AIAA JOURNAL
Vol. 59, No. 1, January 2021

Novel Expression-Template-Based Automatic Differentiation of Fortran Codes for Aerodynamic Optimization

Reza Djeddi* and Kivanc Ekici†
University of Tennessee, Knoxville, Tennessee 37996

https://doi.org/10.2514/1.J059505

An elegant and robust approach for discrete adjoint-based automatic differentiation of the Fortran codes is proposed in this work. The technique aims at bridging the gap within the Fortran programming language that currently lacks certain metaprogramming paradigms that are readily available at compile time in C/C++ codes. More specifically, the present work uses an expression-based tape approach, which is the 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). The proposed expression-template-based approach is incorporated in our in-house AD toolbox, which currently is the only Fortran-based tool in the literature that uses a fixed-point-type operator-overloading adjoint sensitivity analysis. The improved version of the fast automatic differentiation using operator-overloading technique toolbox is then coupled with the in-house unstructured parallel compressible (UNPAC) flow solver for a robust design optimization framework (DOF), 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.

Nomenclature

A	=	airfoil cross-sectional area
b	=	wing semispan
C_D	=	drag coefficient for the airfoil or wing geometry
C_L	=	lift coefficient for the airfoil or wing geometry
C_M	=	moment coefficient for the airfoil or wing geometry
e	=	span efficiency of the wing
$f(x)$	=	unary operation with operand x
$g(x, y)$	=	binary operation with operands x and y
I	=	objective or cost function
$\mathcal{L}$	=	Lagrangian function
S	=	reference semispan wing area
t_i	=	ith active and passive variables in the expression tree
$\boldsymbol{U}$	=	vector of the computational fluid dynamics solutions (conservation variables)
v	=	adjoint tape entries or variables
$\boldsymbol{x}$	=	vector of design variables
x_i	=	ith active variable in the expression tree
$\overline{x}$	=	adjoint of the x variable based on the objective function I; that is, $\overline{x} = (\partial I/\partial x)$
α	=	angle of attack
γ	=	wing twist angle
ϵ	=	small perturbation parameter for finite difference approximations
Λ	=	wing aspect ratio
λ	=	Lagrange multiplier or adjoint solution
ξ, η, ζ	=	parametric coordinates corresponding to the (x, y, z) Cartesian coordinates

I. Introduction

PRESENTED in this work is the development and application of a first-of-its-kind approach to automatically compute discrete adjoint sensitivities for computational fluid dynamics (CFD) codes written in Fortran programming language. The toolbox developed herein that mimics the "expression-template metaprogramming" can be directly and rapidly coupled with many legacy flow solvers used in the aerospace industry. The motivation and the need for this research will be discussed in detail in what follows.

With the advances in computational science and technologies, aerodynamic shape optimization 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 of 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] have 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, whereas in the discrete adjoint approach [6,7], the computational 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 nonapproximative 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 because 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 mode requires the reversal of the entire expression tree for the primal solver, which makes it more challenging to implement. In terms of programming, the reverse AD mode can be performed using source-code transformation [8,9] or

Presented as Paper 2020-0885 at the AIAA Science and Technology Forum and Exposition (AIAA SciTech Forum), Orlando, FL, 6–10 January 2020; received 16 February 2020; revision received 10 July 2020; accepted for publication 31 August 2020; published online 30 November 2020.

*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.

operator overloading (OO), with the latter technique being the focus of the present work. The interested reader is referred to Ref. [10] 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 [11], Adept [12], CppAD [13], and CoDiPack [14] being the most commonly used tools for C/C++ programs and ADF [15], ADOL-F [16], AUTO_DERIV [17], DNAD [18], and dco/Fortran [19] being the most commonly used tools for Fortran programs. More recently, Djeddi and Ekici [20] have developed a novel OO/AD toolbox, called the FDOT (which stands for fast automatic differentiation toolbox based on operator-overloading technique), for automatically differentiating Fortran programs. The 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 solver, the FDOT toolbox has been effectively used in gradient-based optimization problems for designing optimal airfoil and wing topologies [21].

It is worth noting that the 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, both in terms of memory and CPU requirements [20,21]. However, due to the limitations of the Fortran programming language, the original version of the FDOT was at a disadvantage compared to more advanced OO/AD tools developed for C/C++ programs such as CoDiPack [14]. More specifically, the use of "expression templates" as a metaprogramming paradigm in C++ language has enabled 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 the FDOT uses 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 [which stands for unstructured parallel compressible (UNPAC) design optimization framework (DOF)] [21] 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, the 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 used in the FDOT toolbox as well as the design optimization framework are described. Finally, the method 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 [20] is capable of efficient and accurate evaluation of the gradient information for any given primal solver. Simply, it uses 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 [22,23].

Due to the fact that the FDOT uses 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, the 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 [24,25]. 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 preiterative process that handles grid preprocessing and flow initialization, 2) an iterative process that solves the flow equations, and 3) a postiterative 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 *steady* 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 [21].

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 $\boldsymbol{U}$ 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 the FDOT is depicted in Fig. 1.

The main goal of the present work is to improve the computational and memory efficiencies of operator-overload-based adjoint analyses using the Fortran programming language. This would enable the application of the approach to large-scale CFD solvers written in Fortran. As the first step, it is necessary to review the adjoint evaluation process using the FDOT OO/AD tool. When the expression tree is

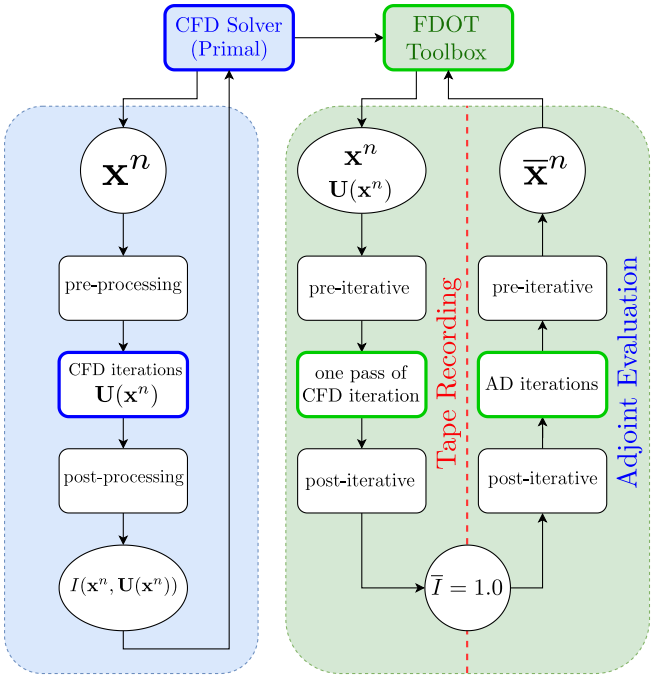


Fig. 1 Flowchart for the FDOT toolbox and its integration into the primal CFD solver (adopted from the works of Djeddi and Ekici [20,21]).

recorded into the operation stack, known as the "OP stack," 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:

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

$$c_2 = g(x, y) \xrightarrow{\text{to OP stack}} \text{index}(x, y), \text{tag}(g(\square, \Delta)), \text{value}(c_2), \text{adjoint}(\overline{c_2}) \tag{2}$$

It must be noted that with the preceding format, each tape entry will require 27 B of memory, of which 4 B are used for each of the argument indices, 1 B is used for an operation tag, 1 B is used for an "iterative" flag, 1 B is used for a "passive" flag, and 16 B are used for storing the primal and adjoint values (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 $\partial f/\partial x$, $\partial g/\partial x$, and $\partial g/\partial y$ 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 for steady design optimization are performed following a fully converged primal flow solution, even the nondifferentiable functions would be locally differentiable. For example, functions such as min and max, which are nonsmooth and nondifferentiable 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., $\partial f/\partial x$, $\partial g/\partial x$, or $\partial g/\partial y$), 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 approach is proposed to calculate the partial derivative information that can greatly enhance the performance of the adjoint evaluation process. By using 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 method, 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 through v_n correspond to k independent and one intermediate variables (with $n = k + 1$) leading 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 preceding expression, we can calculate the adjoints of the independent and intermediate variables as

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

The preceding formula is nothing but a repeated application of the chain rule of differentiation, which is the cornerstone of the automatic or algorithmic differentiation technique. In a CFD solver, each intermediate variable is calculated using an assignment operation (=) that can be defined on a single line or multiple (broken-down) lines of code. Each of these assignment operations can be viewed 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

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

Now, using Eq. (7) and the adjoint formulation [20], we can define the adjoints of independent and intermediate variables as

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

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 [20,21] 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 the reverse direction to calculate the partial derivatives. This partial derivative information is then recorded into a new derived-type class, called the "ET stack" (which stands for expression tape stack), 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 stack is described in the following for the expression shown in Eq. (8):

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

This leads to a memory footprint per tape entry of only 16 B (for double precision, this will include 4 B for the LHS and RHS indices and 8 B for the partial derivative value) compared to the 27 B of the original approach, whereas the length of the tape has also been reduced since only active variables will be accounted for. To fully understand the difference between independent and intermediate variables, let us look at the following expression:

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

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

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

one binary addition (+), one binary subtraction (−), and one binary multiplication (×), i.e.,

$$t_5 = t_1 + t_2$$
$$t_6 = t_3 - t_4$$
$$t_7 = t_5 \times t_6$$

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

$$t_8 = t_7^2$$

$$t_9 = t_8$$

Clearly, recording the tape for the expression shown in Eq. (11) to compute c leads to nine entries in Fortran as opposed to only four entries that will be recorded in the ET stack for each independent variable of x_1 through x_4. It is worth noting that these four independent variables have been defined before this expression, whereas the other variables are simply compiler-defined "intermediate" variables whose adjoints will not be useful beyond the scope of this expression. 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 [26]. This is mainly due to the fact that compilers use temporary or intermediate 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 three and five when averaged for all expressions involved in a nominal CFD solver [26,27].

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

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

As discussed earlier, each expression or statement in a CFD solver leads to the recording of ωn tape entries, with n being the number of operations involved and ω the computational effort factor. However, only a single adjoint value corresponding to the variable on the left-hand side of each expression is ultimately required. Therefore, the adjoints of the intermediate variables do not need to be included, which results in a vector that simply stores a single adjoint value per statement (or expression). The length of this vector, with an 8 B memory footprint per entry for double precision, becomes much shorter compared to the expression tape (ET stack). 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:

1) The OP stack, also known as the operation stack, records one entry per "operation." In the original implementation of the FDOT [20], this is the only recorded tape, which is used for adjoint evaluations (memory footprint: 27 B per entry for double precision).

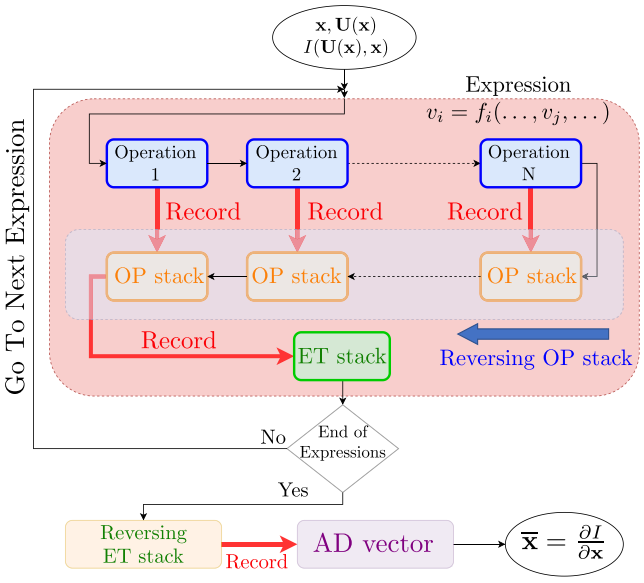


Fig. 2 Flowchart for the expression-based adjoint evaluation in the improved FDOT toolbox.

2) The ET stack, also known as the expression tape, records one entry per active variable per expression (memory footprint: 16 B per entry for double precision).

3) The AD vector, also known as the adjoint vector, records one entry per independent/intermediate variable (memory footprint: 8 B per entry for double precision).

It must be noted that the approach used 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 [14] to significantly enhance the adjoint evaluation performance by using expression templates at compile time. However, none of these capabilities and paradigms are available for Fortran programming language. Nonetheless, the present work focuses on addressing these issues by using 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. UNPAC is a grid-transparent Reynolds-averaged Navier–Stokes solver based on a vertex-based finite volume discretization approach [10,28]. 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 design optimization framework is provided in Fig. 3.

It must be noted that the optimization framework seeks optimal designs via an iterative process that 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 parametrization based on a free-form deformation (FFD) box approach [29].

A. Unconstrained Drag Minimization

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

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

where 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 (BFGS) algorithm [30].

B. 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

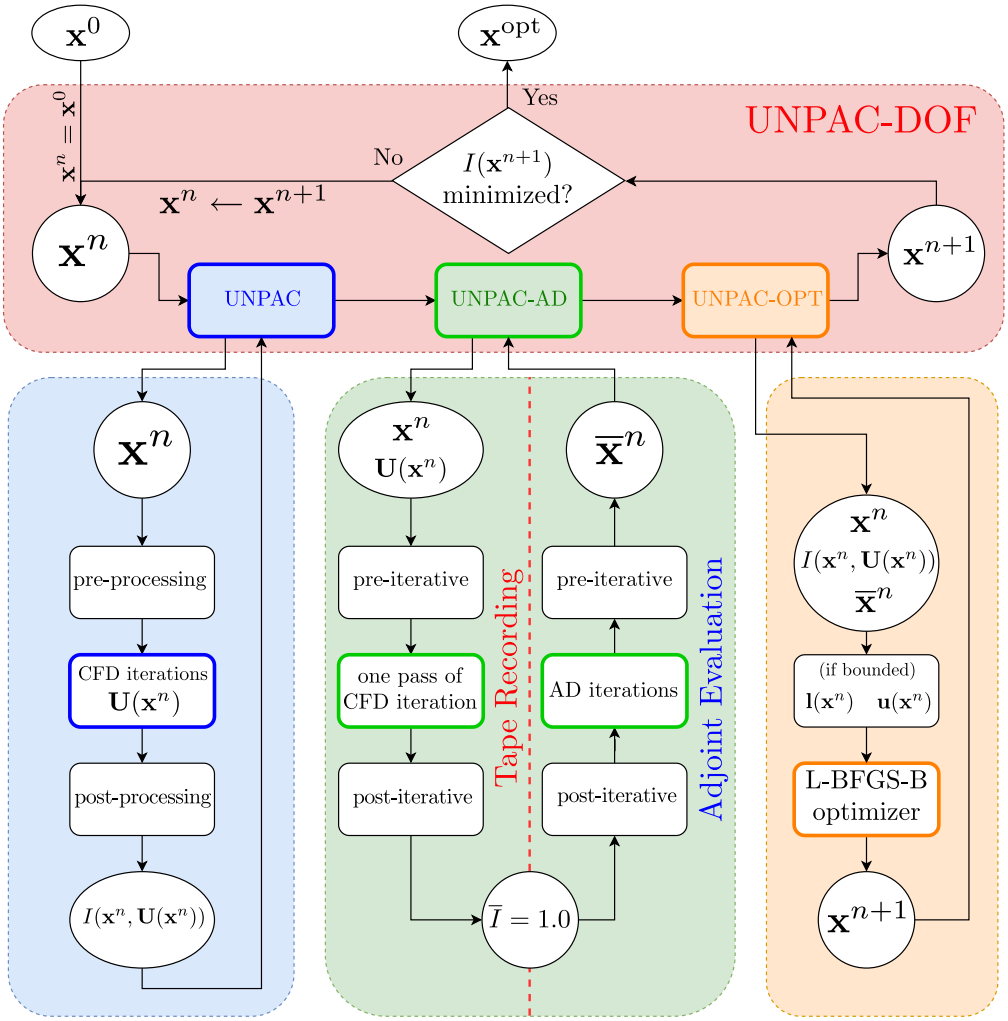


Fig. 3 Flowchart of the UNPAC-DOF; adopted from the work of Djeddi and Ekici [21].

$$\min_{\boldsymbol{x},\alpha} \quad C_D(\boldsymbol{x},\alpha) \tag{14}$$

subject to

$$C_L(\boldsymbol{x},\alpha) = C_L^*$$

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

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

where λ is the Lagrange multiplier. Therefore, to minimize the Lagrangian, the following Karush–Kuhn–Tucker conditions [31] must hold:

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

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

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

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

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

For fixed-lift drag minimization problems, UNPAC-DOF solves Eq. (20) using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bound-constrained (L-BFGS-B) optimizer [30,32]. The optimization is handled in two stages. First, the angle of attack α_{new} is updated through

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

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 $\boldsymbol{x}$ 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 design variables $\boldsymbol{x}_{\text{new}}$.

C. 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; and 2) geometric-based inequality constraints, e.g., a minimum airfoil area (in two dimensions) or wing volume (in three dimensions) must be maintained.

Such an aerodynamic design optimization problem can be described via

$$\min I(\boldsymbol{x}) = C_D(\boldsymbol{x}) - \frac{\partial C_D}{\partial C_L}\left(C_L^{\text{target}} - C_L(\boldsymbol{x})\right) \tag{24}$$

with respect to

$$\boldsymbol{x}, \alpha$$

subject to

$$C_M(\boldsymbol{x}) \geq C_M^{\min}$$
$$A(\boldsymbol{x}) \geq \text{Area}^{\min}$$

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

In such cases, the problem will be characterized in terms of the Karush–Kuhn–Tucker as well as the "geometric optimality," and "Fritz John" conditions [31]. Here, the quadratic programming problem would require the additional gradient information for the equality and inequality constraints with respect to the design variables. As described earlier, the FDOT uses 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 in the following:

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 "preiterative" portion of the tape that handles grid preprocessing. Therefore, by simply rewinding the preiterative portion of the tape, the sensitivity information for the geometric-based constraints is evaluated.

3) Solution-based constraints (e.g., moment coefficient constraint), however, are most often handled similarly 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, $C_D - (\partial C_D/\partial C_L)(C_L^{\text{target}} - C_L)$] and repeated for each additional solution-based constraint.

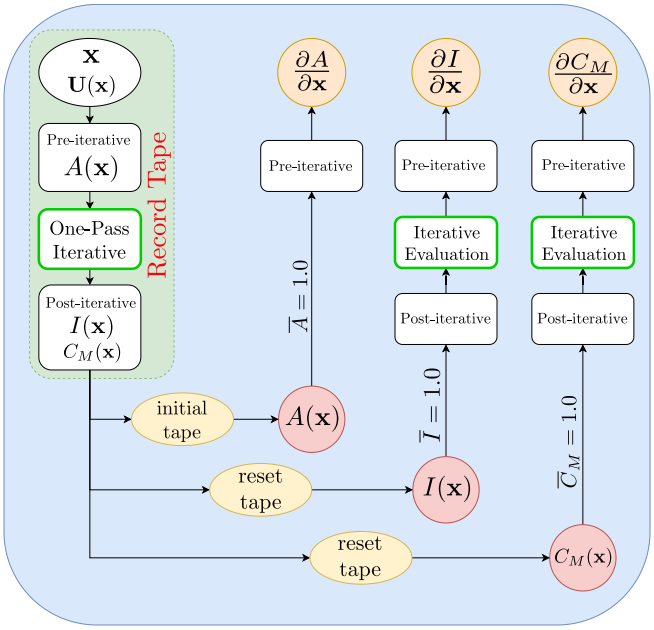


Fig. 4 Procedure for calculating the necessary gradient information for solving the constrained optimization problem described in Eq. (24).

The described procedure for calculating the entire gradient information is also shown in Fig. 4. Here, 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 uses a sequential least-squares quadratic programming (SLSQP) optimizer [33] 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 used 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 lift-constrained drag minimization problem with additional constraints involving the moment coefficient and airfoil area is considered for the turbulent transonic flow past the RAE 2822 airfoil. Next, the aerodynamic shape optimization of three-dimensional wing geometries is considered. In this regard, 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. Constrained Drag Minimization: Transonic RAE 2822 Airfoil

The first 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 can provide insights into the performance of the original and improved versions of the FDOT toolbox in evaluating sensitivity information.

The freestream 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 4800 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

$$\min I(\boldsymbol{x}) = C_D(\boldsymbol{x}) - \frac{\partial C_D}{\partial C_L}\left(C_L^{\text{target}} - C_L(\boldsymbol{x})\right) \tag{25}$$

with respect to

$$\boldsymbol{x}, \alpha$$

subject to

$$C_M(\boldsymbol{x}) \geq -0.092$$
$$A(\boldsymbol{x}) \geq A_{\text{base}}$$

where the target lift is set to $C_L^{\text{target}} = 0.824$, which initially requires an angle of attack of 2.9209 deg 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. Here, a free-form deformation box is used to parametrize the airfoil geometry. The FFD box tightly encloses the RAE 2822 airfoil, and the degrees of the Bernstein polynomials in the ξ and η directions are taken to be 15 and 1, respectively. 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 $\boldsymbol{x}$. 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 1. It must be pointed out that the ET-stack approach used in the improved version of the FDOT toolbox results in a much shorter adjoint tape compared to the original implementation. Clearly, the memory footprint per entry of the adjoint tape is reduced by more than 40% in the ET-stack approach, which proves the robustness of the proposed approach. This reduction results in a very significant reduction in the overall memory footprint of the adjoint solver.

The accuracy of the gradient evaluations using the original FDOT toolbox has been studied in previous works [10,20]. Additionally, since the proposed expression-template approach is theoretically a preaccumulation procedure, it is expected to achieve the same level of accuracy from the improved version of the FDOT toolbox compared to the original implementation. To examine this, the sensitivity of the drag coefficient with respect to the angle of attack is considered. As shown in Table 2, the sensitivity values obtained from the original and improved versions of the FDOT toolbox are compared to the finite difference approximations using a first-order backward difference as well as a second-order central difference scheme. The presented results show the accuracy of the gradient evaluations using the FDOT toolbox. Additionally, it is shown that the proposed technique, while being extremely efficient in terms of memory footprint, has essentially no effect on the accuracy of the gradient calculations.

Table 2 Comparison of the drag sensitivity calculations with regard to angle of attack ($\partial C_D/\partial\alpha$) for the RAE 2822 airfoil case

Calculation	Value
Finite difference (first order)	**0.2784**870397
Finite difference (second order)	**0.278490**3021
FDOT (original)	**0.27849028033512**
FDOT (improved)	**0.27849028033517**

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., the drag coefficient with the lift constraint as well as the moment coefficient. Additionally, the preiterative portion of the tape needs to be reevaluated 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. While this feature makes FDOT a robust sensitivity analysis tool for constrained optimization problems, improving the tape recording and adjoint evaluation procedure by breaking down the tape structures into three smaller stacks is the subject of an ongoing research to further improve the computational efficiency of the adjoint solver.

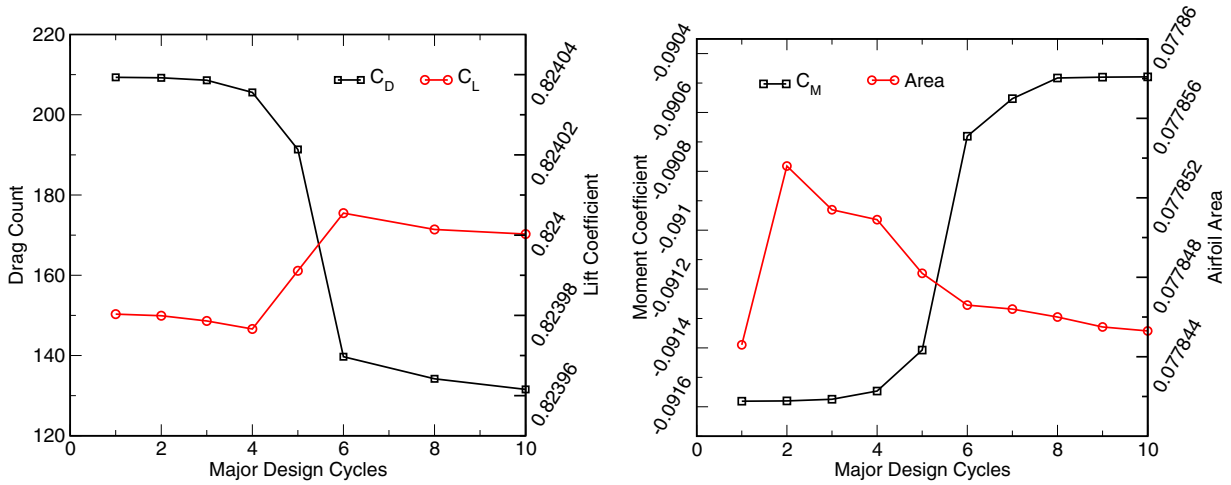
Having presented the performance test results for the improved FDOT toolbox, we can now focus our attention to the aerodynamic design optimization results. First, the convergence histories of the objective function and the constraints for this optimization problem are presented. These results are shown for the drag count, lift coefficient, moment coefficient, and the airfoil area in Fig. 5. As shown here, 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 a 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. 6. 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. As shown in Fig. 6, the shock can be completely eliminated if only the lift constraint is enforced. In a similar optimization study, Lee et al. [34] have shown that by using a lower degree for the Bernstein polynomials, the shock can be further alleviated or even eliminated even in the presence of all constraints. 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 [34].

The weakening of the shock during the design optimization process can also be shown via the Mach number contour plots for the original and optimized cases. These results are shown in Fig. 7 and exhibit a weaker shock-/boundary-layer interaction for the deformed RAE 2822 airfoil, which results in a reduced drag count for this airfoil

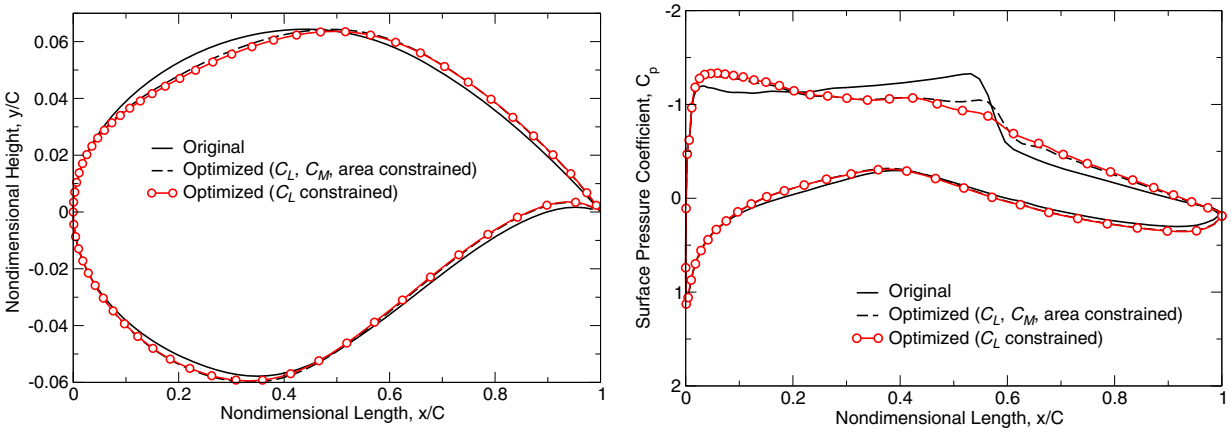
Table 1 Comparison of tape length and memory footprint for the original and improved FDOT toolboxes (transonic RAE 2822 constrained drag minimization problem)

FDOT toolbox	Stack/vector type	Stack/vector length	Memory per entry, B	Memory breakdown, GB	Total memory, GB	Reduction, %
Original	OP stack	326,909,670	27	8.22	8.22	— —
Improved	ET stack	221,958,575	16	3.30	4.28	**47.3**
	AD vector	130,534,725	8	0.98		



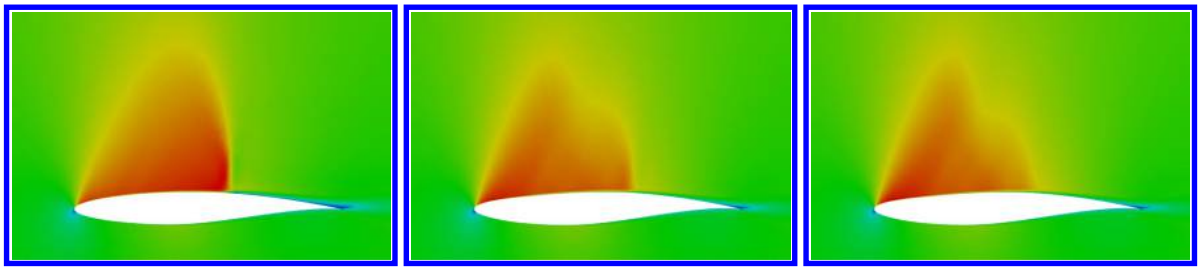
a) C_D and C_L b) C_M and area

Fig. 5 Convergence history of the objective function (C_D) as well as the constraints for the drag minimization of the RAE 2822 airfoil.



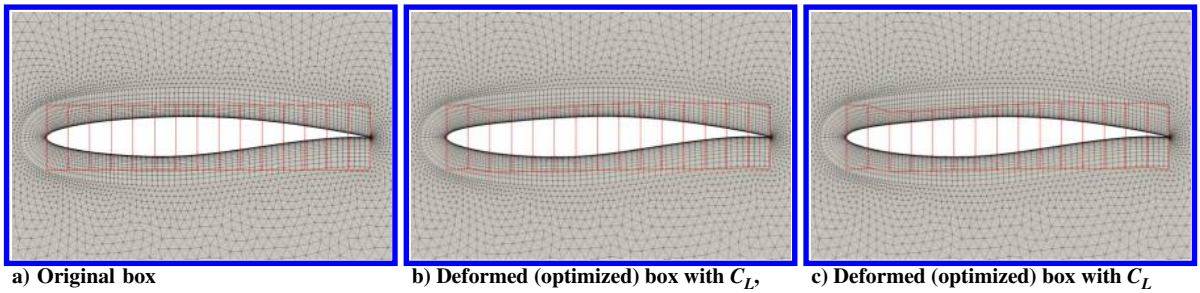
a) Airfoil shape b) Surface pressure coefficient, C_p

Fig. 6 Comparison of RAE 2822 airfoil shape and the surface pressure coefficients for the original and optimized geometries.



a) Original design b) Optimized design with C_L, C_M, and A constraints c) Optimized design with C_L constraint

Fig. 7 Contour field of Mach number for the turbulent transonic flow past RAE 2822 airfoil.



a) Original box b) Deformed (optimized) box with C_L, C_M, and A constraints c) Deformed (optimized) box with C_L constraint

Fig. 8 Original and deformed FFD boxes that parametrize the RAE 2822 airfoil for the constrained drag minimization problem.

Table 3 Normalized CPU time comparisons between the original and the modified (present) FDOT toolboxes for a single adjoint evaluation process

FDOT toolbox	Normalized CPU time	Reduction, %
Original	3.11	— —
Improved (present)	**2.56**	**17.7**

at the present flow conditions. Once again, the case with only the lift constraint is also considered, which shows a complete elimination of the shock that used to form on the suction side of the airfoil. Next, the FFD box deformation as well as the interior mesh deformation are presented in Fig. 8 for this case. It is shown that this 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 costs of running the adjoint solver using the original and improved FDOT toolboxes are described in terms of normalized CPU times with respect to that of the primal solver, and the results are presented in Table 3. As discussed before, with the additional constraints used 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.

It must be noted that the computational time in the early stages of the adjoint solver, which involve recording the OP stack and evaluating the partial derivatives for the ET stack, will be increased due to the preaccumulation procedure proposed in this work. However, with the significant reduction of the tape size, it is possible to achieve huge time savings in the adjoint evaluation process. The breakdowns of the relative CPU times are shown in Fig. 9, which shows the increase in CPU time during stack recording and a decrease in computational time during adjoint evaluations. Since the stack recording process takes only a fraction of the time required for running the adjoint solver, it would be possible to have overall savings in CPU time using the proposed expression-template-based approach.

B. 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 [35,36]. The geometry of the transonic M6 wing is based on the symmetric airfoil sections of the ONERA D type, which have a maximum thickness-to-chord ratio of 10%. The M6 wing has a sweep angle of 30 deg at the leading edge and an aspect ratio of 3.8, and it is tapered with a ratio of 0.562. The flow conditions are set according to the experiments carried out by Schmitt and Charpin [37] with a freestream Mach number of 0.8395 and an angle of attack of 3.06 deg. 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, whereas the unconstrained or lift-constrained drag minimization of this wing has also been studied extensively [38–40].

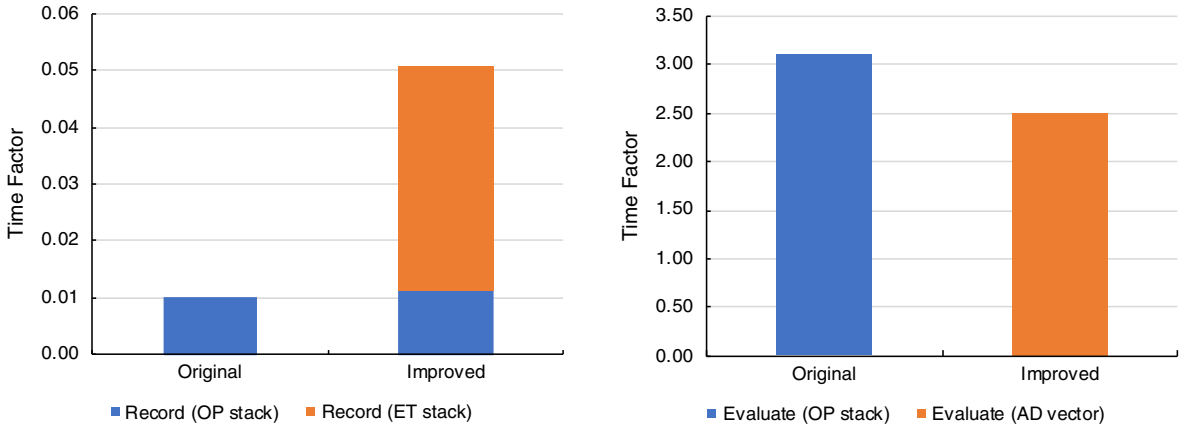
The computational grid used for this study consists of a rectangular outer boundary that extends about 10 mean chord lengths on each side. The far-field and near-field views of the grid used for this test case are shown in Fig. 10. The fully unstructured grid is made of 108,396 nodes and 582,752 tetrahedral 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.

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 4. Once again, the new approach used in the FDOT toolbox, where the expression and adjoint tapes are used, 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 proves that the proposed technique is capable of dramatically improving the memory efficiency of the FDOT toolbox for adjoint sensitivity analysis.

Once again, for the design optimization problem, the wing geometry is parametrized using a hexahedral free-form deformation 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 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 histories of the drag and lift coefficients are plotted for major design cycles in Fig. 11. Here, 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 5.

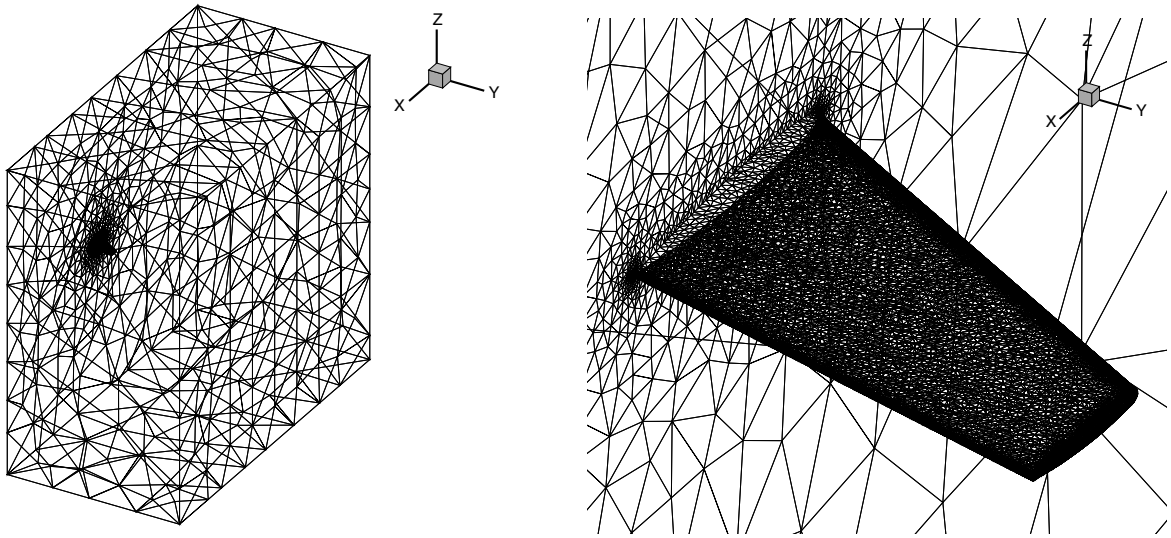
Next, the numerical results obtained using the original ONERA M6 wing geometry are compared against the experimental data of Schmitt and Charpin [37]. These solutions are reported at six different sections along the span of the wing and the distribution of the surface pressure coefficient at each section. These results are presented in Fig. 12, which shows a good agreement between the UNPAC solver results and the experimental data.

Additionally, the surface solutions at the last three spanwise sections are compared for the original and optimized geometries, and the



a) Stack recording relative time factors **b) Adjoint evaluation relative time factors**

Fig. 9 Breakdowns of the relative time factors for the original and improved versions of the FDOT toolbox (RAE 2822 airfoil constrained optimization).



a) Computational domain with 582, 752 cells **b) Surface mesh with 38, 756 triangles**

Fig. 10 Volume and surface meshes used for the transonic flow past the ONERA M6 wing.

Table 4 Comparison of tape length and memory footprint for the original and improved FDOT toolbox (transonic ONERA M6 fixed-lift drag minimization problem)

FDOT toolbox	Stack/vector type	Stack/vector length	Memory per entry, B	Memory breakdown, GB	Total memory, GB	Reduction, %
Original	OP stack	560,731,841	27	14.1	14.1	— —
Improved	ET stack	381,297,651	16	5.68	6.81	**51.7**
	AD vector	151,666,032	8	1.13		

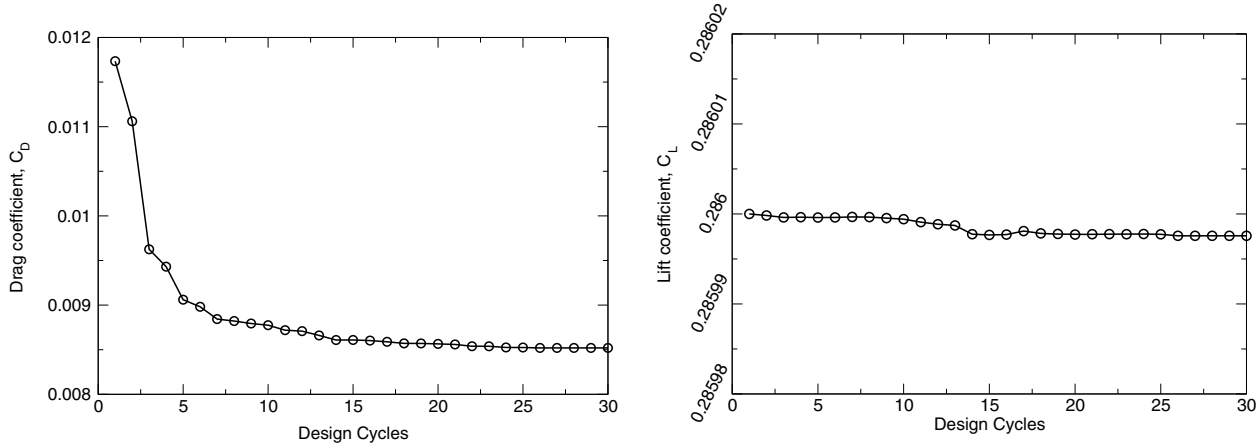
distributions of the surface pressure coefficients at each section are shown in Fig. 13. As shown here, the strong shock close to the tip of the wing is completely eliminated for the optimized geometry. This strong shock is the main contributor to the drag, and its elimination has proved to drastically reduce the drag coefficient. It is worth noting that the slight oscillations that occur around the leading edge of the wing are mainly due to the number and location of the FFD box control points.

Next, the contour plots of the surface pressure are presented for the original and optimized M6 wings. These results are presented in Fig. 14 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. 15.

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 6. As shown here, 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.

C. 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 semispan of $3.06c$, where the last $0.06c$ consists of a



a) Drag coefficient C_D **b) Lift coefficient C_L**

Fig. 11 Convergence histories for the drag and lift coefficients for the fixed-lift drag minimization of the ONERA M6 wing.

Table 5 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

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 freestream Mach number. The optimization problem is described as

$$\min C_D(\gamma(y)) \tag{26}$$

with respect to

$$\gamma(y)$$

subject to

$$C_L(\gamma(y)) = 0.375$$

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 [41] and Lee et al. [34], as well as more recently by Yang and Da Ronch [42] 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 [41] have used computational grids with more than 13 million nodes, whereas Lee et al. [34] 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. To reduce the number of degrees of freedom for the computational mesh, a hybrid grid generation approach is used 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 7.

Once again, the FFD box approach is used to parametrize 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

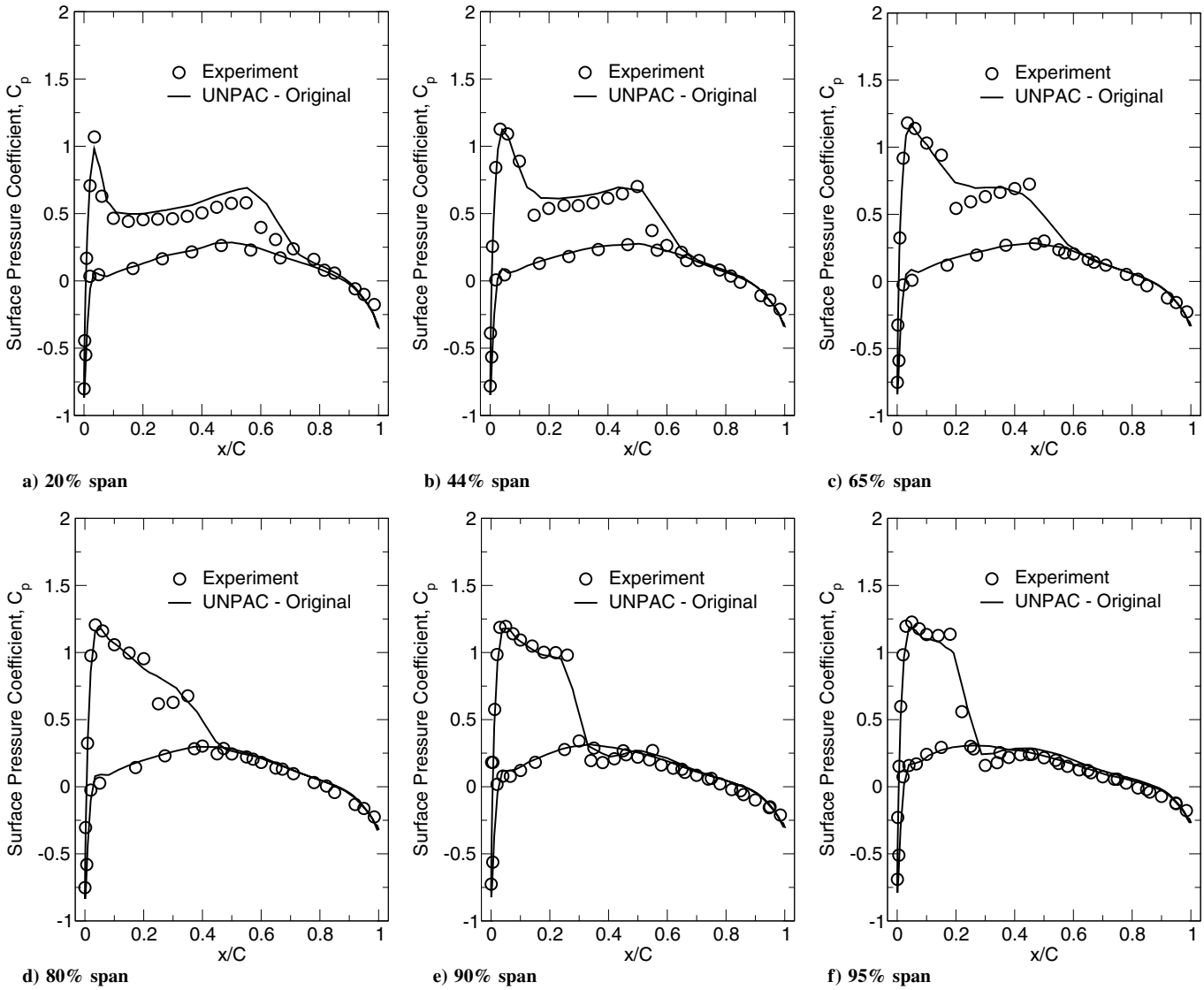
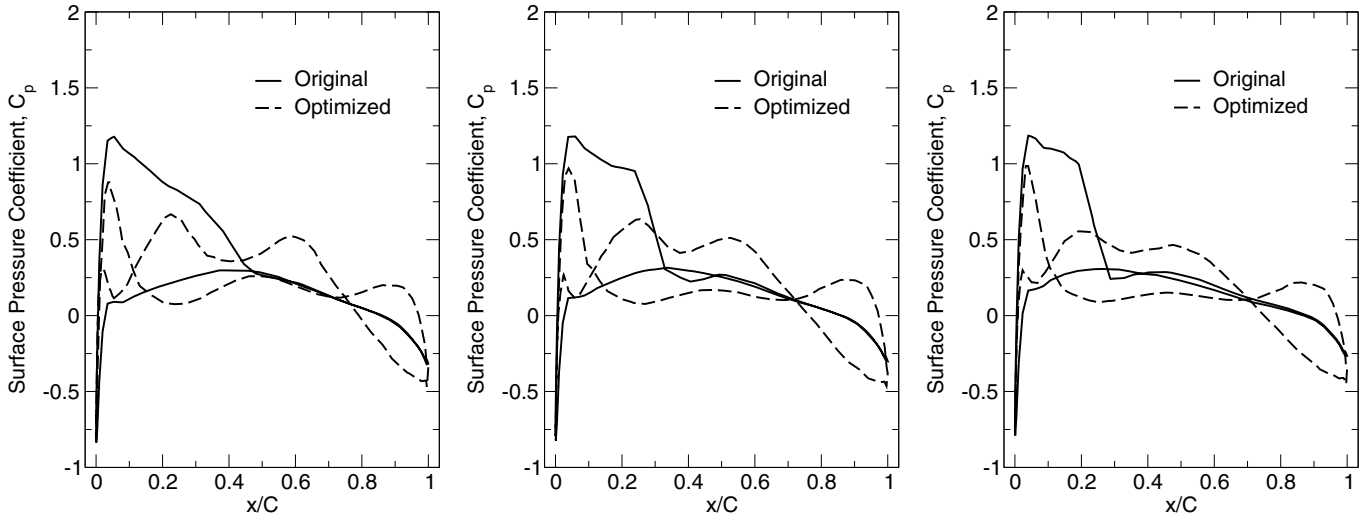
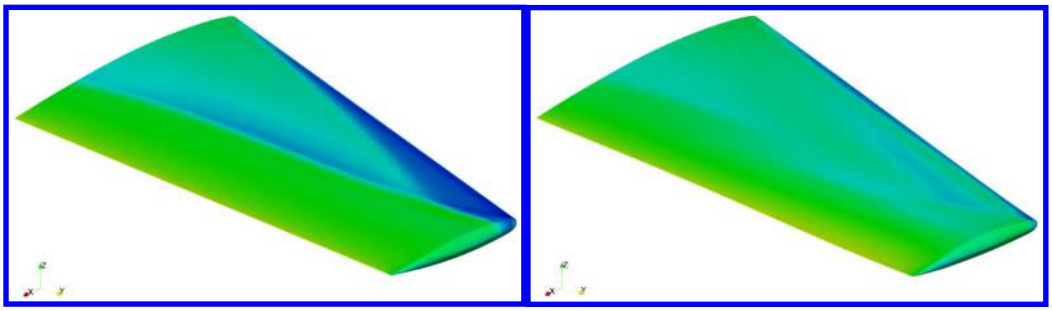


Fig. 12 Surface pressure coefficients for the inviscid transonic flow past ONERA M6 wing compared to experimental data [37].



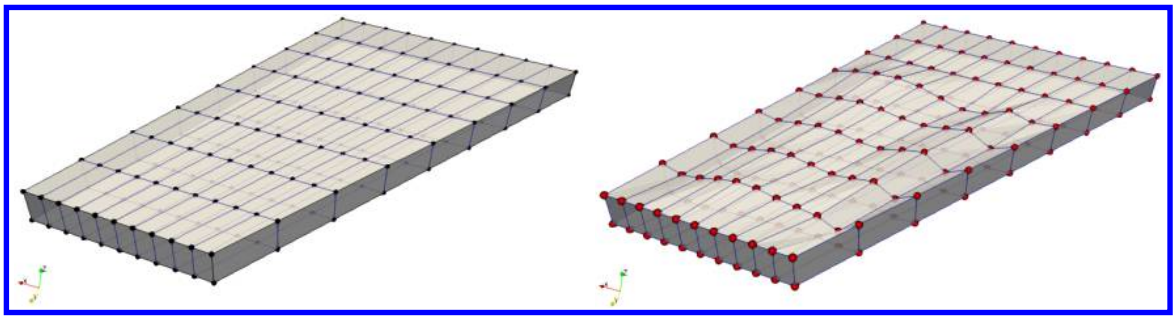
a) 80% span b) 90% span c) 95% span

Fig. 13 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

Fig. 14 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

Fig. 15 Original and deformed (optimized) FFD boxes parametrizing the ONERA M6 wing for the fixed-lift drag minimization problem.

benchmark guidelines for reporting the twist angles (see Fig. 16). 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.

During the design optimization process, the 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.

Table 6 CPU timings for the primal and adjoint solvers 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**

Table 7 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}

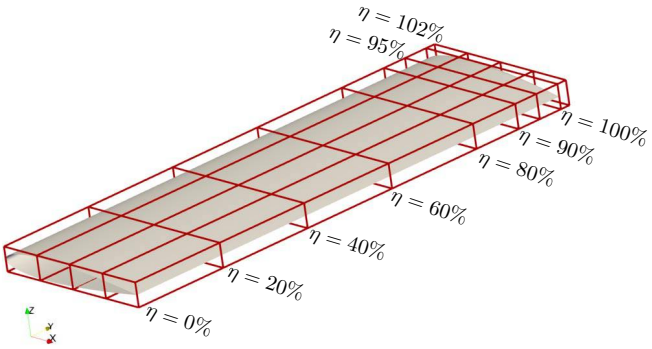


Fig. 16 Free-form deformation box and its control points used to parametrize the rectangular NACA 0012 wing for the twist optimization test case.

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 8 using the original and the improved (present) approaches. Note that only the finest grid (G3) is used for this memory-efficiency study because the tape length and memory footprint are directly related to the number of degrees of freedom of the CFD solver. Once again, it is shown that the proposed ET-stack approach offers a significant reduction in the tape length, which consequently lowers the memory footprint of the adjoint solver. Moreover, the memory footprint reductions for the three-dimensional test cases happen to be more significant compared to the two-dimensional tests. Overall, the ET-stack approach proposed in this work can provide more than a 35% reduction in the tape length and an almost 60% reduction in the memory footprint of the adjoint solver.

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, three grid levels are used to solve the design optimization problem while using the same FFD box and shape parametrization 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 semispan area that will be used for nondimensionalization 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 9. 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. 17.

As shown in Table 9, 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 the best reduction in drag count and improvement in span efficiency are achieved for the G2 grid, whereas the finest grid (G3) provides the highest span efficiency for the NACA 0012 wing.

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. 18, 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 expected and shown in Fig. 18, 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, 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. 19. Also, the untwisted and optimally twisted wing geometries as well as the FFD box deformation are presented in Fig. 20. Clearly, 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.

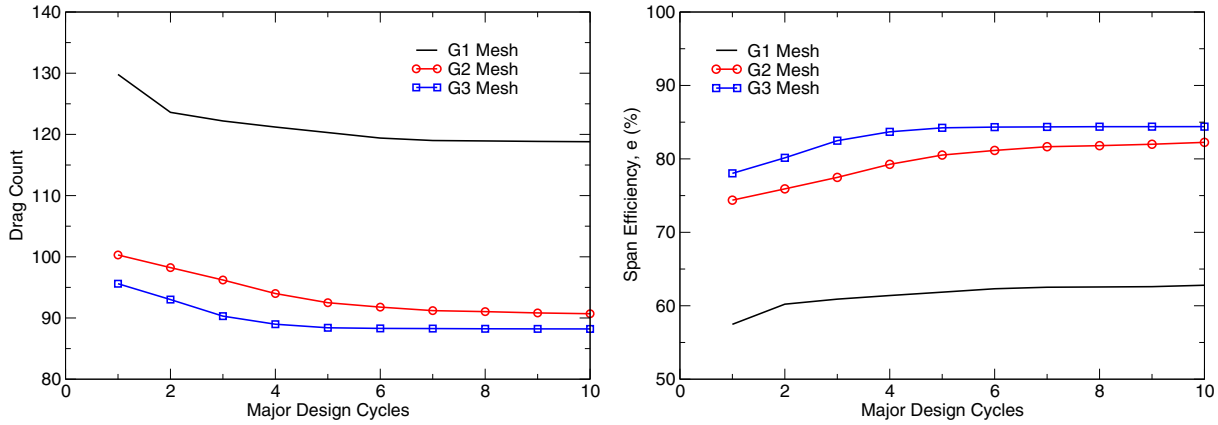
To further study the effect of the optimal twist distribution on the tip vortex of the NACA 0012 wing, contours of the Q criterion are plotted at the $\eta = 100\%$ plane as shown in Fig. 21. The Q criterion presented here clearly shows that the optimized geometry with a negative twist angle at the tip of the wing is leading to a weaker tip vortex that will result in an induced drag reduction and higher span

Table 8 Details of tape length and memory footprint: rectangular NACA 0012 wing twist optimization problem using the finest (G3) grid resolution

FDOT toolbox	Stack/vector type	Stack/vector length	Memory per entry, B	Memory breakdown, GB	Total memory, GB	Reduction, %
Original	OP stack	545,558,068	27	13.8	13.8	— —
Improved	ET stack	350,793,838	16	5.22	5.7	**58.7**
	AD vector	64,424,509	8	0.48		

Table 9 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**



a) Drag coefficient, C_D b) Span efficiency, e

Fig. 17 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.

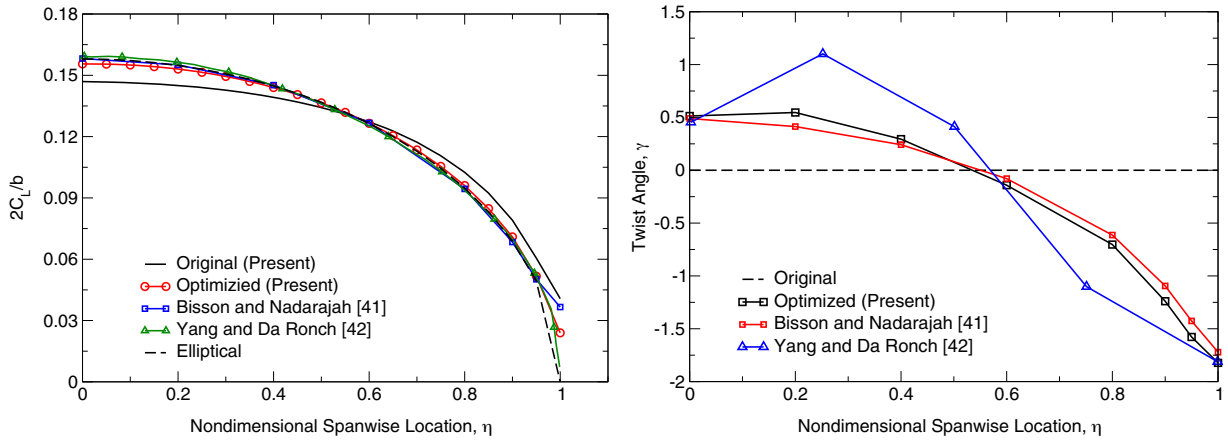


Fig. 18 Sectional lift (left) and twist angle (right) distributions along the wing span.

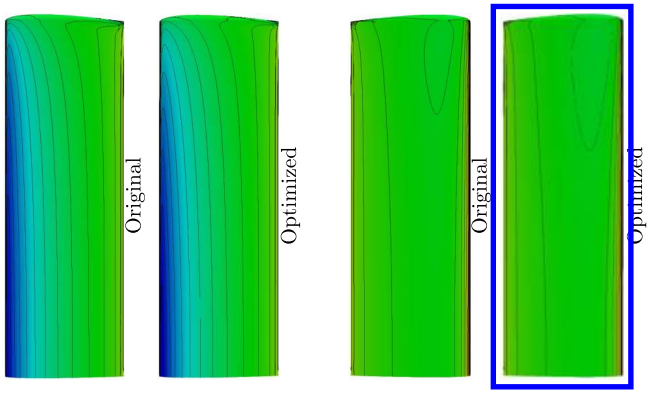
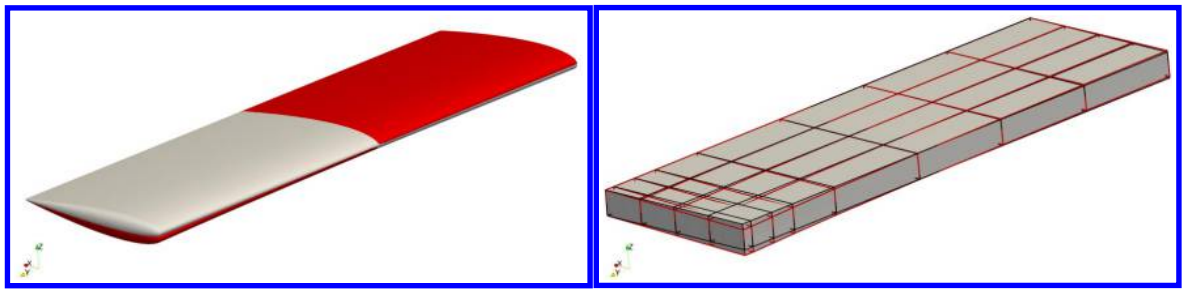


Fig. 19 Contour field of pressure on the top (left) and bottom (right) surfaces of the NACA 0012 wing for the fixed-lift drag minimization (twist optimization) problem.

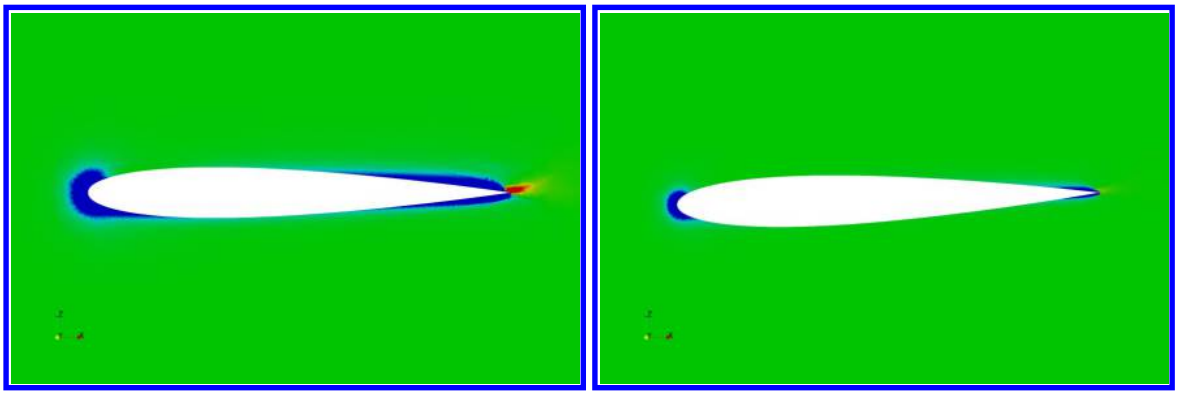
efficiency. It must be noted that these results are obtained using the finest (G3) computational mesh.

Finally, the CPU times of the primal and adjoint solvers are studied using the original (OP stack) and the improved (ET stack) approaches used 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 costs of running the adjoint solver using the original and improved FDOT toolboxes are described in terms of normalized CPU times with respect to that of the primal solver, and the results are presented in Table 10. As discussed before, with the additional lift coefficient constraint used 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 a 17% improvement in the CPU time.



a) Original vs twisted wing b) Initial and deformed FFD boxes

Fig. 20 Original and deformed (optimized) geometries of the wing (left) and the deformed FFD box (right), with the deformed geometry shown in red.



a) Original wing b) Twisted (optimized) wing

Fig. 21 Contours of Q criterion at the $\eta = 100\%$ plane for the original and deformed (twist optimized) geometries of the NACA 0012 wing.

Table 10 Normalized CPU time comparisons between the original and the modified (present) FDOT toolboxes 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 that uses an expression-template approach for recording the tape. The use of the expression templates via metaprogramming paradigms is readily available in the C/C++ programming language, which has resulted in the development of robust AD tools like CoDiPack [14]. However, the Fortran programming language is not equipped with such a feature. Currently, there are still many CFD solvers and legacy codes that are written in Fortran that rely solely on source-code transformation for automatic differentiation. In this work, the idea of the expression template in adjoint sensitivity analysis has been made possible within the Fortran language via the improved version of the operator-overloading-based FDOT toolbox. 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 sides 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 constrained drag minimization problem for the RAE 2822 airfoil subject to turbulent transonic flow is 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-stack approach can effectively improve the computational and memory efficiency of the FDOT toolbox for all design optimization problems studied in this work.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant no. CBET-1803760. The Program Managers are Ron Joslin and 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. 1, 1974, pp. 97–110. https://doi.org/10.1017/S0022112074002023

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

[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. https://doi.org/10.2514/2.271

[4] Anderson, W. K., and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers and Fluids*, Vol. 28, No. 4, 1999, pp. 443–480. https://doi.org/10.1016/S0045-7930(98)00041-3

[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. https://doi.org/10.2514/2.1961

[8] 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. 1–43. https://doi.org/10.1145/2450153.2450158

[9] Bischof, C., Khademi, P., Mauer, A., and Carle, A., "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs," *IEEE Computational Science and Engineering*, Vol. 3, No. 3, 1996, pp. 18–32. https://doi.org/10.1109/99.537089

[10] Djeddi, S., "Towards Adaptive and Grid-Transparent Adjoint-Based Design Optimization Frameworks," Ph.D. Thesis, Univ. of Tennessee, Knoxville, TN, 2018.

[11] 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. https://doi.org/10.1145/229473.229474

[12] 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. 1–0. https://doi.org/10.1145/2560359

[13] Bell, B. M., "CppAD: A Package for C++ Algorithmic Differentiation," *Computational Infrastructure for Operations Research*, Vol. 57, No. 10, 2012, http://www.coin-or.org/CppAD

[14] 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.

[15] 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. https://doi.org/10.1016/j.cpc.2005.01.011

[16] Shiriaev, D., Griewank, A., and Utke, J., "A User Guide to ADOL-F: Automatic Differentiation of Fortran Codes," *Computational Differentiation: Techniques, Applications, and Tools*, Dept. of Mathematics, TU Dresden, 1996, pp. 375–384.

[17] 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. https://doi.org/10.1016/S0010-4655(99)00513-5

[18] 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. https://doi.org/10.1016/j.cpc.2012.12.025

[19] Naumann, U., *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*, Vol. 24, SIAM, Philadelphia, PA, 2012.

[20] 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, Aug. 2019, pp. 159–174. https://doi.org/10.1016/j.ast.2019.05.004

[21] Djeddi, R., and Ekici, K., "Aerodynamic Shape Optimization Framework Based on a Novel Fully-Automated Adjoint Differentiation Toolbox," AIAA Paper 2019-3201, 2019.

[22] Wolfe, P., "Checking the Calculation of Gradients," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 8, No. 4, 1982, pp. 337–343. https://doi.org/10.1145/356012.356013

[23] Baur, W., and Strassen, V., "The Complexity of Partial Derivatives," *Theoretical Computer Science*, Vol. 22, No. 3, 1983, pp. 317–330. https://doi.org/10.1016/0304-3975(83)90110-X

[24] Christianson, B., "Reverse Accumulation and Attractive Fixed Points," *Optimization Methods and Software*, Vol. 3, No. 4, 1994, pp. 311–326. https://doi.org/10.1080/10556789408805572

[25] Christianson, B., "Reverse Accumulation and Implicit Functions," *Optimization Methods and Software*, Vol. 9, No. 4, 1998, pp. 307–322. https://doi.org/10.1080/10556789808805697

[26] Griewank, A., "On Automatic Differentiation," *Mathematical Programming: Recent Developments and Applications*, Vol. 6, No. 6, 1989, pp. 83–107.

[27] Griewank, A., "On Automatic Differentiation and Algorithmic Linearization," *Pesquisa Operacional*, Vol. 34, No. 3, 2014, pp. 621–645. https://doi.org/10.1590/0101-7438.2014.034.03.0621

[28] Djeddi, R., and Ekici, K., "Solution-Based Adaptive Mesh Redistribution Applied to Harmonic Balance Solvers," *Aerospace Science and Technology*, Vol. 84, Jan. 2019, pp. 543–564. https://doi.org/10.1016/j.ast.2018.11.003

[29] Chauhan, D., Chandrashekarappa, P., and Duvigneau, R., "Wing Shape Optimization Using FFD and Twist Parameterization," *12th Aerospace Society of India CFD Symposium*, Bangalore, India, Aug. 2010.

[30] Liu, D. C., and Nocedal, J., "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, Vol. 45, Nos. 1–3, 1989, pp. 503–528. https://doi.org/10.1007/BF01589116

[31] Nocedal, J., and Wright, S., *Numerical Optimization*, Springer Science and Business Media, New York, 2006.

[32] 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. https://doi.org/10.1137/0916069

[33] Kraft, D., "A Software Package for Sequential Quadratic Programming," *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, DFVLR Obersfaffeuhofen, Germany, 1988.

[34] 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.

[35] 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. https://doi.org/10.1016/j.jcp.2008.02.028

[36] Howison, J. C., "Aeroelastic Analysis of a Wind Turbine Blade Using the Harmonic Balance Method," Ph.D. Thesis, Univ. of Tennessee, Knoxville, TN, 2015.

[37] Schmitt, V., and Charpin, F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," *Experimental Data Base for Computer Program Assessment*, Report of the Fluid Dynamics Panel Working Group 04, AGARD AR-138, May 1979.

[38] Reuther, J., and Jameson, A., "Aerodynamic Shape Optimization of Wing and Wing-Body Configurations Using Control Theory," AIAA Paper 1995-0123, 1995.

[39] 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. https://doi.org/10.2514/2.1765

[40] 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. https://doi.org/10.2514/1.J051192

[41] Bisson, F., and Nadarajah, S., "Adjoint-Based Aerodynamic Optimization of Benchmark Problems," AIAA Paper 2014-1948, 2014.

[42] Yang, G., and Da Ronch, A., "Aerodynamic Shape Optimisation of Benchmark Problems Using SU2," AIAA Paper 2018-0412, 2018.

S. Fu
Associate Editor