Aerodynamic Shape Optimization Framework Based on a Novel Fully-Automated Adjoint Differentiation Toolbox

Reza Djeddi* and Kivanc Ekici†

University of Tennessee, Knoxville, Tennessee 37996

A robust and automated design optimization framework is developed in this work. This wrapper program automates the design process by coupling the in-house UNstructured PArallel Compressible (UNPAC) flow solver with a novel toolbox for sensitivity analysis based on the discrete adjoint method. The Fast automatic Differentiation using Operator-overloading Technique (FDOT) toolbox utilizes an advanced recording technique to store the expression tree which can significantly reduce the memory footprint and the computational cost of the adjoint calculations. Additionally, this novel toolbox uses an iterative process to evaluate the sensitivities of the cost function with respect to the entire design space and requires only minimal modifications to the available solver. The design optimization framework, UNPAC-DOF, is then employed for aerodynamic design applications based on a gradient-based optimization algorithm. This framework is used to improve airfoil and wing designs for minimized drag or maximized efficiency.

I. Nomenclature

$\mathcal{V}$	control volume
c	speed of sound
E	total energy
f	objective or cost function
$\mathbf{F}, \mathbf{G}, \mathbf{H}$	flux vectors
h, H	specific and total enthalpy
I	objective or cost function
p	pressure
$\mathbf{R}$	residual vector
$\mathbf{S}$	vector of source terms
t	time
$\mathbf{U}$	vector of conservative variables
$\mathbf{v}$	vector operator
u, v, w	Cartesian velocity components
x, y, z	Cartesian coordinates
γ	specific heat ratio
ψ	adjoint solution vector
$\tilde{\nu}$	working variable for Spalart-Allmaras turbulence model
ρ	density
τ	stress tensor

*Research Assistant Professor and Lecturer, Department of Mechanical, Aerospace and Biomedical Engineering, Professional Member AIAA.

†Associate Professor, Department of Mechanical, Aerospace and Biomedical Engineering, Senior Member AIAA.

II. Introduction

Aerodynamic shape optimization involves the determination of an optimized topology that satisfies certain objectives subject to a set of aerodynamic and/or structural constraints. Traditionally, inverse design methods have been used with the goal of obtaining an optimal shape considering a target aerodynamic property. More recently, design optimization techniques have shifted towards more "direct" methods based on searching the design space for optimum topologies. This approach may lead to novel unconventional designs subject to challenging off-design conditions.

In general, there are two main families of aerodynamic optimization techniques that can be categorized as (1) gradient-based and (2) non-gradient-based approaches, where in the latter, repeated cost function evaluations are required. Using a CFD-based design approach, the computational burden of evaluating the objective (cost) function can be very high even for today's high performance computing resources. Therefore, it is desirable to use gradient-based algorithms in the framework of aerodynamic design optimization. In this approach, however, derivatives (sensitivities) of a cost function with respect to all design variables (that can number in the hundreds to thousands) are required. While "optimum" configurations can be determined after a small number of optimization cycles using the gradient-based approach, the cost and complexity associated with the gradient evaluations can be overwhelming. These issues are even more pronounced in high-fidelity CFD solvers and in the framework of a multidisciplinary design optimization process. Therefore, a fast and efficient sensitivity analysis tool would be a leap forward that can greatly reduce the time and cost of the CFD-based design and topology optimization.

The most important and challenging part of the gradient-based design optimization approach is the calculation of the gradient information. Traditionally, finite difference (FD) method has been used for this purpose although the step-size dilemma associated with this approximative technique have led to the introduction of the complex-step (CS) method.[1] While the step-size is not an issue with the latter approach, the computational cost for both FD and CS techniques is linearly proportional to the number of design variables since repeated objective function evaluations are necessary to obtain the entire gradient information. As an alternative, adjoint methods have been developed[2,3] and their application to aerodynamic design optimization has rapidly grown in the past two decades.[4] The adjoint method can be classified in two categories of (1) discrete[5,6] and (2) continuous[7,8] approaches depending on the order with which the adjoint equations are derived and solved. In the discrete adjoint method, the discretized governing equations are adjoined and solved to obtain gradient information. As a result, the adjoint equations have the same level of computational complexity as the governing equations and the same numerical procedure will be followed for both the primal and adjoint solvers. On the other hand, in the continuous adjoint method, the adjoint method is first applied to the governing equations to derive the continuous adjoint equations which are then discretized and solved. Therefore, special boundary conditions and numerical procedures need to be considered to discretize and solve the continuous adjoint equations.[9] The most important feature of the adjoint methods is that the computational cost of gradient evaluation is independent of the number of design variables which makes this class of technique a prime candidate for aerodynamic design optimization involving hundreds, if not thousands, of design variables.

Development of the complementary discrete adjoint solvers can be substantially simplified using algorithmic or automatic differentiation (AD) tools. Using the systematic application of the chain rule of differentiation, AD tools can provide non-approximated (exact) gradient information. This process can be carried out in (1) forward or (2) reverse modes based on the direction of the gradient propagation. While the forward mode is mathematically straightforward and easy to implement, its computational cost increases linearly with the size of the design space. Therefore, the forward mode of AD may not be suitable for modern aerodynamic shape optimization problems. As an alternative, the reverse mode of AD is used although the computational complexity and memory requirements of this approach need to be addressed. On the programming side, the AD approach can be implemented using (1) source code transformation (SCT)[10–13] and (2) operator overloading (OO).[14–17] Interested readers are referred to the work of Griewank and Walther[18] for a detailed comparison between the two approaches. As the use of object-oriented programming paradigms has gained popularity in Fortran programming, many OO/AD tools have been developed specifically for Fortran. ADF,[19] ADOL-F,[20] AUTO_DERIV,[21] DNAD[22] and more recently dco/fortran[23] are some examples of these tools that are available today.

It must be noted that the memory footprint for the adjoint-based automatic differentiation can become prohibitively high for large-scale CFD problems. One approach to address the memory and computational efficiency issues of the conventional AD tools is the use of fixed-point iterations as originally proposed by

Christianson.[24,25] This iterative approach has been commonly used in SCT/AD applications[26–28] while the two most advanced OO/AD tools for C++ codes, ADOL-C[29] and CoDiPack,[30] have recently adopted a similar strategy for accumulating and evaluating the adjoints while maximizing memory and computational efficiency. Recently, Djeddi and Ekici[31] have developed, for the first time, a Fortran-based OO/AD toolbox based on the discrete adjoint method that uses a similar fixed-point iteration approach for adjoint evaluations. The Fast automatic Differentiation toolbox based on Operator-overloading Technique (FDOT) toolbox[31] is capable of efficiently and accurately calculating the sensitivity information of a cost function with respect to any design variable.

In this work, a design optimization framework, called UNPAC-DOF, is developed.[32] This framework utilizes the novel FDOT toolbox[31] for adjoint sensitivity analysis. The FDOT toolbox provides the gradient information based on the discrete adjoint method with minimal changes required to be made to the available codes. More specifically, FDOT greatly eliminates the memory requirements inherent in existing OO-based tools mentioned earlier. This toolbox is then coupled with an in-house Computational Fluid Dynamics (CFD) solver, called UNPAC, as well as a gradient-based optimization algorithm. The resulting framework is used for robust aerodynamic shape optimization of airfoils and wings. In the following sections, details of the UNPAC solver, the FDOT toolbox, as well as the design optimization framework (UNPAC-DOF) are described. Finally, the framework is applied to a set of different aerodynamic shape optimization problems.

III. UNstructured PArallel Compressible (UNPAC) Solver

To motivate the design optimization framework, the three-dimensional Unsteady Reynolds-Averaged Navier-Stokes (URANS) equations coupled with the Spalart-Allmaras turbulence model[33] are considered here. These conservation laws are written in the differential form as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} = \mathbf{S} \tag{1}$$

where the vector of conservative variables is $\mathbf{U} = [\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{\nu}]^T$ and the vectors of convective and viscous fluxes in three Cartesian directions, $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$, are given as:

$$\mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p - \tau_{xx} \\ \rho uv - \tau_{xy} \\ \rho uw - \tau_{xz} \\ \rho uH - \tau_{xh} \\ \rho u\tilde{\nu} - \tau_{x\tilde{\nu}} \end{bmatrix}, \; \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uv - \tau_{yx} \\ \rho v^2 + p - \tau_{yy} \\ \rho vw - \tau_{yz} \\ \rho vH - \tau_{yh} \\ \rho v\tilde{\nu} - \tau_{y\tilde{\nu}} \end{bmatrix}, \; \mathbf{H} = \begin{bmatrix} \rho w \\ \rho wu - \tau_{zx} \\ \rho wv - \tau_{zy} \\ \rho w^2 + p - \tau_{zz} \\ \rho wH - \tau_{zh} \\ \rho w\tilde{\nu} - \tau_{z\tilde{\nu}} \end{bmatrix}$$

Here, $\tilde{\nu}$ is the viscosity-like working variable of the Spalart-Allmaras turbulence model. Additionally, the source vector, $\mathbf{S}$, has all zero terms except for the last equation where the source term is described by the Spalart-Allmaras turbulence model.[33] The pressure, p, and the total enthalpy, H, are defined in terms of the conservative variables as

$$p = (\gamma - 1)\rho\left[E - \frac{1}{2}(u^2 + v^2 + w^2)\right]$$
$$H = \frac{\rho E + p}{\rho} = \frac{\gamma}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}(u^2 + v^2 + w^2)$$

Moreover, the viscous fluxes in the 3D URANS equations depend on the gradients of the flow velocities, specific enthalpy and the working variable of the turbulence model. The governing equations given in Eq. (1) are discretized using a finite volume method with median-dual, vertex-based control volume approach.[32,34,35] Therefore, the semi-discretized form of the governing equations can be written as

$$\frac{d}{dt}(\mathcal{V}\mathbf{U}) + \mathbf{R}(\mathbf{U}) = \mathbf{0} \tag{2}$$

where $\mathcal{V}$ is the control volume, and $\mathbf{R}$ is the numerical residual that represents the discretization of the spatial terms that include both the convective fluxes and viscous fluxes as well as the turbulence model

source term. Here, the fluxes are defined at the midpoint of an edge. Therefore, the numerical solver loops over all edges to calculate these fluxes, which then get integrated to evaluate the residual at the center of each control volume (defined at grid nodes).

The convective terms are discretized using an upwind scheme based on Roe-fluxes.[36] Additionally, limiter functions of Barth and Jespersen[37] and Venkatakrishnan[38] are used for the upwind convective fluxes in the case of higher-order solution reconstruction at the face center. A second-order central averaging scheme is used for the calculation of the viscous fluxes. The gradients of the flow variables are calculated at the grid nodes using a Green-Gauss method. Furthermore, the solver is parallelized using the message passing interface (MPI) tools with a non-overlapping domain decomposition,[39] and METIS software package[40] is used for partitioning the computational domain. It must be noted that for the steady cases studied in this work, the time-derivative term in Eq. (2) is replaced with a "pseudo-time" derivative in order to march the governing equations to steady-state.

IV. FDOT Toolbox

In this section, details about the fully-automated discrete adjoint toolbox (FDOT) are provided. It must be noted that the original idea was introduced by the authors in a previous publication.[31] As such, the theory is repeated here for completeness, and it closely follows the work presented in Ref.[31]

The goal of the aerodynamic design optimization process is to find the optimal solution (or design) that can minimize the objective function, $I(\mathbf{x}, \mathbf{U}(\mathbf{x}))$, defined in terms of the design variables, $\mathbf{x}$, and the corresponding flow solution, $\mathbf{U}(\mathbf{x})$. As discussed earlier, in the framework of a gradient-based design optimization problem, the gradient of the objective function with respect to the design variables is required. This gradient can be written as the total derivative of the objective function with respect to the design variables:

$$\frac{dI}{d\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} + \frac{\partial I}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} \tag{3}$$

where $\mathbf{x}$ is the vector of design variables. In an Aerodynamic Shape Optimization (ASO) problem, the objective function is usually defined as a scalar (drag or lift coefficient) while the number of design variables can be on the order of $\mathcal{O}(10)$ to $\mathcal{O}(10^2)$ for a two- or three-dimensional test case. On the other hand, the size of the $\mathbf{U}$ vector or the number of degrees of freedom (DOF) for a typical CFD solver can easily be on the order of $\mathcal{O}(10^5)$ to $\mathcal{O}(10^7)$. As a result, evaluating the Jacobian term, $\frac{\partial \mathbf{U}}{\partial \mathbf{x}}$, in Eq. (3) can become very expensive computationally. Therefore, it would be impractical to use a direct approach in sensitivity analysis. It must be noted that the flow solution and the corresponding flow residual (see Eq. [2]) are both defined in terms of the design variables for an ASO problem. Thus, Eq. (2) can be rewritten as

$$\frac{d}{dt}\left(\mathcal{V}\mathbf{U}(\mathbf{x})\right) + \mathbf{R}(\mathbf{x}, \mathbf{U}(\mathbf{x})) = \mathbf{0}. \tag{4}$$

In practice, the converged steady solution for any CFD solver means that the residual vector is driven to zero. Moreover, it can be assumed that the total derivative of the residual vector with respect to the vector of design variables would also vanish when the solution is converged, i.e.,

$$\frac{d\mathbf{R}}{d\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} = 0 \tag{5}$$

By rearranging Eq. (5) and inserting it into Eq. (3), the total derivative of the objective function with respect to the design variables can be rewritten as

$$\frac{dI}{d\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} - \frac{\partial I}{\partial \mathbf{U}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} + \psi^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{6}$$

where ψ is the adjoint solution vector. Calculating the adjoint solution vector is the essense of both discrete and continuous adjoint approaches. The main goal here is to efficiently and accurately evaluate all the partial derivatives in Eq. (6) in order to find the necessary gradient information for the ASO problem.

The FDOT toolbox developed by Djeddi and Ekici[31] utilizes the concept of discrete adjoint sensitivity analysis and the object-oriented programming (OOP) capabilities of the modern Fortran programming language to evaluate the gradient information. By defining a new derived type for real-typed variables and by overloading all the unary and binary operations and intrinsic functions, the FDOT toolbox can be coupled

with any numerical solver to provide the sensitivities (gradients) of the output (objective) function(s) with respect to all design variables. Calculating these gradients is done via an "adjoint evaluation" process whose computational cost is only a small multiple of that of the primal solver.[41,42] It must be noted that in the discrete analysis, the derivatives are propagated in the reverse direction, thus require the complete time history of the primal flow equations to be stored in the memory. Almost all OO/AD tools achieve this by recording the entire expression tree into a derived type class often called the *tape*. This tape is then executed in the reverse order while accumulating the derivatives using the recorded adjoint information. However, this process can lead to intractable memory footprints in a large scale three-dimensional flow solver.

A typical CFD solver can be viewed as a set of three main stages: (1) a pre-iterative process that handles grid preprocessing and flow initialization, (2) an iterative process that solves the flow equations, and (3) a post-iterative process that computes the cost function based on the converged flow solution. Clearly, the most complex part of any CFD solver is the iterative part where numerical schemes and discretization techniques are utilized to obtain flow solution. Consequently, the iterative part of the CFD solver would also be the most memory demanding and computationally expensive portion of the adjoint process. Due to the fact that the iterative process involves repeated evaluations of the flow solution, recording the tape for the entire set of intermediate flow solutions can significantly increase the memory cost of the adjoint solver to the point that it would become impractical. The FDOT toolbox addresses this issue by using a fixed-point iteration approach, as originally proposed by Christianson,[24,25] to evaluate the flow solution adjoints, $\overline{\mathbf{U}} = \frac{\partial I}{\partial \mathbf{U}}$. Here, the iterative process can be written as

$$\mathbf{U}_{k+1} = f(\mathbf{U}_k) \;\text{ with } \mathbf{U}_k \to \mathbf{U} \text{ as } k \to N \tag{7}$$

where $\mathbf{U}_k$ is the flow solution at the k-th iteration. Assuming that N iterations are required for the primal solver to reach the desired level of accuracy, the adjoint evaluation process can be viewed as a reverse accumulation of the flow solution adjoints such that

Algorithm 1 Reverse flow adjoint accumulation

1: *loop*:
2: $k = N \to 1$ (decrement 1)
3: $\overline{\mathbf{U}_k} \mathrel{+}= (\partial f/\partial \mathbf{U}_k) * \overline{\mathbf{U}_{k+1}}$
4: $\overline{\mathbf{U}_{k+1}} \leftarrow \overline{\mathbf{U}_k}$

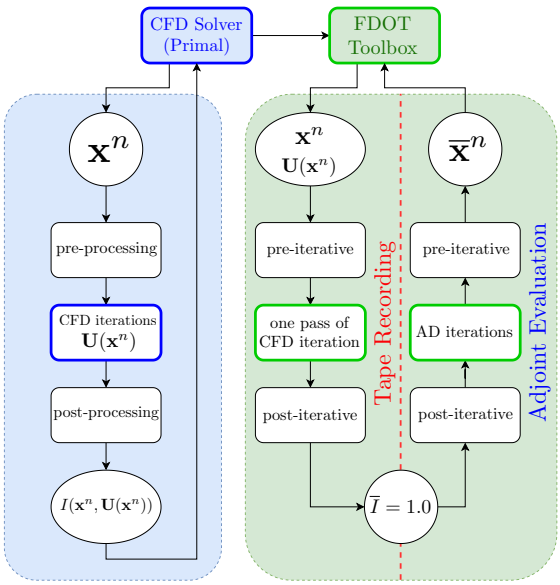


Figure 1. Flowchart for the FDOT toolbox and its integration into the primal CFD solver (adopted from Djeddi and Ekici[31]).

An important feature of the FDOT toolbox is that it requires minimal changes to an already existing primal solver to obtain the adjoint version of that solver. One of the modifications that is necessary is the

declaration of the flow solution variables, **U**, in the FDOT toolbox while also marking the start and end of the iterative part using a "checkpointing" function. The overall scheme demonstrating the coupling of the primal solver to the FDOT toolbox for the discrete adjoint analysis is depicted in Figure 1.

V. UNPAC-DOF: Design Optimization Framework

Traditionally, aerodynamic design process has heavily relied on experimental wind tunnel tests and engineering judgment. With the advent of computational fluid dynamics, numerical shape optimization has been made possible without expensive and cost-prohibitive experiments and wind tunnel tests. Over the years, robust design methodologies have been proposed in aerodynamic shape optimization. Additionally, a whole field has been devoted to developing algorithms and black-box software packages used for numerical optimization.[43]

In this work, the gradient-based optimization approach is considered. UNPAC is used to obtain primal (or flow) solutions. Furthermore, the FDOT toolbox and the CFD code are integrated into the UNPAC-AD framework to compute the gradient information. Finally, the UNPAC-OPT wrapper program is developed to perform design optimization.[32]

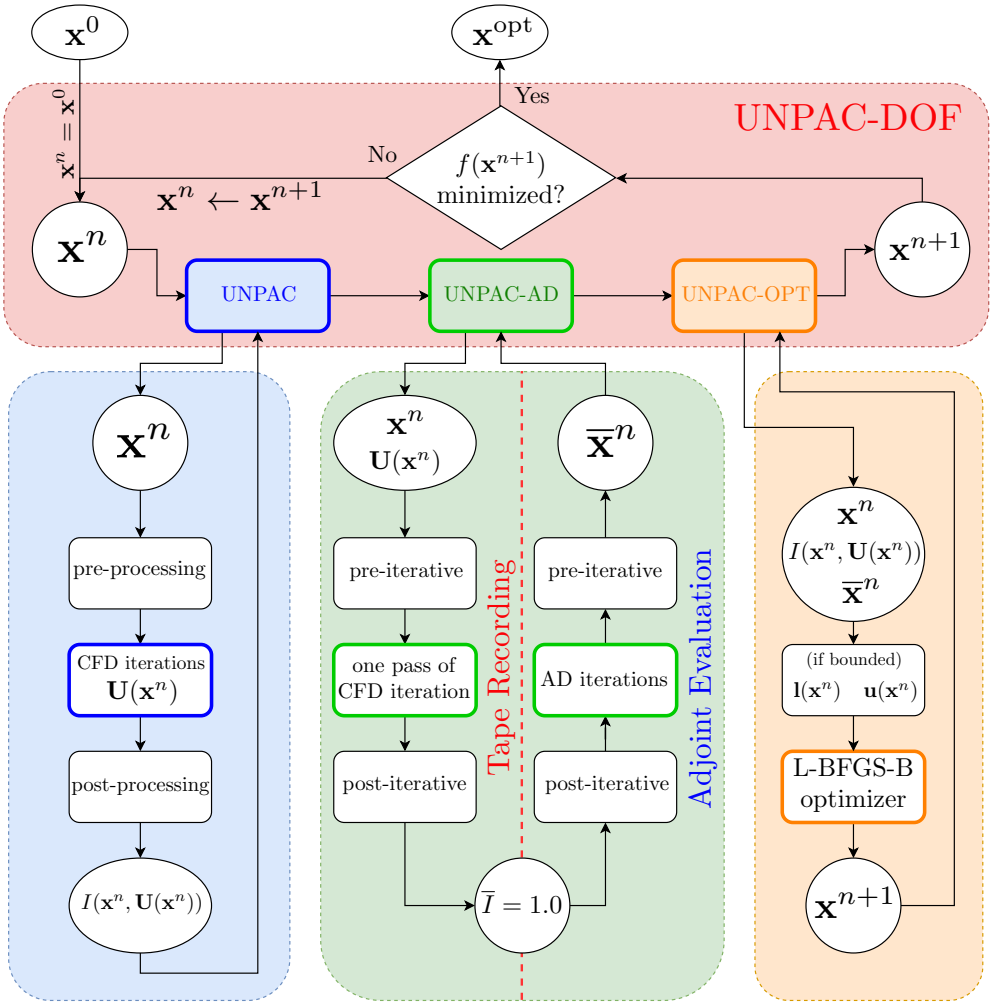


Figure 2. Flowchart of the UNPAC Design Optimization Framework (UNPAC-DOF) and its three main components: UNPAC, UNPAC-AD, and UNPAC-OPT.

Simply, UNPAC-OPT uses a quasi-Newton method for optimization in both unbounded and bound constrained modes subject to upper and/or lower bounds for the design variables. The schematic of the present **UNPAC D**esign **O**ptimization **F**ramework (UNPAC-DOF) is provided in Figure 2. The optimization frame-

work seeks for optimal designs via an iterative process in which the following steps are considered for each design cycle:

1. The nominal CFD solver (UNPAC) is run to obtain the flow solution. In the first cycle, the initial set of design variables is used while in the subsequent cycles, the new values for the design variables (solution of the optimizer) are utilized. In the case of shape optimization, the design variables define the new geometry which will then be used to deform the computational mesh. This process is described in Section A.

2. Using the flow solution, the UNPAC-AD solver is initiated. First, this solver executes a single pass of the CFD solver to record the expression tree as a tape. Later, the tape is rewound in the reverse mode with a novel fixed-point approach to evaluate the adjoints of all derived-type variables (including intermediate and ultimately design variables).

3. The adjoint solutions obtained from the UNPAC-AD solver are then input into the UNPAC-OPT program. Here, the gradient information is passed on to a quasi-Newton optimizer, details of which are presented in Section B, to obtain the new set of design variables. The new topology is then returned back to the UNPAC solver for the next design cycle. The optimization process is repeated until either the desired number of design cycles is reached or the desired tolerance for the optimal solution has been achieved.

In general, any set of design variables can be used for the optimization process. For certain optimization problems, these variables can be the angle of attack, free-stream Mach number, etc. For the case of shape optimization, the design variables can be taken to be the surface points defining the geometry. Compared to the cases where shape parameterization is utilized, the surface points theoretically define a complete design space. However, the use of surface mesh points as design variables can potentially lead to unsmooth profiles due to high frequency modes. This problem is circumvented using a smoothing approach similar to that proposed by Huang and Ekici.[28]

As for the objective function, although any desired definition can be easily implemented, three possible options are considered in this work. These are namely the:

1. Drag coefficient, C_D (minimized by default)

2. Lift coefficient, C_L (maximized by default)

3. Lift-to-drag ratio or efficiency, C_L/C_D (maximized by default)

The wrapper program for the design optimization framework uses bash scripts to automate the process of running nominal and adjoint solvers. Additionally, it runs scripts to organize solution data into different folders for each design cycle. It must be noted that the nominal solver can be run in serial or parallel mode while the adjoint solver is currently run only in serial mode. The parallelization of the adjoint solver is the subject of ongoing research and will be addressed in future works.

A. Shape Deformation

In aerodynamic shape optimization, it is common to use shape parameterization techniques where the focus is shifted from the actual grid points defining the geometry to a certain number of variables controlling the parameterized geometry. In this approach, a large number of design variables (on the order of hundreds to tens of thousands) can be reduced down to a fraction at the expense of limiting the design space.

In practice, Hicks-Henne bump functions,[44] B-Splines (NURBS),[45] and Free Form Deformation (FFD)[46] techniques are commonly used for the purpose of shape parameterization. However, if not tuned correctly, these geometrical representations can lead to cases where the design variables would not form a complete design space, causing the optimizer getting trapped at local optima. As an alternative, the mesh points can be directly used as the design variables given the fact that the cost of the adjoint solver is independent of the number of design variables. On the other hand, using surface points as design variables for aerodynamic shape optimization would require smoothing procedures that can be challenging for arbitrary 3D geometries. In the UNPAC design optimization framework, both the surface points and the FFD box shape parameterization techniques are implemented where the former approach is mostly used for 2D cases while the latter is used for both 2D and 3D aerodynamic shape optimization problems.

1. Surface Mesh Points

As discussed earlier, the use of mesh points can lead to unsmooth profiles which, at extreme conditions, can even cause convergence issues for the nominal solver. In the framework of steepest descent optimizers, Jameson[47] and Castonguay and Nadarajah[48] have utilized a smoother technique based on the Sobolev inner product to smooth the gradient information. However, for the Newton and quasi-Newton optimizers, smoothing the gradient information can cause gradient inaccuracies that can negatively affect the performance of the optimization algorithm. In this regard, Huang and Ekici[28] have proposed smoothing the surface perturbations using an implicit smoother before applying them to the design variables from the previous design cycle. In this work, a similar approach is utilized which follows the same procedure used for the implicit residual smoothing. First, the perturbation of the design variable i is described as

$$\Delta \mathbf{x}_i = \mathbf{x}_i^{n+1} - \mathbf{x}_i^n \tag{8}$$

where $\mathbf{x}^n$ and $\mathbf{x}^{n+1}$ are the design variables at two subsequent design cycles n and $n+1$, respectively. Here, the smoothed perturbation at node i is defined based on a pseudo-Laplacian of the perturbations at neighboring nodes via

$$\Delta \mathbf{x}_i^* + \epsilon \sum_{j=1}^{\mathrm{Ngb}_i} [\Delta \mathbf{x}_i^* - \Delta \mathbf{x}_j] = \Delta \mathbf{x}_i \tag{9}$$

where $\Delta \mathbf{x}_i$ and $\Delta \mathbf{x}_i^*$ are the original (unsmoothed) and smoothed perturbations of the design variable i, respectively. It is worth mentioning that the pseudo-Laplacian only includes neighbors of node i that lies on the surface. Finally, the design variables at design cycle $n+1$ are updated through

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta \mathbf{x}^* \tag{10}$$

It must be noted that the smoothing parameter, ϵ, has a value between 0.3 and 0.8 with larger values of ϵ leading to an increased smoothing of the perturbations. In this work, a smoothing parameter of 0.5 is used. As will be shown later in the results section, this approach ultimately results in a smooth geometry deformation during design updates. Finally, the smoothed design variables are then passed on to the nominal solver to obtain the flow solution in the next design cycle. A radial basis function (RBF) approach is used to perform volume mesh deformation considering the displacements of the design variables as the control points of the RBF system of equations.[49–52]

2. Free-Form Deformation (FFD) Box

Originally used in computer graphics, the FFD box technique enables the deformation of a two- or three-dimensional geometries regardless of the actual shape of the object. As the name suggests, the FFD approach involves a box enclosing the geometry where equi-spaced points, known as FFD control points, are defined on the faces of this box. The classical Bézier parameterization of the surface points is then used to define the location of each surface mesh point in the Cartesian coordinate system as a function of all the FFD control points.

Here, a 3D hexahedral lattice is assumed to enclose the design geometry where a local coordinate system (ξ, η, ζ) can be defined within this hexahedral lattice with $(\xi, \eta, \zeta) \in [0,1] \times [0,1] \times [0,1]$. Therefore, the location of each surface mesh point, $\mathbf{x}_s$, can be defined by a third-order Bézier tensor product involving the location of the FFD control points, $\mathbf{x}_{\mathrm{cp}}$, such that:[46]

$$\mathbf{x}_s = \sum_{i=0}^{\mathrm{NI}} \sum_{j=0}^{\mathrm{NJ}} \sum_{k=0}^{\mathrm{NK}} B_i^{\mathrm{NI}}(\xi_s)\, B_j^{\mathrm{NJ}}(\eta_s)\, B_k^{\mathrm{NK}}(\zeta_s)\ \ \mathbf{x}_{\mathrm{cp}} \tag{11}$$

where $B_i^{\mathrm{NI}}(\xi_s)$, $B_j^{\mathrm{NJ}}(\eta_s)$, and $B_k^{\mathrm{NK}}(\zeta_s)$ are the Bernstein polynomials of order NI, NJ, and NK defined as

$$B_p^n(s) = C_p^n s^p (1-s)^{n-p} \tag{12}$$

where C_p^n is the binomial coefficient given as

$$C_p^n = \begin{pmatrix} n \\ p \end{pmatrix} = \frac{n!}{p!(n-p)!} \, . \tag{13}$$

For any given FFD box, the location of the equi-spaced control points can be easily defined by simply seeding points in each direction depending on the order of the Bézier curve, i.e., NI, NJ, and NK. The main challenge here is to define the parametric coordinates corresponding to each surface mesh point, i.e., (ξ, η, ζ). As can be seen from Eq. (11), one should recover the original geometry when the correct definition of these parametric coordinates is used with the unperturbed FFD box (control points at their original position). Therefore, starting from an initial guess for the parametric coordinates, an iterative approach based on a Newton's search algorithm is used that solves the least-squares problem to find the correct parametric coordinates.

As described earlier, in cases with shape parameterization using the FFD box approach, the location of the FFD box control points are used as the design variables instead of the surface mesh points. Therefore, at the end of each design cycle, the new location of the control points can be used to define the new location of the surface mesh points. The deformed surface geometry is then extrapolated to the interior nodes within the computational domain using an RBF-based[35,51] mesh deformation technique. As mentioned earlier, when using the surface mesh points as the design variables, it is necessary to smooth their perturbations in order to avoid getting a non-smooth deformed geometry. However, shape parameterization using the Bézier surfaces does not require any special treatment and smoothing for the design variables during optimization cycles.

B. Optimization Algorithm

In general, most numerical optimization techniques involve an iterative process by considering a sequence of intermediate solutions for design variables, $\mathbf{x}^n$, that will theoretically converge to the minimizer of function $f(\mathbf{x})$ at an optimal solution, $\mathbf{x}^{\text{opt}}$. The goal here is to use the information at $\mathbf{x}^n$ to find the next estimate $\mathbf{x}^{n+1}$ such that $f(\mathbf{x}^{n+1}) < f(\mathbf{x}^n)$.

The UNPAC-DOF framework developed in this work utilizes a quasi-Newton optimization algorithm for the following bound constrained minimization problem

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{14}$$

where $\mathbf{x}$ is the vector of N design variables bounded by the lower, $\mathbf{l}$, and upper, $\mathbf{u}$, bounds and f is a differentiable scalar objective or the cost function. The L-BFGS-B[53] tool written in Fortran 77 language is used as a black-box optimizer which is based on the bound constrained version of the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm.[54] Due to an efficient Hessian approximation technique incorporated in the optimizer, it can be used for large-scale optimization problems with limited memory footprint. Additionally, the toolbox can be used for both unbounded and bound constrained problems. For bound constrained optimization, the method uses a simple gradient technique to determine free and fixed variables (according to the per-variable constant lower and upper bounds). Ultimately, the L-BFGS method is applied to the free variables via an iterative process. At each optimization cycle, the L-BFGS-B optimizer receives the current design variables and their bounds (if any), the gradient vector, and the value of the cost function. Upon successful termination, the optimizer returns the new values of the design variables.

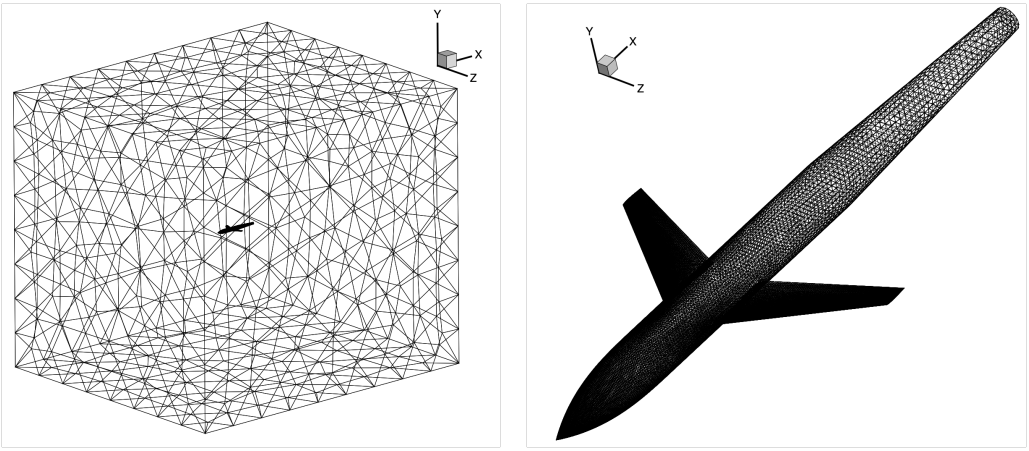
VI. Numerical Results

Before presenting the results from the UNPAC design optimization framework, the flow solution obtained using the primal UNPAC solver needs to be validated and verified (V&V). For this reason the flow around an RAE "A" wing-body configuration is considered first. Next, the UNPAC-DOF framework is used for aerodynamic shape optimization of two well-known airfoils. First, the lift-to-drag ratio or the efficiency of a NACA0012 airfoil operating at inviscid subsonic flow regime is maximized. For this purpose, the unbounded as well as bound constrained optimization techniques are considered. Next, the starting with a NACA0012 airfoil operating at inviscid transonic flow regime, the airfoil shape is optimized so as to minimize

the drag coefficient. Following these, the turbulent flow past the National Renewable Energy Laboratory (NREL) S809 wind turbine cross-section is considered and shape optimization is carried out with the goal of increasing the efficiency of the blade section by maximizing the lift-to-drag ratio at a certain operating condition. Finally, the UNPAC-DOF is extended to 3D aerodynamic shape optimization where the ONERA M6 wing is optimized with the aim of reducing the drag in transonic flow regime.

A. CFD Solver Validation: RAE "A" Wing-Body Configuration

For solver validation, the inviscid flow past the RAE "A" wing-body configuration is considered. The experimental data for this setup are taken from AGARD-AR-138 report[55] and "Case 6" with a free-stream Mach number of 0.9 and an angle of attack of 1.0 degree is chosen. The RAE "A" wing for this configuration has an aspect ratio of 5.5, a leading-edge sweep angle of 36.7 degree, a trailing-edge sweep angle of 22.3 degree, and a taper ratio of 0.375. This untwisted wing is made of RAE 101 symmetrical airfoil cross-sections. A rectangular outer boundary that extends $3L$ in each direction is used (where L is the total length of the wing-body plane). This grid is consisted of $459,487$ tetrahedral cells and is shown in Figure 3.



(a) Far-field view (b) Near-field view

Figure 3. Far-field and near-field views of the computational grid used for the RAE "A" wing-body configuration test case.

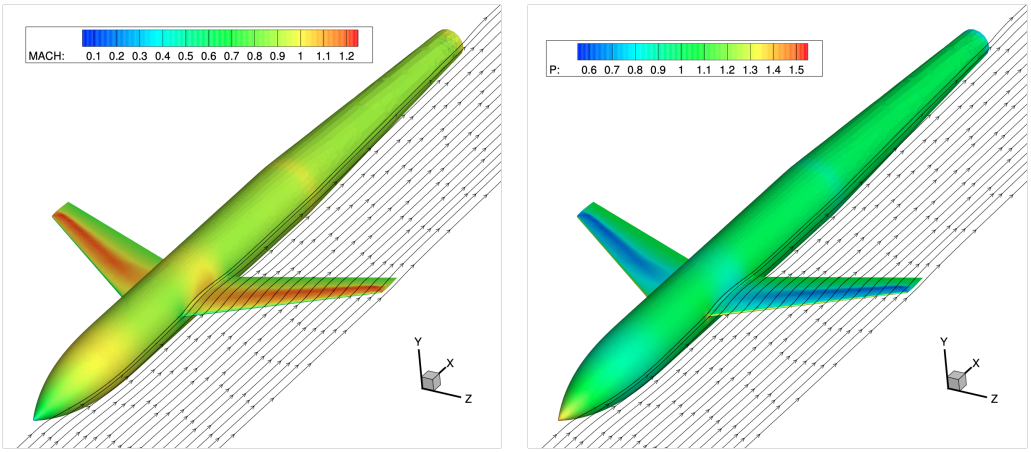
First, the Mach number and pressure contour distributions on the surface of the wing-body configuration are shown in Figure 4. For the transonic flow considered here, the shock formations on the suction sides of the RAE wings can be clearly seen. Additionally, the stream-traces are shown in Figure 4 for the flow past the left wing.

In order to validate the flow solutions for this case, the surface pressure coefficient distributions at 6 span-wise locations along the wing are considered. Additionally, the surface pressure coefficients at two longitudinal sections are presented. The two longitudinal sections follow the AGARD-AR-138[55] report and are defined at $\phi = 90$ and $\phi = 15$ degree angles with respect to the $x - z$ plane. It must be noted that the $\phi = 90$ degree plane is essentially the symmetry plane of the wing-body configuration or the $x - y$ plane for the present setup. The surface pressure coefficients from the CFD solver and the experimental data are given in Figures 5 and 6.

As can be seen, there is good agreement between the obtained numerical results and the experimental data for all sections. It must be noted that the slight disagreements in the suction pressure amplitudes for span-wise sections close to the tip of the RAE wing are consistent with the results available in the literature for this case.[56]

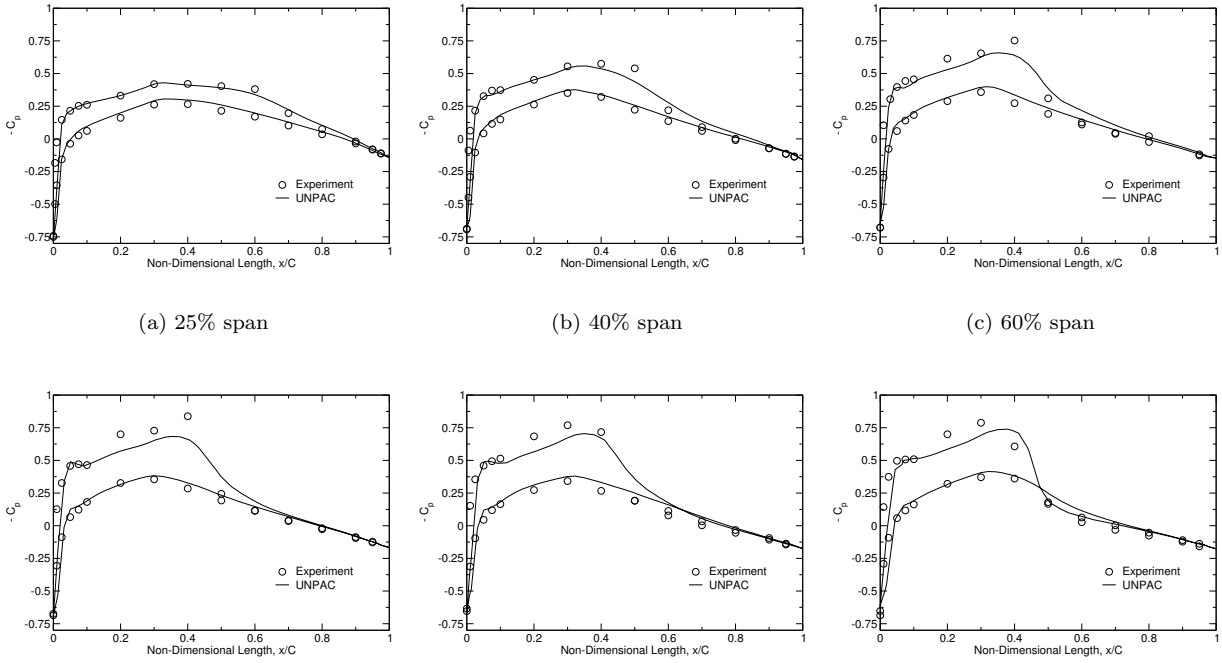
B. Efficiency Maximization: Subsonic NACA0012 Airfoil

As the first optimization test case, the lift-to-drag maximization of the NACA0012 airfoil is sought. Here, the inviscid subsonic flow at $M = 0.5$ with an angle of attack of $\alpha = 2.0$ degrees is considered. The computational



(a) Mach number contours (b) Pressure contours

Figure 4. Contours of Mach number and pressure on the surface of the wing-body configuration along with the stream-traces about the left wing for the RAE "A" wing-body configuration case with $M_\infty = 0.9$ and AoA $= 1.0$ deg.



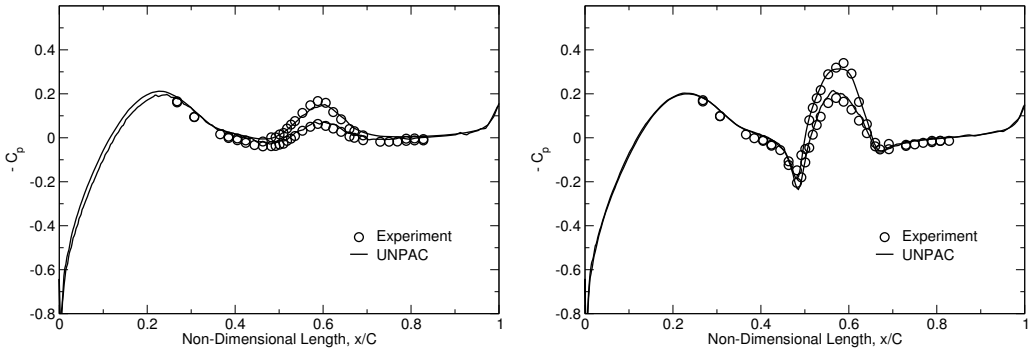
(a) 25% span (b) 40% span (c) 60% span

(d) 75% span (e) 85% span (f) 92.5% span

Figure 5. Surface pressure coefficient, C_p, distributions for six span-wise locations along the wing for the RAE "A" wing-body configuration test case.

grid used for this case is a fully unstructured mesh with $10,216$ triangular elements.

Initially, an unbounded or unconstrained efficiency maximization is sought. For this reason, the objective function is taken to be the ratio of lift coefficient, C_L, to drag coefficient, C_D, which is also known as the aerodynamic efficiency. Since the optimizer solves a standard minimization problem, the cost function is taken to be negative of the ratio in order to maximize its value, i.e.,



(a) 90-deg section, x-y plane

(b) 15-deg section

Figure 6. Surface pressure coefficient, C_p, distributions for two longitudinal sections along the body at $\phi = 90$ degree and $\phi = 15$ degree for the RAE "A" wing-body configuration test case.

$$f(\mathbf{x}, \mathbf{U}(\mathbf{x})) = -\frac{C_L}{C_D} \tag{15}$$

where $\mathbf{x}$ and $\mathbf{U}(\mathbf{x})$ are the design variables and the flow solutions at the corresponding design cycle, respectively. Design variables are taken to be the y-coordinates of the grid nodes on the surface of the airfoil which result in vertical movement of the nodes throughout the design optimization process.

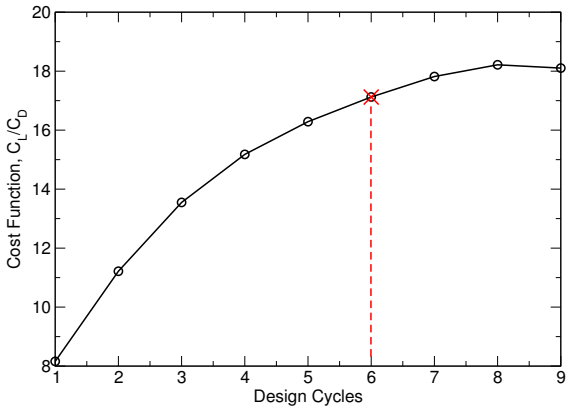


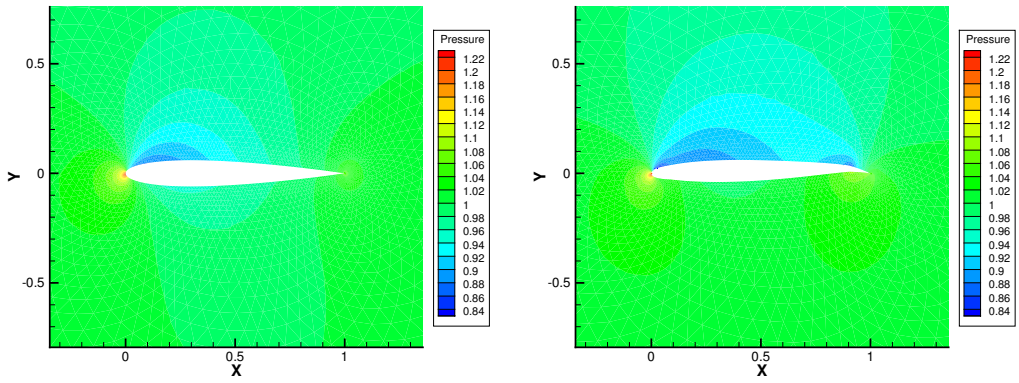
Figure 7. Convergence history of the lift-to-drag ratio for the unbounded efficiency optimization of NACA0012 airfoil.

As will be shown later, unbounded lift-to-drag ratio maximization can lead to extreme deformations in the airfoil geometry that can eventually cause a zero thickness at some point along the airfoil chord. As a result, for the first unbounded optimization case, the optimizer is intentionally setup in a way that will slow down the line search towards the optimal solution. Convergence history of the unbounded optimization test case for the NACA0012 airfoil operating at inviscid subsonic flow regime is shown in Figure 7.

For the unbounded optimization case and after six design cycles, the airfoil becomes significantly thinner and has an apparent camber. The lift-to-drag ratio increases for almost 110% after 6 design cycles. The contour plots of pressure for the original and optimized (design cycle 6) airfoils are shown in Figure 8.

Additionally, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils are shown in Figure 9. As can be seen, there is an extreme movement close to the trailing edge region. Moreover, the airfoil thickness is reduced for about 18% from its original value after 6 design cycles.

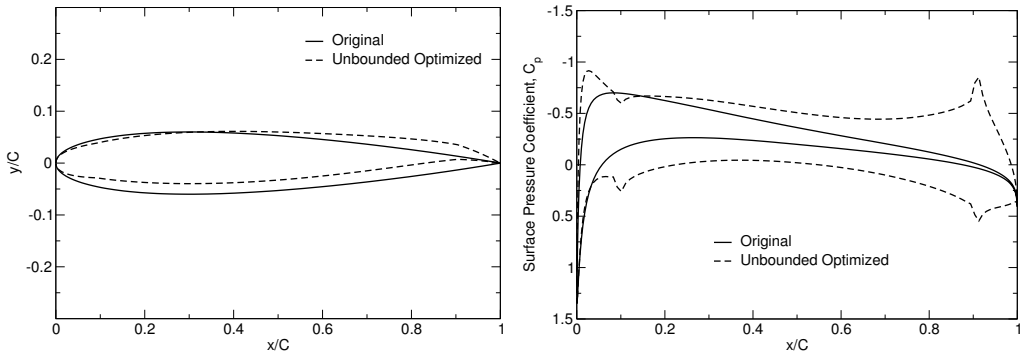
As shown earlier, for the unbounded optimization case, the sixth design cycle was chosen as the final optimized design although the value of the cost function appears to be increasing in the subsequent design



(a) Original Design

(b) Unbounded optimized (design cycle 6)

Figure 8. Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.



(a) Airfoil shape

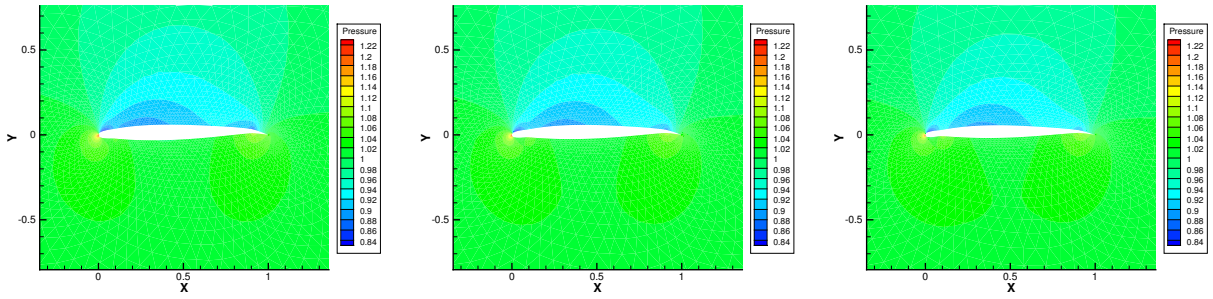
(b) Surface pressure coefficient, C_p

Figure 9. Comparison of airfoil shape and the surface pressure coefficients for the original and unbounded optimized geometries.

cycles before starting to decrease at design cycle 9. The contour plots of pressure for design cycles seven, eight, and nine are also shown in Figure 10. It can be clearly seen that the airfoil thickness decreases significantly around 10% chord length.

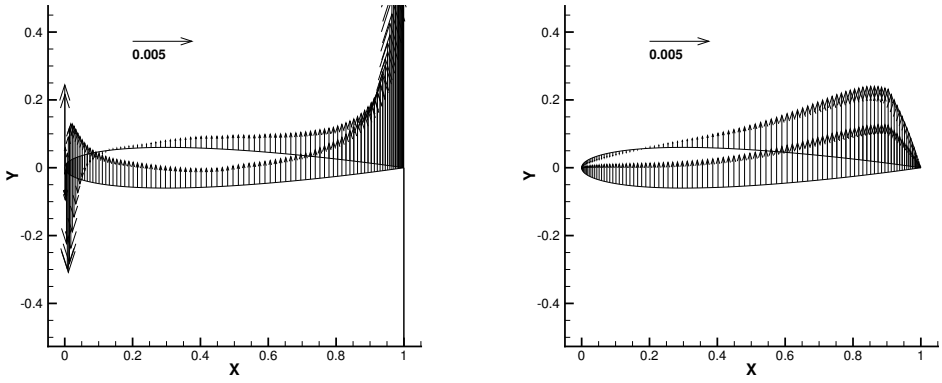
As discussed earlier, a necessary step to preserve a smooth geometry throughout the optimization process is to smooth the surface perturbations obtained at the end of each design cycle. The original unsmoothed perturbations are compared to the ones after smoothing and the results are presented as perturbation vectors in Figure 11. It can be clearly seen that the magnitude of perturbations are very large in the sharp trailing edge region. Also, both leading and trailing edges have very sharp node movements initially. After the smoothing, these perturbations become much smoother and their magnitudes get smaller, leading to a more gradual surface deformation. Additionally, the effect of the under-relaxation approach applied to the 10% chord length region close to the trailing edge can be seen in Figure 11. This under-relaxation approach has proven to be necessary for bounding the node movements close to the trailing edge in order to avoid extreme shape deformations.

Next, the bound constrained optimization test cases are considered. Here, the original unbounded test case is once again included for the sake of comparison. However, unlike the previous unbounded test case,



(a) Design cycle 7 (b) Design cycle 8 (c) Design cycle 9

Figure 10. Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.



(a) Unsmoothed perturbations (b) Smoothed perturbations

Figure 11. Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the NACA0012 airfoil geometry (lift-to-drag maximization case).

optimal line search settings are utilized here. Three bound limits are used for the constrained optimization test cases where the y-coordinates of the grid nodes are bounded by 10%, 20%, and 50% of their original value. The convergence history of the objective function for the unbounded and the three bound constrained tests are shown in Figure 12.

It is apparent that the relaxation in the bound limit improves the cost function. However, the significant reduction in the airfoil thickness can make the unconventional design unfit for manufacturing. Additionally, while the unbounded case results in a significantly improved design after 6 cycles, the value of the cost function drops dramatically right after this point. The convergence histories for the lift and drag coefficients are also presented in Figure 13.

While the objective for these optimization test cases was to maximize the lift-to-drag ratio, it can be seen from Figure 13 that the 10% bounded case results in an increase in the lift coefficient and a decrease in the drag coefficient which is exactly what is desired in aerodynamic shape optimization. The convergence histories for the lift-to-drag ratio, lift coefficient, and drag coefficient for this bounded case are plotted separately from the other optimization test cases and are shown in Figure 14.

Next, the comparison between the geometry as well as the surface pressure coefficients for the original

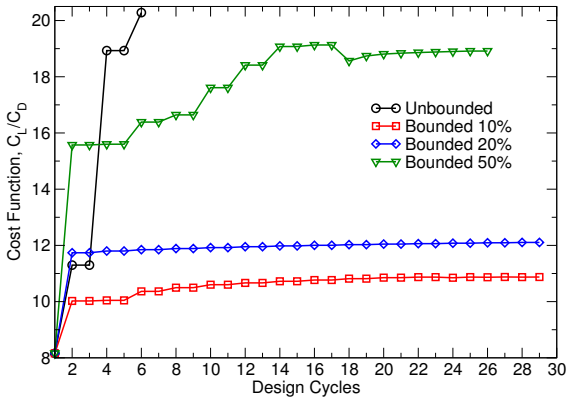
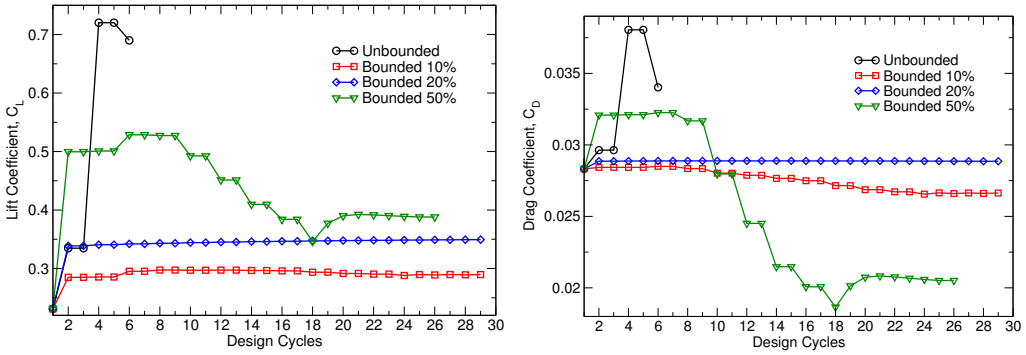
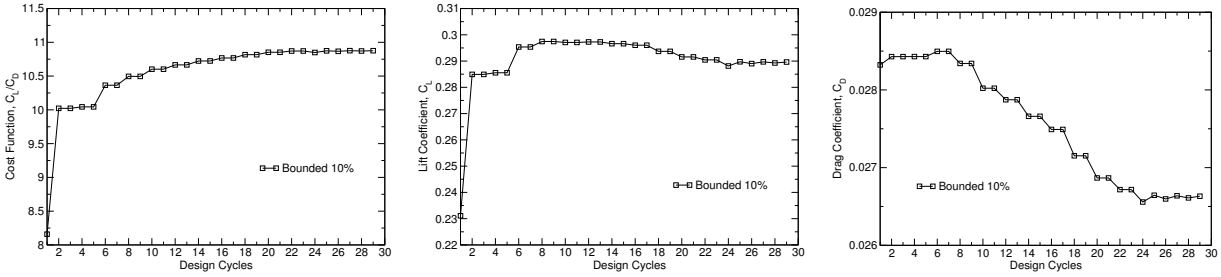


Figure 12. Convergence history of the lift-to-drag ratio for the bounded and unbounded efficiency optimization of NACA0012 airfoil.



(a) Lift coefficient, C_L (b) Drag coefficient, C_D

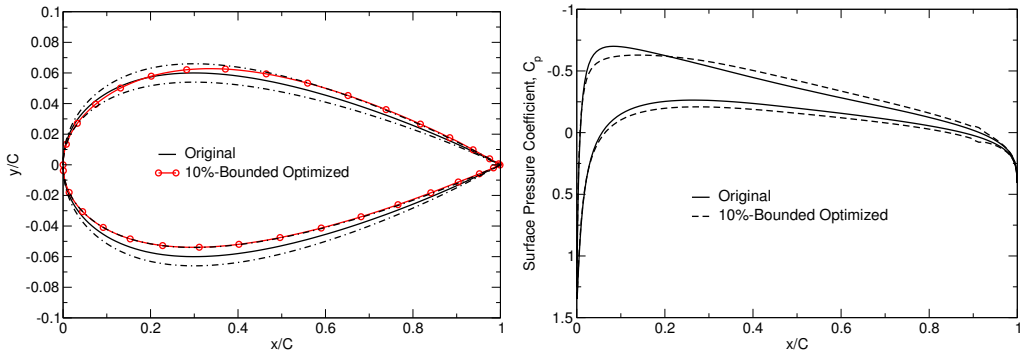
Figure 13. Convergence histories of the lift and drag coefficients for the bounded and unbounded efficiency optimization of NACA0012 airfoil.



(a) Objective, C_L/C_D (b) Lift coefficient, C_L (c) Drag coefficient, C_D

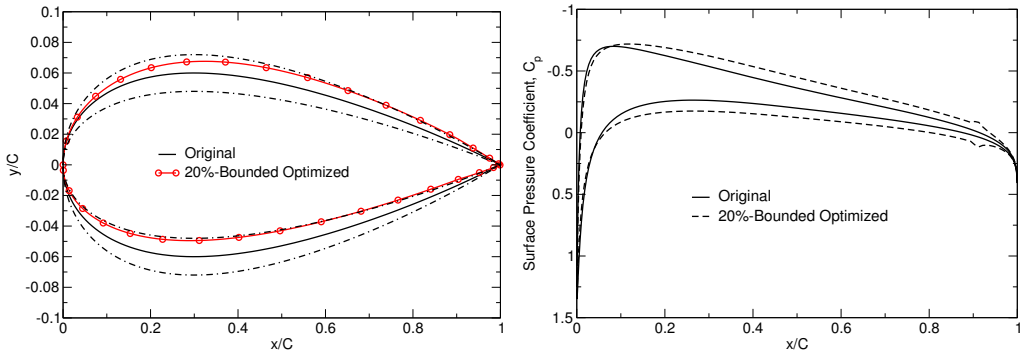
Figure 14. Convergence histories of the objective function, lift, and drag coefficients for the 10% bound constrained efficiency optimization of NACA0012 airfoil.

and optimized airfoils are shown in Figures 15 through 17 for the three bound constrained test cases studied here. As intended, in all three cases the final airfoil shape is bounded by the upper and lower bounds specified for the optimizer. Additionally, the movements in the trailing edge region are much larger in the case of 50%-bound since the node movements are much less limited for this case compared to the other two



(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 15. Comparison of airfoil shape and the surface pressure coefficients for the original and 10%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).



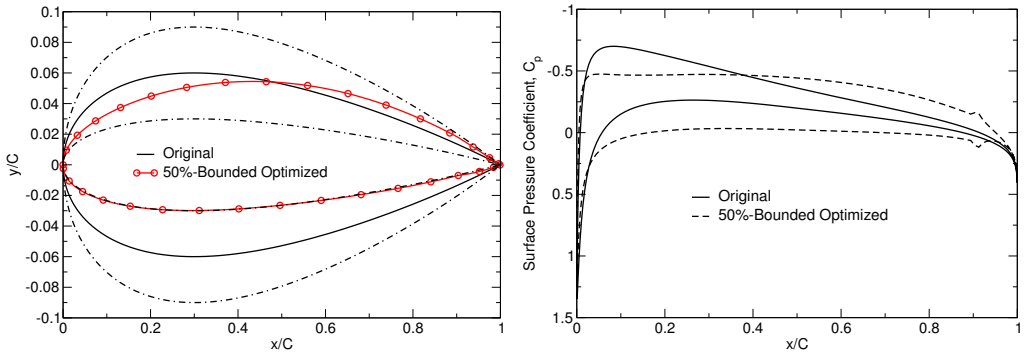
(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 16. Comparison of airfoil shape and the surface pressure coefficients for the original and 20%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).

bounded test cases.

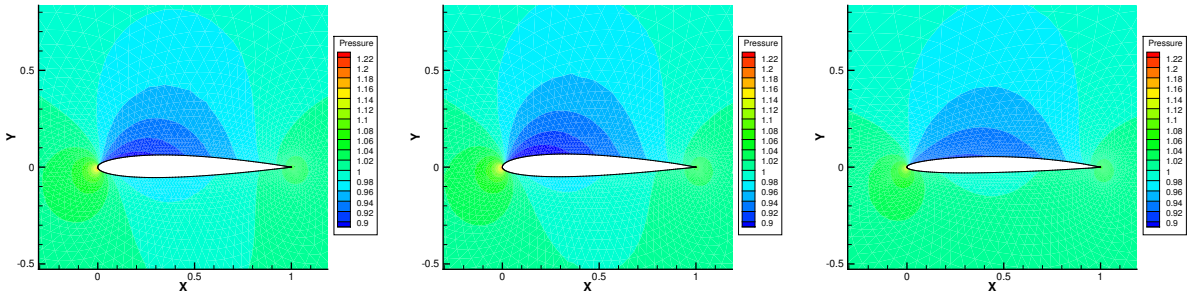
Additionally, the contour plots of pressure for the original, three optimized airfoils are shown in Figure 18. Once again, it is noted that after each design update, the existing computational mesh is deformed based on the topological changes on the surface. This approach eliminates the need to generate a new mesh after each design cycle. Similar to what is used in the r-adaptive mesh relocation technique,[32] the mesh deformation for design optimization is performed using the RBF approach with a support radius that can efficiently relocate grid nodes to conform the deformed geometry of the airfoil. As an example, the original and deformed grids for the 50%-bounded case are shown in Figure 19. Since no re-meshing is required in this approach, the quality of the original mesh is preserved. Additionally, for the turbulent test cases, the initial $y+$ value is retained throughout the design cycles which is necessary for accurate turbulent solutions. As shown earlier in this work, the present RBF-based mesh deformation approach is capable of efficiently relocating grid points without leading to any inverted cells.

Finally, it must be mentioned that the memory footprint of the adjoint solver for this test case is about 200 MBytes and the computational cost of running the adjoint solver in each design cycle is about 1.5 times that of the nominal flow solver.



(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 17. Comparison of airfoil shape and the surface pressure coefficients for the original and 50%-bounded optimized geometries (lower and upper bounds are shown in dashed lines).



(a) 10%-bounded optimized (b) 20%-bounded optimized (c) 50%-bounded optimized

Figure 18. Contour field of pressure for the inviscid subsonic flow past NACA0012 airfoil at $M = 0.5$ and AOA $= 2.0$ deg.

C. Drag Minimization: Transonic NACA0012 Airfoil

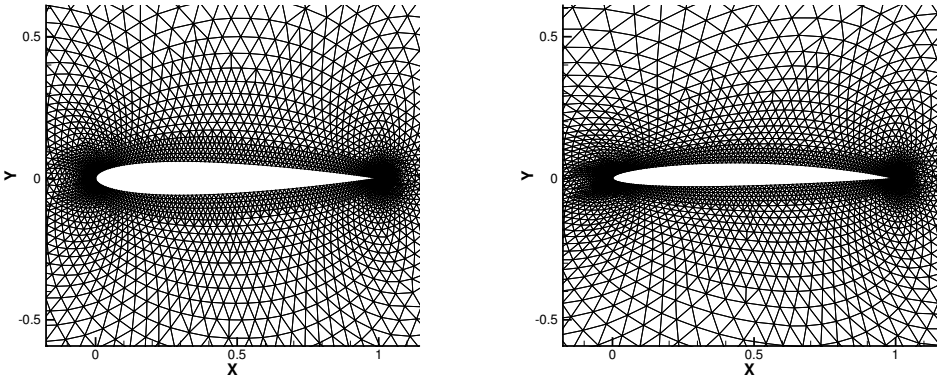
The next optimization test case considered here is the drag minimization of a NACA0012 airfoil subject to an inviscid transonic flow. This case has a free-stream Mach number of 0.8 and an angle of attack of 1.25 degrees. The same unstructured grid and solver settings used earlier are adopted here. The cost or the objective function is taken to be the drag coefficient, C_D, which is sought to be minimized and the design variables are taken to be the y-coordinates of the grid points on the surface of the airfoil. Additionally, only the unconstrained optimization is considered in this section.

First, the history of the objective function (C_D) is plotted for different design cycles in Figure 20. As can be seen, there is a steady drop in the drag with a 95% drop in the first 5 design cycles. Here, the optimization process is continued for 20 design cycles although most of the reduction in the drag coefficient has been achieved within the first 5 cycles.

The contour plots of pressure for the original and the optimized airfoils are shown in Figure 21. Also, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils are shown in Figure 22.

The inviscid transonic flow past the NACA0012 airfoil leads to the formation of a strong shock on the suction side and a weaker shock on the pressure side.[32] As can be seen in Figures 21 and 22, the drag minimization leads to the elimination of these shocks on both sides of the airfoil.

Due to the fact that the mesh nodes are used as the design variables, it is again necessary to perform



(a) Original geometry grid (b) Deformed geometry grid

Figure 19. Original and deformed grids obtained using the RBF technique for the NACA0012 airfoil in the 50%-bounded optimization test case.

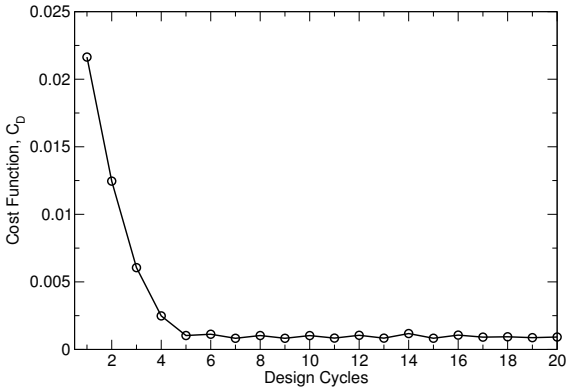
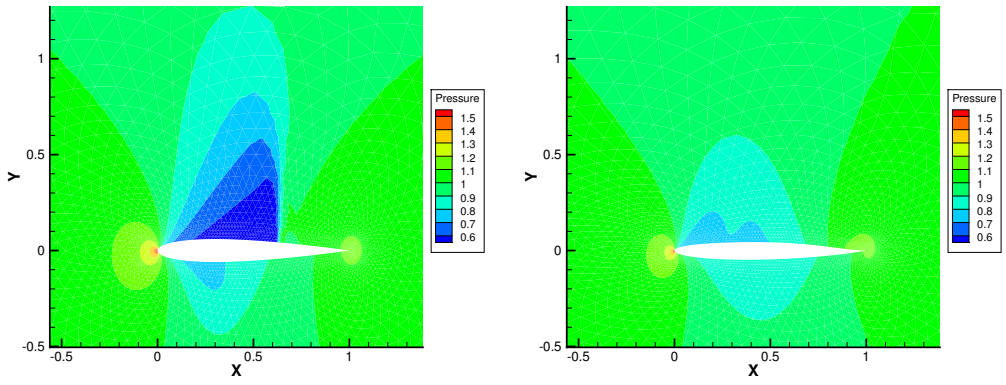


Figure 20. Convergence history of the objective function for the unbounded drag minimization of NACA0012 airfoil.

smoothing before applying the surface deformations. Therefore, the original unsmoothed perturbations are compared against the ones after smoothing iterations and the results are shown in terms of perturbation vectors in Figure 23. As can be seen, the unsmoothed perturbations have high frequency modes that can lead to jagged surface deformations. With the use of the smoothing procedure, these surface perturbations are smoothed significantly while the overall geometry deformation features are retained. Additionally, an under-relaxation is applied to the 10% chord length region close to the trailing edge, and its effects can be clearly seen in Figure 23. Again, this under-relaxation approach has proven to be necessary for bounding the node movements close to the sharp trailing edge in order to avoid non-physical shape deformations.

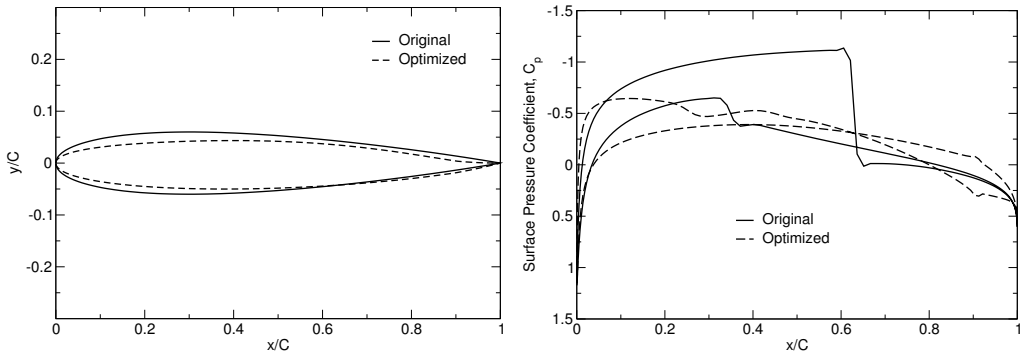
As described earlier in section A, the use of surface points as design variables is preferred for 2D test cases while shape parameterization using the FFD box technique is normally used for 3D cases. However, in order to compare the two approaches directly, the FFD box technique is also used for the present case. The 2D FFD box used here is defined by four corner nodes and is extended from $[-0.001, 1.0001]$ in the x-direction and from $[-0.2, 0.2]$ in the y-direction. Also, the orders of the Bézier curves in $x-$ and y-directions are taken to be NI = 20 and NJ = 1 which translates to 21 control points in the x-direction and 2 control points in the y-direction for a total of 42 control points.

Here, the same unconstrained optimization problem is considered to minimize the drag coefficient. The y coordinates of the FFD box control points are chosen as design variables. Additionally, in order to maintain the effective angle of attack, the leading and trailing edge points of the airfoil are fixed. That is, the FFD



(a) Original Design (b) Unbounded optimized

Figure 21. Contour field of pressure for the inviscid transonic flow past NACA0012 airfoil at $M = 0.8$ and AOA $= 1.25$ deg.

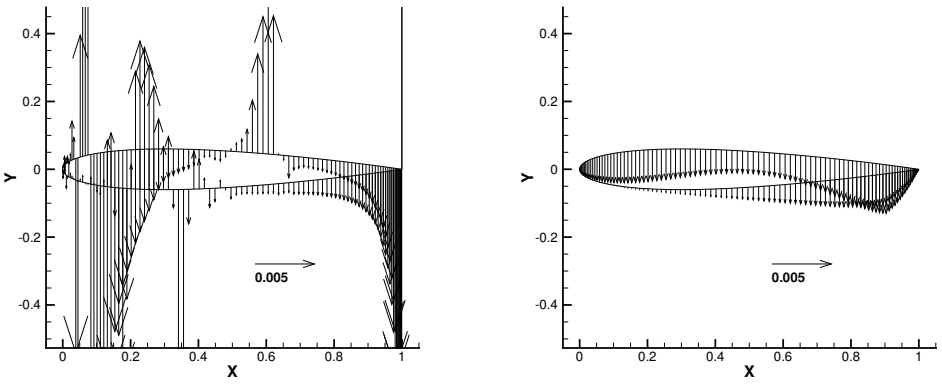


(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 22. Comparison of airfoil shape and the surface pressure coefficients for the original and optimized geometries.

box control points at $\xi = 0$ and $\xi = 1$ are kept fixed. This leads to a total of 38 design variables in the optimization as opposed to the 200 surface points used earlier. However, it must be noted again that the computational cost of the adjoint solver is essentially independent of the number of design variables. Initially, the history of the objective function (C_D) is plotted for major design cycles in Figure 24. Once again, an improvement in the drag coefficient is observed during the first 10 design cycles. The comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils is provided in Figure 25.

In order to compare the results of optimization from using the surface points as design variables or utilizing the FFD parameterization, the convergence of the cost function from the two techniques are plotted against each other (see Figure 26). As can be seen, the FFD parameterization approach leads to a slower reduction of the objective function (drag coefficient) compared to the case with surface points used as design variables. However, it must be noted that in the FFD box approach, displacements of the FFD control point are reduced (under-relaxed) in the first 10 cycles as the optimizer leads to large step sizes in the initial line search iterations. A very interesting finding from this optimization problem is that, while both approaches lead to the same minimized drag value, the case using the FFD parameterization achieves the same objective



(a) Unsmoothed perturbations

(b) Smoothed perturbations

Figure 23. Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the NACA0012 airfoil geometry (drag minimization case).

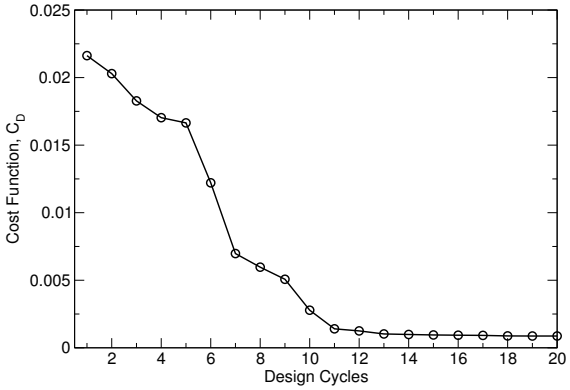


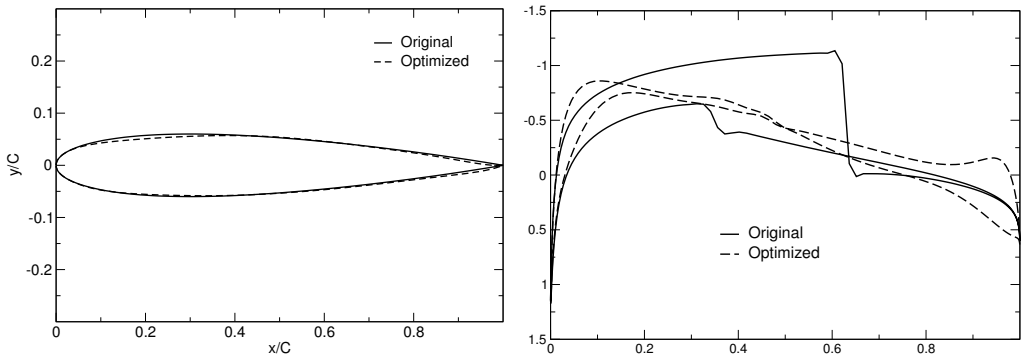
Figure 24. Convergence history of the objective function for the unbounded drag minimization of NACA0012 airfoil using the FFD parameterization approach.

Table 1. Drag coefficient, C_D, and maximum thickness, $t_{\max}/c$, comparisons for transonic NACA 0012 drag minimization problem.

Geometry	C_D	Reduction	$t_{\max}/c$	Reduction
Original	2.1638E-2	-	0.120	-
Optimized (Surface Points)	9.1755E-4	95.7%	0.093	22.5%
Optimized (FFD Box)	8.7265E-4	95.9%	0.115	4.2%

without drastically decreasing the maximum thickness of the NACA 0012 airfoil. As presented in Table 1, the optimization problem using the FFD parameterization approach leads to almost the same objective function (even slightly smaller drag coefficient) while only reducing the maximum thickness by 4.2% compared to the 22.5% thickness reduction in the former approach.

Additionally, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized airfoils (from both approaches) are shown in Figure 27. As can be seen, the optimized geometry using the FFD parameterization approach exhibits two suction regions on both sides of the airfoil close to the leading edge while both strong and weak shocks have disappeared. This phenomenon can be



(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 25. Comparison of airfoil shape and the surface pressure coefficients for the original and optimized geometries using the FFD parameterization approach.

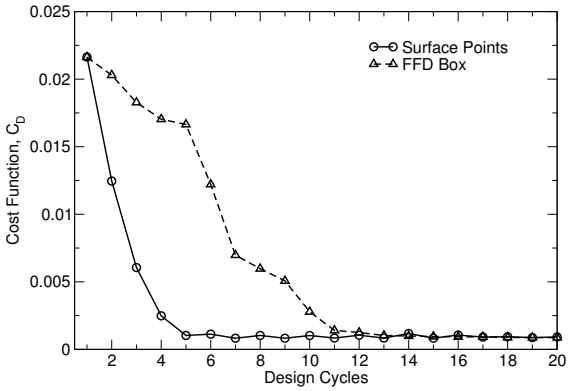
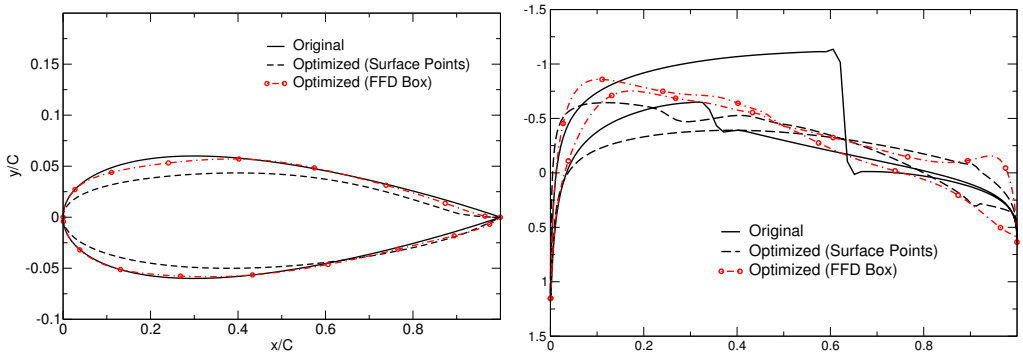


Figure 26. Comparison of the convergence histories of the objective function for the unbounded drag minimization of NACA0012 airfoil using the two different geometrical definition approaches.

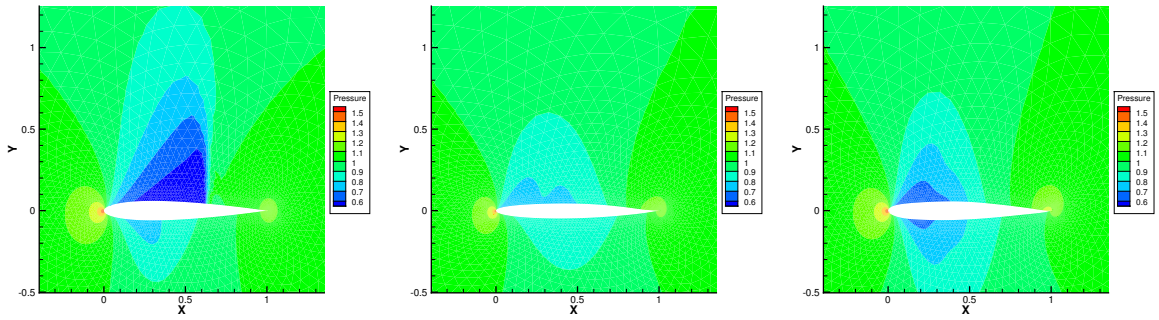
seen more clearly in the contour plots of the pressure field shown in Figure 28.



(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 27. Comparison of airfoil shape and the surface pressure coefficients for the original and optimized geometries using the FFD parameterization approach.

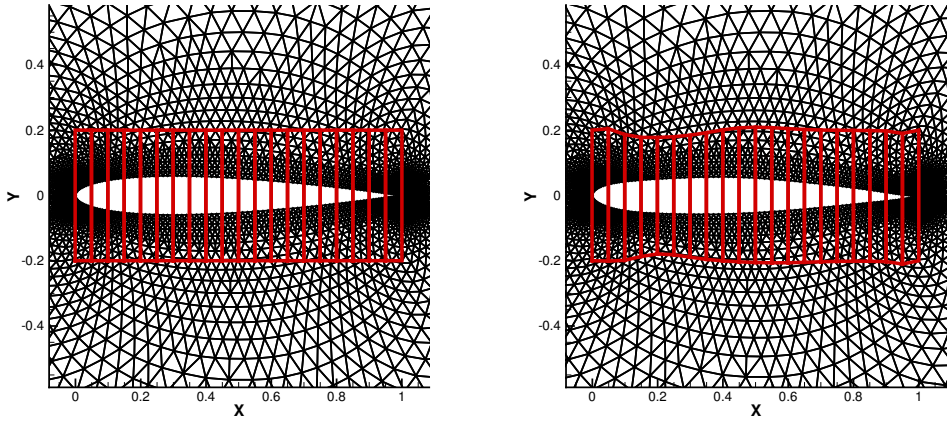
Finally, the FFD box deformation is presented in Figure 29 for this drag minimization test case. Maximum



(a) Original Design (b) Optimized (Surface Points) (c) Optimized (FFD Box)

Figure 28. Contour field of pressure for the inviscid transonic flow past NACA0012 airfoil at $M = 0.8$ and AOA $= 1.25$ deg.

deformations of the FFD box are observed around a quarter-chord aft of the airfoil leading edge. Here, the displacements of the FFD box control points are translated into geometry deformations, and consequently, a reduction in the drag coefficient. Once again, it must be noted that the memory footprint of the adjoint solver for this case is about 800 MBytes and the computational cost of running the adjoint solver in each design cycle is about 2.5× that of the nominal flow solver for the same number of iterations.



(a) Original Box (b) Deformed (Optimized) Box

Figure 29. Original and deformed FFD box that parameterizes the NACA 0012 airfoil for the transonic drag minimization problem.

D. Efficiency Maximization: NREL S809 Wind Turbine Blade Section

Next, the efficiency maximization of a wind turbine blade section in the viscous flow regime is considered. The National Renewable Energy Laboratory (NREL) Phase VI wind turbine has been widely used as a test bed for CFD analysis as well as design optimization.[57] This particular horizontal-axis wind turbine (HAWT) is made of two tapered-twisted blades with an S809 airfoil cross-section. This airfoil has been developed by Airfoils, Inc.[58] and has been optimized with the goal of maximizing power production. As a result of these optimization studies, the NREL Phase VI wind turbine blade has been designed with a linear taper and a nonlinear twist distribution along the span while using the S809 airfoil exclusively from the root all the way to the tip of the blade.

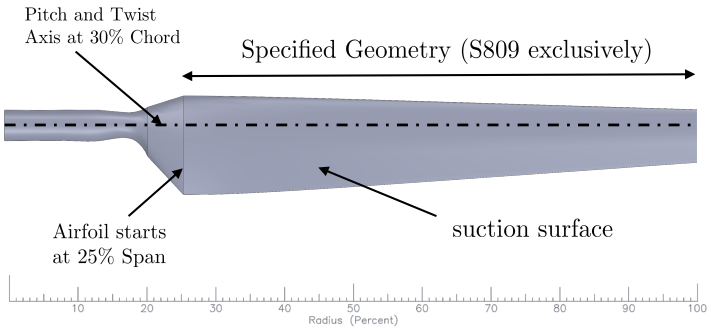
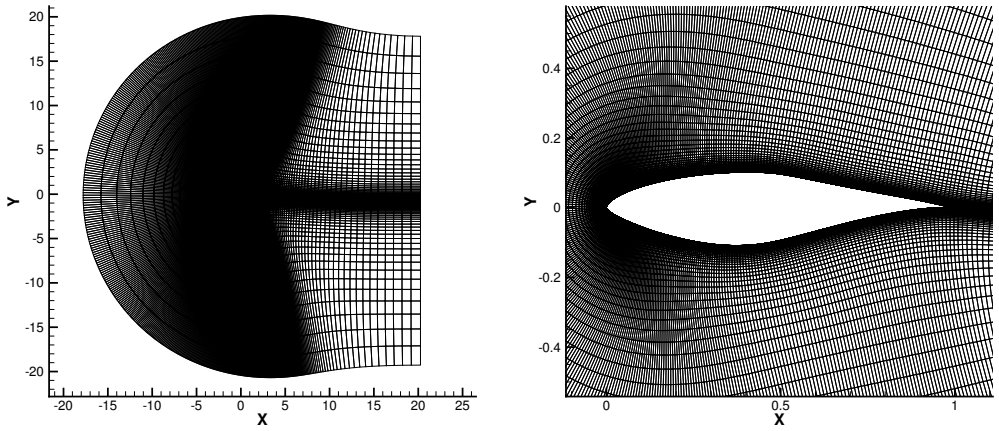


Figure 30. Geometry of the NREL Phase VI horizontal-axis wind turbine (HAWT) blade with S809 airfoil as blade cross-section.

The geometry of the NREL Phase VI wind turbine blade is shown in Figure 30. As can be seen, 75% of the span is covered with the standard S809 airfoils while 15% of the span length has a semi-S809 cross-section in an area that is formed by lofting an S809 airfoil into a circular section. The standard S809 airfoil has a 21% maximum thickness at 39.5% chord length and is designed as a laminar flow airfoil. Here, the goal is to maximize the efficiency (or the lift-to-drag ratio) of the S809 airfoil operating at a 4.1 degree angle of attack, a free-stream Mach number of $M_\infty = 0.1$ and a Reynolds number of $\mathrm{Re} = 1 \times 10^6$. The computational grid used for this study has a $y+$ value of less than 0.5 and is shown in Figure 31.



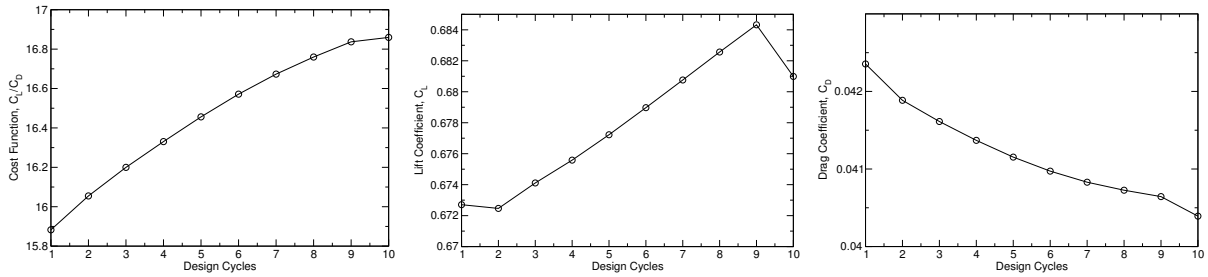
(a) far-field grid

(b) near-field grid

Figure 31. Computational grid used for the lift-to-drag maximization test case for the S809 airfoil.

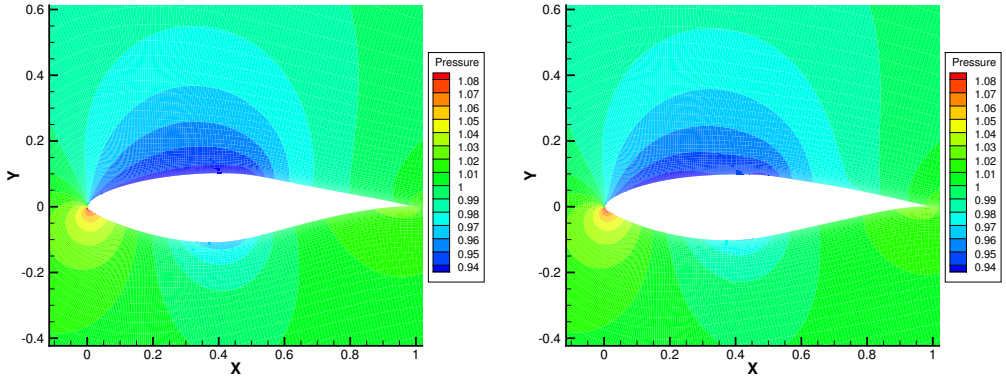
Here, an unconstrained optimization is considered and the objective function is once again taken to be $f = -C_L/C_D$ in order to maximize the lift-to-drag ratio. The optimization process is continued for 10 design cycles and the convergence histories of the objective function as well as the lift and drag coefficients are shown in Figure 32. As can be seen, there is a steady increase in the lift-to-drag ratio with a maximum of 6.2% increase in the efficiency. Similarly, the lift coefficient is increased 2.7% whereas the drag coefficient is decreased about 4.6% compared to the original values.

The contour plots of pressure for the original and optimized airfoils are presented next in Figure 33. Also, the comparison between the geometry as well as the surface pressure coefficients for the original and optimized S809 airfoils are shown in Figure 34. Here, the surface pressure distributions are also compared to the experimental data of Ramsay.[59] For this case, there is a 4.4% reduction in the maximum thickness



(a) Lift-to-drag ratio, C_L/C_D (b) Lift coefficient, C_L (c) Drag coefficient, C_D

Figure 32. Convergence histories of the lift-to-drag ratio as well as lift and drag coefficients for the unbounded efficiency maximization of the S809 airfoil.



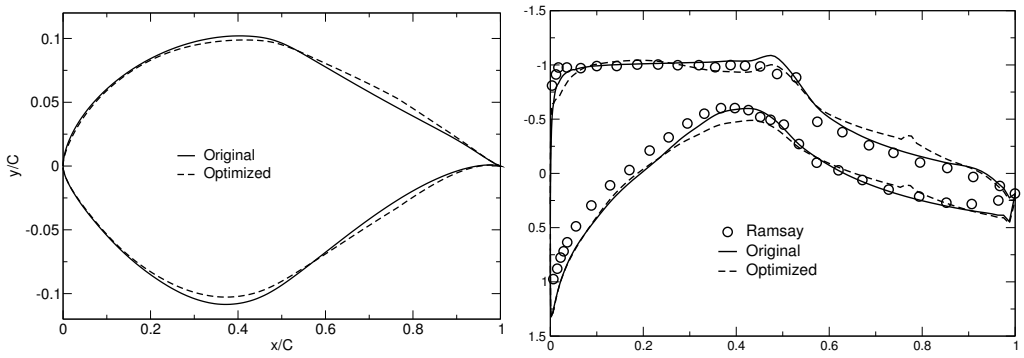
(a) Original Design (b) Unbounded optimized

Figure 33. Contour field of pressure for the turbulent flow past S809 airfoil at $M = 0.1$, $\mathbf{Re} = 10^6$, and AOA = 4.1 deg.

after 10 design cycles. Although an unconstrained optimization was used here, the reduction in the airfoil thickness is well within the desired bound limits.

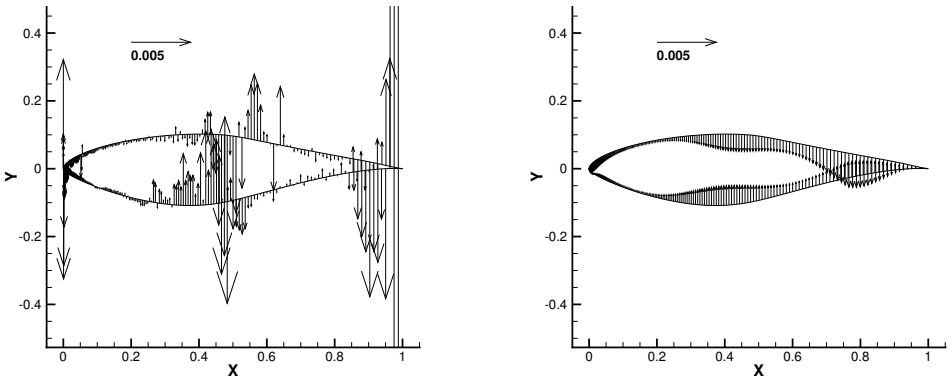
Similar to the previous optimization test cases, the effects of surface perturbation smoothing has been analyzed next. Here, the original unsmoothed perturbations are once again compared to the ones after smoothing iterations and the results are shown in terms of perturbation vectors in Figure 35. Although the magnitudes of the surface perturbations are not large for this case (compared to the previous test cases), there are high frequency modes involved which can lead to an unsmooth geometry. With the addition of the smoothing procedure, these surface perturbations are smoothed significantly while the overall geometry deformation features are retained. Additionally, the under-relaxation approach applied to the 10% chord length region close to the trailing edge has been at play and its effects can be clearly seen in Figure 35. The under-relaxation approach used in this work has proven to be necessary for bounding the node movements close to the sharp trailing edge while maintaining a smooth shape deformation near the trailing edge.

Finally, it must be noted that the memory footprint of the adjoint solver for this case is about 1.4 GBytes and the computational cost of running the adjoint solver in each design cycle is about 2.8 times that of the primal solver.



(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 34. Comparison of S809 airfoil shape and the surface pressure coefficients for the original and optimized geometries.



(a) Unsmoothed perturbations (b) Smoothed perturbations

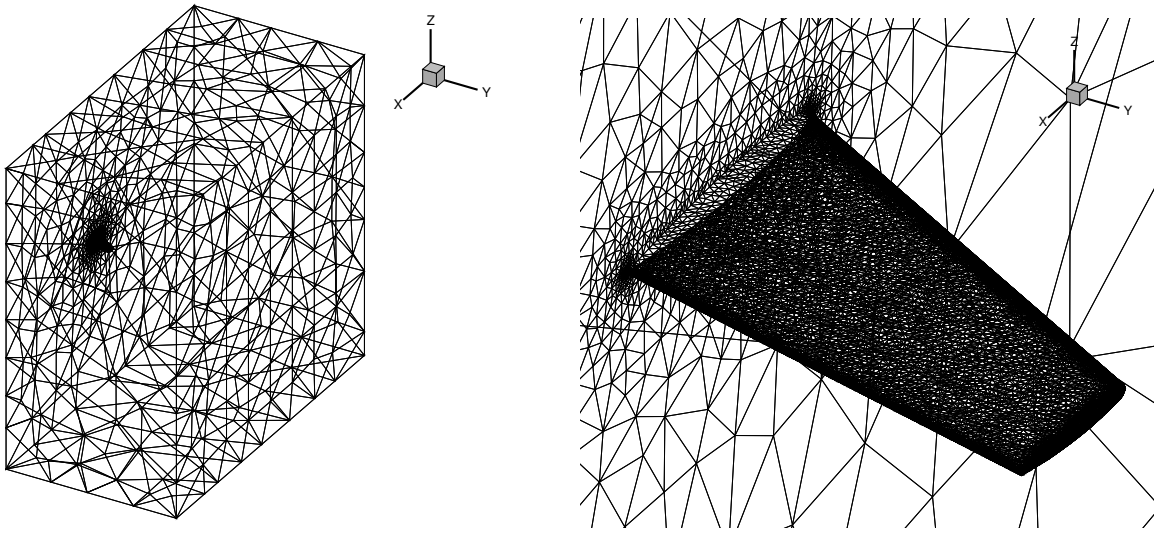
Figure 35. Comparison of surface perturbations for the original (unsmoothed) and the smoothed cases in vector notation at the first design cycle for the S809 airfoil geometry (lift-to-drag maximization case).

E. Drag Minimization: Transonic ONERA M6 Wing

As the last test case, the drag minimization problem for the inviscid transonic ONERA M6 wing is investigated. This configuration has been studied extensively and often used as a classical benchmark test case to validate three-dimensional CFD solvers. More recently, the drag minimization of the ONERA M6 wing has become a standard test case for aerodynamic shape optimization studies.[60–62]

The aerodynamics of this wing involve a region of supersonic flow as well as a special shock formation known as the lambda shock.[63,64] The geometry of the transonic M6 wing is based on the symmetric airfoil sections of type ONERA D which have a maximum thickness-to-chord ratio of 10%. The M6 wing has a sweep angle of 30 degrees at the leading edge and an aspect ratio of 3.8 and is tapered with a ratio of 0.562. The flow conditions are set according to the experiments carried out by Schmitt and Charpin[65] with a free-stream Mach number of 0.8395 and an angle of attack of 3.06°.

Here, a rectangular outer boundary that extends about 10 mean chord lengths on each side is used. The far-field and near-field views of the grid used for the ONERA M6 wing case are provided in Figure 36. This grid is made of 108,396 nodes and 582,752 tetrahedral elements. Also, on the surface of the wing there are 38,756 triangular elements. A symmetry boundary condition is used on the root-plane and far-field boundary conditions are used for the rest of the outer boundaries. Here, the inviscid (free-slip) wall boundary condition



(a) Computational domain with 582, 752 cells

(b) Surface mesh with 38, 756 triangles

Figure 36. Volume and surface meshes used for the transonic flow past ONERA M6 wing.

is imposed on the surface of the wing.

Numerical results obtained using the original ONERA M6 wing geometry are compared against the experimental data of Schmitt and Charpin.[65] These solutions are reported at 6 different sections along the span of the wing and the distribution of the surface pressure coefficient at each section are shown in Figure 37. As can be seen in Figure 37, there is a good agreement between the UNPAC solver results and the experimental data although the double shock formation at the 65% and 80% span is not captured accurately. This disagreement is very likely due to the relatively coarse mesh used for this case.

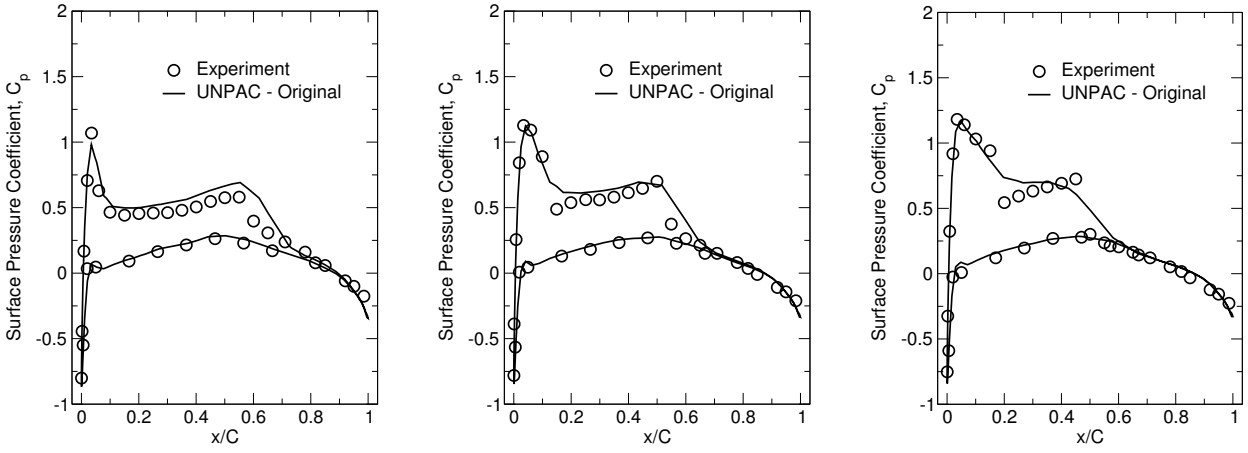
Next, the geometry is parameterized using a hexahedral Free Form Deformation (FFD) box that encloses the wing. Since the M6 wing has different sweep angles on the leading and trailing edges, the FFD box is also swept to tightly enclose the wing geometry. As a result, the FFD box is no longer a parallelepiped and the definitions of the parametric coordinates for the surface points need to be found using the search algorithm described earlier in section A. Additionally, the degrees of the Bézier curves in (ξ, η, ζ) directions are taken to be $(10, 8, 1)$ which translate to 11, 9, and 2 control points in each parametric coordinate. The history of the objective function (C_D) is plotted for major design cycles in Figure 38. As can be seen, a steady drop in the objective function is observed with the drag coefficient being reduced by almost 28% after 30 major design cycles. This result is also presented in Table 2.

Table 2. Drag coefficient, C_D, comparisons for transonic ONERA M6 drag minimization problem.

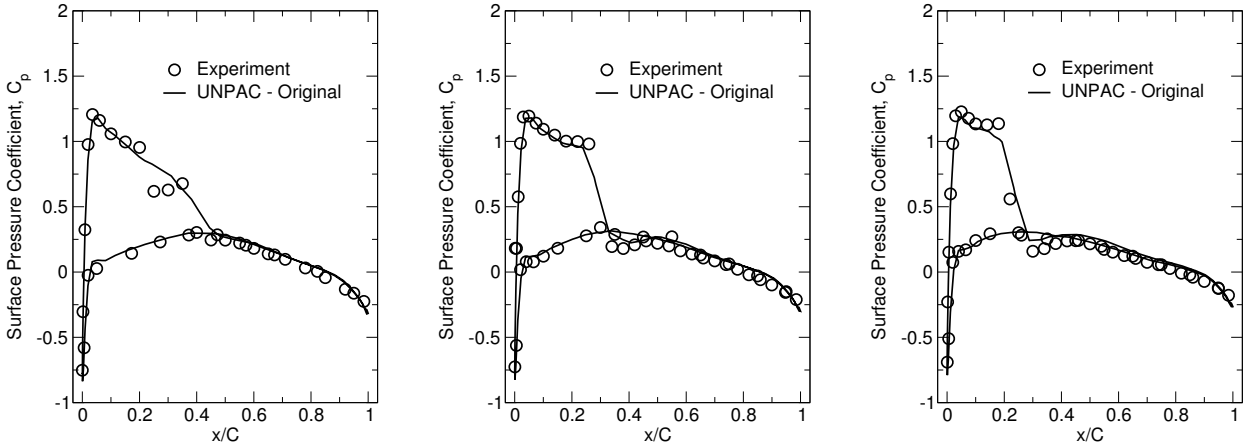
Geometry	C_D	Reduction
Original	1.1734E-2	-
Optimized	8.4601E-3	27.9%

Additionally, the surface solutions at last 3 span-wise sections are compared for the original and optimized geometries and the distribution of the surface pressure coefficients at each section are shown in Figure 39. As can be seen, the strong shock close to the tip of the wing is completely eliminated for the optimized geometry. This strong shock is the main contributor to the drag and its elimination has proved to drastically reduce the drag coefficient.

Next, the contour plots of the surface pressure are presented for the original and optimized M6 wing. These results are presented in Figure 40 and clearly show the elimination of the lambda-shock feature from the transonic ONERA M6 wing. Also, the original and deformed FFD boxes for this unconstrained drag



(a) 20% span (b) 44% span (c) 65% span



(d) 80% span (e) 90% span (f) 95% span

Figure 37. Surface pressure coefficients for the inviscid transonic flow past ONERA M6 wing compared to experimental data.[65]

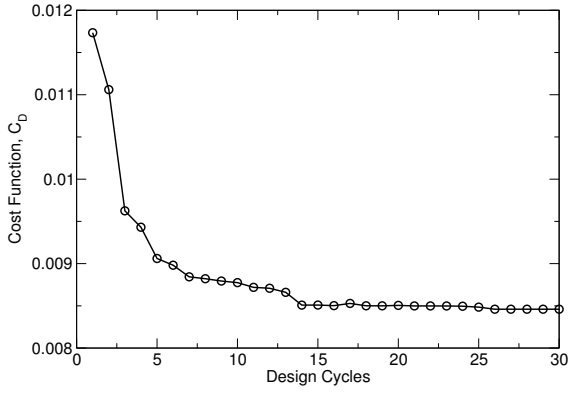
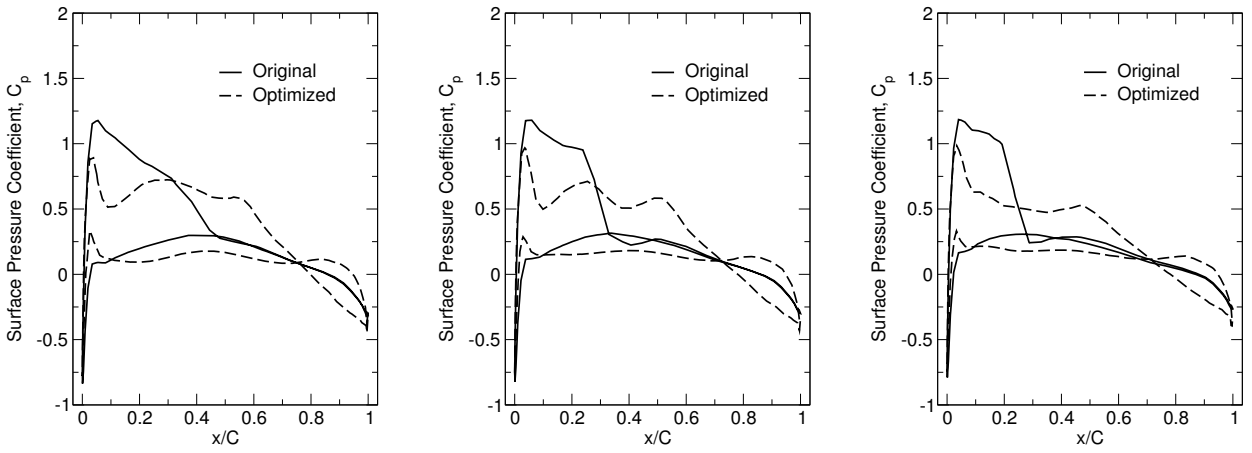


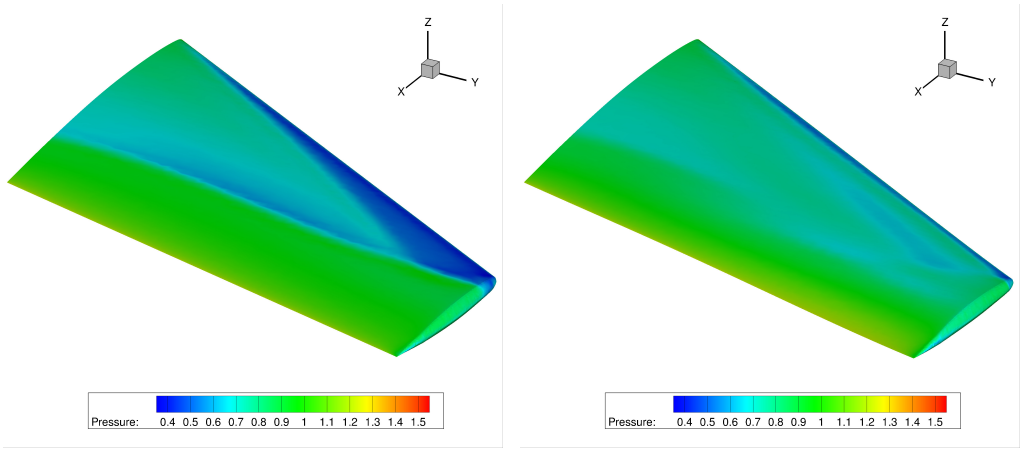
Figure 38. Convergence history of the objective function for the drag minimization of the ONERA M6 wing.



(a) 80% span (b) 90% span (c) 95% span

Figure 39. Comparison of the surface pressure coefficients between the original and optimized geometries for the drag minimization of the inviscid transonic ONERA M6 wing.

minimization problem are presented in Figure 41.



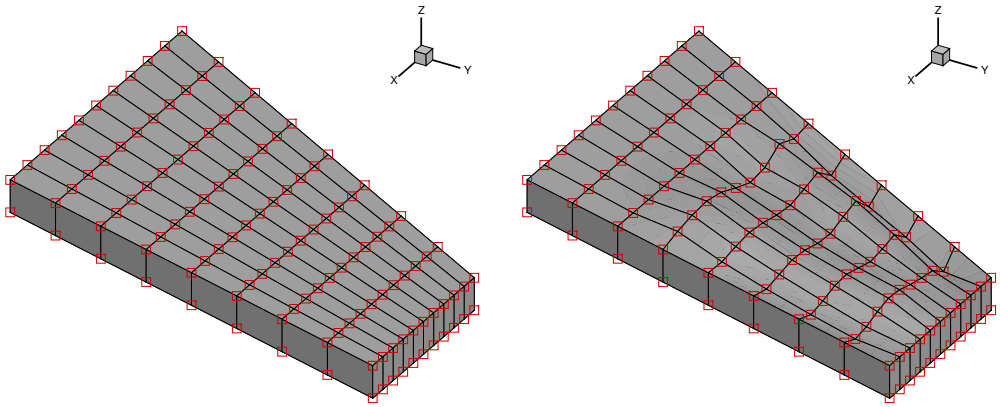
(a) Original Design (b) Optimized Design

Figure 40. Contour field of pressure on the surface of the ONERA M6 wing for the unconstrained drag minimization problem.

It must be noted that for this case with 650,376 degrees-of-freedom (DOF), the size of the recorded tape is about 14.1 GBytes. Also, the CPU time for the adjoint solver is around 4.2 times that of the primal CFD solver which, once again, proves the computational efficiency of the FDOT toolbox for adjoint-based sensitivity analysis (see Table 3).

VII. Conclusions

In this work, the **UN**structured **PA**rallel **C**ompressible **D**esign **O**ptimization **F**ramework (UNPAC-DOF) is developed. The adjoint computations are performed using a novel and fully-automated sensitivity analysis toolbox, FDOT. This toolbox is written in Fortran and is based on an operator-overloading version of the automatic differentiation. The primal and adjoint solvers are coupled together as part of a wrapper program



(a) Original Box (b) Deformed (Optimized) Box

Figure 41. The original and deformed (optimized) geometry of the FFD box parameterizing the ONERA M6 wing for the drag minimization problem.

Table 3. CPU timings and memory footprints for the primal and adjoint solvers, inviscid transonic flow past ONERA M6 wing.

Solver Type	CPU Time (m)	Normalized CPU Time	Memory (Tape)
Primal (CFD)	17.21	1.0	-
FDOT	72.28	**4.2**	14.1 GB

that handles the aerodynamic shape optimization procedure. The adjoint solver uses the fully-converged CFD solution to determine the gradient information using the discrete adjoint method. The values of the design variables as well as the sensitivities of the objective function with respect to these design variables are passe to the gradient-based L-BFGS-B optimizer to update the design. The developed design optimization framework is applied to various 2D and 3D test cases involving the NACA 0012 and the S809 airfoils as well as the ONERA M6 wing. In all test cases studied, the relative cost of the adjoint solver is found to be around 2-4 times that of the primal (CFD) solver. Moreover, the adjoint solver has a manageable memory footprint which can be easily handled by most modern personal computers.

VIII. Acknowledgments

This material is based upon work supported by the National Science Foundation under grants No: CBET-1803760 and No: CBET-1150332. The authors greatly appreciate the support provided.

References

[1]Martins, J. R., Sturdza, P., and Alonso, J. J., "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 29, No. 3, 2003, pp. 245–262.

[2]Pironneau, O., "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 01, 1974, pp. 97–110.

[3]Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[4]Elliott, J. and Peraire, J., "Practical three-dimensional aerodynamic design and optimization using unstructured meshes," *AIAA Journal*, Vol. 35, No. 9, 1997, pp. 1479–1485.

[5]Nadarajah, S. and Jameson, A., "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization," AIAA Paper 2000-0667, 2000.

[6]Giles, M. B., Duta, M. C., Müller, J.-D., and Pierce, N. A., "Algorithm developments for discrete adjoint methods,"

AIAA Journal, Vol. 41, No. 2, 2003, pp. 198–205.

[7]Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation," *Computers & Fluids*, Vol. 28, No. 4, 1999, pp. 443–480.

[8]Kirn, S., Alonso, J. J., and Jameson, A., "Design optimization of high-lift configurations using a viscous continuous adjoint method," AIAA Paper 2002-0844, 2002.

[9]Giles, M. B. and Pierce, N. A., "An introduction to the adjoint approach to design," *Flow, turbulence and combustion*, Vol. 65, No. 3-4, 2000, pp. 393–415.

[10]Naumann, U., Utke, J., Heimbach, P., Hill, C., Ozyurt, D., Wunsch, C., Fagan, M., Tallent, N., and Strout, M., "Adjoint code by source transformation with OpenAD/F," *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics, Egmond aan Zee, The Netherlands, September 5-8, 2006*, Delft University of Technology; European Community on Computational Methods in Applied Sciences (ECCOMAS), 2006.

[11]Hascoet, L. and Pascual, V., "The Tapenade Automatic Differentiation tool: Principles, model, and specification," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 39, No. 3, 2013, pp. 20.

[12]Giering, R. and Kaminski, T., "Recipes for adjoint code construction," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 24, No. 4, 1998, pp. 437–474.

[13]Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P., "ADIFOR–generating derivative codes from Fortran programs," *Scientific Programming*, Vol. 1, No. 1, 1992, pp. 11–29.

[14]Griewank, A., Juedes, D., and Utke, J., "Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 22, No. 2, 1996, pp. 131–167.

[15]Hogan, R. J., "Fast reverse-mode automatic differentiation using expression templates in C++," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 40, No. 4, 2014, pp. 26.

[16]Bell, B. M., "CppAD: A package for C++ algorithmic differentiation," *Computational Infrastructure for Operations Research*, Vol. 57, 2012.

[17]Albring, T., Sagebaum, M., and Gauger, N. R., "Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework," AIAA Paper 2015-3240, 2015.

[18]Griewank, A. and Walther, A., *Evaluating derivatives: principles and techniques of algorithmic differentiation*, Vol. 105, Siam, 2008.

[19]Straka, C. W., "ADF95: Tool for automatic differentiation of a FORTRAN code designed for large numbers of independent variables," *Computer Physics Communications*, Vol. 168, No. 2, 2005, pp. 123–139.

[20]Shiriaev, D. and Griewank, A., "ADOL-F: Automatic differentiation of Fortran codes," *Computational Differentiation: Techniques, Applications, and Tools*, 1996, pp. 375–384.

[21]Stamatiadis, S., Prosmiti, R., and Farantos, S., "AUTO_DERIV: Tool for automatic differentiation of a FORTRAN code," *Computer Physics Communications*, Vol. 127, No. 2, 2000, pp. 343–355.

[22]Yu, W. and Blair, M., "DNAD, a simple tool for automatic differentiation of Fortran codes using dual numbers," *Computer Physics Communications*, Vol. 184, No. 5, 2013, pp. 1446–1452.

[23]Naumann, U., *The art of differentiating computer programs: An introduction to algorithmic differentiation*, Vol. 24, SIAM, 2012.

[24]Christianson, B., "Reverse accumulation and attractive fixed points," *Optimization Methods and Software*, Vol. 3, No. 4, 1994, pp. 311–326.

[25]Christianson, B., "Reverse accumulation and implicit functions," *Optimization Methods and Software*, Vol. 9, No. 4, 1998, pp. 307–322.

[26]Thomas, J., Dowell, E., and Hall, K. C., "Discrete adjoint method for nonlinear aeroelastic sensitivities for compressible and viscous flows," AIAA Paper 2013-1860, 2013.

[27]Huang, H. and Ekici, K., "An efficient harmonic balance method for unsteady flows in cascades," *Aerospace Science and Technology*, Vol. 29, No. 1, 2013, pp. 144–154.

[28]Huang, H. and Ekici, K., "A discrete adjoint harmonic balance method for turbomachinery shape optimization," *Aerospace Science and Technology*, Vol. 39, 2014, pp. 481–490.

[29]Walther, A. and Griewank, A., "Getting Started with ADOL-C." *Combinatorial Scientific Computing*, 2009, pp. 181–202.

[30]Sagebaum, M., Albring, T., and Gauger, N. R., "High-Performance Derivative Computations using CoDiPack," *arXiv preprint arXiv:1709.07229*, 2017.

[31]Djeddi, R. and Ekici, K., "FDOT: A Fast, Memory-Efficient and Automated Approach for Discrete Adjoint Sensitivity Analysis using the Operator Overloading Technique," *Aerospace Science and Technology*, 2019 article in press.

[32]Djeddi, S., *Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks*, Ph.D. thesis, University of Tennessee, 2018.

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

[34]Blazek, J., *Computational fluid dynamics: principles and applications*, Butterworth-Heinemann, Oxford, UK, 2015.

[35]Djeddi, R. and Ekici, K., "An Adaptive Mesh Redistribution Approach for Time-Spectral/Harmonic-Balance Flow Solvers," AIAA Paper 2018-3245, 2018.

[36]Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[37]Barth, T. J. and Jespersen, D. C., "The design and application of upwind schemes on unstructured meshes," AIAA Paper 1989-0366, 1989.

[38]Venkatakrishnan, V., "Convergence to steady state solutions of the Euler equations on unstructured grids with limiters," *Journal of Computational Physics*, Vol. 118, No. 1, 1995, pp. 120–130.

[39]Zhao, L. and Zhang, C., "A Parallel Unstructured Finite-Volume Method for All-Speed Flows," *Numerical Heat Transfer, Part B: Fundamentals*, Vol. 65, No. 4, 2014, pp. 336–358.

[40]Karypis, G. and Kumar, V., "A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, 1998.

[41]Wolfe, P., "Checking the calculation of gradients," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 8, No. 4, 1982, pp. 337–343.

[42]Baur, W. and Strassen, V., "The complexity of partial derivatives," *Theoretical Computer Science*, Vol. 22, No. 3, 1983, pp. 317–330.

[43]Mani, K., *Application of the Discrete Adjoint Method to Coupled Multidisciplinary Unsteady Flow Problems for Error Estimation and Optimization*, Ph.D. thesis, University of Wyoming, 2009.

[44]Hicks, R. M. and Henne, P. A., "Wing design by numerical optimization," *Journal of Aircraft*, Vol. 15, No. 7, 1978, pp. 407–412.

[45]Lépine, J., Guibault, F., Trépanier, J.-Y., and Pépin, F., "Optimized nonuniform rational B-Spline geometrical representation for aerodynamic design of wings," *AIAA Journal*, Vol. 39, No. 11, 2001, pp. 2033–2041.

[46]Chauhan, D., Chandrashekarappa, P., and Duvigneau, R., "Wing shape optimization using FFD and twist parameterization," *12th Aerospace Society of India CFD Symposium*, 2010.

[47]Jameson, A., "Aerodynamic shape optimization using the adjoint method," *Lectures at the Von Karman Institute, Brussels*, 2003.

[48]Castonguay, P. and Nadarajah, S., "Effect of shape parameterization on aerodynamic shape optimization," AIAA Paper 2007-59, 2007.

[49]Wendland, H., "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in Computational Mathematics*, Vol. 4, No. 1, 1995, pp. 389–396.

[50]De Boer, A., Van der Schoot, M., and Bijl, H., "Mesh deformation based on radial basis function interpolation," *Computers & Structures*, Vol. 85, No. 11, 2007, pp. 784–795.

[51]Rendall, T. C. and Allen, C. B., "Efficient mesh motion using radial basis functions with data reduction algorithms," *Journal of Computational Physics*, Vol. 228, No. 17, 2009, pp. 6231–6249.

[52]Djeddi, R. and Ekici, K., "Solution-based adaptive mesh redistribution applied to harmonic balance solvers," *Aerospace Science and Technology*, Vol. 84, 2019, pp. 543–564.

[53]Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C., "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, Vol. 16, No. 5, 1995, pp. 1190–1208.

[54]Liu, D. C. and Nocedal, J., "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, Vol. 45, No. 1-3, 1989, pp. 503–528.

[55]Barche, J. et al., "Experimental data base for computer program assessment," AGARD-report AGARD-AR-138, North Atlantic Treaty Organization, Advisory Group for Aerospace Research and Development, 1979.

[56]Carlson, L. A., Ratcliff, R. R., Gally, T. A., and Campbell, R. L., "Inverse wing design in transonic flow including viscous interaction," NASA Report N89-20947, 1989.

[57]Simms, D., Schreck, S., Hand, M., and Fingersh, L. J., "NREL unsteady aerodynamics experiment in the NASA-Ames wind tunnel: A comparison of predictions to measurements," *National Renewable Energy Laboratory Colorado, USA*, 2001.

[58]Somers, D. M., "Design and experimental results for the S809 airfoil," Tech. rep., National Renewable Energy Lab., Golden, CO (United States), 1997.

[59]Ramsay, R., Hoffman, M., and Gregorek, G., "Effects of Grid Roughness and Pitch Oscillations on the S809 Airfoil," *Technical Paper 442-7817, NREL*, 1995.

[60]Reuther, J. and Jameson, A., "Aerodynamic shape optimization of wing and wing-body configurations using control theory," AIAA Paper 1995-0123, 1995.

[61]Nielsen, E. J. and Anderson, W. K., "Recent improvements in aerodynamic design optimization on unstructured meshes," *AIAA journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[62]Leung, T. M. and Zingg, D. W., "Aerodynamic shape optimization of wings using a parallel newton-krylov approach," *AIAA journal*, Vol. 50, No. 3, 2012, pp. 540–550.

[63]Ekici, K., Hall, K. C., and Dowell, E. H., "Computationally fast harmonic balance methods for unsteady aerodynamic predictions of helicopter rotors," *Journal of Computational Physics*, Vol. 227, No. 12, 2008, pp. 6206–6225.

[64]Howison, J. C., *Aeroelastic Analysis of a Wind Turbine Blade Using the Harmonic Balance Method*, Ph.D. thesis, University of Tennessee, 2015.

[65]Schmitt, V. and Charpin, F., "Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers," *Experimental Data Base for Computer Program Assessment*, Vol. 4, 1979.