PC}}$ has the exact same block matrix structure as in Eq. 33. This strategy is effective for solving the adjoint equation, as reported in He et al. [59, 62].

D. MACH: A high-fidelity gradient-based optimization framework

In this paper, we use MACH [§], an open-source, high-fidelity gradient-based optimization framework, to perform aerodynamic shape optimization. The MACH framework consists of the following modules:

- **Flow simulation**. MACH supports multiple flow solvers (ADflow [53, 58] and OpenFOAM) for simulating turbulent flows. In this study, the time-spectral flow solver pisoTSDAFoam, described in Secs. II.A and II.B is used to simulate unsteady, incompressible turbulent flow.
- **Adjoint derivative computation**. MACH supports multiple adjoint solvers (ADflow [53, 58] and DAFoam [59, 60]) for efficient derivative computation. The time-spectral adjoint solver pisoTSDAFoam described in Sec. II.C is used.
- **Geometry parameterization**. MACH uses the free-form deformation (FFD) approach [63] to parameterize the design surface geometry through the `pyGeo` package. [¶] This method embeds the design surface into a tri-variate B-spline volume. We then manipulate the surface geometry by moving the FFD points at the surface of that volume. We also use `pyGeo` to compute the values and derivatives of the geometric constraints ($\boldsymbol{c}$ and $\mathrm{d}\boldsymbol{c}/\mathrm{d}\boldsymbol{x}$, respectively). An example of FFD control points can be seen in Fig. 1.
- **Volume mesh deformation**. MACH uses the analytic inverse distance algorithm, similar to that described by Luke et al. [64], to deform the volume mesh through the `IDWarp` package. [‖] The advantage of this approach is that it applied to both structured and unstructured meshes. Moreover, this approach better preserves mesh

[§]`https://github.com/mdolab/mach-aero`
[¶]`https://github.com/mdolab/pygeo`
[‖]`https://github.com/mdolab/idwarp`

Table 1 Unsteady aerodynamic optimization setup for the NACA0012 airfoil, which has 21 design variables and 23 constraints.

	Function or variable	Description	Quantity
minimize	$\overline{C}_D$	Averaged Drag coefficient	
with respect to	Δy	Displacement of FFD points in the vertical direction	20
	α	Angle of attack	1
		Total design variables	**21**
subject to	$\overline{C}_L$=0.5	Averaged lift-coefficient constraint	1
	$t \geq 0.8 t_{\text{baseline}}$	Minimum-thickness constraint	20
	$\Delta z_{\text{LE}}^{\text{upper}} = -\Delta z_{\text{LE}}^{\text{lower}}$	Fixed leading-edge constraint	1
	$\Delta z_{\text{TE}}^{\text{upper}} = -\Delta z_{\text{TE}}^{\text{lower}}$	Fixed trailing-edge constraint	1
	-0.1 m $< \Delta z <$ 0.1 m	Design variable bounds	
		Total constraints	**23**

orthogonality in the boundary layer, compared with the methods based on the radial basis function [64].

- **Optimizer**. We use `pyOptSparse` [**], an extension of pyOpt [65], for setting constrained nonlinear optimization problems. The SNOPT [66] optimizer is used, which adopts the sequential quadratic programming (SQP) algorithm for optimization. The constraints are handled by formulating them into the Lagrangian function, and the Hessian of the Lagrangian is approximated by using a Broyden–Fletcher–Goldfarb–Shanno update [67].

III. Results and Discussion

A. Flow and adjoint configurations

We use the NACA0012 airfoil as our benchmark geometry. The chord (c) is set to be 1 m, and the span is set to be 0.1 m with one cell in the spanwise (z) direction. The flow condition is incompressible with Reynolds number 10^6 and Mach number 0.03 (U_∞=10 m s^{-1}). We generate a coarse structured mesh with 5 280 cells using the pyHyp[††] package and the computational domain extends 20 chords from the surface (Fig. 1). The averaged y^+ is 80.5, so the wall function is used.

To generate periodic wakes, we set a box oscillating in the vertical (y) direction. The box is located at 0.5c upstream of the airfoil (Fig. 1). This is done by adding an oscillating source term S in the x (streamwise) component of the momentum equation (2). To ensure numerical stability, we smooth the source term using the hyperbolic tangent function:

$$S = \frac{1}{8} A_x S_x S_y S_z, \text{ with } S_x = F(x - x_C, L_x/2),\ S_y = F(y - y_C, L_y/2),\ S_z = F(z - z_C, L_z/2), \tag{34}$$

where A_x is the magnitude of the source term, x, y, and z are the cell-center coordinate of a finite volume cell, $L_x = 0.1c$, $L_y = 0.1c$, $L_z = 0.1c$ are the sizes of the box, and $x_C = -0.5c$, $y_C = 0$, $z_C = 0.05c$ are the initial centers of the box. The shape function F is defined as:

$$F(d, l) = \tanh[\beta(d + l)] - \tanh[\beta(d - l)], \tag{35}$$

where $\beta = 10$ is a coefficient that controls the smoothness of the S term. As mentioned before, the center of the box is oscillating in the y direction $y_C = B\sin(2\pi f t)$ with a time period ($T = 1/f = 0.5c/U_\infty = 0.05$ s) half of the flow-over time and and an amplitude $B = 0.2c$. The choice of A_x determines the strength of the wake, in this study, we set $A_x = 1.0$ such that the wake has a velocity magnitude that is 10% higher than the mainstream velocity.

All the simulations are run on a desktop workstation with one Intel i7-4770 CPU core (3.4 GHz) in serial. We run the time-accurate simulations using the OpenFOAM's built-in flow solver pisoFoam. The time step is set to be 1×10^{-4} s such that the CFL number is kept less than 1.0. The second order backward differentiation scheme is used for the time-derivative term. The second-order linear-upwind scheme [68] is used to differentiate the divergence terms,

[**] `https://github.com/mdolab/pyoptsparse`

[††] `https://github.com/mdolab/pyhyp`

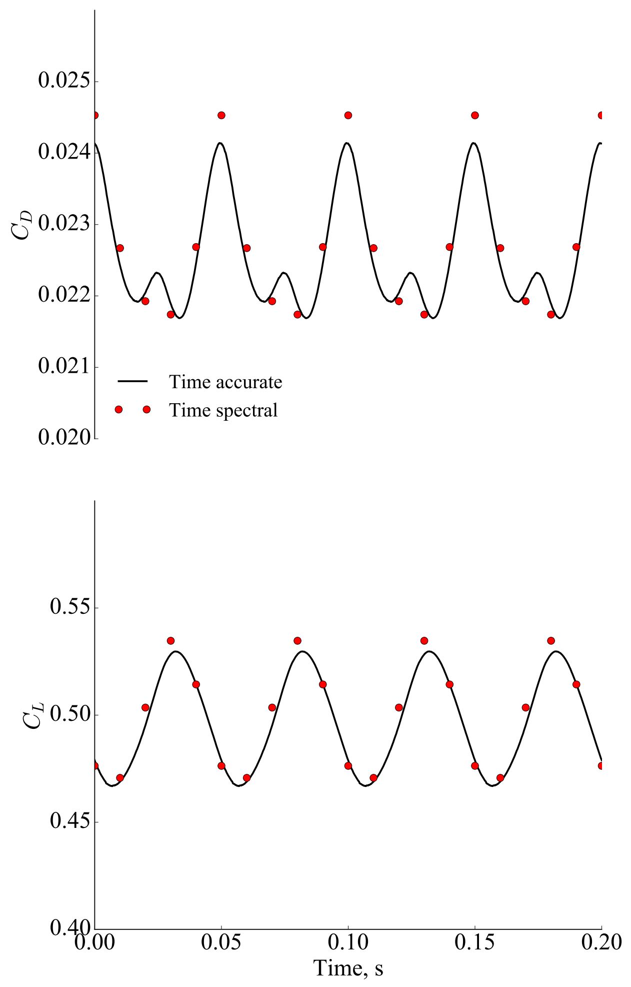

Fig. 2 The time-series of C_D and C_L predicted by the time-spectral method reasonably matches the ones simulated by the time-accurate method.

whereas the central differential scheme is selected for the diffusion terms. The simulations are run for 5 s (physical time) in total. For time-spectral simulations, we use five time instances and the time step is $\Delta t = T/N = 1 \times 10^{-2}$ s. The simulations are run for 400 s (physical time) in total. For the adjoint computation, we require the residual of the adjoint linear equations to drop five orders of magnitudes.

The optimization configuration is summarized in Table 1, the objective function is the averaged drag coefficient ($\overline{C_D}$) and the nominal flight condition is at averaged lift coefficient ($\overline{C_L}$) of 0.5. The design variables are the angle of attack (α) and 20 FFD points moving in the vertical (y) direction (Fig. 1). The angle of attack is changed by pitching the airfoil with respect to its quarter chord, as opposed to changing the direction of free stream velocity typically used in steady-flow optimization. This strategy ensures the free stream is parallel to the wakes (x direction).

To ensure a practical design, we require the airfoil thickness to be no less than 80% of its baseline value. This is done by monitoring the airfoil thickness at 20 sampling points along the chord. In addition, we use two linear constraints to fix the leading the trailing edges. The averaged lift coefficient is constrained to be equal to 0.5.

B. Flow and adjoint performance

To evaluate the performance of time-spectral flow solver, we first compare the time-series of C_D and C_L computed from the time-accurate and time-spectral solvers, as shown in Fig. 2. In the time-series of C_L computed by the time-accurate method, we observe a dominant frequency with a time period of 0.05 s, which is correlated to the prescribed frequency of the oscillating wake. The amplitude of the C_L oscillation is about 10% of its mean value. For

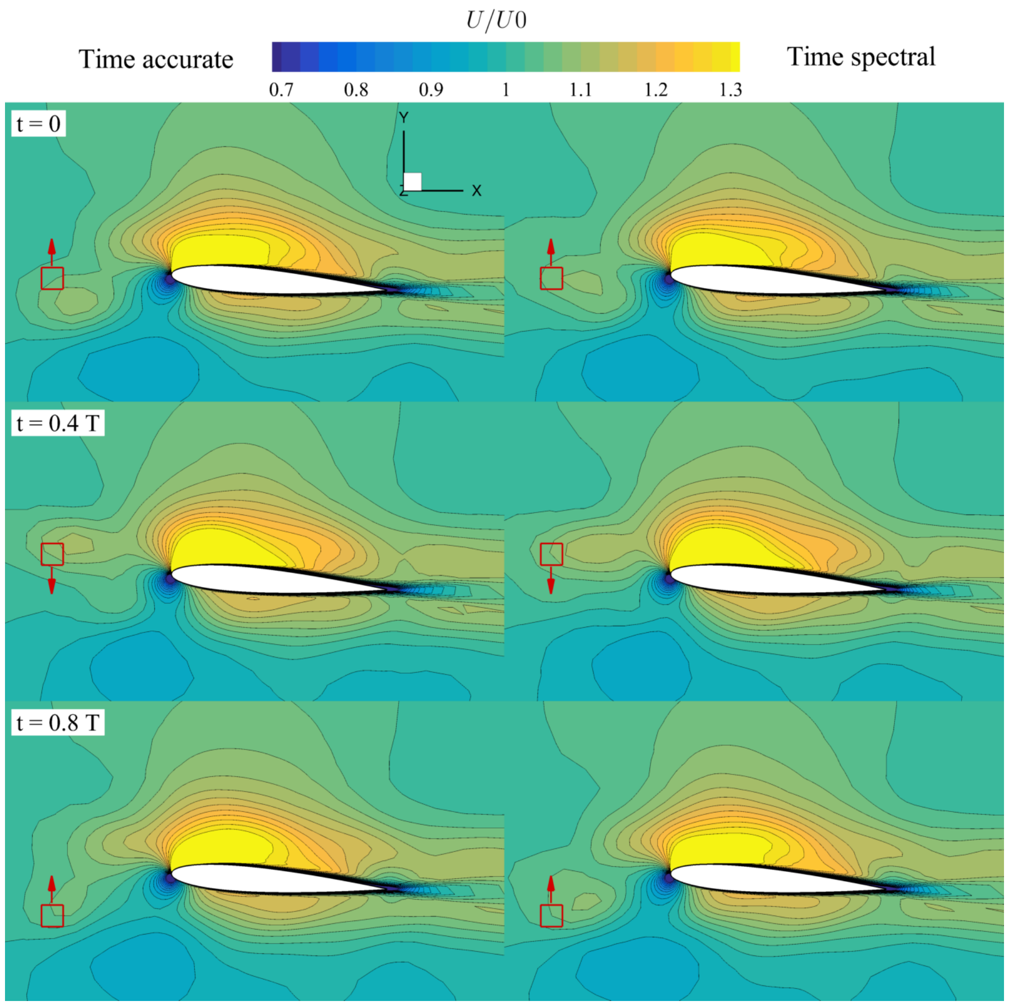

Fig. 3 The velocity contours predicted by the time-spectral method reasonably matches the ones simulated by the time-accurate method.

C_D, we observe a similar dominant frequency; however, in contrast to C_L, there are two amplitudes. The larger and smaller amplitudes are of 10% and 3% of the mean C_D value, respectively, and their phase has a shift of about 0.025 s (half time-period).

Generally, the C_D and C_L values predicted by the time-spectral method agree with the time-series computed with time-accurate simulations. To further confirm this, we compare the velocity contours between the time-accurate and time-spectral methods, as shown in Fig. 3. Again, the time-spectral simulations qualitatively capture the velocity distribution at each time instance. Since the time-spectral method provides more accurate time derivatives than the backward differentiation used in the time-accurate method, we expect the time-spectral simulation results are more accurate.

Table 2 shows the run time and peak memory usage for the time-accurate and time-spectral flow simulations, as well as the time-spectral adjoint computation. The time-spectral flow simulation is 40% faster than the time-accurate flow simulation. In addition, the time-spectral adjoint computation is faster than the flow simulation with an adjoint-flow run time ratio of 0.8. In terms of memory usage, the time-spectral flow simulation takes only 30% more memory than the time-accurate one. However, for the time-spectral adjoint, we need 8.32 GB memory, primarily because we assemble the whole Jacobian containing all the time instances (Eq. 19).

To verify the time-spectral adjoint accuracy, we compare the derivatives computed from the adjoint solver with the reference values computed directly from finite differencing. We use the derivatives from the top ten FFD points, as shown in Tables 3 and 4. Generally, the adjoint derivatives match the reference values. The value-weighted averaged

Table 2 Run time and memory usage comparison between the time-accurate flow simulation, time-spectral flow simulation, and time-spectral adjoint computation.

	Time-accurate flow	Time-spectral flow	Time-spectral adjoint
Run time	1975 s	1195 s	911 s
Peak memory	0.10 GB	0.13 GB	8.32 GB

Table 3 The total derivatives computed by the time-spectral adjoint method match the references with the value-weighted averaged errors being 3.5%.

FFD #	Reference	Adjoint
1	0.01393287	$0.01\underline{406124}$
2	0.00921630	$0.00\underline{864564}$
3	0.00321114	$0.00\underline{277542}$
4	0.00605408	$0.00\underline{587909}$
5	0.00685826	$0.006\underline{74317}$
6	0.00789065	$0.007\underline{73631}$
7	0.01299993	$0.01\underline{264514}$
8	0.02067202	$0.020\underline{00364}$
9	0.02699674	$0.026\underline{04181}$
10	−0.06752914	$-0.06\underline{501389}$

Table 4 The total derivatives computed by the time-spectral adjoint method match the references with the value-weighted averaged errors being 0.8%.

FFD #	Reference	Adjoint
1	−0.01639367	$-0.01\underline{739969}$
2	0.20743961	$0.2\underline{1038701}$
3	0.30894715	$0.3\underline{1176138}$
4	0.34930939	$0.3\underline{5117792}$
5	0.37164342	$0.37\underline{430646}$
6	0.50600053	$0.5\underline{1016325}$
7	0.95271738	$0.9\underline{6006269}$
8	1.63748314	$1.6\underline{5024483}$
9	2.21544151	$2.2\underline{3413826}$
10	−6.98295271	$-\underline{7.03555221}$

errors are 3.5% and 0.8% for C_D and C_L, respectively. The errors in the time-spectral adjoint are primarily caused by ignoring the off-diagonal blocks of p and ϕ in the adjoint equations, as mentioned in Sec. II.C.

C. Aerodynamic optimization of NACA0012 under periodic wakes

The convergence history of C_D and optimality is shown in Fig. 4. The optimization converges in 20 iterations, and C_D is reduced by 10.5%. Figure 5 compares the baseline and optimized airfoil profiles. Compared with the baseline NACA0012 profile, the airfoil thickness is reduced. Moreover, the optimized design uses a camber and a smaller angle of attack to reduce the aerodynamic drag.

A comparison of pressure profiles between the baseline and optimized designs is shown in Fig. 6. The pressure profiles show a smoother loading in the optimized design, as opposed to the high-loading near the leading edges observed in the baseline design. This is achieved by reducing the angle of attack and changing the airfoil profiles. This

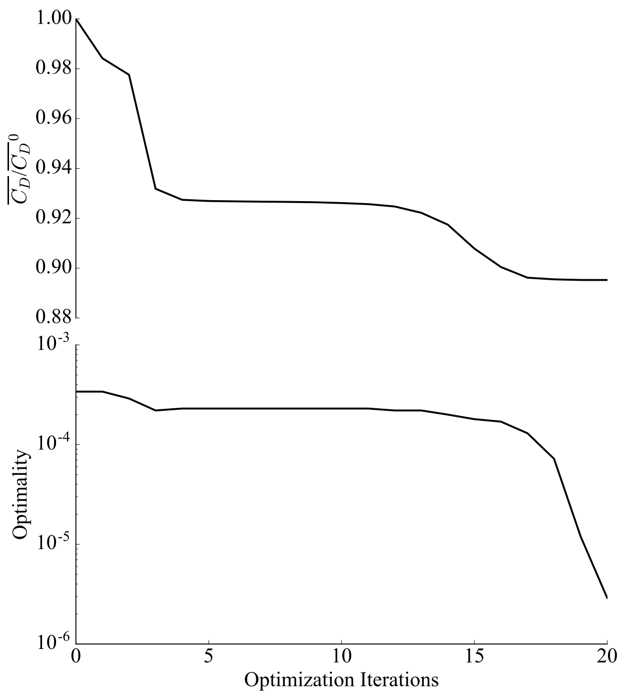

Fig. 4 Optimization history for C_D and optimality; both of them converge well.

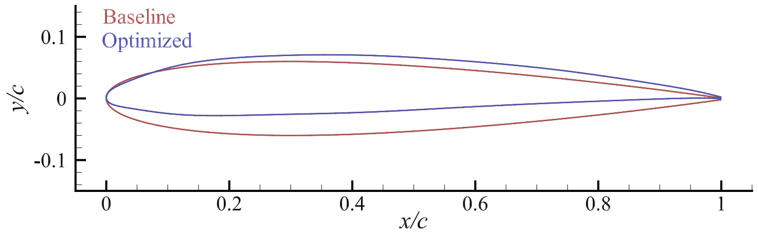

Fig. 5 Comparison of airfoil profile between the baseline and optimized designs, both of which are rotated to zero angle of attack.

feature is also observed in the velocity contour comparison in Fig. 7, where a smoother flow acceleration is observed in the optimized design.

Finally, we verify our optimization results by re-running the baseline and optimized designs using the time-accurate method, as shown in Fig. 8. The time-accurate simulations reproduce the drag reduction predicted by the time-spectral optimization. In addition, the lift constraint is satisfied.

IV. Conclusion

In this paper, we develop an efficient time-spectral method for aerodynamic analysis and design under periodic wakes. We propose a fully-segregated, block Gauss–Siedel method to solve the time-spectral flow equations, which requires low memory overhead and minimal modification to the original code structure. We then implement the discrete adjoint method to the time-spectral solver, and the diagonal block matrices in the adjoint linear equations are assembled by using an object-oriented adjoint framework DAFoam.

We use the NACA0012 airfoil as our baseline geometry. We add an oscillating source term in the momentum equation to mimic the periodic wake, originated from the upstream of the airfoil. The simulated time-series and velocity contours from the time-spectral method reasonably match the ones simulated by the time-accurate method. The time-spectral flow simulation is faster than the time-accurate simulation. In addition, the time-spectral adjoint computation is faster than the time-spectral flow simulation. However, the time-spectral adjoint computation requires a

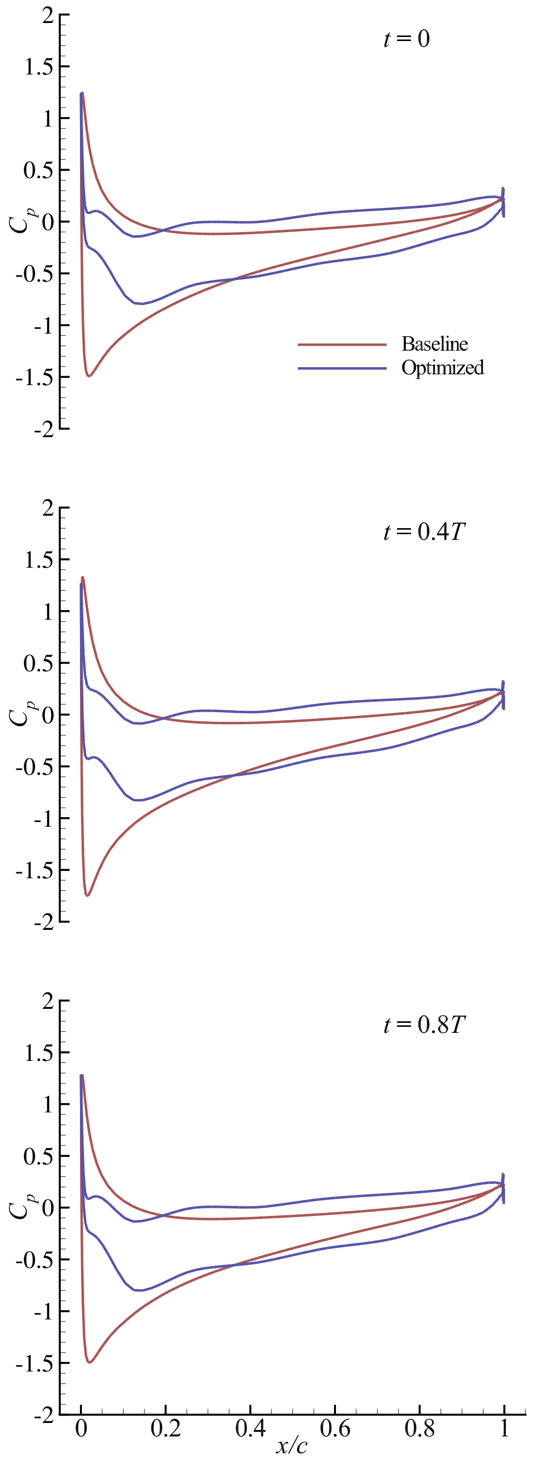

Fig. 6 Comparison of pressure profiles between the baseline and optimized design..

large memory overhead, because we explicitly assemble and store the Jacobian matrices for all the time instances.

Finally, we incorporate the implemented time-spectral flow and adjoint solvers into a gradient-based optimization framework MACH. We then perform an aerodynamic shape optimization under periodic wakes. We obtain a 10.5% drag reduction, which is primarily achieved by using a cambered airfoil shape and a smaller angle of attack. The optimization

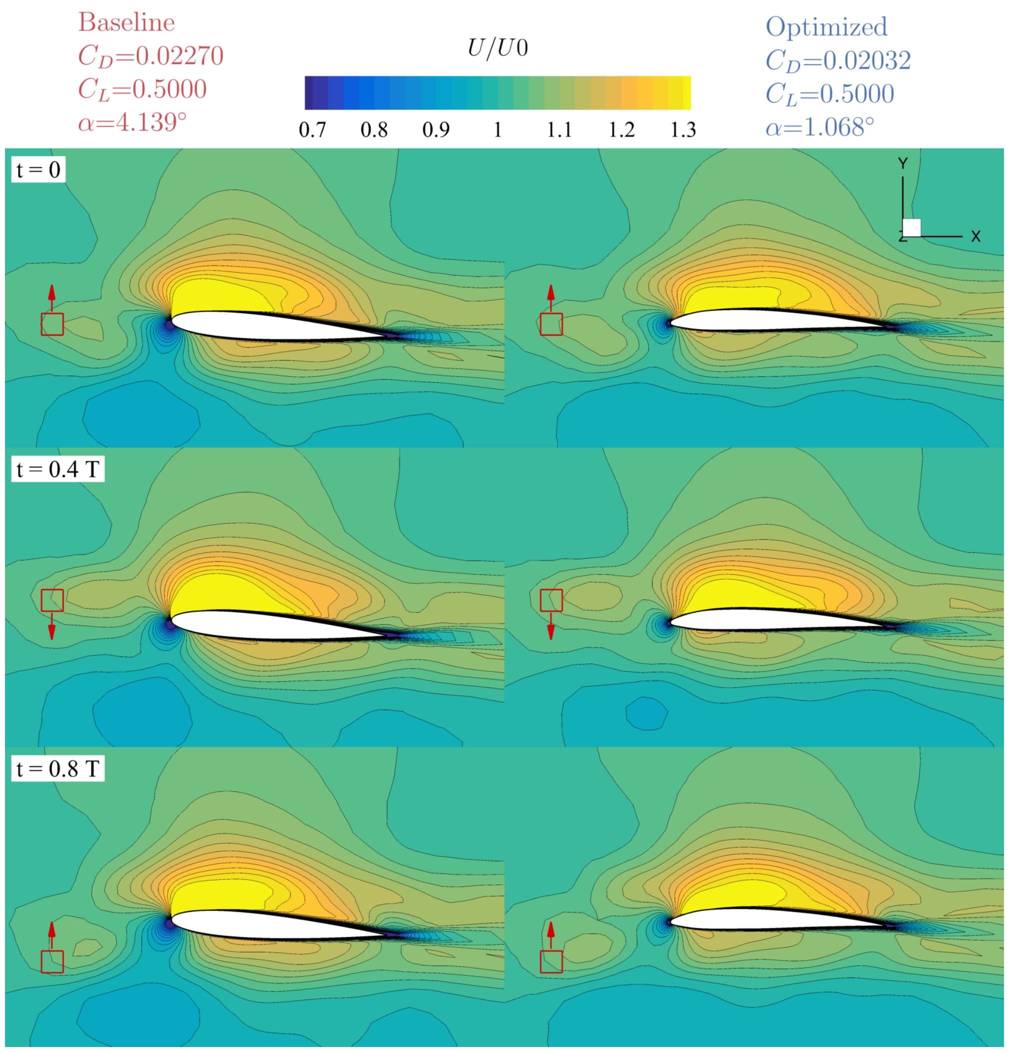

Fig. 7 Comparison of velocity contours between the baseline and optimized design. Drag reduces by 10.5% and the lift constraint is satisfied.

results are verified by re-running the baseline and optimized designs using the time-accurate method.

The proposed method has the potential to reduce the computational cost for analyzing and designing engineering systems that involve unsteady wakes, e.g., propeller-wing interaction for electric aircraft, blade-tower interaction for wind turbines, and hull-propeller interaction for marine vessels.

Acknowledgments

The computations were done in the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation Grant number ACI-1548562.

References

[1] Kim, H. D., Perry, A. T., and Ansell, P. J., "A review of distributed electric propulsion concepts for air vehicle technology," *2018 AIAA/IEEE Electric Aircraft Technologies Symposium (EATS)*, IEEE, 2018, pp. 1–21.

[2] Brelje, B., and Martins, J. R. R. A., "Electric, Hybrid, and Turboelectric Fixed-Wing Aircraft: A Review of Concepts, Models, and Design Approaches," *Progress in Aerospace Sciences*, Vol. 104, 2019, pp. 1–19. doi:10.1016/j.paerosci.2018.06.004.

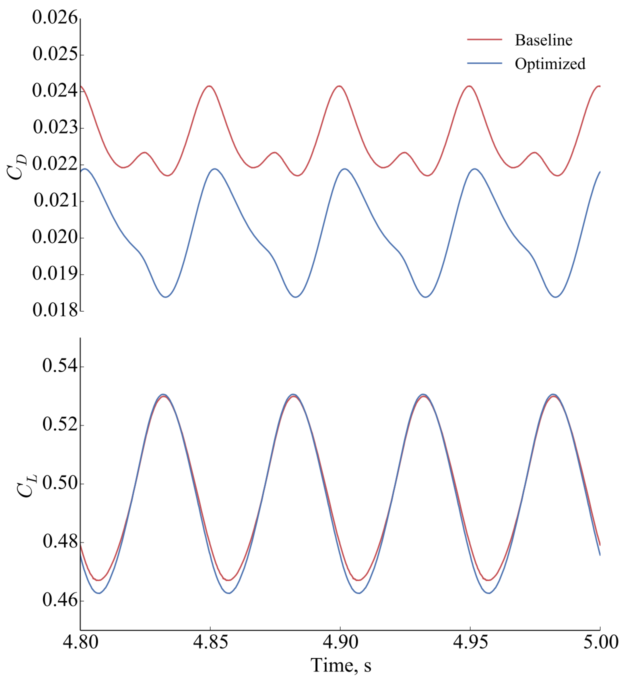

Fig. 8 Comparison of C_D and C_L time-series between the baseline and optimized design. The time-series are computed by the time-accurate method.

[3] Arnone, A., and Pacciani, R., "Rotor-stator interaction analysis using the Navier–Stokes equations and a multigrid method," *Journal of Turbomachinery*, Vol. 118, 1996, pp. 679–689.

[4] He, L., Chen, T., Wells, R., Li, Y., and Ning, W., "Analysis of rotor-rotor and stator-stator interferences in multi-stage turbomachines," *Journal of Turbomachery*, Vol. 124, No. 4, 2002, pp. 564–571.

[5] McMullen, M., Jameson, A., and Alonso, J., "Application of a non-linear frequency domain solver to the Euler and Navier-Stokes equations," *40th AIAA Aerospace Sciences Meeting & Exhibit*, 2002, p. 120.

[6] Hall, K. C., Thomas, J. P., and Clark, W. S., "Computation of Unsteady Nonlinear Flows in Cascades Using a Harmonic Balance Technique," *AIAA Journal*, Vol. 40, No. 5, 2002, pp. 879–886. doi:10.2514/2.1754, URL `http://arc.aiaa.org/doi/abs/10.2514/2.1754`.

[7] Cvijetic, G., Jasak, H., and Vukcevic, V., "Finite volume implementation of the harmonic balance method for periodic non-linear flows," *54th AIAA aerospace sciences meeting*, 2016, p. 0070.

[8] McMullen, M., "The Application of Non-Linear Frequency Domain Methods to the Euler and Navier-Stokes Equations," Ph.D. thesis, Stanford University, Mar. 2003. URL `citeseer.ist.psu.edu/mcmullen03application.html`.

[9] Gopinath, A. K., and Jameson, A., "Time Spectral Method for Periodic Unsteady Computations over Two- Three Dimensional Bodies," *AIAA Paper* 2005-1220, 2005.

[10] Ekici, K., and Hall, K. C., "Harmonic Balance Analysis of Limit Cycle Oscillations in Turbomachinery," *AIAA Journal*, Vol. 49, No. 7, 2011, pp. 1478–1487. doi:10.2514/1.j050858.

[11] Mader, C. A., and Martins, J. R. R. A., "Derivatives for Time-Spectral Computational Fluid Dynamics Using an Automatic Differentiation Adjoint," *AIAA Journal*, Vol. 50, No. 12, 2012, pp. 2809–2819. doi:10.2514/1.J051658.

[12] Luder, A. J., "A Block-Jacobi Time-Spectral Method For Incompressible Flow," Ph.D. thesis, University of Michigan, Ann Arbor, MI, 2013.

[13] Mundis, N., and Mavriplis, D., "Quasi-periodic Time Spectral Method for Aeroelastic Flutter Analysis," 2013. URL `http://dx.doi.org/10.2514/6.2013-638`.

[14] Yao, W., and Marques, S., "Prediction of Transonic Limit-Cycle Oscillations Using an Aeroelastic Harmonic Balance Method," *AIAA Journal*, 2015. doi:10.2514/1.J053565.

[15] Ma, C., Su, X., and Yuan, X., "An efficient unsteady adjoint optimization system for multistage turbomachinery," *Journal of Turbomachinery*, Vol. 139, No. 1, 2017, p. 011003.

[16] Rubino, A., Pini, M., Colonna, P., Albring, T., Nimmagadda, S., Economon, T., and Alonso, J., "Adjoint-based fluid dynamic design optimization in quasi-periodic unsteady flow problems using a harmonic balance method," *Journal of Computational Physics*, Vol. 372, 2018, pp. 220–235.

[17] He, S., Jonsson, E., Mader, C. A., and Martins, J. R. R. A., "A Coupled Newton–Krylov Time Spectral Solver for Flutter Prediction," *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, American Institute of Aeronautics and Astronautics, Kissimmee, FL, 2018. doi:10.2514/6.2018-2149.

[18] Puente, R., Corral, R., and Parra, J., "Comparison between aerodynamic designs obtained by human driven and automatic procedures," *Aerospace Science and Technology*, Vol. 72, 2018, pp. 443–454.

[19] Martins, J. R. R. A., and Hwang, J. T., "Multidisciplinary Design Optimization of Aircraft Configurations—Part 1: A modular coupled adjoint approach," Lecture series, Von Karman Institute for Fluid Dynamics, Rode Saint Genèse, Belgium, May 2016. ISSN0377-8312.

[20] Yu, Y., Lyu, Z., Xu, Z., and Martins, J. R. R. A., "On the Influence of Optimization Algorithm and Starting Design on Wing Aerodynamic Shape Optimization," *Aerospace Science and Technology*, Vol. 75, 2018, pp. 183–199. doi:10.1016/j.ast.2018.01.016.

[21] Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260. doi:10.1007/BF01061285.

[22] Nielsen, E. J., and Anderson, W. K., "Aerodynamic Design Optimization on Unstructured Meshes Using the Navier–Stokes Equations," *AIAA Journal*, Vol. 37, No. 11, 1999, pp. 1411–1419. doi:10.2514/2.640.

[23] Jameson, A., "Aerodynamic shape optimization using the adjoint method," Lecture series, Von Karman Institute for Fluid Dynamics, Rode Saint Genèse, Belgium, 2003.

[24] Mavriplis, D. J., "Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes," *AIAA Journal*, Vol. 45, No. 4, 2007, pp. 741–750. doi:10.2514/1.22743.

[25] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and van der Weide, E., "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, 2008, pp. 863–873. doi:10.2514/1.29123.

[26] Mader, C. A., "Stability-Constrained Aerodynamic Shape Optimization with Applications to Flying Wings," Ph.D. thesis, University of Toronto, August 2012.

[27] Lyu, Z., Kenway, G. K. W., and Martins, J. R. R. A., "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," *AIAA Journal*, Vol. 53, No. 4, 2015, pp. 968–985. doi:10.2514/1.J053318.

[28] Kenway, G. K. W., and Martins, J. R. R. A., "Multipoint Aerodynamic Shape Optimization Investigations of the Common Research Model Wing," *AIAA Journal*, Vol. 54, No. 1, 2016, pp. 113–128. doi:10.2514/1.J054154.

[29] Secco, N. R., Jasa, J. P., Kenway, G. K. W., and Martins, J. R. R. A., "Component-based Geometry Manipulation for Aerodynamic Shape Optimization with Overset Meshes," *AIAA Journal*, Vol. 56, No. 9, 2018, pp. 3667–3679. doi:10.2514/1.J056550.

[30] Bons, N. P., He, X., Mader, C. A., and Martins, J. R. R. A., "Multimodality in Aerodynamic Wing Design Optimization," *AIAA Journal*, Vol. 57, No. 3, 2019, pp. 1004–1018. doi:10.2514/1.J057294.

[31] He, X., Li, J., Mader, C. A., Yildirim, A., and Martins, J. R. R. A., "Robust aerodynamic shape optimization—from a circle to an airfoil," *Aerospace Science and Technology*, Vol. 87, 2019, pp. 48–61. doi:10.1016/j.ast.2019.01.051.

[32] Leader, M. K., Chin, T. W., and Kennedy, G. J., "High-Resolution Topology Optimization with Stress and Natural Frequency Constraints," *AIAA Journal*, Vol. 57, No. 8, 2019. doi:10.2514/1.J057777.

[33] Anderson, E. M., Bhuiyan, F. H., Mavriplis, D. J., and Fertig, R. S., "Adjoint-Based High-Fidelity Structural Optimization of Wind-Turbine Blade for Load Stress Minimization," *AIAA Journal*, Vol. 57, No. 9, 2019. doi:10.2514/1.J057756.

[34] Zhang, P., Lu, J., Song, L., and Feng, Z., "Study on continuous adjoint optimization with turbulence models for aerodynamic performance and heat transfer in turbomachinery cascades," *International Journal of Heat and Mass Transfer*, Vol. 104, 2017, pp. 1069–1082. doi:10.1016/j.ijheatmasstransfer.2016.08.103.

[35] Gkaragkounis, K., Papoutsis-Kiachagias, E., and Giannakoglou, K., "The continuous adjoint method for shape optimization in Conjugate Heat Transfer problems with turbulent incompressible flows," *Applied Thermal Engineering*, Vol. 140, 2018, pp. 351–362. doi:10.1016/j.applthermaleng.2018.05.054.

[36] Martins, J. R. R. A., and Lambe, A. B., "Multidisciplinary Design Optimization: A Survey of Architectures," *AIAA Journal*,

Vol. 51, No. 9, 2013, pp. 2049–2075. doi:10.2514/1.J051895.

[37] Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 523–530. doi:10.2514/1.11478.

[38] Kenway, G. K. W., Kennedy, G. J., and Martins, J. R. R. A., "Scalable Parallel Approach for High-Fidelity Steady-State Aeroelastic Analysis and Derivative Computations," *AIAA Journal*, Vol. 52, No. 5, 2014, pp. 935–951. doi:10.2514/1.J052255.

[39] Brooks, T. R., Kenway, G. K. W., and Martins, J. R. R. A., "Benchmark Aerostructural Models for the Study of Transonic Aircraft Wings," *AIAA Journal*, Vol. 56, No. 7, 2018, pp. 2840–2855. doi:10.2514/1.J056603.

[40] Burdette, D. A., and Martins, J. R. R. A., "Design of a Transonic Wing with an Adaptive Morphing Trailing Edge via Aerostructural Optimization," *Aerospace Science and Technology*, Vol. 81, 2018, pp. 192–203. doi:10.1016/j.ast.2018.08.004.

[41] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "Aerothermal optimization of internal cooling passages using a discrete adjoint method," *AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, Atlanta, GA, 2018. doi:10.2514/6.2018-4080.

[42] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "Aerothermal Optimization of a Ribbed U-Bend Cooling Channel Using the Adjoint Method," *International Journal of Heat and Mass Transfer*, 2019. Submitted.

[43] Gray, J. S., Mader, C. A., Kenway, G. K. W., and Martins, J. R. R. A., "Modeling Boundary Layer Ingestion Using a Coupled Aeropropulsive Analysis," *Journal of Aircraft*, Vol. 55, No. 3, 2018, pp. 1191–1199. doi:10.2514/1.C034601.

[44] Gray, J. S., and Martins, J. R. R. A., "Coupled Aeropropulsive Design Optimization of a Boundary-Layer Ingestion Propulsor," *The Aeronautical Journal*, Vol. 123, No. 1259, 2019, pp. 121–137. doi:10.1017/aer.2018.120.

[45] Garg, N., Kenway, G. K. W., Lyu, Z., Martins, J. R. R. A., and Young, Y. L., "High-fidelity Hydrodynamic Shape Optimization of a 3-D Hydrofoil," *Journal of Ship Research*, Vol. 59, No. 4, 2015, pp. 209–226. doi:10.5957/JOSR.59.4.150046.

[46] Garg, N., Kenway, G. K. W., Martins, J. R. R. A., and Young, Y. L., "High-fidelity Multipoint Hydrostructural Optimization of a 3-D Hydrofoil," *Journal of Fluids and Structures*, Vol. 71, 2017, pp. 15–39. doi:10.1016/j.jfluidstructs.2017.02.001.

[47] Fabiano, E., and Mavriplis, D., "Adjoint-based aeroacoustic design-optimization of flexible rotors in forward flight," *Journal of the American Helicopter Society*, Vol. 62, No. 4, 2017, pp. 1–17.

[48] Kiviaho, J. F., Jacobson, K., Smith, M. J., and Kennedy, G., "Application of a Time-Accurate Aeroelastic Coupling Framework to Flutter-Constrained Design Optimization," *2018 Multidisciplinary Analysis and Optimization Conference*, 2018, p. 2932.

[49] Ntanakas, G., Meyer, M., and Giannakoglou, K. C., "Employing the Time-Domain Unsteady Discrete Adjoint Method for Shape Optimization of Three-Dimensional Multirow Turbomachinery Configurations," *Journal of Turbomachinery*, Vol. 140, No. 8, 2018, p. 081006.

[50] Franco, M., Persson, P.-O., Pazner, W., and Zahr, M. J., "An Adjoint Method using Fully Implicit Runge-Kutta Schemes for Optimization of Flow Problems," *AIAA Scitech 2019 Forum*, 2019, p. 0351.

[51] Sen, A., Towara, M., and Naumann, U., "A discrete adjoint version of an unsteady incompressible solver for OpenFOAM using algorithmic differentiation," *11th World Congress on Computational Mechanics*, 2014.

[52] Nemili, A., Özkaya, E., Gauger, N. R., Kramer, F., and Thiele, F., "Accurate discrete adjoint approach for optimal active separation control," *AIAA Journal*, 2017, pp. 3016–3026.

[53] Kenway, G. K. W., Mader, C. A., He, P., and Martins, J. R. R. A., "Effective Adjoint Approaches for Computational Fluid Dynamics," *Progress in Aerospace Sciences*, Vol. 110, 2019, p. 100542. doi:10.1016/j.paerosci.2019.05.002.

[54] Spalart, P., and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992. doi:10.2514/6.1992-439.

[55] Issa, R. I., "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of Computational Physics*, Vol. 62, No. 1, 1986, pp. 40–65.

[56] Jasak, H., "Error analysis and estimation for finite volume method with applications to fluid flow," Ph.D. thesis, Imperial College of Science, Technology and Medicine, 1996.

[57] Rhie, C., and Chow, W. L., "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA Journal*, Vol. 21, No. 11, 1983, pp. 1525–1532. doi:10.2514/3.8284.

[58] Yildirim, A., Kenway, G. K. W., Mader, C. A., and Martins, J. R. R. A., "A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations," *Journal of Computational Physics*, Vol. 397, 2019, p. 108741. doi:10.1016/j.jcp.2019.06.018.

[59] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "An Aerodynamic Design Optimization Framework Using a Discrete Adjoint Approach with OpenFOAM," *Computers & Fluids*, Vol. 168, 2018, pp. 285–303. doi:10.1016/j.compfluid.2018.04.012.

[60] He, P., Mader, C. A., Martins, J. R. R. A., and Maki, K. J., "DAFoam: An Open-Source Adjoint Framework for Multidisciplinary Design Optimization with OpenFOAM," *AIAA Journal*, 2019. doi:10.2514/1.J058853, (In press).

[61] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018. URL `http://www.mcs.anl.gov/petsc`.

[62] He, P., Martins, J. R. R. A., Mader, C. A., and Maki, K., "Aerothermal Optimization of a Ribbed U-Bend Cooling Channel Using the Adjoint Method," *International Journal of Heat and Mass Transfer*, Vol. 140, 2019, pp. 152–172. doi:10.1016/j.ijheatmasstransfer.2019.05.075.

[63] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, Fort Worth, TX, 2010. doi: 10.2514/6.2010-9231.

[64] Luke, E., Collins, E., and Blades, E., "A Fast Mesh Deformation Method Using Explicit Interpolation," *Journal of Computational Physics*, Vol. 231, No. 2, 2012, pp. 586–601. doi:10.1016/j.jcp.2011.09.021.

[65] Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, 2012, pp. 101–118. doi:10.1007/s00158-011-0666-3.

[66] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[67] Nocedal, J., and Wright, S. J., *Numerical Optimization*, 2nd ed., Springer-Verlag, 2006.

[68] Warming, R., and Beam, R. M., "Upwind second-order difference schemes and applications in aerodynamic flows," *AIAA Journal*, Vol. 14, No. 9, 1976, pp. 1241–1249.