AIAA Science and Technology Forum and Exposition (AIAA SciTech Forum), 6-10 January 2020, Orlando, FL

Memory-Efficient Adjoint Sensitivity Analysis for Aerodynamic Shape Optimization

Reza Djeddi* and Kivanc Ekici†

University of Tennessee, Knoxville, Tennessee 37996

An improved version of the Fast automatic Differentiation using Operator-overloading Technique (FDOT) toolbox is developed in this work. The enhanced sensitivity analysis toolbox utilizes an expression-based tape approach – a first-of-its-kind implementation in Fortran programming language – that can significantly reduce the memory footprint while improving the computational efficiency of the adjoint-based automatic differentiation (AD). In the proposed approach, the partial derivatives are calculated for each expression using the reverse adjoint accumulation for the active variables involved on the right-hand-side of that expression. The recorded partial derivative information is then used in a very efficient adjoint evaluation process to calculate the entire Jacobian information. The enhanced toolbox is coupled with the in-house UNstructured PArallel Compressible (UNPAC) flow solver for a robust design optimization framework, called UNPAC-DOF. The efficiency and robustness of the proposed technique and the resulting framework are tested for aerodynamic shape optimization problems applied to airfoil and wing geometries.

I. Introduction

With the advances in computational science and technologies, aerodynamic shape optimization (ASO) has become a viable tool for designing efficient systems that involve optimized topologies satisfying certain aerodynamic objectives. In particular, gradient-based design optimization techniques that rely on accurate gradient or sensitivity information have grown in popularity due to their robustness in determining an "optimal" solution in a handful design cycles. However, developing an efficient gradient-based design optimization framework in terms of computational and memory requirements can be very challenging.

The introduction of the adjoint methods[1,2] in control theory and their application to fluid dynamics problems[3] has provided an efficient tool for gradient and sensitivity calculations. The adjoint method can be implemented in two different forms depending on the order with which the discretization and variation steps are performed. In the continuous adjoint approach[4,5] the variations are taken first before discretizing the resulting set of equations while in the discrete adjoint approach[6,7] the process is reversed. Of particular interest in the present work is the discrete adjoint method where the cost function is first augmented with the flow equations – using the Lagrange multipliers – before taking their variations. The latter step is generally performed using automatic or algorithmic differentiation (AD) which systematically applies the chain rule of differentiation to the discretized equations.

Depending on the direction in which the derivatives are propagated, the automatic differentiation can be performed in (1) forward/tangent or (2) reverse/adjoint modes. It must be noted that both modes of AD provide non-approximative and highly accurate gradient information for all design variables involved in the optimization problem. In the present context, we focus on the reverse mode of AD as its computational cost is around three to five times that of the primal solver, i.e., the flow solver, and independent of the number of design variables. However, the reverse AD would require the reversal of the entire expression tree for the primal solver which makes it more challenging to implement. In terms of programming, the reverse

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

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

AD can be performed using source code transformation (SCT) or operator overloading (OO) with the latter technique being the focus of the present work. Interested reader is referred to Ref.[8] for the details of both approaches.

In the OO/AD approach, the entire expression tree is recorded in a derived type class called the *tape* which is used at a later stage for adjoint evaluations. Many OO/AD tools are available depending on the programming language that is used with ADOL-C,[9] Adept,[10] CppAD,[11] and CoDiPack[12] being the most commonly used tools for C/C++ programs and ADF,[13] ADOL-F,[14] AUTO_DERIV,[15] DNAD,[16] and dco/fortran[17] for Fortran programs. More recently, Djeddi and Ekici[18] have developed a novel OO/AD toolbox, called FDOT, for automatically differentiating Fortran programs. The Fast automatic Differentiation toolbox based on Operator-overloading Technique (FDOT) uses the operator overloading capabilities of the modern Fortran language to provide an efficient and fully-automated framework for gradient and sensitivity calculations. Coupled with an in-house Computational Fluid Dynamics (CFD) solver, the FDOT toolbox has been effectively used in gradient-based optimization problems for designing optimal airfoil and wing topologies.[19]

It is worth noting that our in-house FDOT toolbox is the only available OO/AD tool with advanced memory handling for Fortran programs and it is proven to be very efficient in the use of computational and memory resources.[18,19] However, due to the limitations of the Fortran programming language, FDOT is at a disadvantage compared to more advanced OO/AD tools developed for C/C++ programs such as ADOL-C[9] and CoDiPack.[12] More specifically, the use of "expression templates" as a metaprogramming paradigm in C++ language has enabled ADOL-C and CoDiPack to significantly reduce the memory footprint associated with the operator overloading-based automatic differentiation.

In this work, a modified and more robust version of the FDOT toolbox is developed. This enhanced version of FDOT utilizes a novel expression-based tape approach, which greatly improves the memory and computational efficiency of the OO/AD toolbox. The improved toolbox is then incorporated into a design optimization framework, called UNPAC-DOF,[19] to efficiently and accurately calculate the sensitivity information of a cost function with respect to any design variable. Together with our in-house CFD solver as well as a gradient-based optimization algorithm, UNPAC-DOF offers a robust aerodynamic shape optimization framework for improving the design of airfoils and wings. In the following sections, details of the novel approach utilized in 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.

II. Discrete Adjoint-Based Sensitivity Analysis

As discussed in the previous section, the FDOT toolbox developed by Djeddi and Ekici[18] is capable of efficient and accurate evaluation of the gradient information for any given primal solver. Simply, it utilizes the concept of discrete adjoint sensitivity analysis and the object-oriented programming paradigms. At its core, the FDOT toolbox introduces a new derived type for real-typed variables, called **AReal**, while overloading all unary and binary operations and intrinsic functions handling any combination of **AReal** and other types of variables. This enables one to couple any numerical solver with the FDOT toolbox to obtain the gradients or sensitivities of the output (objective) function(s) with respect to all independent (design) or intermediate variables. The process of gradient calculation involves an "adjoint evaluation" process and its computational cost is only a small multiple of that of the primal solver.[20,21]

Due to the fact that FDOT utilizes the reverse mode of automatic differentiation, the derivatives are propagated in the reverse direction compared to the primal flow solver. Normally, this would require the complete time history of the primal flow equations to be stored in the memory as what is often called the *tape*. During the adjoint evaluation process, the recorded tape is executed in the reverse order while the derivatives are accumulated based on the recorded adjoint information. This process is common to almost all OO/AD tools and is a significant factor that affects the efficiency and efficacy of the automatic differentiation toolbox. More specifically, the memory footprint of the recorded tape can become intractable for a large scale three-dimensional flow solver that involves a significant number of expressions being executed at run time.

To the best of the authors' knowledge, FDOT is the first OO/AD tool developed for Fortran programming language that reduces the memory requirements by incorporating an iterative process similar to a fixed-point iteration approach originally proposed by Christianson.[22,23] For a successful implementation of this idea, the FDOT toolbox takes advantage of the fact that most CFD solvers are made up of three major parts

including (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. Since the iterative stage is known to be the most complex part of any CFD solver, using a fixed-point iteration approach can eliminate the need for the repeated evaluations of the flow solution and recording unnecessary information in the *tape*. Therefore, the memory footprint of the tape can be greatly reduced by recording only a single pass of the adjoint solver provided that the CFD solver is fully converged. This iterative approach for adjoint accumulation is described in more details in a previous work.[19]

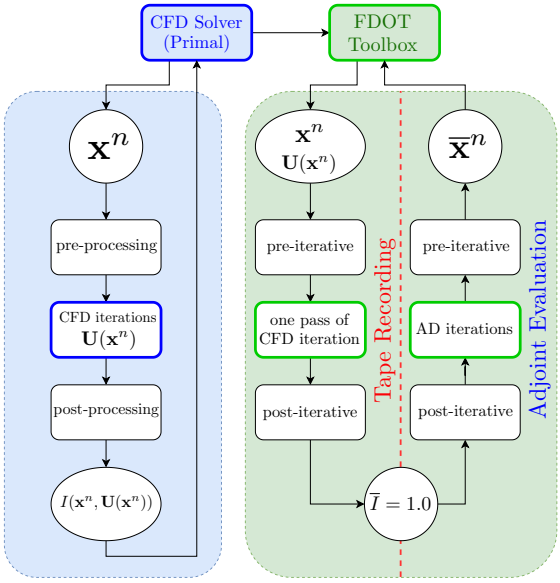


Figure 1. Flowchart for the FDOT toolbox and its integration into the primal CFD solver (adopted from Djeddi and Ekici[18,19]).

One of many attractive features of the FDOT toolbox that makes it a robust OO/AD tool is that it requires minimal changes to an already existing primal solver in order to obtain the adjoint counterpart of that solver. By declaring the flow solution variables, $\mathbf{U}$, while also marking the start and end of the iterative part using a "checkpointing" function, the FDOT toolbox can be efficiently used for adjoint evaluations while being fully-automated with minimal user interventions. The overall scheme demonstrating the process required for coupling the primal solver to FDOT is depicted in Fig. 1.

A. Improved Efficiency Using the Expression and Adjoint Tapes

The main goal of the present work is to improve the computational and memory efficiency of the original FDOT toolbox in order to make it more robust for large-scale CFD problems. As the first step, it is necessary to review the adjoint evaluation process using the FDOT OO/AD tool. When the expression tree is recorded into the "OP-tape", the indices of the arguments (one for unary and two for binary operations) are also recorded. As an example, let us assume the following unary and binary operations:

$$c_1 = f(x) \xrightarrow{\text{to OP-tape}} \textbf{index}(x),\ \textbf{tag}(f(\square)),\ \textbf{value}(c_1),\ \textbf{adjoint}(\overline{c_1}) \tag{1}$$

$$c_2 = g(x,y) \xrightarrow{\text{to OP-tape}} \textbf{index}(x,y),\ \textbf{tag}(g(\square,\triangle)),\ \textbf{value}(c_2),\ \textbf{adjoint}(\overline{c_2}) \tag{2}$$

It must be noted that with the above format, each tape entry will require 27 Bytes of memory (for double precision computations). The recorded tape is then used during adjoint evaluations to calculate the adjoint values for all active variables involved in the two operations, i.e.,

$$\overline{x} = \overline{x} + \frac{\partial g}{\partial x}\overline{c_2} \tag{3}$$

$$\overline{y} = \overline{y} + \frac{\partial g}{\partial y}\overline{c_2} \tag{4}$$

$$\overline{x} = \overline{x} + \frac{\partial f}{\partial x}\overline{c_1} \tag{5}$$

which means $\frac{\partial f}{\partial x}$, $\frac{\partial g}{\partial x}$, and $\frac{\partial g}{\partial y}$ derivatives should be defined as a function of the input arguments. This can be easily done for any differentiable function that handles unary or binary operations. Since the adjoint evaluations are performed following a fully-converged primal flow solution, even the non-differentiable functions would be locally differentiable. For example, min and max functions, which are non-smooth and non-differentiable at certain points, can be viewed as simple assignment operations depending on the known values for their two passed arguments.

Needless to say, different functions and operations would have their own derivatives defined with respect to their input arguments. Therefore, during the adjoint evaluations, for each tape entry, we need to use a *switch/case* which will determine the exact derivative, i.e., $\frac{\partial f}{\partial x}$, $\frac{\partial g}{\partial x}$, or $\frac{\partial g}{\partial y}$, depending on the **tag** stored in the tape. This process can slow down adjoint evaluations especially in cases where the length of the recorded tape is very large.

In this work, a new technique is proposed to calculate the partial derivative information that can greatly enhance the performance of the adjoint evaluation process. By utilizing this new approach, the length of the tape can be significantly reduced while also reducing the memory footprint per tape entry. Additionally, the *switch/case* structure is eliminated, which improves the performance of the iterative adjoint calculation significantly. To motivate the development of the proposed approach, let us assume that the entire CFD solver can be written as a set of functions applied to a number of independent (design) variables as well as intermediate variables to finally achieve the objective function. Therefore, we can write

$$v_{n+1} = f_n(v_1, v_2, v_3, \ldots, v_n) \tag{6}$$

where v_1, v_2, ..., and v_n are the n independent and intermediate variables leading up to v_{n+1} which can be assumed to be the objective or cost function, i.e., $I = v_{n+1}$. By applying the chain rule of differentiation to the above expression, we can calculate the adjoints of the independent and intermediate variables as

$$\frac{\partial I}{\partial v_m} = \frac{\partial v_{n+1}}{\partial v_m} = \prod_{i=m}^{n} \frac{\partial v_{i+1}}{\partial v_i} \quad \text{for any } m \leq n \tag{7}$$

The above formula is nothing but a repeated application of the chain rule of differentiation which is the cornerstone of the automatic or algorithmic differentiation (AD) technique. In a CFD solver, each intermediate variable is calculated using an assignment operation (=) that can be defined on a single or multiple (broken-down) lines of code. Each of these assignment operations can be seen as an expression where on the left-hand-side (LHS) we have v_i and on the right-hand-side (RHS), we have independent and/or intermediate variables, v_j, that have been defined previously, i.e., $j < i$. Ultimately, the result of the operations performed on all the prior variables, v_j, is then assigned to the LHS variable, v_i. Therefore, we can write

$$v_i = f_i(v_1, v_2, \ldots, v_j) \quad \text{for} \quad j < i \tag{8}$$

Now, using Eq. (7) and the adjoint formulation,[18] we can define the adjoints of independent and intermediate variables as

$$\overline{v}_j = \overline{v}_j + \frac{\partial f_i}{\partial v_j}\overline{v}_i \quad \text{for} \quad j < i \tag{9}$$

The idea here is to calculate and store the partial derivatives for any expression as they appear in the adjoint solver. This feature has been incorporated into the FDOT toolbox[18,19] such that any time an assignment operator (=) is executed, it is assumed that one full expression has been defined. Thus, the recorded tape for that specific expression tree is replayed in reverse direction to calculate the

partial derivatives. This partial derivative information is then recorded in to a new derived type class, called "ET-tape", along with the indices of the RHS variables (independent and intermediate) as well as the index of the LHS variable (result of the expression). As an example, the recording process for the "ET-tape" is described below for the expression shown in Eq. (8).

$$v_i = f_i(\dots, v_j, \dots) \quad \text{for each} \quad j < i \tag{10}$$
$$\xrightarrow{\text{to ET-tape}} \quad \textbf{index}(\text{RHS} = v_j, \text{LHS} = v_i), \textbf{adjoint}(\overline{v}_j = \frac{\partial f_i}{\partial v_j})$$

This leads to a memory footprint per tape entry of only 16 Bytes (for double precision) compared to the 27 Bytes of the original approach while the length of the tape has also been reduced since only active variables will be accounted for. To fully understand the difference between active and passive variables, let us look at the following expression

$$c = f(x_1, x_2, x_3, x_4) = ((x_1 + x_2) * (x_3 - x_4))^2 \tag{11}$$

where four independent variables, x_1 through x_4, are involved resulting in the LHS variable, c. The above expression is recorded in the original FDOT toolbox as four assignment operations (=), i.e.,

$$t_1 = x_1, \quad t_2 = x_2, \quad t_3 = x_3, \quad t_4 = x_4,$$

one binary addition (+), one binary subtraction (-), and one binary multiplication (*), i.e.,

$$t_5 = t_1 + t_2$$
$$t_6 = t_3 - t_4$$
$$t_7 = t_5 * t_6$$

and ultimately, one unary power (^) and a final assignment (=) operation, i.e.,

$$t_8 = t_7^2$$
$$t_9 = t_8$$

As can be seen, recording the tape for the expression shown in Eq. (11) leads to nine entries as opposed to only four entries which will be recorded in the "ET-tape" for each independent variable x_1 through x_4. These four variables are also known as active variables since they have been defined prior to this expression while the other variables are simply compiler-defined virtual variables that need not be adjoined. In general, any expression that involves n overloaded operators can result in ωn tape entries where ω can be viewed as a computational effort factor.[24] This is mainly due to the fact that compilers use temporary variables at run-time to execute unary and binary operations before reaching the end of an expression defined by an assignment operator. It is worth noting that ω is usually between 3-5 when averaged for all expressions involved in a nominal CFD solver.[24,25]

Having the partial derivative information, we can now move on to the actual adjoint evaluation process. Here, the adjoint of each independent and intermediate variable is defined using the repeated use of the adjoint formula applied to Eq. (8) to get

$$\overline{v}_k = \overline{v}_k + \frac{\partial f_i}{\partial v_k}\overline{v}_i \quad \text{where} \;\; k = 1, \dots, j \;\; \text{and} \;\; j < i \tag{12}$$

Although the number of expressions in a CFD solver are much more than the number of independent and intermediate variables, the adjoint values are only calculated for the independent/intermediate variables and not for all expressions. As a result, the tape that stores the adjoint values (8 Bytes per entry for double precision) will be even much shorter compared to the expression tape (ET-tape). The first-of-its-kind scheme demonstrating the proposed expression-based approach for adjoint evaluations is shown in Fig. 2. In summary, the following tapes are defined in the FDOT toolbox:

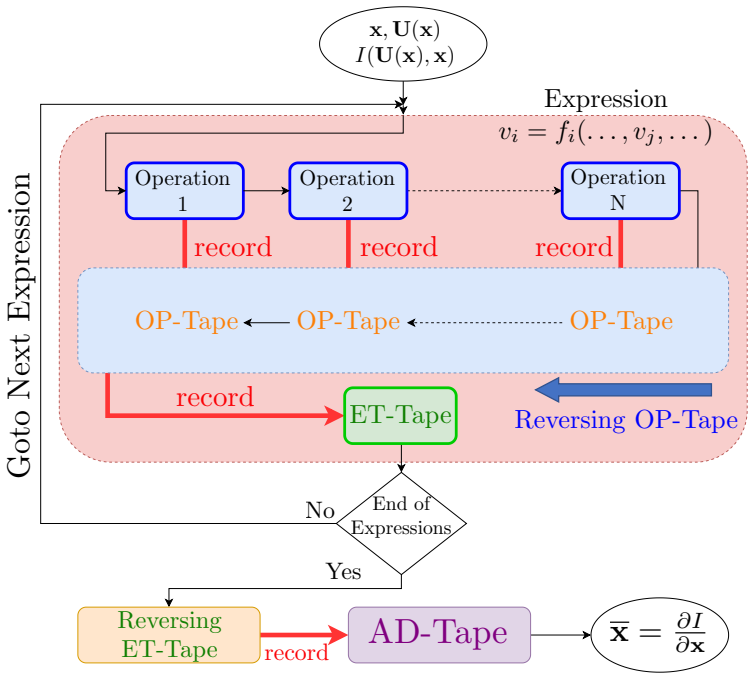


Figure 2. Flowchart for the expresion-based adjoint evaluation in the improved FDOT toolbox.

1. **OP-tape**: also known as the **operation tape** records one entry per "operation." In the original implementation of FDOT,[18] this is the only recorded tape, which is used for adjoint evaluations. (memory footprint: 27 Bytes per entry for double precision)

2. **ET-tape**: also known as the **expression tape** records one entry per active variable per expression. (memory footprint: 16 Bytes per entry for double precision)

3. **AD-tape**: also known as the **adjoint tape** records one entry per independent/intermediate variable. (memory footprint: 8 Bytes per entry for double precision)

It must be noted that the approach utilized here is, in some ways, very similar to the "expression templates" implemented in some OO/AD tools based on C++ programming language. The use of C++ template metaprogramming paradigms has enabled the developers of OO/AD tools such as CoDiPack[12] and ADOL-C[9] to significantly enhance the adjoint evaluation performance by utilizing expression templates at compile time. However, none of the these capabilities and paradigms are available for modern Fortran programming language. Nonetheless, the present work focuses on addressing these issues by utilizing a very robust adjoint evaluation process that is shown to improve the computational and memory efficiency of this AD tool.

III. UNPAC-DOF: Design Optimization Framework

Developing a robust and advanced CFD solver for complex and high-fidelity aerodynamic simulations is vital for a design optimization framework. **UN**structured **PA**rallel **C**ompressible (UNPAC) is a grid-transparent Reynolds-Averaged Navier-Stokes (RANS) solver based on a vertex-based finite volume discretization approach.[8,26] This in-house solver is coupled to the FDOT toolbox to automatically generate the adjoint versions of the primal solver (UNPAC-AD). Additionally, a gradient-based optimization wrapper program, called UNPAC-OPT, is developed to automate the aerodynamic shape optimization process. The UNPAC-OPT program 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 **UNPAC D**esign **O**ptimization **F**ramework (UNPAC-DOF) is provided in Fig. 3.

It must be noted that the optimization framework seeks for optimal designs via an iterative process

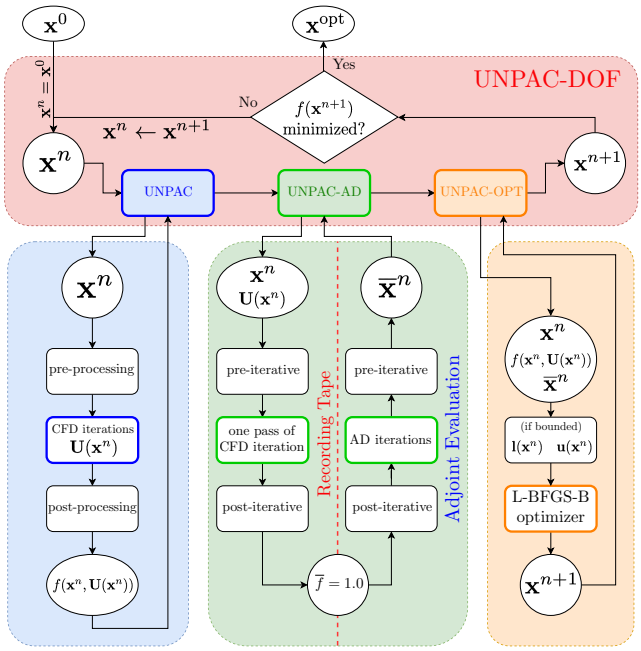


Figure 3. Flowchart of the UNPAC Design Optimization Framework (UNPAC-DOF) and its three main components: UNPAC, UNPAC-AD, and UNPAC-OPT (adopted from Djeddi and Ekici[19]).

which involves (1) calculation of the flow solution using the UNPAC solver, (2) evaluation of the gradient information using the UNPAC-AD solver, and finally (3) solution of the quasi-Newton optimization problem using the UNPAC-OPT program. Aerodynamic shape optimization is performed by either using the surface points or shape parameterization based on a Free-Form Deformation (FFD) box approach.[27]

A. Optimization Algorithm

1. Unconstrained Drag Minimization

Generally speaking, an aerodynamic design optimization problem seeks to iteratively determine the optimal solution for the set of design variables, $\mathbf{x}$, that minimizes an objective function, $I(\mathbf{x})$. Let us assume that the goal is to minimize the drag coefficient for an airfoil or a wing where the geometry is parameterized using an FFD box defined by a set of control points, $\mathbf{x}$. This optimization problem can be written as

$$\min_{\mathbf{x}} \quad C_D(\mathbf{x}) \tag{13}$$

where $\mathbf{x}$ is the vector of N control points or design variables. In the UNPAC-DOF framework, this optimization problem is solved using the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm.[28]

2. Lift-Constrained Drag Minimization

The unconstrained drag minimization problem is commonly used in aerodynamic shape optimization where the sole objective is to determine the optimal topology so as to minimize the overall drag coefficient (C_D). However, minimizing drag can also lead to a reduced overall lift coefficient (C_L) in many cases. This ultimately leads to a reduced efficiency, C_L/C_D. To circumvent this issue, the drag minimization problems are often constrained to a fixed-lift condition. In these cases, the angle of attack, α, is also treated as one of the design variables and the optimization problem is redefined as

$$\begin{aligned} \min_{\mathbf{x},\alpha} \quad & C_D(\mathbf{x}, \alpha) \\ \text{subject to} \quad & C_L(\mathbf{x}, \alpha) = C_L^* \end{aligned} \tag{14}$$

where C_L^* is a user-defined target value for the lift coefficient. As the first step, the constrained optimization problem [Eq. (14)] is rewritten as

$$\mathcal{L}(\mathbf{x}, \alpha, \lambda) = C_D(\mathbf{x}, \alpha) - \lambda \left[C_L^* - C_L(\mathbf{x}, \alpha)\right] \tag{15}$$

where λ is the Lagrange multiplier. Therefore, to minimize the Lagrangian, the following Karush-Kuhn-Tucker (KKT) conditions[29] must hold

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial C_D}{\partial \mathbf{x}} + \lambda \frac{\partial C_L}{\partial \mathbf{x}} = 0 \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \frac{\partial C_D}{\partial \alpha} + \lambda \frac{\partial C_L}{\partial \alpha} = 0 \tag{17}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = C_L(\mathbf{x}, \alpha) - C_L^* = 0 \tag{18}$$

Clearly, the last condition of optimality described in Eq. (18) is the actual constraint defined in the original minimization problem (see Eq. [14]). By rearranging Eq. (17), one can isolate the Lagrange multiplier such that

$$\lambda = -\frac{\partial C_D}{\partial \alpha} \left(\frac{\partial C_L}{\partial \alpha}\right)^{-1} = -\frac{\partial C_D}{\partial C_L} \tag{19}$$

which also satisfies Eq. (16). This results in the Lagrangian dual problem described as

$$\min_{\mathbf{x}, \alpha} \quad \mathcal{L}(\mathbf{x}, \alpha) = C_D(\mathbf{x}, \alpha) + \frac{\partial C_D}{\partial C_L} \left(C_L^* - C_L(\mathbf{x}, \alpha)\right) \tag{20}$$

For fixed-lift drag minimization problems, UNPAC-DOF solves Eq. (20) using the L-BFGS-B optimizer.[28,30] The optimization is handled in two stages. First, the angle of attack, α_{new}, is updated through

$$\alpha_{\text{new}} = \alpha + \left(\frac{\partial C_L}{\partial \alpha}\right)^{-1} \left(C_L^* - C_L(\mathbf{x}, \alpha)\right) \tag{21}$$

where a finite-difference approach is used to approximate the sensitivity of the lift coefficient to the angle attack, i.e.,

$$\frac{\partial C_L}{\partial \alpha} \approx \frac{C_L(\alpha + \epsilon) - C_L(\alpha)}{\epsilon} \tag{22}$$

where ϵ is a small value set to 10^{-8} in this work. At the same time, perturbing the angle of attack by ϵ, the sensitivity of the drag coefficient to the lift coefficient can also be approximated in a similar fashion, i.e.,

$$\frac{\partial C_D}{\partial C_L} \approx \frac{C_D(\alpha + \epsilon) - C_D(\alpha)}{C_L(\alpha + \epsilon) - C_L(\alpha)} \tag{23}$$

Next, this sensitivity is incorporated into the objective function [Eq. (20)] to minimize the drag coefficient. It is worth noting that the unconstrained minimization problem described in Eq. (20) can be also viewed as a "penalty" method with a variable penalty factor for various design stages. At each optimization cycle, current design variables ($\mathbf{x}$) as well as the gradient vector and the value of the cost function are passed to the L-BFGS-B optimizer, which outputs the updates for the the design variables, $\mathbf{x}_{\text{new}}$.

3. Lift-Constrained Drag Minimization with Additional Solution- and Geometric-Based Constraints

In the past couple of decades, the number of design parameters as well as the complexity of the CFD-based simulation and design tools have both increased dramatically. Therefore, aerodynamic shape optimization problems have sought optimal designs that are subject to multiple constraints over a more restrictive design space. As shown earlier, when only equality constraints are present, the method of "Lagrange multipliers" can be used to convert the design problem into an unconstrained problem. However, in more advanced aerodynamic shape optimization problems, one may need to deal with inequality constraints. As an example, a lift-constrained drag minimization problem can be further constrained by:

1. Solution-based inequality constraints, e.g., a minimum moment coefficient must be maintained.
2. Geometric-based inequality constraints, e.g., a minimum airfoil area (in 2D) or wing volume (in 3D) must be maintained.

Such an aerodynamic design optimization problem can be described via

$$\begin{aligned} \min\ I(\mathbf{x}) \ &= \ C_D(\mathbf{x}) - \frac{\partial C_D}{\partial C_L}\left(C_L^{\text{target}} - C_L(\mathbf{x})\right) \\ \text{w.r.t.} \quad & \mathbf{x},\ \alpha \\ \text{subject to} \quad & C_M(\mathbf{x}) \geq C_M^{\min} \\ & A(\mathbf{x}) \geq \text{Area}^{\min} \end{aligned} \tag{24}$$

where, as described earlier, the original objective function, i.e., drag coefficient, is augmented by the lift constraint.

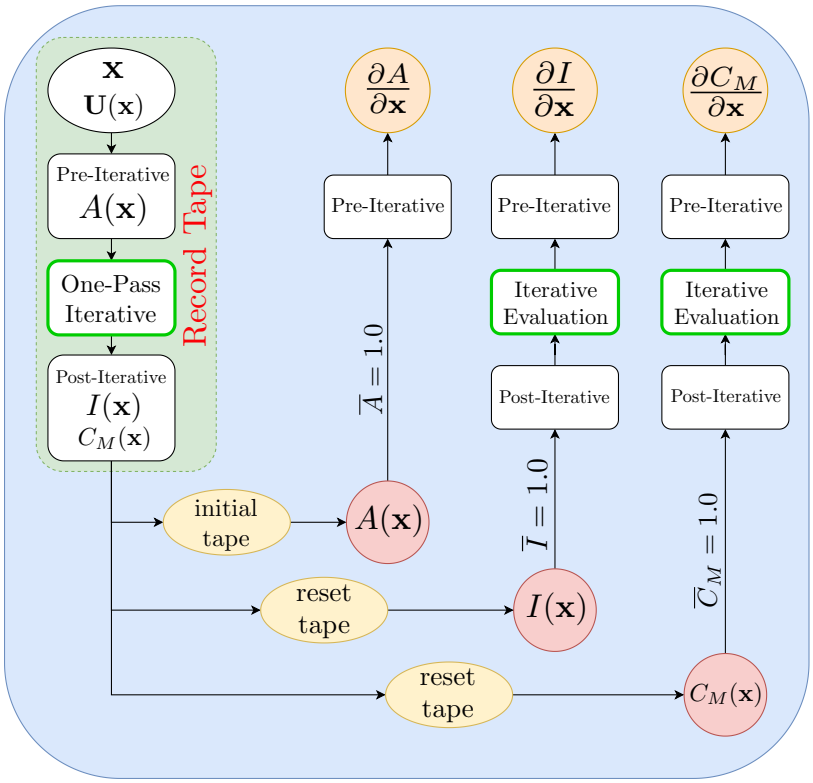


Figure 4. Procedure for calculating the necessary gradient information for solving the constrained optimization problem described in Eq. 24.

In such cases, the problem will be characterized in terms of the Karush-Kuhn-Tucker (KKT) as well as the "geometric optimality," and "Fritz John (FJ)" conditions.[29] Here, the Quadratic Programming (QP) problem would require the additional gradient information for the equality and inequality constraints with respect to the design variables. As described earlier, FDOT utilizes the fixed-point iteration approach by recording one pass of the CFD solver in a tape. As the first step in the tape evaluation process, the adjoint of the objective function is set to "one" and the tape is rewound to calculate the sensitivity information. The recorded tape is reused for calculating the additional gradient information in the process described below:

1. During the tape recording process, the objective function as well as the equality and inequality constraints are marked (their location or index in the tape is stored).

2. Since the geometric-based constraints are often only a function of the computational grid, their sensitivities only rely on the "pre-iterative" portion of the tape which handles grid pre-processing. Therefore, by simply rewinding the pre-iterative portion of the tape, the sensitivity information for the geometric-based constraints are evaluated.

3. Solution-based constraints, e.g., moment coefficient constraint, however, are most often handled similar to the objective function, e.g., drag coefficient. Therefore, the iterative tape evaluation process is executed once for the objective function (C_D or the augmented functional, $C_D - \frac{\partial C_D}{\partial C_L}(C_L^{\text{target}} - C_L)$) and repeated for each additional solution-based constraint.

The described procedure for calculating the entire gradient information is also shown in Fig. 4. As can be seen, the adjoint information in the tape is reset after each gradient evaluation before reevaluating the tape for the next quantity of interest. The UNPAC-DOF utilizes a Sequential Least-Squares Quadratic Programming (SLSQP) optimizer[31] for the constraint optimization problems described here.

IV. Aerodynamic Shape Optimization Results

In this section, the UNPAC-DOF framework is used for aerodynamic shape optimization of various airfoil and wing geometries. The novel memory-efficient version of the FDOT toolbox is utilized to perform adjoint-based sensitivity analysis. The new expression-based approach in the FDOT toolbox is compared to the original implementation via performance gain studies in terms of memory footprint and CPU times. Initially, the unconstrained drag minimization problem is solved for the NACA 0012 airfoil subject to inviscid transonic flow conditions. Next, the lift-constrained drag minimization problem with additional constraints involving moment coefficient and airfoil area is considered for the turbulent transonic flow past the RAE 2822 airfoil. This section also includes aerodynamic shape optimization of three-dimensional wing geometries. First, the drag minimization problem for the transonic ONERA M6 wing at a target lift coefficient is studied. Finally, the lift-constrained drag minimization problem with respect to the twist angle distribution along the wing span is considered for a rectangular NACA 0012 wing geometry.

A. Unconstrained Drag Minimization: Transonic NACA 0012 Airfoil

The first optimization test considered here is the unconstrained drag minimization of a NACA 0012 airfoil subject to an inviscid transonic flow. For this classical test case, the free-stream Mach number is 0.8 and the angle of attack of is 1.25 degrees. A fully unstructured grid with $10,216$ triangular elements, which is shown in Fig. 5, is used in this work. A second order Roe scheme is used for the discretization of the convective fluxes and the Green-Gauss method is used to calculate the flow gradients for the upwind flux terms.

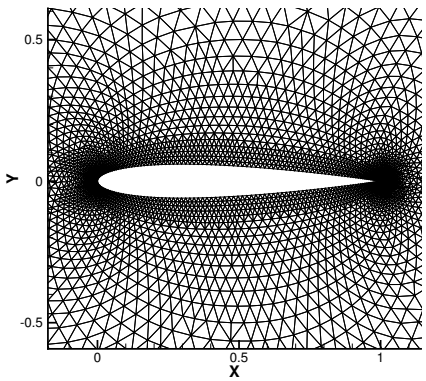


Figure 5. Unstructured grid used for the unconstrained drag minimization of the NACA 0012 airfoil.

The FFD box approach is used for the parameterization of the airfoil geometry. This FFD box is defined by four corners and is varied 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 into 21 control points in the x-direction and 2 control points in the y-direction for a total of 42 control points. Here, the unconstrained drag minimization problem is considered to minimize the drag coefficient. As for the design variables, the y coordinates of the FFD box control points are chosen with the control points at $\xi = 0$ and $\xi = 1$ fixed in order to maintain the effective angle of attack.

First, we need to study the effectiveness of the novel expression-based approach in reducing the memory footprint of the FDOT toolbox. For this reason, the length of the recorded tape as well as the memory footprint are presented for both approaches. These results are shown in Table 1. As can be seen, the ET-Tape used in the improved version of the FDOT toolbox has a much shorter length compared to the OP-Tape used in the original implementation. Also, as discussed earlier, the memory footprint per entry of the ET-Tape is almost 41% lower compared to that of the OP-Tape, thus resulting in an even more significant reduction in the overall memory footprint of the adjoint solver.

Table 1. Comparison of tape length and memory footprint for the original and improved FDOT toolbox (transonic NACA 0012 unconstrained drag minimization problem).

FDOT Toolbox	Tape Length	Reduction %	Memory (MBytes)	Reduction %
Original	31,738,803[a]	-	817	-
Improved	22,535,137[b]	**29.0**	413	**49.4**

Note: [a] OP-Tape, [b] ET-Tape

Having presented the performance improvements, we now turn our attention to the optimization results. First, the convergence history of the objective function (C_D) is plotted for major design cycles in Fig. 6. As can be seen, a significant reduction in the drag coefficient is observed during the first 10 design cycles with an ultimate reduction of almost 96%. Additionally, the comparison between the original and the optimized geometries as well as the distribution of the surface pressure coefficients are provided in Fig. 7.

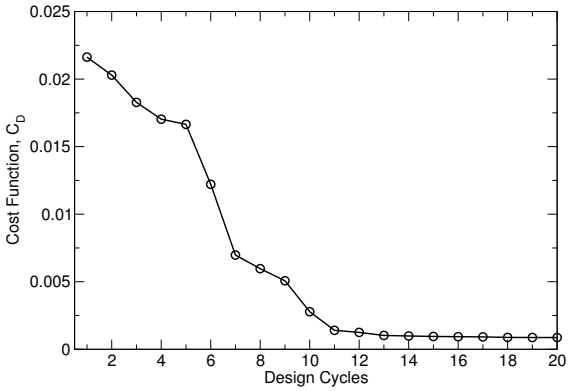
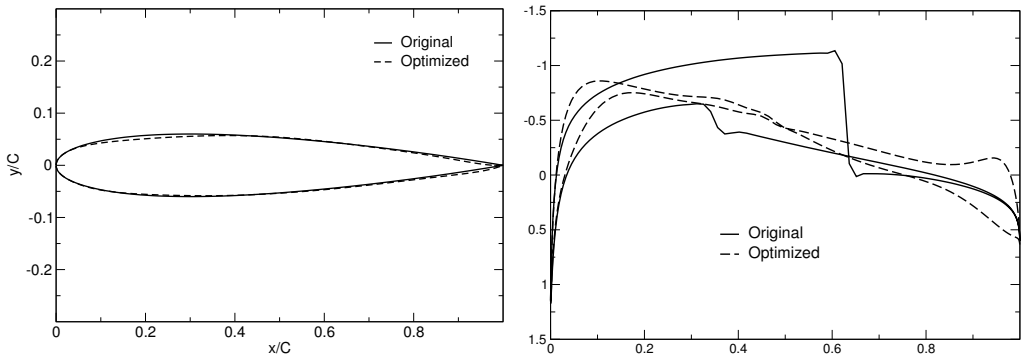


Figure 6. Convergence history of the objective function for the unconstrained drag minimization of NACA 0012 airfoil.

Table 2. Drag coefficient, C_D, and maximum thickness, $t_{\max}/c$, comparisons for transonic NACA 0012 unconstrained drag minimization problem.

Geometry	C_D	Reduction	$t_{\max}/c$	Reduction
Original	2.1638E-2	-	0.120	-
Optimized	8.7265E-4	95.9%	0.115	4.2%

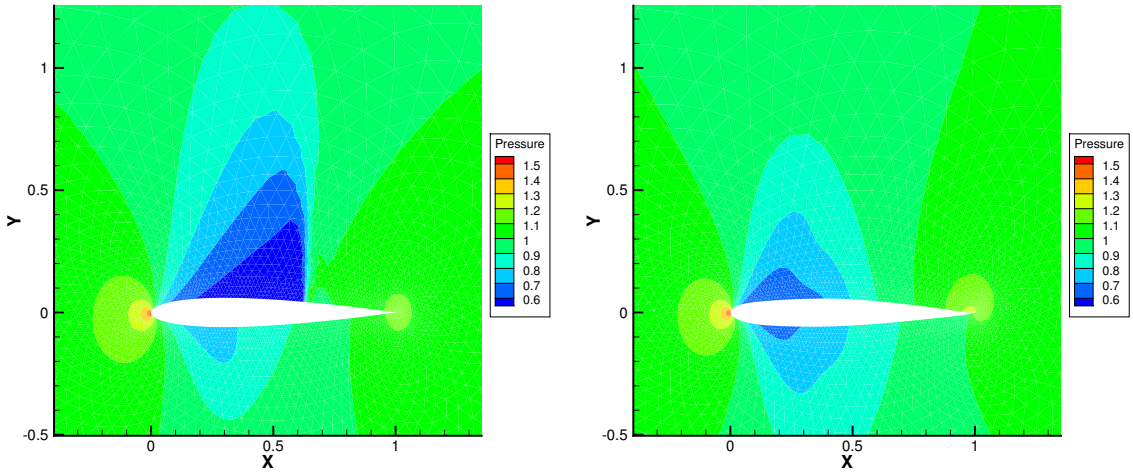
It is interesting to note that while the drag coefficient is significantly reduced, there is only 4.2% reduction in the maximum thickness of the NACA 0012 airfoil as presented in Table 2. The inviscid transonic flow past the NACA 0012 airfoil leads to the formation of a strong shock on the suction side and a weaker shock



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

Figure 7. Comparison of airfoil shape and the surface pressure coefficients for the original and optimized geometries.

on the pressure side.[8] As can be seen in Figs. 7 and 8, the present unconstrained drag minimization leads to the elimination of these shocks on both sides of the airfoil.

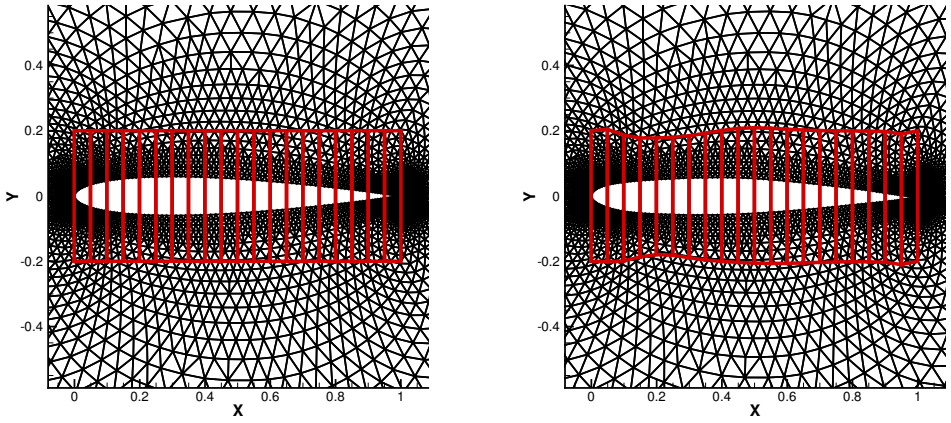


(a) Original Design (b) Optimized Design

Figure 8. Contour field of pressure for the inviscid transonic flow past NACA 0012 airfoil at $M = 0.8$ and AOA $= 1.25$ deg.

As discussed earlier, the FFD box approach is used here for the purpose of shape parameterization and the control points of the FFD box are used as the design variables for the optimization problem. The displacements of these control points will describe the deformed airfoil geometry which is then used to drive the interior mesh deformation. Here, the FFD box deformation is presented in Fig. 9 for the present unconstrained drag minimization test case. As can be seen, the maximum deformations of the FFD box are observed around the quarter-chord of the airfoil.

Finally, the efficiency of the current expression-based approach is studied. The computational cost of the adjoint solver using the original approach and the novel approach are presented in Table 3. It must be noted that the data the adjoint computation times are normalized with respect to that of the primal solver. As can be seen, the proposed technique provides almost 16% improvement in the CPU time and using both approaches. Also, the computational cost of the adjoint solver in each design cycle is around 2 to 2.5 times



(a) Original Box

(b) Deformed (Optimized) Box

Figure 9. Original and deformed FFD box that parameterizes the NACA 0012 airfoil for the unconstrained drag minimization problem.

the cost of the primal flow solver.

Table 3. Normalized CPU time comparisons between the original and the modified (present) FDOT toolbox.

FDOT Toolbox	Normalized CPU Time	Reduction (%)
Original	2.50	-
Improved (Present)	**2.11**	**15.6**

B. Constrained Drag Minimization: Transonic RAE 2822 Airfoil

The second optimization test case studied in this work deals with the constrained drag minimization of the RAE 2822 airfoil in turbulent transonic flow. As discussed earlier in Sec. III, the additional constraints for the optimization problem require the calculation of extra gradient information that will be provided to the SLSQP optimizer. As a result, this case is even more attractable as it can provide further insights into the performance of the original and improved versions of the FDOT toolbox in evaluating sensitivity information.

The free-stream Mach number for this case is 0.734 at a Reynolds number of 6.5 million. A hybrid grid with $22,842$ cells is considered which consists of $4,800$ quadrilateral elements in the near-field region and $13,937$ triangular elements for the rest of the domain that is extended for 100-chord lengths away from the airfoil surface. The goal of the optimization problem is to minimize the drag coefficient while maintaining a target lift coefficient. As discussed before, the lift-constrained drag minimization problem can be redefined as an unconstrained optimization problem with the addition of the angle of attack to the list of design variables while also augmenting the objective function by the "equality" lift coefficient constraint. For this case, however, two "inequality" constraints are also considered with the optimization problem defined as

$$\begin{aligned} \min\ & I(\mathbf{x}) \;=\; C_D(\mathbf{x}) - \frac{\partial C_D}{\partial C_L}\left(C_L^{\text{target}} - C_L(\mathbf{x})\right) \\ \text{w.r.t.}\quad & \mathbf{x},\ \alpha \\ \text{subject to}\quad & C_M(\mathbf{x}) \geq -0.092 \\ & A(\mathbf{x}) \geq A_{\text{base}} \end{aligned} \tag{25}$$

where the target lift is set to $C_L^{\text{target}} = 0.824$ which initially requires an angle of attack of 2.9209° to be satisfied. It must be noted that the area of the original RAE 2822 airfoil is $A_{\text{base}} = 0.077845c^2$ which is used as the minimum area of the airfoil during the shape optimization process. Once again, a free-form deformation (FFD) box is used to parameterize the airfoil geometry. The FFD box tightly encloses the RAE 2822 airfoil and the degrees of the Bernstein polynomials in ξ and η directions are taken to be 15 and 1, respectively. In order to fix the leading and trailing edges of the airfoil during the shape optimization cycles, the first and last rows of the FFD box control points are frozen. Also, the control points of the FFD box are only allowed to move in the y-direction. Therefore, the y coordinates of the remaining 28 control points are considered as the geometrical design variables, $\mathbf{x}$. With the addition of the angle of attack as an extra variable, the total number of design variables for this constrained optimization problem would be 29.

Before presenting the design optimization results, the effectiveness of the proposed expression-based approach in reducing the memory footprint of the FDOT toolbox is studied. In this regard, the length of the recorded tape as well as the memory footprint are compared for both approaches with the results shown in Table 4. As can be seen, the ET-Tape approach used in the improved version of the FDOT toolbox results in a much shorter adjoint tape compared to the original implementation. Once again, the memory footprint per entry of the adjoint tape is reduced by more than 40% in the ET-Tape approach. This reduction results in a very significant reduction in the overall memory footprint of the adjoint solver.

Table 4. Comparison of tape length and memory footprint for the original and improved FDOT toolbox (transonic RAE 2822 constrained drag minimization problem).

FDOT Toolbox	Tape Length	Reduction %	Memory (GBytes)	Reduction %
Original	326,909,670[a]	-	8.12	-
Improved	221,958,575[b]	**32.1**	4.28	**47.3**

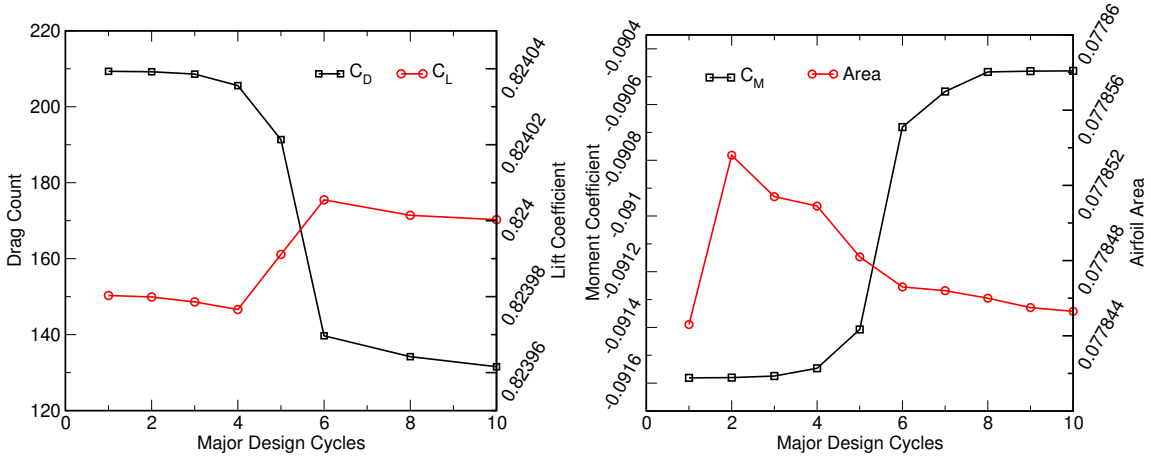
Note: [a] OP-Tape, [b] ET-Tape

Note that for the present constrained optimization problem, according to the discussion in Sec. III and the flowchart shown in Fig. 4, the entire tape needs to be evaluated twice for the augmented objective function, i.e., drag coefficient with the lift constraint, as well as the moment coefficient. Additionally, the pre-iterative portion of the tape needs to be re-evaluated for the geometric-based constraint, i.e., the airfoil area. However, since the originally recorded tape is reused for these evaluation processes, the tape length and the subsequent memory footprint will not be increased at all. This feature makes FDOT an even more robust sensitivity analysis tool for PDE-constrained optimization problems.

Having presented the performance test results for the improved FDOT toolbox, we can now focus our attention to the aerodynamic design optimization results. First, the convergence histories of the objective function and the constraints for this optimization problem are presented. These results are shown for the drag count, lift coefficient, moment coefficient, and the airfoil area in Fig. 10. As can be seen, the drag count has been steadily reduced during the design optimization cycles while the target lift coefficient of 0.824 is closely maintained. Additionally, the moment coefficient is not only kept above the minimum requirement for all design cycles but is also increased slightly. Finally, the area of the airfoil is kept almost unchanged during the design optimization cycles. It must be noted that the present constrained drag minimization problem has led to more than 37% reduction in the drag count while maintaining the target lift coefficient. As a result, the efficiency of the RAE 2822 airfoil has been increased by about 60%.

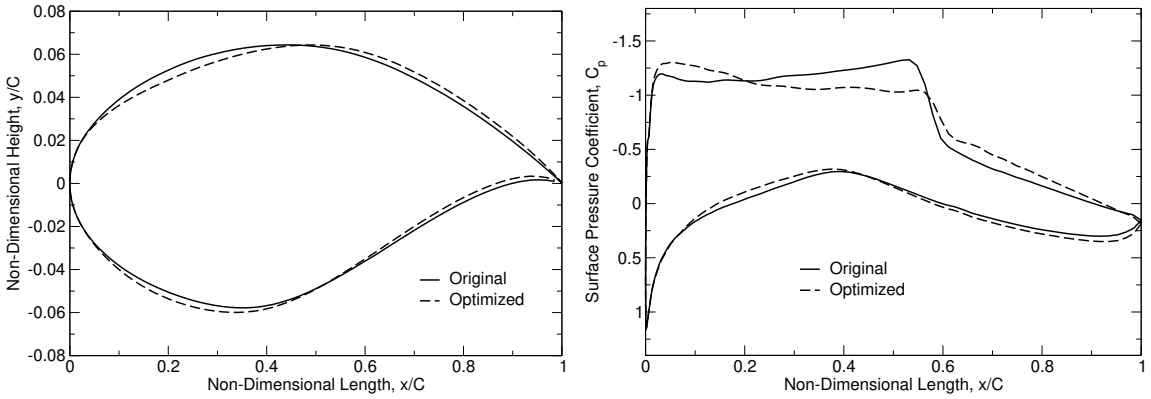
Next, the airfoil shape and the surface pressure coefficient distributions are compared for the original and optimized geometries. These results are shown in Fig. 11. For this case, the strength of the shock on the suction side is significantly reduced. However, the shock is not fully eliminated which can be associated with the fact that the present drag minimization problem is constrained by the lift and moment coefficients as well as the airfoil area. In a similar optimization study, Lee et al.[32] have shown that by using a lower degree for the Bernstein polynomials, the shock can be further alleviated or even eliminated. However, it must be noted that using fewer number of design variables can, in some cases, lead to pressure oscillations on the bottom surface of the RAE 2822 airfoil.[32]

The weakening of the shock during the design optimization process can be also shown via the Mach number contour plots for the original and optimized cases. These results are shown in Fig. 12 and, as



(a) C_D and C_L (b) C_M and Area

Figure 10. Convergence history of the objective function (C_D) as well as the constraints for the drag minimization of the RAE 2822 airfoi.

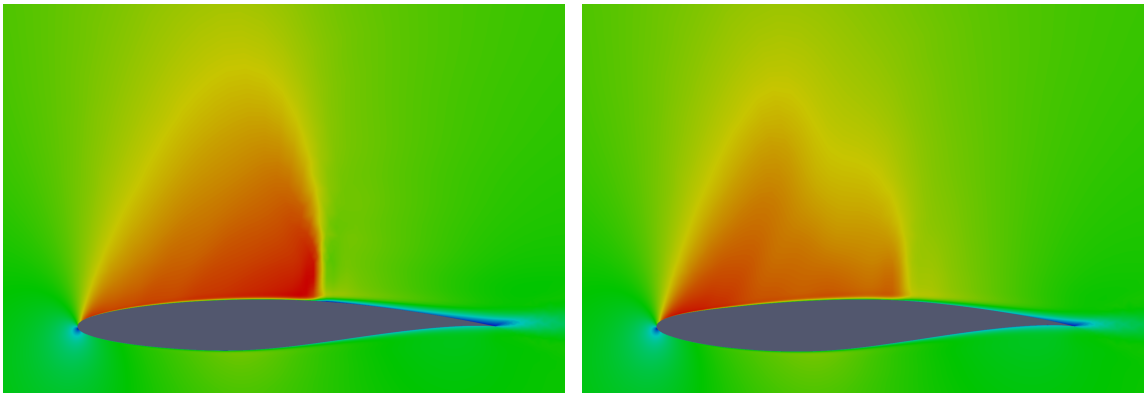


(a) Airfoil shape (b) Surface pressure coefficient, C_p

Figure 11. Comparison of RAE 2822 airfoil shape and the surface pressure coefficients for the original and optimized geometries.

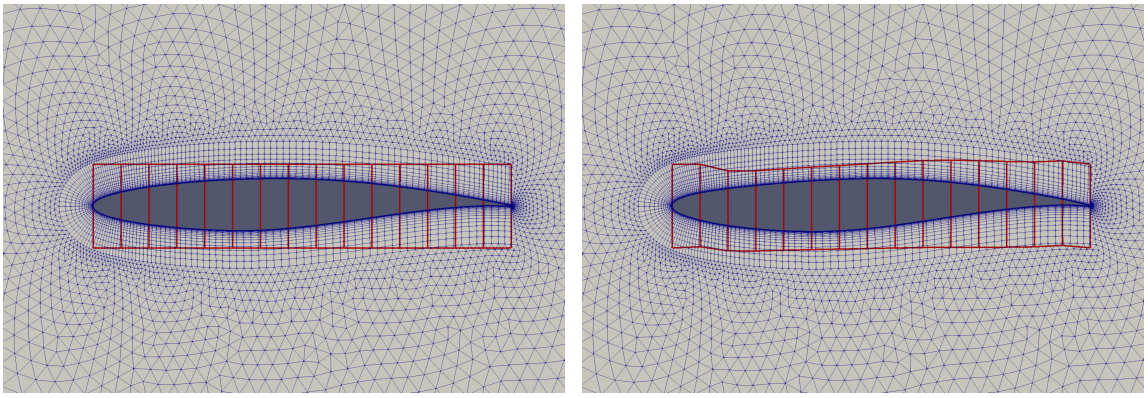
can be seen, the geometry deformation for the RAE 2822 airfoil has led to a weaker shock-boundary-layer interaction which results in a reduced drag count for this airfoil at the present flow conditions. Next, the FFD box deformation as well as the interior mesh deformation are presented in Fig. 13 for this case. It can be seen that the particular optimization leads to the maximum deformation of the FFD box around the quarter-chord of the airfoil.

Finally, the efficiency of the FDOT toolbox using the original and the improved approaches is studied. The computational cost of running the adjoint solver using the original and improved FDOT toolbox is described in terms of normalized CPU times with respect to that of the primal solver and the results are presented in Table 5. As discussed before, with the additional constraints utilized for this optimization problem, it is required to reevaluate the recorded tape several times. Therefore, the overall computational cost of the adjoint solver will be linearly scaled by the number of solution-based constraints. However, the computational cost of a single adjoint evaluation is still only a small multiple of that of the primal flow solver. Additionally, the proposed technique provides an almost 18% improvement in the CPU time.



(a) Original Design (b) Optimized Design

Figure 12. Contour field of Mach number for the turbulent transonic flow past RAE 2822 airfoil.



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

Figure 13. Original and deformed FFD box that parameterizes the RAE 2822 airfoil for the constrained drag minimization problem.

Table 5. Normalized CPU time comparisons between the original and the modified (present) FDOT toolbox for a single adjoint evaluation process.

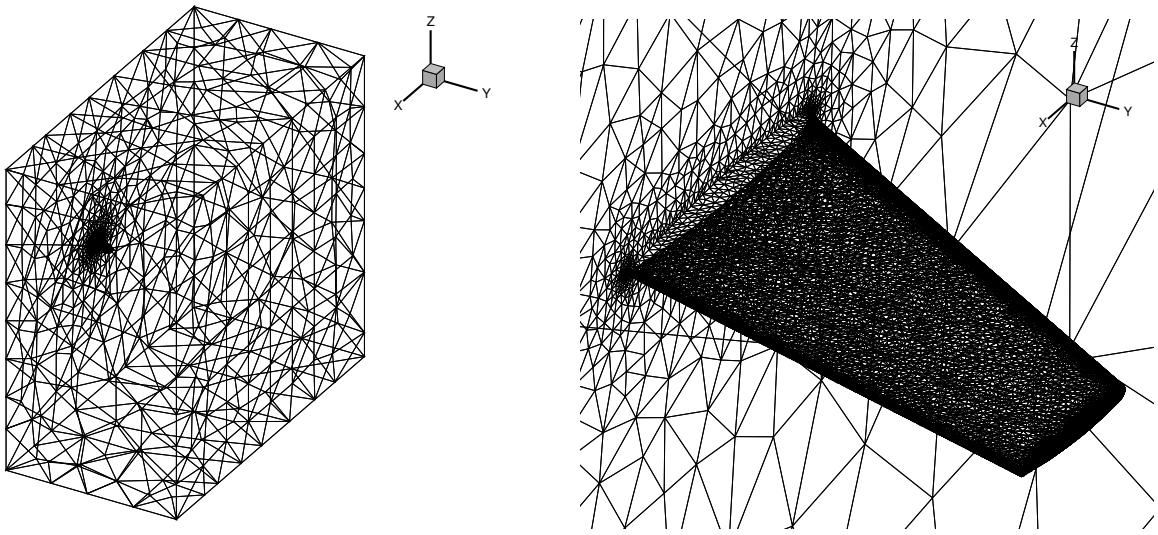
FDOT Toolbox	Normalized CPU Time	Reduction (%)
Original	3.11	-
Improved (Present)	**2.56**	**17.7**

C. Fixed-Lift Drag Minimization: Transonic ONERA M6 Wing

Next, the fixed-lift drag minimization problem for the inviscid transonic ONERA M6 wing is investigated. The aerodynamics of this wing involve a region of supersonic flow including a special shock formation known as the lambda shock.[33,34] 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[35] with a free-stream

Mach number of 0.8395 and an angle of attack of 3.06°. It must be noted that the transonic flow past the ONERA M6 wing has been used in the literature as a standard benchmark test case for the purpose of validation and verification of the CFD solvers while the unconstrained or lift-constrained drag minimization of this wing has also been studied extensively.[36–38]

The computational grid used for this study consists of a rectangular outer boundary that extends about 10 mean chord lengths on each side. The far-field and near-field views of the grid used for this test case are shown in Fig. 14. The fully unstructured grid is made of 108,396 nodes and 582,752 tetrahedral elements with 38,756 triangular elements on the surface of the wing. Here, a symmetry boundary condition is used on the root-plane and far-field boundary conditions are used for the rest of the outer boundaries.



(a) Computational domain with 582,752 cells

(b) Surface mesh with 38,756 triangles

Figure 14. Volume and surface meshes used for the transonic flow past ONERA M6 wing.

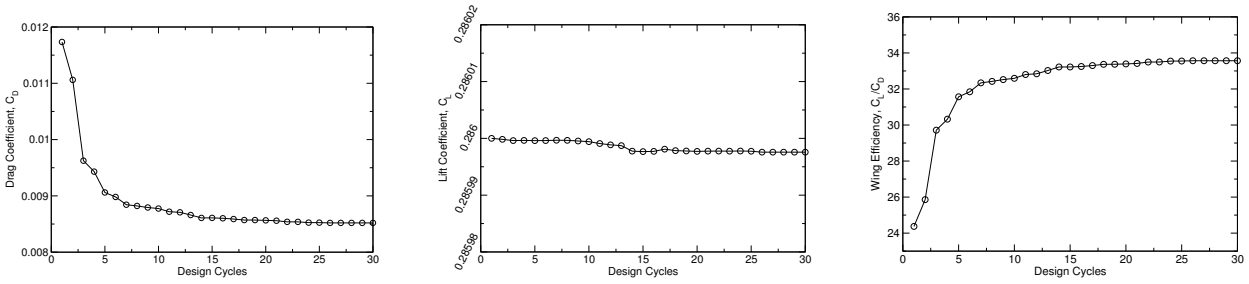
Before presenting the optimization results, the performance metrics of the FDOT toolbox in terms of the memory footprint as well as the tape length are presented in Table 6. As can be seen, the new approach used in the FDOT toolbox, where the expression and adjoint tapes are utilized, results in significant reductions in both the tape length and the memory footprint of the adjoint solver. More specifically, the expression-based approach leads to 32% reduction in the tape length with a memory footprint reduction of almost 52%. This, once again, proves that the proposed technique is capable of dramatically improving the memory efficiency of the FDOT toolbox for adjoint sensitivity analysis.

Table 6. Comparison of tape length and memory footprint for the original and improved FDOT toolbox (transonic ONERA M6 fixed-lift drag minimization problem).

FDOT Toolbox	Tape Length	Reduction %	Memory (GBytes)	Reduction %
Original	560,731,841[a]	-	14.1	-
Improved	381,297,651[b]	**32.0**	6.81	**51.7**

Note: [a] OP-Tape, [b] ET-Tape

Once again, for the design optimization problem, the wing geometry is parameterized using a hexahedral Free Form Deformation (FFD) box. The FFD box is swept in order to tightly enclose the ONERA M6 wing that has different sweep angles on the leading and trailing edges. 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



(a) Drag Coefficient, C_D (b) Lift Coefficient, C_L (c) Wing Efficiency, C_L/C_D

Figure 15. Convergence histories for the drag and lift coefficients and the efficiency for the fixed-lift drag minimization of the ONERA M6 wing.

coordinate. For the constrained optimization problem studied in this work, the target value for the lift coefficient is set to $C_L^* = 0.268$. The convergence history of the drag and lift coefficients as well as the wing efficiency are plotted for major design cycles in Fig. 15. As can be seen, a steady drop in the objective function is observed with the drag coefficient being reduced by 27.4% after 30 major design cycles. Since the lift coefficient is constrained at $C_L^* = 0.268$, the efficiency of the wing is also increased by almost 38%. These results are also presented in Table 7.

Table 7. Optimization results for the transonic ONERA M6 fixed-lift drag minimization problem.

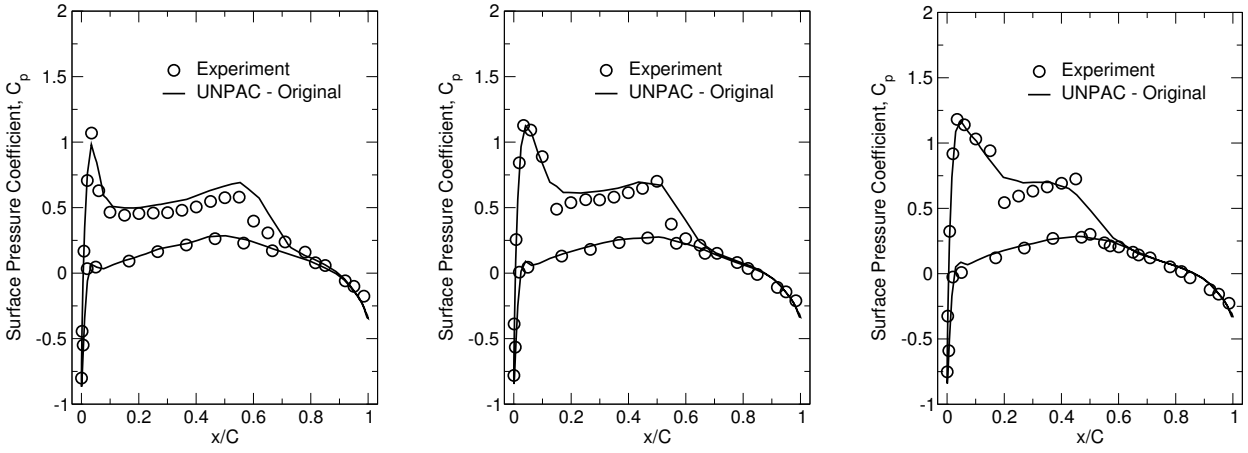
Geometry	C_D	Reduction	Efficiency	Improvement
Original	1.1734E-2	-	24.37	-
Optimized	8.5201E-3	27.4%	33.56	37.7%

Next, the numerical results obtained using the original ONERA M6 wing geometry are compared against the experimental data of Schmitt and Charpin.[35] 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. These results are presented in Fig. 16 which show a good agreement between the UNPAC solver results and the experimental data.

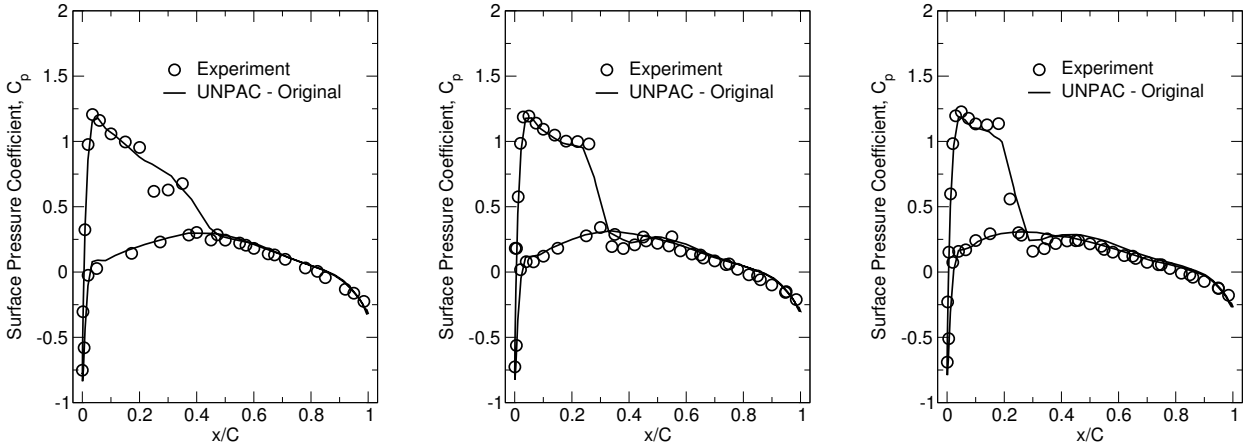
Additionally, the surface solutions at the last 3 span-wise sections are compared for the original and optimized geometries and the distribution of the surface pressure coefficients at each section are shown in Fig. 17. As can be seen, the strong shock close to the tip of the wing is completely eliminated 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 Fig. 18 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 minimization problem are presented in Fig. 19.

Finally, the computational performance of the adjoint solver using the FDOT toolbox is studied in terms of CPU time. Here, the original and the improved versions of the FDOT toolbox are compared against each other with the results presented in Table 8. As can be seen, the CPU time for the adjoint solver using the original implementation of the FDOT toolbox is around 4.2 times that of the primal CFD solver. However, using the proposed technique in the improved version of the FDOT toolbox, this normalized CPU time is reduced by almost 22% to only 3.3 times that of the primal CFD solver. This result once again proves the computational efficiency of the improved FDOT toolbox for adjoint-based sensitivity analysis.



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



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

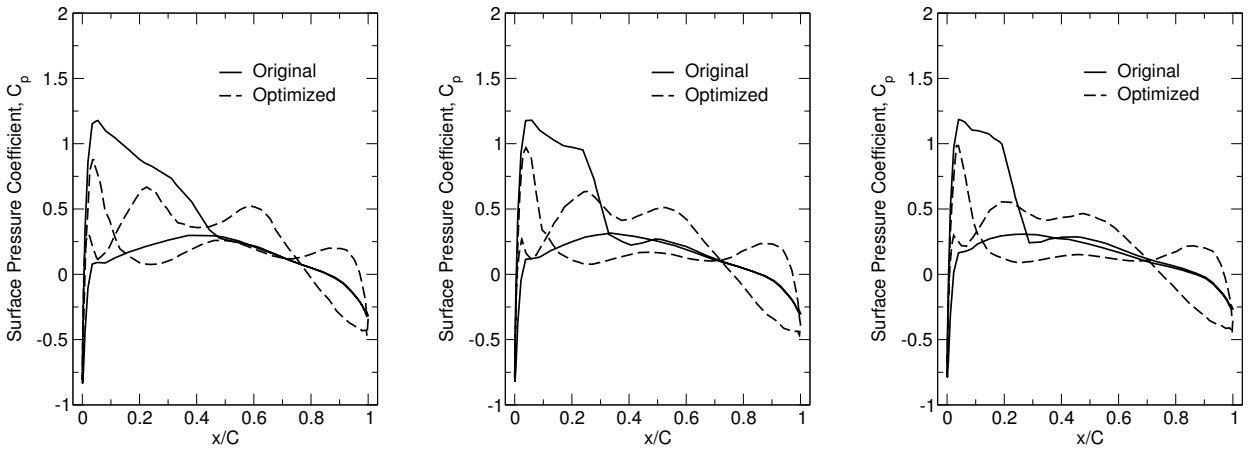
Figure 16. Surface pressure coefficients for the inviscid transonic flow past ONERA M6 wing compared to experimental data.[35]

Table 8. CPU timings for the primal and adjoint solvers (using the original and improved versions of the FDOT toolbox) for the fixed-lift drag minimization of the ONERA M6 wing.

Solver Type	CPU Time (m)	Normalized CPU Time	Reduction %
Primal (CFD)	17.21	1.0	-
FDOT (Original)	72.28	**4.2**	-
FDOT (Improved)	56.80	**3.3**	**21.9**

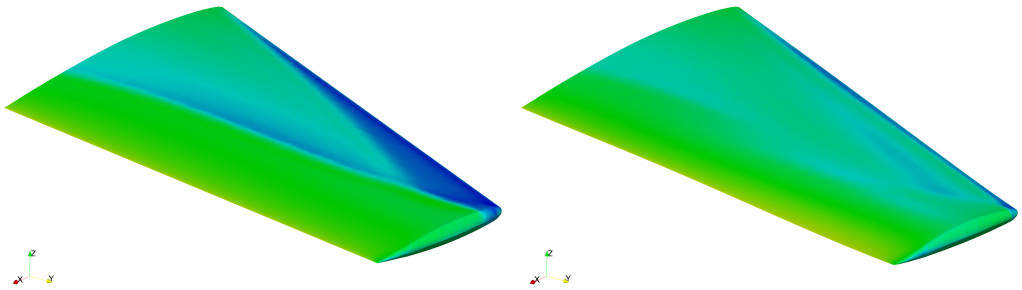
D. Drag Minimization Subject to Twist Distribution: Rectangular NACA 0012 Wing

The last optimization problem studied in this work is the drag minimization of an untwisted and untapered rectangular wing with a NACA 0012 cross-section. The rectangular wing has a sharp trailing edge with a semi-span of 3.06c where the last 0.06c is consisted of a rounded tip created from revolving the NACA 0012 profile around the tip camber line. This wing is subject to an inviscid subsonic flow at a 0.5 free-stream



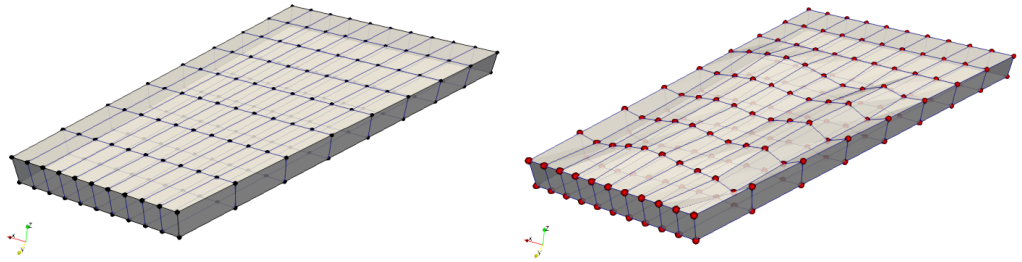
(a) 80% span (b) 90% span (c) 95% span

Figure 17. Comparison of the surface pressure coefficients between the original and optimized geometries for the fixed-lift drag minimization of the inviscid transonic ONERA M6 wing.



(a) Original Design (b) Optimized Design

Figure 18. Contour field of pressure on the surface of the ONERA M6 wing for the fixed-lift drag minimization problem.



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

Figure 19. The original and deformed (optimized) geometry of the FFD box parameterizing the ONERA M6 wing for the fixed-lift drag minimization problem.

Mach number. The optimization problem is described as

$$\begin{aligned} &\min\ C_D(\gamma(y)) \\ \text{w.r.t.}\quad &\gamma(y) \\ \text{subject to}\quad &C_L(\gamma(y)) = 0.375 \end{aligned} \tag{26}$$

where $\gamma(y)$ is the spanwise distribution of the twist angle which is described about the trailing edge. Here, the equality constraint for the target lift describes a fixed-lift drag minimization problem. The goal here is to optimize the twist angle distribution along the wing span to minimize the tip vortex that can lead to a reduction in the induced drag. This case has been studied by Bisson and Nadarajah,[39] Lee et al.,[32] and more recently by Yang and Da Ronch[40] based on the descriptions provided by the Aerodynamic Design Optimization Discussion Group (ADODG).

As reported in the literature, the computational mesh for this case has a direct effect on the prediction of the induced drag. As a result, Bisson and Nadarajah[39] have utilized computational grids with more than 13 million nodes while Lee et al.[32] have even used structured grids with more than 87 million nodes to study this case that involves an inviscid subsonic flow. Therefore, in this work, different grid resolutions are considered to closely study the effects of computational mesh on the primal CFD solutions as well as the design optimization results. In order to reduce the number of degrees of freedom for the computational mesh, a hybrid grid generation approach is utilized where pyramid cells are used close to the wing surface with tetrahedral elements filling the remainder of the computational domain. Three computational meshes, named G1, G2, and G3, are generated with the grid statistics provided in Table 9.

Table 9. Grid statistics and parameters for the rectangular NACA 0012 wing drag minimization problem.

Grid No.	No. of Nodes	No. of Elements	Tetrahedral Cells	Pyramid Cells	Minimum Wall Spacing
G1	126,204	686,396	674,516	11,880	4.0×10^{-3}
G2	287,934	1,570,050	1,543,230	26,820	2.0×10^{-3}
G3	533,721	2,935,147	2,887,387	47,760	1.0×10^{-3}

Once again, the FFD box approach is used to parameterize the wing geometry for the purpose of design optimization. The FFD box is taken to be a simple rectangular parallelepiped that tightly encloses the rectangular wing including the rounded tip. Unlike the previous applications of the FFD box, where equally-spaced control points were used in parametric coordinates, the η-planes for the present FFD box are defined at the specific locations defined by the ADODG benchmark guidelines for reporting the twist angles (see Fig. 20). This results in an unequally-spaced distribution of the FFD box control points along the wing span with the degrees of the Bernstein polynomial taken to be $(4, 8, 1)$ in (ξ, η, ζ) parametric directions.

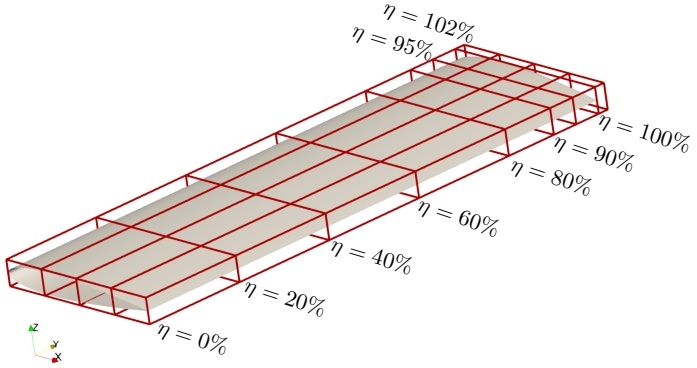


Figure 20. The free-form deformation (FFD) box and its control points used to parameterize the rectangular NACA 0012 wing for the twist optimization test case.

During the design optimization process, FFD box control points on each η plane are rotated about the trailing edge according to the twist angle at that specific station. According to the ADODG guidelines for

this optimization test case, the angle of attack will be fixed and the only design variables used will be the twist angles at the $\eta = 0\%, 20\%, 40\%, 60\%, 80\%, 90\%, 95\%$, and 100% spanwise locations which means that the wing is allowed to be twisted at the root. Additionally, the twist angle at the last η plane, i.e., $\eta = 102\%$, is set to be equal to that of the $\eta = 100\%$ plane to avoid any twist within the rounded tip region. Since the angle of attack is kept fixed for this optimization problem, the equality lift constraint must be incorporated directly into the SLSQP optimizer. This would require the calculation of the sensitivities of the lift coefficient in addition to the gradient information of the objective function, i.e., the drag coefficient, which results in two adjoint evaluation processes during each design cycle.

Initially, the performance of the FDOT toolbox in terms of the memory efficiency is studied with the memory footprint and tape length information presented in Table 10 using the original and the improved (present) approach. Note that only the finest grid (G3) is used for this memory efficiency study as the tape length and memory footprint are directly related to the number of degrees of freedom of the CFD solver. Once again, it can be seen that the proposed ET-Tape approach can significantly reduce the tape length which consequently lowers the memory footprint of the adjoint solver. Moreover, it must be noted that the memory footprint reductions for the three-dimensional test cases happen to be more significant compared to the two-dimensional tests. Overall, the ET-Tape approach proposed in this work can provide more than 35% reduction in the tape length and an almost 60% reduction in the memory footprint of the adjoint solver.

Table 10. Comparison of tape length and memory footprint for the original and improved FDOT toolbox: rectangular NACA 0012 wing twist optimization problem using the finest (G3) grid resolution.

FDOT Toolbox	Tape Length	Reduction %	Memory (GBytes)	Reduction %
Original	545,558,068[a]	-	13.8	-
Improved	350,793,838[b]	**35.7**	5.7	**58.7**

Note: [a] OP-Tape, [b] ET-Tape

Next, the optimization results are presented for the drag minimization of the rectangular NACA 0012 wing with respect to the spanwise twist angle distribution. As explained earlier, for this case, the three grid levels are used to solve the design optimization problem while using the same FFD box and shape parameterization settings. A very important factor to determine the performance of a three-dimensional wing is "span efficiency". This parameter is defined as

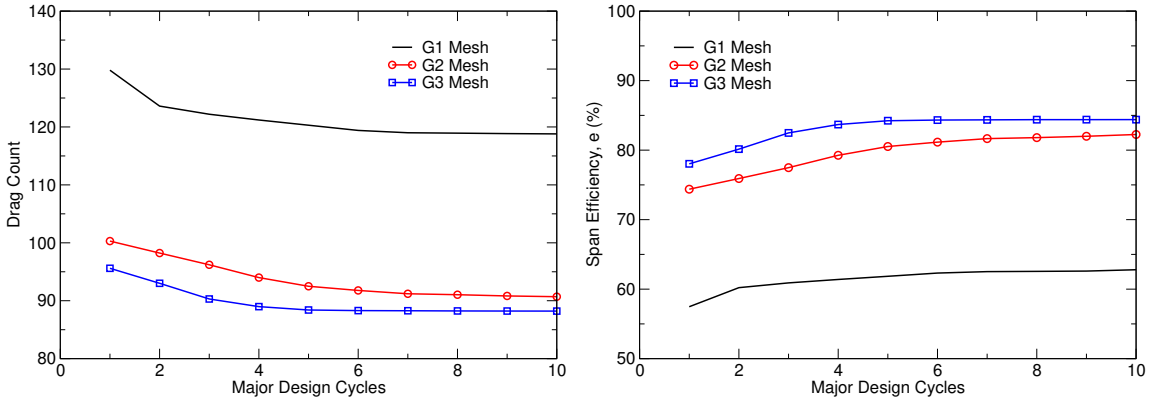
$$e = \frac{C_L^2}{\pi \Lambda C_D} \tag{27}$$

where Λ is the wing aspect ratio which is $\Lambda = b^2/(2S) = 6$ for the present case with S being the reference semi-span area that will be used for non-dimensionalization as well as for the calculation of the lift and drag coefficients. Results in terms of drag count and the span efficiency for the present twist optimization problem using the three grid resolutions are presented in Table 11. Also, the convergence histories for the drag count and the span efficiency for the three computational grids used in this study are shown in Fig. 21.

Table 11. Optimization results for the rectangular NACA 0012 wing twist optimization problem.

Grid	Geometry	C_L	Drag Count	Reduction	Span Efficiency	Improvement
G1	Original	0.3750	12.98	-	0.5747	-
G1	Optimized	0.3751	11.88	**8.47%**	0.6279	**9.25%**
G2	Original	0.3750	10.03	-	0.7438	-
G2	Optimized	0.3749	9.07	**9.57%**	0.8225	**10.58%**
G3	Original	0.3750	9.56	-	0.7803	-
G3	Optimized	0.3750	8.84	**7.53%**	0.8439	**8.15%**

As can be seen in Table 11, the lift coefficient is maintained within ± 0.001 of the target value for all these cases. Additionally, the drag count reductions between 7.5% and 9.6% are observed for these cases. The improvements in terms of the span efficiency are slightly more pronounced due to the fact that the lift coefficient is maintained while drag is reduced. It must be noted that best reduction in drag count and improvement in span efficiency are achieved for the G2 grid while the finest grid (G3) provides the highest span efficiency for the NACA 0012 wing.



(a) Drag Coefficient, C_D

(b) Span Efficiency, e

Figure 21. Convergence histories for the drag coefficient and the span efficiency for the twist optimization of the rectangular NACA 0012 wing using three different grid resolutions.

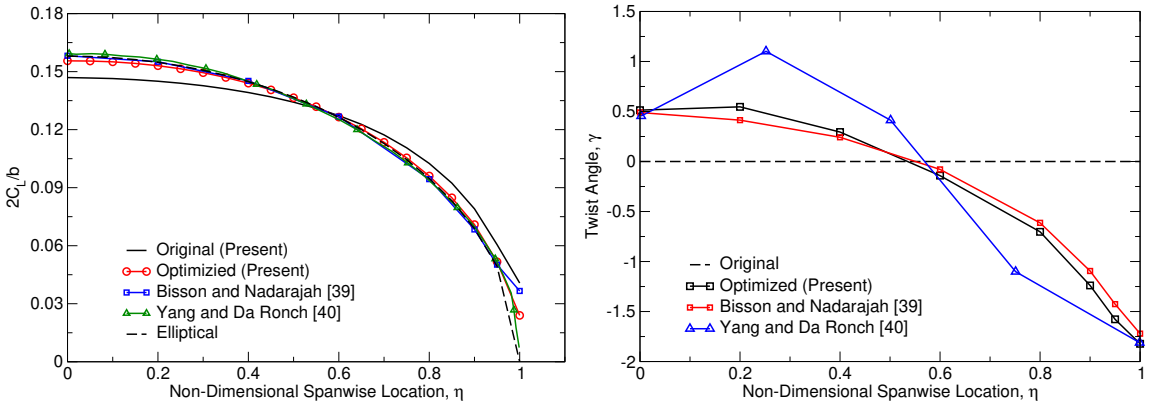


Figure 22. Sectional lift (left) and twist angle (right) distributions along the wing span for the original and optimized geometries compared to the results of Bisson and Nadarajah[39] and Yang and Da Ronch:[40] lift-constrained drag minimization problem for the rectangular NACA 0012 wing using the finest (G3) grid resolution.

Next, the sectional lift, $2C_L/b$, distributions as well as the twist angles, $\gamma(y)$, along the wing span are presented for the original and optimized geometries using the finest (G3) grid resolution. These are shown in Fig. 22 where the sectional lift distributions are also compared to the elliptic sectional lift distribution. According to the lifting-line theory, the elliptical wing with elliptic sectional lift distribution has the maximum, i.e., 100%, span efficiency. As can be seen in Fig. 22, and as expected, the sectional lift distributions for the optimized wing (with the optimal twist angle distribution) are very close to the elliptical curve except close to the tip of the wing. Additionally, the twist angle distribution along the wing span shows that the NACA 0012 twisted upward (positive twist angle) about the trailing edge closer to the root of the wing to increase the sectional lift. Moreover, the wing is twisted downward (negative twist angle) after about $\eta = 60\%$ to reduce the sectional lift and, thus, reducing the induced drag component.

Next, the pressure contour fields on the top and bottom surfaces of the NACA 0012 wing for the original and optimized geometries are plotted and the results are shown in Fig. 23. Also, the untwisted and optimally-twisted wing geometries as well as the FFD box deformation are presented in Fig. 24. As can be seen, the lift-constrained drag minimization problem studied here has been able to find the optimal twist angle distribution along the wing span that can provide an "almost-elliptic" lift distribution, resulting in an induced drag reduction.

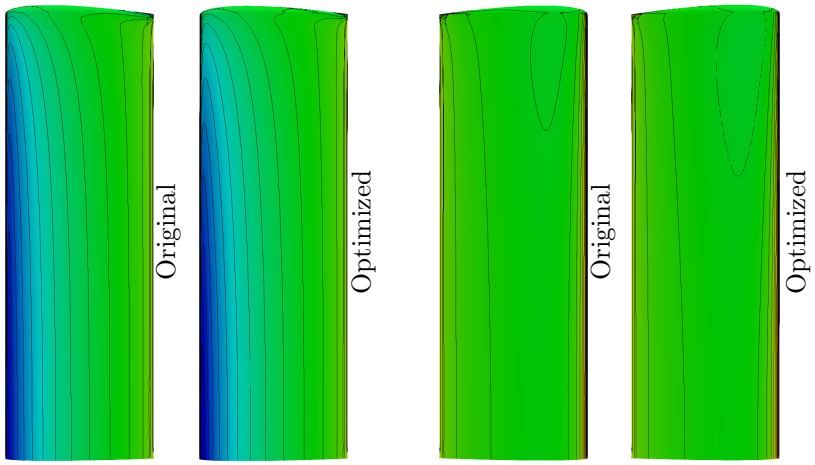
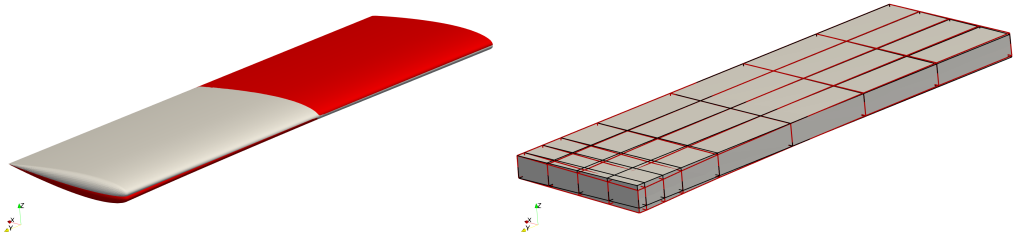


Figure 23. Contour field of pressure on the top (left) and bottom (right) surfaces of the NACA 0012 wing for the fixed-lift drag minimization (twist optimization) problem.



(a) Original vs. Twisted Wing (b) Initial and Deformed FFD Boxes

Figure 24. The original and deformed (optimized) geometry of the wing (left) and the deformed FFD box (right) for the twist optimization problem (deformed geometry is shown in red): rectangular NACA 0012 wing.

In order to further study the effect of the optimal twist distribution on the tip vortex of the NACA 0012 wing, the stream-traces are plotted at the $\eta = 100\%$ plane as shown in Fig. 25. As can be seen, the optimized geometry that has a negative twist angle at the tip of the wing is leading to a weaker tip vortex that will result in an induced drag reduction and higher span efficiency. It must be noted that these results are obtained using the finest (G3) computational mesh.

Finally, the CPU times of the primal and adjoint solvers are studied using the original (OP-Tape) and the improved (ET-Tape) approaches utilized in the FDOT toolbox to further study the robustness of the proposed technique in enhancing the computational efficiency of the adjoint solver. Here, the computational cost of running the adjoint solver using the original and improved FDOT toolbox is described in terms of normalized CPU times with respect to that of the primal solver and the results are presented in Table 12. As discussed before, with the additional lift coefficient constraint utilized for this optimization problem, it is required to evaluate the recorded tape twice. Therefore, the overall computational cost of the adjoint solver will be exactly doubled. However, the computational cost of a single adjoint evaluation is still only a small multiple of that of the primal flow solver. Moreover, the proposed technique is capable of providing more than 17% improvement in the CPU time.

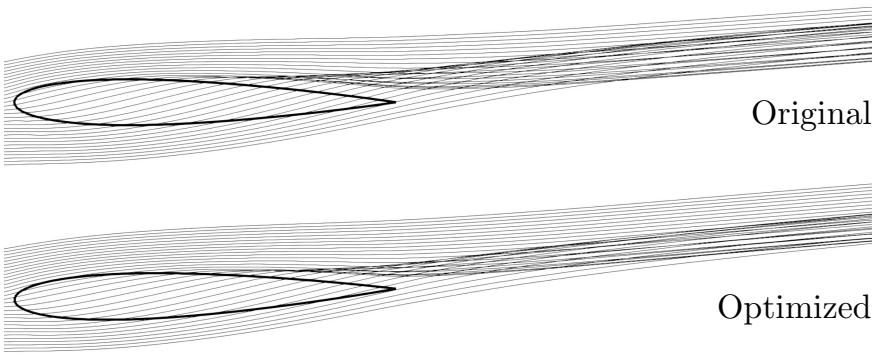


Figure 25. Stream-traces at the $\eta = 100\%$ plane for the original and deformed (twist optimized) geometries of the NACA 0012 wing showing the tip vortex forming downstream of the wing.

Table 12. Normalized CPU time comparisons between the original and the modified (present) FDOT toolbox for a single adjoint evaluation process.

FDOT Toolbox	Normalized CPU Time	Reduction (%)
Original	3.96	-
Improved (Present)	**3.27**	**17.6**

V. Conclusions

In this paper an improved version of the FDOT toolbox is developed which utilizes an expression-template approach for recording the tape. The proposed technique calculates the partial derivatives for each expression using the standard adjoint accumulation approach. Here, only the adjoints of the active variables on the right-hand-side of each expression are stored in the tape and the rest of the intermediate variables are removed from the memory. This process can significantly reduce the length of the tape, thus, resulting in significant reductions of the memory footprint for the adjoint solver.

The enhanced FDOT toolbox is applied to several aerodynamic design optimization problems with various levels of complexity. First, the unconstrained drag minimization of the NACA 0012 airfoil subject to inviscid transonic flow conditions was considered. Next, the constrained drag minimization problem for the RAE 2822 airfoil subject to turbulent transonic flow was studied where the objective function minimization is constrained with an equality constraint involving a target lift coefficient as well as two inequality constraints that require the moment coefficient and the airfoil area to be kept greater than desired values.

Another test case studied in this work is the lift-constrained drag minimization of the transonic ONERA M6 wing. Finally, the drag minimization of a rectangular NACA 0012 wing at a constant lift coefficient with respect to the twist angle distribution along the wing span is sought. It is shown that the proposed ET-Tape approach can effectively improve the computational and memory efficiency of the FDOT toolbox for all design optimization problems studied in this work.

VI. Acknowledgments

This material is based upon work supported by the National Science Foundation under grant No: CBET-1803760. The program managers are Dr. Ron Joslin and Dr. Shahab Shojaei-Zadeh. The authors greatly appreciate the support provided.

References

[1]Pironneau, O., "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, Vol. 64, No. 01, 1974, pp. 97–110.
[2]Jameson, A., "Aerodynamic design via control theory," *Journal of Scientific Computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[3]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.

[4]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.

[5]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.

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

[7]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.

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

[9]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.

[10]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.

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

[12]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.

[13]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.

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

[15]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.

[16]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.

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

[18]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*, Vol. 91, 2019, pp. 159–174.

[19]Djeddi, R. and Ekici, K., "Aerodynamic Shape Optimization Framework Based on a Novel Fully-Automated Adjoint Differentiation Toolbox," AIAA Paper 2019-0000, 2019.

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

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

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

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

[24]Griewank, A. et al., "On automatic differentiation," *Mathematical Programming: recent developments and applications*, Vol. 6, No. 6, 1989, pp. 83–107.

[25]Griewank, A., "On automatic differentiation and algorithmic linearization," *Pesquisa Operacional*, Vol. 34, No. 3, 2014, pp. 621–645.

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

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

[28]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.

[29]Nocedal, J. and Wright, S., *Numerical optimization*, Springer Science & Business Media, 2006.

[30]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.

[31]Kraft, D., "A software package for sequential quadratic programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[32]Lee, C., Koo, D., Telidetzki, K., Buckley, H., Gagnon, H., and Zingg, D. W., "Aerodynamic shape optimization of benchmark problems using jetstream," AIAA Paper 2015-0262, 2015.

[33]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.

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

[35]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.

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

[37]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.

[38]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.

[39]Bisson, F. and Nadarajah, S., "Adjoint-based aerodynamic optimization of benchmark problems," AIAA Paper 2014-1948, 2014.

[40]Bisson, F. and Nadarajah, S., "Aerodynamic shape optimisation of benchmark problems using SU2," AIAA Paper 2018-0412, 2018.