Efficient Discrete Adjoint Approach Based on the Approximate Newton-Krylov Method for Aerodynamic Design Optimization

Reza Djeddi[*] and Kivanc Ekici[†]

University of Tennessee, Knoxville, Tennessee 37996

An extremely efficient, robust, and fully-automated sensitivity analysis toolbox based on the discrete adjoint approach is developed in this work. The proposed framework relies on an in-house automatic differentiation (AD) toolbox, called FDOT, that was originally developed by the authors. The first-of-its-kind FDOT toolbox uses the operator-overloading technique to automatically differentiate Fortran codes while utilizing advanced techniques for computational and memory efficiency without sacrificing the fully-automated features. The present work aims at extending the AD toolbox using a recursive technique for evaluating the partial derivatives. The resulting partial derivatives are required for solving the equations arising from the application of the discrete adjoint method to the primal CFD solver. Improving upon the earlier version of the novel Discrete Adjoint based on a Recursive Technique (DART) toolbox that was originally developed by the authors, this approach is modified to utilize an Approximate Newton-Krylov (ANK) solution technique. This modification is shown to significantly increase the computational efficiency of the toolbox, especially for large-scale complex flow problems involving realistic geometries. At its core, the DART toolbox combines both the adjoint and tangent modes of automatic differentiation resulting in a very efficient discrete adjoint framework. Finally, the computed adjoint flow field is used for calculating the total derivatives of the quantities of interest which can be directly used for gradient-based design optimization. It must be noted that the entire discrete adjoint process is done internally by the DART toolbox with little to no user intervention. The improved version developed in this work is ultimately used to improve airfoil and wing designs for minimized drag or maximized efficiency.

I. Introduction

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

The most important and challenging part of the gradient-based design optimization approach is the calculation of the gradient information. Traditionally, finite difference (FD) method has been used for

[*]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. Copyright by the authors.

this purpose although the step-size dilemma associated with this approximative technique have led to the introduction of the complex-step (CS) method.[1] While the step-size is not an issue with the latter approach, the computational cost for both FD and CS techniques is linearly proportional to the number of design variables since repeated objective function evaluations are necessary to obtain the entire gradient information. It must be noted that the FD or CS techniques only provide "approximations" to the desired derivatives. As an alternative, the algorithmic or automatic differentiation (AD) based on the chain rule result in analytically evaluated derivative values without any truncation error.

Automatic differentiation has two basic modes of operation known as forward/tangent or reverse/adjoint corresponding to a *top-down* or a *bottom-up* strategy of accumulating partial derivatives, respectively. It is important to note that the computational cost of the tangent AD is proportional to the number of independent variables while the computational cost of the adjoint AD is a function of the number of dependent quantities of interest. An in-depth discussion about the advantages and disadvantages of each approach is presented in Ref.[2] These methods have been extensively used for aerodynamic design optimization in the past two decades.[3–5] The adjoint method can be classified in two categories of (1) discrete[6,7] and (2) continuous[8,9] approaches depending on the order with which the adjoint equations are derived and solved. In the discrete approach, the discretized governing equations are adjoined and solved to obtain gradient information. As a result, the adjoint equations have the same level of computational complexity as the governing (primal) equations and the same numerical procedure is followed for both the primal and adjoint solvers. On the other hand, in the continuous adjoint method, the adjoint method is first applied to the governing equations to derive the continuous adjoint equations which are then discretized and solved. Therefore, special boundary conditions and numerical procedures need to be considered to discretize and solve the continuous adjoint equations[10] adding extra complexity and effort to the process. The most important feature of the adjoint methods is that the computational cost of gradient evaluation is independent of the number of design variables which makes this class of techniques a prime candidate for aerodynamic design optimization involving hundreds, if not thousands, of design variables.

On the programming side, the AD approach can be implemented using (1) source code transformation (SCT)[11–14] and (2) operator overloading (OO).[15–18] Interested readers are referred to the work of Griewank and Walther[19] for a detailed comparison between the two approaches. As discussed earlier, the development of the complementary sensitivity analysis solvers can be substantially simplified using automatic differentiation which can be carried out in forward or reverse modes based on the direction of the gradient propagation. Recently, Djeddi and Ekici[20] have developed a Fortran-based OO/AD toolbox based on the discrete adjoint method that uses a fixed-point iteration approach[21,22] for adjoint evaluations. The $\underline{\text{F}}$ast automatic $\underline{\text{D}}$ifferentiation based on $\underline{\text{O}}$perator-overloading $\underline{\text{T}}$echnique (FDOT) toolbox[20] is capable of efficiently and accurately calculating the sensitivity information of a cost function with respect to any design variable. While the adjoint mode of AD was the focus of the earlier research involving the FDOT toolbox,[2,20,23,24] the present work focuses on the application of both tangent and adjoint AD modes for developing an efficient and extremely robust sensitivity analysis tool that directly forms and solves the linearized (adjoint) equations.

In the present work, the novel $\underline{\text{D}}$iscrete $\underline{\text{A}}$djoint based on the $\underline{\text{R}}$ecursive $\underline{\text{T}}$echnique (DART) toolbox for gradient calculation, that was developed previously by the authors,[25] is further improved to utilize an iterative technique based on the Newton-Krylov method while still taking advantage of the robust and "recursive" approach that was at the core of the proposed method. The enhanced toolbox uses the matrix-free version of the Generalized Minimal RESidual (GMRES) method[26] where the product of the transposed Jacobian matrix and an arbitrary vector is simply calculated using the adjoint mode of the automatic differentiation instead of forming the high-order full Jacobian matrix. As the GMRES method focuses on solving the exact Jacobian of the adjoint system, which typically happens to be very ill-conditioned, a preconditioner matrix is formed. However, the proposed technique utilizes a direct sparse matrix solver to *exactly* factorize the preconditioner matrix. This technique allows us to simply rely on the first-order Jacobian matrix for the preconditioner which results in an *Approximate* Newton-Krylov (ANK) solution technique. It must be noted that the developed toolbox also uses an effective graph coloring technique that significantly reduces the number of adjoint tape evaluations for evaluating the preconditioner matrix. Additionally, the forward mode of AD is utilized to calculate the partial derivatives that determine the sensitivity of the residual vector with respect to the design variables. Ultimately, the proposed technique uses the calculated adjoint field to evaluate the total derivative of the objective function with respect to the set of design variables to be used for gradient-based aerodynamic shape optimization. The robustness of the proposed approach is studied through a few sensitivity analysis and design optimization test cases involving airfoils and wings in the transonic flow regime.

II. Governing Equations

This section introduces the governing equations for the primal flow CFD solver along with the numerical design optimization problem. Additionally, adjoint flow equations based on the discrete adjoint approach are presented and the conventional automatic differentiation approach using the FDOT toolbox is described.

A. Primal Flow Equations

To motivate the development of the design optimization framework, the three-dimensional Unsteady Reynolds-Averaged Navier-Stokes (URANS) equations augmented with the Spalart-Allmaras turbulence model[27] are considered. These conservation laws can be written in the differential form as

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

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

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

Here, $\tilde{\nu}$ is the viscosity-like working variable of the Spalart-Allmaras turbulence model. Additionally, the source vector, $\mathbf{S}$, has all zero terms except for the last equation where the source term is described by the Spalart-Allmaras turbulence model.[27] The pressure, p, and the total enthalpy, H, are defined in terms of the conservative variables as

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

Moreover, the viscous fluxes in the 3D URANS equations depend on the gradients of the flow velocities, specific enthalpy and the working variable of the turbulence model. The governing equations given in Eq. (1) are discretized using a finite volume method with median-dual, vertex-based control volume approach.[2,28,29] Therefore, the semi-discretized form of the governing equations can be written as

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

where $\mathcal{V}$ is the control volume, and $\mathbf{R}$ is the numerical residual that represents the discretization of the spatial terms that include both the convective fluxes and viscous fluxes as well as the turbulence model source term. Here, the fluxes are defined at the midpoint of an edge. Therefore, the numerical solver loops over all edges to calculate these fluxes, which then get integrated to evaluate the residual at the center of each control volume (defined at grid nodes).

The convective terms are discretized using an upwind-biased scheme based on Roe-fluxes.[30] Additionally, limiter functions of Barth and Jespersen[31] and Venkatakrishnan[32] are used for the convective fluxes in the case of second-order solution reconstruction at the face center. Additionally, a second-order central averaging scheme is used for the calculation of the viscous fluxes. The gradients of the flow variables are calculated at the grid nodes using the Green-Gauss method. Furthermore, the solver is parallelized using the message passing interface (MPI) tools with a non-overlapping domain decomposition,[33] and METIS software package[34] is used for partitioning the computational domain. It must be noted that for the steady cases studied in this work, the time-derivative term in Eq. (2) is replaced with a "pseudo-time" derivative in order to march the governing equations to steady-state. The current primal (CFD) solver uses either an explicit multi-stage RK scheme or a semi-implicit Lower/Upper Symmetric Gauss Seidel (LU-SGS) technique[35] for this purpose.

B. Optimization Problem and Adjoint Flow Equations

Before presenting the adjoint flow equations, let us first consider the minimization problem for an objective function $I\left(\mathbf{x}, \mathbf{Q}\left(\mathbf{x}\right)\right)$, defined as

$$\begin{gathered}\min_{\mathbf{x}} I(\mathbf{x}, \mathbf{Q}(\mathbf{x})) \\ \text{subject to } \mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x})) = 0\end{gathered} \tag{3}$$

where $\mathbf{x}$ is the vector of design variables and, as described earlier, $\mathbf{Q}$ is the vector of flow solutions (conservation variables) with $\mathbf{R}$ representing the vector of flow residuals for the primal governing equations. As an example, in a typical aerodynamic design optimization problem, the objective function can be defined as an unconstrained or a lift-constrained drag coefficient. Additionally, the vector of design variables may include the parameters or control points of a shape parameterization technique, i.e., $\mathbf{x}_{\text{shape}} = f(\mathbf{x})$, to promote a smooth search in the design space. Here, $\mathbf{x}_{\text{shape}}$ can be regarded as the vector of coordinates for the grid points that define the geometry of an airfoil, a wing, or even a wing-body-pylon-nacelle (WBPN) configuration.

The optimization problem defined here aims at identifying the "optimal" design, $\mathbf{x}_{\text{optimal}}$, that minimizes the objective function subject to a "fully converged" or *feasible*, primal CFD solution, i.e., $\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x})) = 0$. Using the method of Lagrange multipliers, the minimization problem given in Eq. (3) can be reformulated as a functional such that

$$\mathcal{L}(\mathbf{x}, \mathbf{Q}, \boldsymbol{\psi}) = I(\mathbf{x}, \mathbf{Q}(\mathbf{x})) + \boldsymbol{\psi}^T \mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x})) \tag{4}$$

where $\boldsymbol{\psi}$ is the vector of adjoint solutions. The next step is to minimize the Lagrangian functional, $\mathcal{L}(\mathbf{x}, \mathbf{Q}, \boldsymbol{\psi})$, which can be accomplished using the Karush-Kuhn-Tucker (KKT) optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\psi}} = \mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x})) = 0 \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} + \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} = 0 \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Q}} = \frac{\partial I}{\partial \mathbf{Q}} + \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} = 0 \tag{7}$$

Here, the first KKT condition is the original primal governing equations presented earlier in Eq. (3). Additionally, the second KKT condition is nothing but the total derivative of the original objective function with respect to the vector of design variables. This can be proven by writing the total derivative or sensitivity as

$$\frac{dI}{d\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} + \frac{\partial I}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \mathbf{x}} \tag{8}$$

It must be noted that the last term in Eq. (8) is very expensive to calculate. In the discrete adjoint approach, the general assumption is that the governing equations for the primal flow are satisfied, which would require converging the primal CFD solver to machine precision. Therefore, based on the assumption of the converged primal solution, i.e., $\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x})) = 0$, one can show that

$$\mathbf{R} = 0 \quad \rightarrow \quad \frac{d\mathbf{R}}{d\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \mathbf{x}} = 0 \quad \rightarrow \quad \frac{\partial \mathbf{Q}}{\partial \mathbf{x}} = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{9}$$

which is the cornerstone of the discrete adjoint approach. Here, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, is the Jacobian of the primal solver. By rearranging and inserting Eq. (9) into Eq. (8), we can rewrite the total derivative as

$$\frac{dI}{d\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} \underbrace{-\frac{\partial I}{\partial \mathbf{Q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^{-1}}_{\Lambda} \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{10}$$

It can be easily shown that the (Λ) term in Eq. (10) is identical to the transposed adjoint solution vector, $\boldsymbol{\psi}^T$, in Eqs. (4) and [(5)-(7)], such that

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Q}} = \frac{\partial I}{\partial \mathbf{Q}} + \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} = 0 \quad \rightarrow \quad \boldsymbol{\psi}^T = -\frac{\partial I}{\partial \mathbf{Q}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^{-1} \tag{11}$$

Typically, the inverse of the Jacobian matrix, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, is not found explicitly. Instead, we solve the corresponding linear system with the appropriate right-hand-side vector by transposing the Jacobian which yields the *adjoint equations*

$$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T \boldsymbol{\psi} = -\frac{\partial I}{\partial \mathbf{Q}}^T \tag{12}$$

The above equation is also known as the "adjoint" flow equation which is the focus of the present work. Next, the calculated adjoint vector is substituted into Eq. (10) to compute the total derivative

$$\frac{dI}{d\mathbf{x}} = \frac{\partial I}{\partial \mathbf{x}} + \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{13}$$

A very important point to make here is that the design variables $\mathbf{x}$ do no appear in Eq. (12) and the linear system of equations **only needs to be solved once** for each quantity of interest or objective function (I). This means that the computational cost of the adjoint method is only proportional to the number of objective functions and is independent of the number of design variables. This subtle but very important aspect of the adjoint approach makes it advantageous for large-scale aerodynamic design optimization problems which typically involve very few quantities of interest but can potentially consider a very large number of design variables.

C. Automatic Differentiation using the FDOT Toolbox

As described earlier, the aerodynamic design optimization problem described in Eq. (3) requires the calculation of the total derivative, $dI/d\mathbf{x}$. Therefore, the main goal here is to efficiently and accurately evaluate all the partial derivatives in Eqs. (8)-(13) in order to find the necessary gradient information for the design optimization problem. The FDOT toolbox developed by Djeddi and Ekici[20] utilizes the concept of discrete adjoint sensitivity analysis and the object-oriented programming (OOP) capabilities of the modern Fortran programming language to evaluate the gradient information. By defining a new derived type for real-typed variables and by overloading all the unary and binary operations and intrinsic functions, the FDOT toolbox can be coupled with any numerical solver to provide the sensitivities (gradients) of the output (objective) function(s) with respect to all design variables. Calculating these gradients is done via an "adjoint evaluation" process whose computational cost is only a small multiple of that of the primal solver.[36,37] It must be noted that in the discrete analysis, the derivatives are propagated in the reverse direction, thus require the complete time history of the primal flow equations to be stored in the memory. Almost all OO/AD tools achieve this by recording the entire expression tree into a derived type class often called the *tape*. This tape is then executed in the reverse order while accumulating the derivatives using the recorded adjoint information. Additionally, the FDOT toolbox utilizes a fixed-point iteration approach, as originally proposed by Christianson,[21,22] as well as a novel expression-template-based approach[24] to efficiently calculate the necessary gradient information.

An important feature of the FDOT toolbox is that it can be used in an almost "black-box" fashion requiring only minimal changes to an already existing primal solver to obtain the adjoint version of that solver. One of the modifications that is necessary is the declaration of the flow solution variables, $\mathbf{Q}$, in the FDOT toolbox while also marking the start and end of the iterative part using a "checkpointing" function. Details of the FDOT toolbox and the novel techniques used for automatically differentiating a CFD solver based on the discrete adjoint approach are presented in Refs.[20,24] It must be noted that in the original work of Djeddi and Ekici,[20] the actual adjoint flow field is not explicitly calculated since the FDOT toolbox is used for automatically differentiating the linearized iterative flow solver. In the present work, however, we focus on a novel and robust technique for forming and solving the adjoint system (Eq. [12]) by efficiently and accurately linearizing the CFD solver while still using the advanced FDOT toolbox for automatically differentiating the primal flow code. Details of this new approach will be described in the following section.

III. DART: Discrete Adjoint based on the Recursive Technique

As discussed earlier, the focus of the present work is the development of a new technique for efficient calculation of the total derivatives required in the gradient-based design optimization problems. In this section, technical details of the new DART approach will be presented.

A. Flow Solver Linearization in the Original Approach (DART ver. 1.0)

In the previous section, the derivation process for the discrete adjoint flow equations was presented. In the original version of the DART toolbox,[25] the numerical procedure used for calculating the total derivatives consisted of four major steps. First, the Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$, and the flow residual sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$, matrices need to be formed. Afterwards, the partial derivatives of the objective function with respect to the flow solution, $\partial I/\partial\mathbf{Q}$, as well as the design variables, $\partial I/\partial\mathbf{x}$, are calculated. Next, the linear system of equations (12) is solved to obtain the adjoint solution vector $\boldsymbol{\psi}$. Finally, the adjoint solution is used in Eq. (13) to compute the total derivative. These steps are summarized as:

1. Calculate the partial derivatives required to determine and assemble $\partial\mathbf{R}/\partial\mathbf{Q}$ (reverse mode) and $\partial\mathbf{R}/\partial\mathbf{x}$ (forward mode) using AD procedures 1 and 2, respectively.
2. Calculate the partial derivatives required to determine $\partial I/\partial\mathbf{Q}$ (reverse mode) and $\partial I/\partial\mathbf{x}$ vectors using AD procedure 3.
3. Solve the adjoint equation, $[\partial\mathbf{R}/\partial\mathbf{Q}]^T\boldsymbol{\psi} = -[\partial I/\partial\mathbf{Q}]^T$, to determine $\boldsymbol{\psi}$.
4. Calculate the total derivative, $dI/d\mathbf{x} = \partial I/\partial\mathbf{x} + \boldsymbol{\psi}^T[\partial\mathbf{R}/\partial\mathbf{x}]$.

As can be seen, the steps described above require the calculation of several partial derivatives. The goal here is to use an automatic differentiation approach to calculate these derivatives with high accuracy and in a quick manner. Therefore, FDOT is utilized for algorithmic differentiation of the flow solver. As discussed before, this toolbox enables OO/AD by recording the expression tree into a tape (also known as an operation stack or *OP-Stack*). Next, a specific variable will be used as the "dependent" quantity which would require *seeding* its adjoint value as one (1.0) while initializing all the other adjoint values to zero. Finally, the recorded tape is rewound for adjoint evaluation to obtain the sensitivities or gradients of the *seeded* quantity of interest with respect to all intermediate and independent variables.

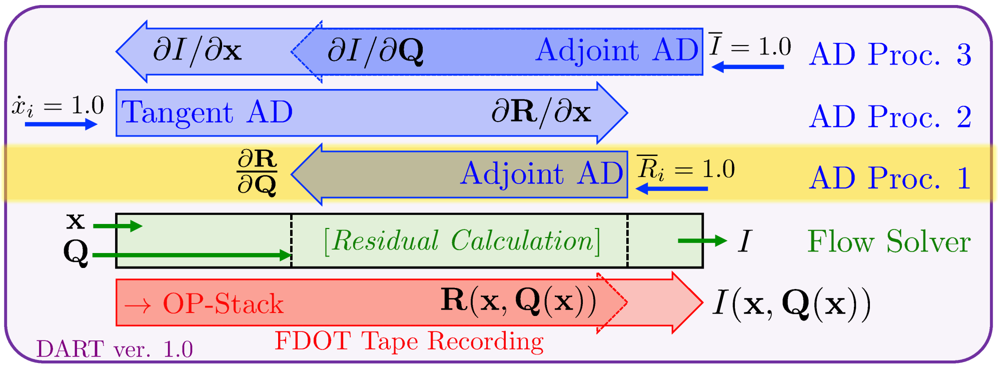

$\partial I/\partial\mathbf{x}$
$\partial I/\partial\mathbf{Q}$
Adjoint AD
$\bar{I} = 1.0$
AD Proc. 3
$\dot{x}_i = 1.0$
Tangent AD
$\partial\mathbf{R}/\partial\mathbf{x}$
AD Proc. 2
$\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}$
Adjoint AD
$\bar{R}_i = 1.0$
AD Proc. 1
$\mathbf{x}$
$\mathbf{Q}$
[Residual Calculation]
I
Flow Solver
→ OP-Stack
$\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$
$I(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$
DART ver. 1.0
FDOT Tape Recording

Figure 1. Linearization of the primal code in the original DART toolbox (ver. 1.0) with the FDOT toolbox recording the expression tree into the OP-Stack. Note the adjoint and tangent modes of automatic differentiation that are used for calculating the necessary partial derivatives for the DART approach.

The linearization process for the CFD solver involves running a slightly "modified" version of the primal code as shown in Fig. (1) for the original version of the DART toolbox.[25] It must be noted that in previous works involving the use of the FDOT toolbox for automatic differentiation and sensitivity analysis, the fixed-point iteration procedure was followed while the expression tree was recorded with the beginning and end of the iterative section marked via a "checkpointing" function.[20,24] For the DART toolbox, on the other hand, the adjoint code includes a very simplified version of the numerical procedure where only the total residual of the CFD solver is automatically differentiated. As can be seen, the design variables as well as the fully-converged flow solution are used to ultimately determine the residual vector, $\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$, and the objective function, $I(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$, during the program execution while the expression tree is being recorded into the OP-Stack.

As discussed before and as depicted in Fig. (1), the execution of the adjoint code involves the calculation of the residual vector and the objective function in addition to the OP-Stack being recorded in the memory. With the recorded OP-Stack, it is possible to efficiently and accurately determine all the partial derivatives involved in calculating the main ingredients of the proposed approach which involve $\partial\mathbf{R}/\partial\mathbf{Q}$ and $\partial\mathbf{R}/\partial\mathbf{x}$ matrices as well as the $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$ vectors. It is also worth noting that the most important step in the DART toolbox is the calculation of the non-zero entries of the sparse Jacobian matrix, $\partial\mathbf{R}/\partial\mathbf{Q}$, which is done via the first AD procedure (as highlighted in Fig. [1]). While the original DART toolbox proved to be a very robust and efficient tool for discrete adjoint sensitivity analysis, the cost of forming the flow Jacobian matrix, $\partial\mathbf{R}/\partial\mathbf{Q}$, as well as the computational overhead for solving the adjoint equations [Eq. (12)] using a "direct" sparse solver such as MUMPS[38,39] can be significantly high, especially for large-scale and complex flow problems. Therefore, in the present work, the original DART toolbox is further improved by utilizing an iterative solution technique which will be described next.

B. Approximate Newton-Krylov Solution Approach (DART ver. 2.0)

Unlike the original version of the toolbox, the improved version utilizes an iterative approach based on the Newton-Krylov method to solve the adjoint equations (Eq. [12]). In this regard, the matrix-free version of the GMRES[26] method is used which relies on an adjoint mode of the automatic differentiation to efficiently and accurately calculate the matrix-vector products without the need for the exact high-order flow Jacobian matrix to be formed.[25] On the other hand, the Newton-Krylov methods require a preconditioner matrix whenever the flow Jacobian matrix is ill-conditioned, which happens to be the case for the RANS equations (Eq. [1]) that are solved for aerodynamics simulations. As a result, while this approach is called matrix-free or Jacobian-free GMRES, a preconditioner matrix needs to be explicitly formed and stored. Additionally, in the present work, we rely on the first-order flow Jacobian to be used as a preconditioner, which needs to be factorized using a direct sparse linear solver. The combination of the first-order flow Jacobian used for the preconditioner matrix and the second-order matrix-free Jacobian used at the core of the GMRES method leads to an "Approximate Newton-Krylov" (ANK) approach.[40] Therefore, the right-preconditioned adjoint system solved in the proposed DART toolbox reads

$$\left(\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}^T\left[\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}_{\mathrm{PC}}^T\right]^{-1}\right)\left(\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}_{\mathrm{PC}}^T\boldsymbol{\psi}\right) = -\frac{\partial I}{\partial\mathbf{Q}}^T \tag{14}$$

where $\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}_{\mathrm{PC}}$ is the preconditioner matrix based on the first-order flow Jacobian. While a *left* preconditioner can also be used in a very similar fashion, numerical results have shown a slightly better performance achieved with the *right* preconditioning. This is mainly due to the fact that the right-preconditioned GMRES aims at minimizing the original residual vector for the linear system while the left-preconditined GMRES tries to minimize the "preconditioned" residual. This subtle but important difference can reportedly lead to significantly different convergence behaviors, and in some cases, stability issues[40,41] for the GMRES method. Therefore, for all cases presented in this work, a right-preconditioned GMRES method is utilized.

The linearization process used in this work is shown in Fig. (2). It is seen that the method involves two separate steps where in the first one the preconditioner matrix is calculated and formed while the second step is very similar to the one followed in the original DART toolbox. However, in the current version, instead of calculating and forming the second-order flow Jacobian matrix (as was done in the earlier work[25]), the Jacobian-free GMRES method is used, requiring only the calculation of a matrix-vector product. This product involves the *transposed* flow Jacobian and an arbitrary vector (as will be described later) and the result of this multiplication is obtained using a **single** adjoint automatic differentiation. The steps required for the improved version of this toolbox are summarized as follows:

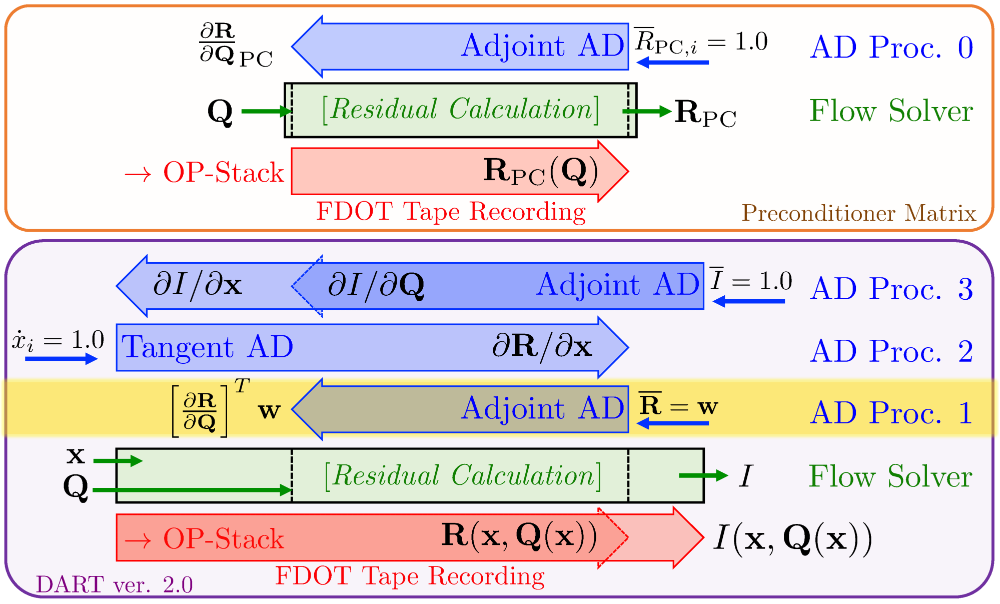

$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}_{\mathrm{PC}}$
Adjoint AD
$\overline{R}_{\mathrm{PC},i} = 1.0$
AD Proc. 0
$\mathbf{Q}$
[Residual Calculation]
$\mathbf{R}_{\mathrm{PC}}$
Flow Solver
→ OP-Stack
$\mathbf{R}_{\mathrm{PC}}(\mathbf{Q})$
FDOT Tape Recording
Preconditioner Matrix
$\partial I/\partial \mathbf{x}$
$\partial I/\partial \mathbf{Q}$
Adjoint AD
$\overline{I} = 1.0$
AD Proc. 3
$\dot{x}_i = 1.0$
Tangent AD
$\partial \mathbf{R}/\partial \mathbf{x}$
AD Proc. 2
$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \mathbf{w}$
Adjoint AD
$\overline{\mathbf{R}} = \mathbf{w}$
AD Proc. 1
$\mathbf{x}$
$\mathbf{Q}$
[Residual Calculation]
I
Flow Solver
→ OP-Stack
$\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$
$I(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$
DART ver. 2.0
FDOT Tape Recording

Figure 2. Linearization of the primal code in the newly improved DART toolbox (ver. 2.0) with the FDOT toolbox recording the expression trees into the OP-Stack for both the first-order and second-order flow residuals.

1. Calculate the first-order flow Jacobian, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}_{\mathrm{PC}}$ (using AD procedure 0)
2. Calculate the product of the flow Jacobian transposed, $[\partial \mathbf{R}/\partial \mathbf{Q}]^T$, and an arbitrary vector, $\mathbf{w}$, using the adjoint mode of AD (procedure 1 in Fig. [2]).
3. Calculate the $\partial \mathbf{R}/\partial \mathbf{x}$ matrix using the tangent mode of AD (procedure 2 in Fig. [2])
4. Calculate the partial derivatives required to determine $\partial I/\partial \mathbf{Q}$ and $\partial I/\partial \mathbf{x}$ vectors (using AD procedure 3)
5. Solve the preconditioned adjoint equation (Eq. [14]) using the **matrix-free GMRES** method to determine $\boldsymbol{\psi}$
6. Calculate the total derivative, $dI/d\mathbf{x} = \partial I/\partial \mathbf{x} + \boldsymbol{\psi}^T[\partial \mathbf{R}/\partial \mathbf{x}]$

As can be seen, there are two main differences between the original DART toolbox[25] and the newly improved version developed in this work. These include: (1) the calculation of a preconditioner matrix based on the first-order flow Jacobian, and (2) solving the adjoint equations (Eq. [14]) using the preconditioned matrix-free GMRES method. In the following sections, the numerical procedures as well as the programming considerations for both of these steps are described and explained in details.

C. Preconditioner Matrix, $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}_{\mathbf{PC}}\right]^T$, via an Adjoint/Reverse AD Approach

The calculation of the high-order flow Jacobian was at the core of the original version of the DART toolbox. However, as explained earlier, the newly improved DART toolbox developed in the present work focuses on the first-order flow Jacobian which will be formed, stored, and factorized in order to obtain the preconditioner matrix for the GMRES solution technique. It must be noted that the first-order flow Jacobian was chosen as it has a much smaller bandwidth with significantly fewer non-zero terms compared to the second-order flow Jacobian, which happens to have a higher bandwidth and a lower sparsity. Nevertheless, a naive approach for calculating the partial derivative terms of the flow Jacobian matrix would require $\mathcal{O}(\mathrm{N_{DOF}} \times \mathrm{N_{DOF}})$ calculations where $\mathrm{N_{DOF}}$ is the number of degrees-of-freedom (DOF) or the number of rows in the Jacobian matrix. Therefore, similar to the earlier work of the authors in developing the DART toolbox,[25] the sparsity pattern is taken into account to minimize the number of partial derivative calculations and storage for this highly sparse preconditioner matrix.

The vertex-based, mixed-element, unstructured/hybrid grid approach utilized in UNPAC relies on first-order and second-order stencils. In the simplest form used for the first-order accurate Euler solutions, the stencil only includes direct neighbors or "*distance-1*" nodes. On the other hand, the second-order Roe scheme needs solution reconstruction at the face medians which requires gradient information to be available at all direct neighbors. In UNPAC, a Green-Gauss approach is used for gradient calculation which also includes direct neighbors for each node to determine the gradient vector at that specific node. Therefore, the second-order stencil would ultimately involve neighbors-of-neighbors or "*distance-2*" nodes. It must be noted that the RANS solutions would also rely on the same nodal structure with distance-2 nodes included in the stencil.

1. Graph Coloring

Graph coloring (also known as graph labeling or vertex coloring) is a technique used for assigning labels or colors to specific nodes that meet a certain "distancing" criteria.[42] As an example, "distance-1" coloring leads to a pattern where no two adjacent nodes have the same color. Generally speaking, the adjacency condition can include distance-k neighbors which will result in a distance-k coloring. It must be noted that the goal of the graph coloring approach is to find the "minimum" number of graph colors that will meet the adjacency criteria. As discussed in the previous section, the numerical stencils lead to a known block structure for the highly sparse Jacobian matrix. Additionally, a graph coloring technique can be used to identify independent subsets of grid nodes such that nodes of the same color would not share any nodes in their respective stencils. This technique is particularly important in that it allows the computation of sensitivities for many variables simultaneously where more than one variable can be perturbed (or "seeded") at the same time.

Here, the residual stencil is used to define an adjacency matrix that will be utilized to compute a unique set of colors that covers the entire domain with non-overlapping residual computations. This is done by partitioning all rows of a matrix into different structurally orthogonal subgroups (aka *colors*) such that, in one structurally orthogonal subgroup, no two rows have a nonzero entry in a shared column. As mentioned earlier, graph coloring allows simultaneous perturbation of multiple rows of the Jacobian by using the adjoint mode of AD since no residuals in a given color overlap. Similarly, the column-based graph coloring approach has been used in conjunction with various forward-mode techniques such as finite-difference,[43] complex-step,[44,45] as well as the forward mode of AD.[46,47] Interestingly, the use of graph coloring significantly reduces the number of residual evaluations to a very small number, i.e., N_{color}.

As mentioned earlier, the first-order stencil includes all distance-1 neighbors while the second-order stencil relies on the inclusion of all distance-2 neighbors. However, for the nodes of the same colors to be completely independent (i.e., structurally orthogonal) from each other, it is necessary to avoid having any overlap between all the nodes in their respective stencil. This means that the first-order stencil would require a distance-2 graph coloring of the computational grid. Consequently, the second-order stencil would require a distance-4 graph coloring. The graph coloring in general is an NP-hard problem while distance-k coloring with $k > 2$ being an NP-complete problem. In many previous efforts, the use of graph coloring for Jacobian calculations has been limited mostly to structured grid solvers[46,47] where the coloring scheme relies heavily on the structured and repeating pattern of the node adjacency and connectivity matrix. For unstructured or hybrid grid cases, however, the only remaining option is to use a "greedy" graph coloring technique[45] to determine distance-4 colors. As an example, the graph coloring is applied to an O-typed structured grid with 4,160 nodes around the NACA 0012 airfoil which results in only 8 colors for the distance-2 graph coloring but 87 colors for the distance-4 graph coloring technique which are both much smaller than the number of nodes, 4,160.

A very important feature of the newly proposed or improved version of the DART toolbox is the fact that only the first-order flow Jacobian is explicitly formed in the calculation of the preconditioner matrix. This means that we can easily rely on distance-2 graph coloring schemes which happens to provide the most optimal graph coloring of the computational mesh that can potentially lead to the least number of colors, thus significantly improving the computational efficiency of the Jacobian calculations. As described for the structured mesh used with the NACA 0012 airfoil, the distance-2 graph coloring can result in a more than ten-fold reduction in the number of graph colors compared to the distance-4 approach. Therefore, it is possible to calculate the preconditioner matrix at a fraction of the cost of calculating the second-order flow Jacobian as was previously done in the original version of the DART toolbox. It is worth noting that the number of distance-2 graph colors is typically on the order of $\mathcal{O}(10)$ to $\mathcal{O}(100)$ for the two- and three-dimensional test cases studied with the new DART approach.

2. Adjoint/Reverse AD

As discussed before, naively evaluating the non-zero terms of the sparse Jacobian matrix using the adjoint mode of AD would require an extensive number of tape reversal and adjoint evaluations. Therefore, with the aid of graph coloring, it is possible to significantly reduce this number, thus improving the efficiency of the gradient calculations. This idea is shown in Fig. (3) where rows of the same color are "seeded" at the same time to initiate the tape reversal. In the reverse mode of AD, the adjoints of the residuals belonging to nodes (i) of the same color (c) are set to 1.0 and the recorded OP-Stack is reversed. During the tape reversal, the adjoint values are propagated backward from the residuals, $\mathbf{R}_{c,i}$, toward the flow variables, $\mathbf{Q}_{j_i}$, where j_i refers to nodes j belonging to the compact stencil at node i. Finally, the partial derivatives, $\partial\mathbf{R}_{c,i}/\partial\mathbf{Q}_{j_i}$, are evaluated and the rows of the Jacobian matrix are filled according to the pre-determined sparsity pattern.

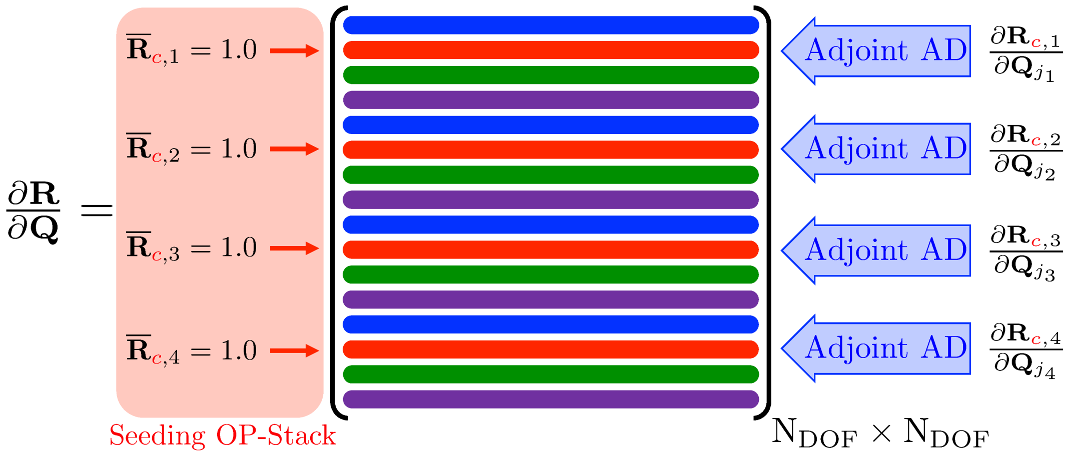

$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} =$
$\overline{\mathbf{R}}_{c,1} = 1.0$
$\overline{\mathbf{R}}_{c,2} = 1.0$
$\overline{\mathbf{R}}_{c,3} = 1.0$
$\overline{\mathbf{R}}_{c,4} = 1.0$
Seeding OP-Stack
Adjoint AD
$\frac{\partial \mathbf{R}_{c,1}}{\partial \mathbf{Q}_{j_1}}$
Adjoint AD
$\frac{\partial \mathbf{R}_{c,2}}{\partial \mathbf{Q}_{j_2}}$
Adjoint AD
$\frac{\partial \mathbf{R}_{c,3}}{\partial \mathbf{Q}_{j_3}}$
Adjoint AD
$\frac{\partial \mathbf{R}_{c,4}}{\partial \mathbf{Q}_{j_4}}$
$\mathrm{N_{DOF}} \times \mathrm{N_{DOF}}$

Figure 3. The adjoint mode of AD used in conjunction with a graph coloring approach for efficient evaluation of the Jacobian matrix. For this example, it is assumed that there are only 4 colors in the partitioned matrix with rows of the same color being seeded and evaluated simultaneously.

It is important to note that at the end of tape reversal and adjoint evaluation process for each color, the adjoint information is reset before seeding a new set of residuals for the next graph color. Additionally, it must be noted that the graph colors only relate to each node in the computational grid with a total of $\mathrm{N_{node}}$ while $\mathrm{N_{eqn}}$ conservation variables are defined at each node to determine flow solutions. Therefore, the total number of degrees-of-freedom would be defined as $\mathrm{N_{DOF}} = \mathrm{N_{node}} \times \mathrm{N_{eqn}}$. However, the total number of tape reversal and adjoint evaluations, $\mathrm{N_{AD}}$, required for calculating the sparse Jacobian matrix would be

$$\mathrm{N_{AD}} = \mathrm{N_{color}} \times \mathrm{N_{eqn}} \quad \ll \quad \mathrm{N_{DOF}} = \mathrm{N_{node}} \times \mathrm{N_{eqn}} \tag{15}$$

Apparently, the use of graph coloring can lead to a very efficient approach for calculating the Jacobian matrix based on the adjoint mode of AD. Note that the first-order Jacobian matrix described in this section must be first **transposed** before being **factorized** so that it can be used as a preconditioner for the GMRES algorithm. In this work, the MUltifrontal Massively Parallel Solver (MUMPS) package[38, 39] is used for this purpose and the numerical precodure will be described in the following section.

D. Matrix-Free Preconditioned GMRES Solution Technique

After calculating the preconditioner matrix, the approximate Newton-Krylov approach is used to solve the right-preconditioned adjoint equations (Eq. [14]). As discussed earlier, the matrix-free GMRES method is utilized for this purpose. For a typical linear system, $\mathbf{Ax} = \mathbf{b}$, the right-preconditioned GMRES method aims at solving

$$\left(\mathbf{AP}^{-1}\right)(\mathbf{Px}) = \mathbf{b} \tag{16}$$

where $\mathbf{P}$ is the preconditioner matrix (right-preconditioning). The procedure utilized for the restarted GMRES method, also known as the GMRES(k) method,[48] is described in Algorithm 1 for the case with m vectors in the Krylov subspace[26] defined as

$$K_m = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\} \tag{17}$$

where $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ and $\mathbf{x}_0$ is an initial guess for the solution vector.

Algorithm 1: Algorithm for the restarted GMRES(k) method.

```
Result: Solution vector, $\mathbf{x}$
Size of the Krylov subspace: $m$;
Choose an "initial" solution vector, $\mathbf{x}_0$;
while not converged do
    calculate the residual vector, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$;
    calculate the first Krylov vector, $\mathbf{v}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|}$;

    Step 1: Arnoldi Algorithm:

    for $j \leftarrow 1$ to $m$ do
        $\mathbf{w}_j = \mathbf{A} \cdot \mathbf{P}^{-1} \cdot \mathbf{v}_j$ ;
        $h_{j+1,1} = \|\mathbf{w}_j\|$ ;
        $\mathbf{v}_{j+1} = \frac{\mathbf{w}_j}{h_{j+1,1}}$ ;
    end
    Define $\mathbf{V}_m = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$
    Set the $(m+1) \times m$ upper Hessenberg matrix, $\overline{\mathbf{H}}_m = [h_{i,j}]$

    Step 2: Least-Squares Problem:

    Solve for the optimal weighting coefficients, $\mathbf{y}_m$, in
        $y_m = \min_y \left\|\boldsymbol{\beta}\mathbf{e} - \overline{\mathbf{H}}_m \mathbf{y}_m\right\|$
    where
        $e_i$: $i$-th vector in the standard basis of $\mathbb{R}^{n+1}$
        $\beta_i = \|\mathbf{b} - \mathbf{A}\mathbf{x}_i\|$

    Step 3: Update Solution:

    Calculate the new solution via
        $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{P}^{-1} \cdot \mathbf{V}_m \cdot \mathbf{y}_m$

    if converged then
        stop
    else
        set $\mathbf{x}_0 = \mathbf{x}_m$
        restart GMRES algorithm with the new initial vector, $\mathbf{x}_0$
    end
end
```

The GMRES solution algorithm involves the calculation of the $\mathbf{P}^{-1}\mathbf{v}$ products during the Arnoldi algorithm as well as in solution update procedure. This requires the inversion and, subsequently, the multiplication of the preconditioner matrix by an arbitrary vector, v. Moreover, the GMRES algorithm requires the calculation of the $\mathbf{A}\mathbf{w}$ products with $\mathbf{w}$ being an arbitrary vector. In the present work, two extremely efficient approaches are utilized to calculate these terms which are described next.

1. *Calculating* $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \mathbf{w}$ *products*

As explained in the previous section, the GMRES algorithm requires the calculation of the $\mathbf{Aw}$ products where $\mathbf{A}$ is the linear system matrix and $\mathbf{w}$ is an arbitrary vector. In the case of the adjoint equations (Eq. [12]), the system matrix is defined as the transpose of the flow Jacobian, i.e., $\mathbf{A} = \left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T$. As was shown in the original DART approach,[25] explicitly forming the flow Jacobian matrix can become prohibitively costly for very large-scale problems even with the aid of efficient approaches that rely on a graph coloring technique. This can be circumvented as the GMRES method does not necessarily require the calculation of the flow Jacobian itself and it only requires the product of the flow Jacobian and an arbitrary vector. For the GMRES method used for solving the primal flow equations, the product of the linear system matrix and the arbitrary vector can be easily calculated using a finite-difference-typed approximation[49, 50] such that

$$\mathbf{Ax} = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\mathbf{w} \approx \frac{\mathbf{R}\left(\mathbf{Q} + \epsilon \mathbf{w}\right) - \mathbf{R}\left(\mathbf{Q}\right)}{\epsilon} \tag{18}$$

where ϵ is a carefully-defined perturbation parameter.[40, 49] As can be seen, this approximation simply requires *one* additional residual calculations with the "perturbed" flow solution vector. However, in the case of the adjoint equations, the linear system matrix is the **transpose** of the flow Jacobian matrix which does not allow us to use a similar approach for approximating the matrix-vector product.

Therefore, in order to perform the matrix-free GMRES iterations for solving the adjoint equations, we rely on the adjoint mode of the automatic differentiation as was originally proposed by Kenway et al.[47] To motivate this approach, let us examine the adjoint (or reverse) mode of AD applied to an arbitrary vector function $\mathbf{y} = \mathbf{y}\left(\mathbf{x}\right)$ where

$$\overline{\mathbf{x}} = \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right]^T \overline{\mathbf{y}} \quad \Rightarrow \quad \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_N \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_N}{\partial x_1} & \frac{\partial y_N}{\partial x_2} & \cdots & \frac{\partial y_N}{\partial x_N} \end{bmatrix}^T \begin{bmatrix} \overline{y}_1 \\ \overline{y}_2 \\ \vdots \\ \overline{y}_N \end{bmatrix} \tag{19}$$

In the above equation, the adjoint Jacobian matrix includes the entire partial derivative information required for linearizing the $\mathbf{y} = \mathbf{y}\left(\mathbf{x}\right)$ function while the $\overline{\mathbf{y}}$ is known as the "seed" or "input" vector. In a standard adjoint-based automatic differentiation, the desired derivatives for the n-th output function are calculated by setting the input seed vector such that

$$\overline{\mathbf{y}} = \left[\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n, \ldots, \overline{y}_N\right]^T = \left[0, 0, \ldots, 1, \ldots, 0\right]^T. \tag{20}$$

Using the exact same idea, we can show that the matrix-vector product necessary for the GMRES algorithm in solving the adjoint equations can be evaluated by simply seeding the the adjoint tape evaluation process with the arbitrary vector $\mathbf{w}$, i.e.

$$\overline{\mathbf{x}} = \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right]^T \overline{\mathbf{y}} \quad : \quad \overline{\mathbf{Q}} = \left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \overline{\mathbf{R}} \quad \Rightarrow \quad \overline{\mathbf{R}} = \mathbf{w} \tag{21}$$

Using this approach, the desired matrix-vector product, $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \mathbf{w}$, can be calculated very efficiently and accurately (non-approximative) using a single tape evaluation based on the adjoint mode of AD.

2. *Calculating* $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T_{PC}\right]^{-1} \mathbf{v}$ *products*

In addition to the matrix-vector product involving the transposed flow Jacobian that was described in the previous section, the right-preconditioned matrix-free GMRES algorithm also requires the $\mathbf{P}^{-1}\mathbf{v}$ products to be calculated. As discussed earlier, in the case of the adjoint equations (Eq. [14]), the preconditioner matrix, $\mathbf{P}$, is defined as the transpose of the first-order flow Jacobian. Therefore, it is necessary to form, transpose, and invert the preconditioner matrix followed by a matrix-vector multiplication to calculate the $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T_{\text{PC}}\right]^{-1} \mathbf{v}$ products. In the present work, we rely on a robust procedure for calculating this product using a "direct" linear solver called MUMPS[38, 39] that is capable of efficiently calculating $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ for a typical linear system, $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is a sparse and non-symmetric matrix. Additionally, the MUMPS solver can be easily configured to solve for the $\mathbf{A}^T\mathbf{x} = \mathbf{b}$ system instead. Therefore, the following steps are taken to efficiently and accurately (down to machine precision) calculate the $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T_{\text{PC}}\right]^{-1} \mathbf{v}$ products as required during the preconditioned GMRES algorithm:

1. Record the expression tree into the operation stack (OP-Stack) only for the residual calculation section of the flow solver based on a "first-order" scheme (see Fig. [2]).
2. Use a distance-2 graph coloring approach to efficiently calculate the non-zero elements of the $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}_{\text{PC}}$ matrix.
3. Run the MUMPS package to only **pre-process** and **factorize** the $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T_{\text{PC}}\right]$ matrix. Note that the MUMPS solver is configured to solve for the "transposed" system.
4. For any arbitrary vector $\mathbf{v}$, set the RHS vector of the linear system to $\mathbf{b} = \mathbf{v}$ and run the MUMPS solver in the **solve** mode to calculate the $\left(\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}^T_{\text{PC}}\right]^{-1} \mathbf{v}\right)$ product

It is very important to note that the proposed approach only requires us to factorize the preconditioner matrix once. This is very important for the linear adjoint equations as the preconditioner matrix does not change during the GMRES solution procedure. Therefore, the factorization procedure is only executed once for the preconditioner matrix with the factorized system stored in the memory for any subsequent calls to the MUMPS solver. As can be clearly seen, the formation and factorization of the preconditioner matrix will lead to an added memory footprint for the DART toolbox. However, as will be shown later, the use of the proposed ANK solution technique can greatly enhance the computational efficiency of the DART toolbox, especially for large-scale problems.

E. Flow Residual Sensitivity Matrix, $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$, via a Tangent/Forward AD Approach

As described in the previous section, the adjoint mode of the AD is used to evaluate the preconditioner matrix (based on the first-order flow Jacobian) where rows of the same color are evaluated simultaneously. Since the Jacobian $\partial \mathbf{R}/\partial \mathbf{Q}$ is a square matrix, it does not necessarily matter whether an adjoint or a tangent mode of AD is used. However, the flow residual sensitivity matrix, $\partial \mathbf{R}/\partial \mathbf{x}$, is non-square whose size is $\text{N}_{\text{DOF}} \times \text{N}_{\text{dv}}$ where N_{dv} refers to the number of design variables used in the optimization problem. Since in most aerodynamic design optimization problems, the number of design variables is several orders of magnitude fewer than the number of degrees-of-freedom, i.e., $\text{N}_{\text{dv}} \ll \text{N}_{\text{DOF}}$, it would be beneficial to use the forward mode of AD to evaluate the $\partial \mathbf{R}/\partial \mathbf{x}$ matrix. The numerical procedure that is followed in the DART toolbox for calculating this matrix will be described next.

The FDOT toolbox developed by Djeddi and Ekici,[20] is designed to use the adjoint mode of AD based on the operator-overloading technique. However, in this work, the capabilities of the FDOT toolbox are enhanced so that the forward mode of AD could also be handled. Generally, the forward mode of OO/AD is performed via a repeated execution of the overloaded computer code without the need for recording the expression tree in the memory. However, in the present work, a novel approach is utilized that can enable the forward-mode AD without any necessary modifications or user interventions to the flow solver code. The proposed DART toolbox uses the already recorded OP-Stack to handle forward derivative propagation. The process starts by "seeding" the OP-Stack for each independent variable, i.e., the design variables, $\mathbf{x}_i$, and evaluating partial derivatives for each unary and binary operation. The tape evaluation process will continue all the way to the end of the OP-Stack which would result in the desired $\partial \mathbf{R}/\partial \mathbf{x}_i$ terms. This process is also shown in Fig. (4). As can be seen, the entire $\partial \mathbf{R}/\partial \mathbf{x}$ matrix can be evaluated by performing N_{dv} tape sweeps in the forward direction based on the tangent mode of AD.

F. Efficient Adjoint-based Sensitivity Analysis using DART

The main objective of the proposed DART approach is to utilize the automatic differentiation capabilities of the FDOT toolbox within the framework of an efficient discrete adjoint-based sensitivity analysis. As a result, the automatically differentiated version of the CFD solver, i.e., **UNPAC_AD**, will be redesigned as

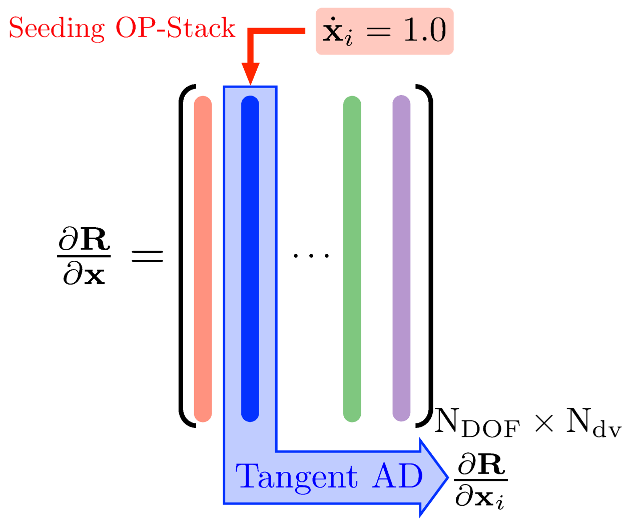

Seeding OP-Stack
$\dot{\mathbf{x}}_i = 1.0$
$\frac{\partial \mathbf{R}}{\partial \mathbf{x}} =$
$\mathrm{N_{DOF}} \times \mathrm{N_{dv}}$
Tangent AD
$\frac{\partial \mathbf{R}}{\partial \mathbf{x}_i}$

Figure 4. The tangent mode of AD used for efficient evaluation of the $\partial \mathbf{R}/\partial \mathbf{x}$ matrix.

duplicate copy of the primal code coupled with the FDOT module with some minor modifications which will be described in the following:

- **Pre-processing stage:** This stage involves reading the computational grid, performing grid metric calculations, as well as reading in the converged primal flow solution, and will be identical to the primal version of the code.
- **Residual calculation stage:** Unlike the primal CFD solver, which involves an iterative update procedure, **UNPAC_AD** code will only include a subroutine that calculates the total residual vector by performing the spatial discretization via a method of lines. As a matter of fact, this subroutine is already used as part of the iterative update procedure that is followed in the primal code. However, the primal code also includes several additional subroutines, e.g., local timestepping, residual smoothing, RK stage updates, etc., which are no longer necessary in the **UNPAC_AD** version of the code.
- **Post-processing stage:** This stage will be identical to the primal version of the code and involves the calculation of the quantities of interest based on the design variables as well as the flow solution.

The entire procedure described above will be automatically differentiated using the overloaded operations that are handled by the FDOT toolbox and the entire expression tree will be recorded into the OP-Stack. Additionally, in the DART approach, the matrix of node adjacency will also be necessary for determining the sparsity pattern as well as coloring the computational graph used in the calculation of the preconditioner matrix. As discussed earlier, the adjoint solution is ultimately obtained using the preconditioned GMRES approach (Algorithm [1]). However, the adjoint equation (14) also relies on the $\partial I/\partial \mathbf{x}$ as its right-hand-side vector. Therefore, the additional steps performed in the DART toolbox that will lead to the calculation of this vector as well as ultimately calculating the total derivatives are described next.

1. *Evaluating Objective-Function-Based Vectors*

Having the non-square $\partial \mathbf{R}/\partial \mathbf{x}$ matrix evaluated, it is now time to calculate the two remaining ingredients for the proposed discrete adjoint approach. Here, we need to calculate the $\partial I/\partial \mathbf{Q}$ and $\partial I/\partial \mathbf{x}$ vectors which will rely on the definition of the objective function or the quantity of interest. It is very important to note that none of the previous calculations relied on the definition of the quantity of interest. Therefore, the matrix calculations described earlier will be done regardless of the choice of objective function.

In recent years, 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. 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.[51] Here, the Quadratic Programming (QP) problem would require the additional gradient information for the equality and inequality constraints with respect to the design variables. In such cases, the proposed DART approach can be utilized as a very efficient tool for sensitivity/gradient calculation since only the process described herein for determining the $\partial I/\partial \mathbf{Q}$ and $\partial I/\partial \mathbf{x}$ vectors would need to be repeated for each quantity of interest. As such, the adjoint mode of AD will be utilized where the objective function or quantity of interest I will be used to "seed" the recorded OP-Stack. After tape reversal and adjoint evaluations, the partial derivatives necessary for evaluating the $\partial I/\partial \mathbf{Q}$ and $\partial I/\partial \mathbf{x}$ vectors will be calculated (AD procedure 3 as shown in Fig. [2]).

2. *Total Derivative Calculation*

As shown earlier, the adjoint Eq. (14) will be solved to obtain the adjoint flow field where only the right-hand-side vector, $\partial I/\partial \mathbf{Q}$, relies on the quantity of interest or the choice of objective function. This means that, for a multi-objective design optimization problem, the adjoint equations will be solved for each objective function by simply substituting the right-hand-side of the linear system. It is important to note that in the earlier version of the DART toolbox where a "direct" approach was utilized for solving the adjoint equations, it is possible to perform the factorization on the $[\partial \mathbf{R}/\partial \mathbf{Q}]^T$ matrix only once and then reusing the factorized transposed Jacobian with different right-hand-side vectors to obtain the corresponding adjoint solutions. However, this will not be possible in the proposed improved version described in this work. That being said, the multi-objective optimization problems are not the focus of the present work.

Ultimately, having the adjoint vector, we will perform a matrix-vector multiplication as well as a final vector addition, according to Eq. (13), to evaluate the final total derivative vector. Once again, this process will be repeated for each quantity of interest (objective function and/or the optimization constraints). Finally, the gradient information will be used in a quadratic programming problem to obtain the new set of design variables. In the present work, the L-BFGS-B[52] and the SLSQP[53] optimizers are used for unconstrained/bound-constrained and PDE-constrained problems, respectively. The design update process will be repeated until the optimality condition (Eq. [6]) has been satisfied based on a preset tolerance.

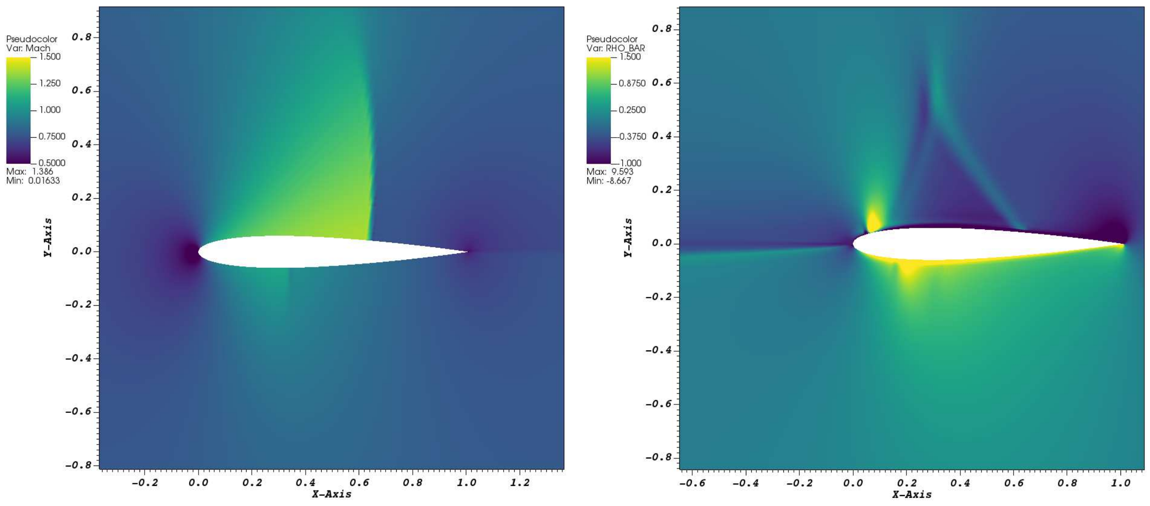
IV. Results

In this section, we present our numerical results obtained using the proposed DART approach. First, the developed toolbox is utilized to perform sensitivity analysis for the NACA0012 airfoil as well as the ONERA M6 wing - both subject to inviscid transonic flows. These two test cases enable us to study the performance of the newly proposed DART toolbox in discrete adjoint-based gradient calculations in addition to conducting a detailed memory footprint and computational effort analysis. Additionally, comparisons are made between the original DART toolbox and the improved version presented in this work that utilizes the approximate Newton-Krylov approach for iteratively solving the adjoint system. Finally, the lift-constrained drag minimization of the RAE 2822 airfoil with turbulent transonic flow is studied.

A. Inviscid Transonic NACA 0012 Airfoil

The first test case presented here involves the discrete adjoint sensitivity analysis for a symmetric NACA0012 airfoil. Here, Euler solutions of the inviscid transonic flow are considered with a free-stream Mach number of 0.8 and an angle of attack of 1.25 degrees. An O-typed structured grid with 256×257 nodes is considered[54] and a Free-Form-Deformation (FFD) box[23] with 20 control points is used for shape parameterization. In order to motivate adjoint solutions, the drag coefficient is considered to be the quantity of interest or the objective function while the y-coordinates of the FFD box control points are used as the design variables. The goal here is to study the performance of the DART approach in calculating the adjoint flow field as well as the total derivative evaluation where both the original and the newly improved versions of the toolbox are utilized.

First, the primal and adjoint flow fields are presented in Fig. (5) which includes the contours of Mach number for the primal solution as well as the drag adjoints of the density. The *flow reversal*, which is the hallmark of the adjoint method, is evident in Fig. 5(b) with the triangular adjoint shock structure clearly visible. This reversed shock formation in the adjoint flow field hints the significant sensitivity of the drag coefficient to the flow field upstream of the shock given the fact that the strong shock forming on the suction side of the NACA 0012 airfoil acts as the main contributor to the total drag coefficient.



(a) Absolute Mach number contours

(b) Adjoint of density, $\overline{\rho}$, contours

Figure 5. Absolute Mach number as well as the adjoint density contour fields obtained from the primal and adjoint solvers for the inviscid transonic NACA 0012 airfoil case. The quantity of interest for this sensitivity analysis is the drag coefficient.

Table 1. Peak memory breakdowns for the sensitivity analysis of the transonic NACA 0012 airfoil using different discrete adjoint tools.

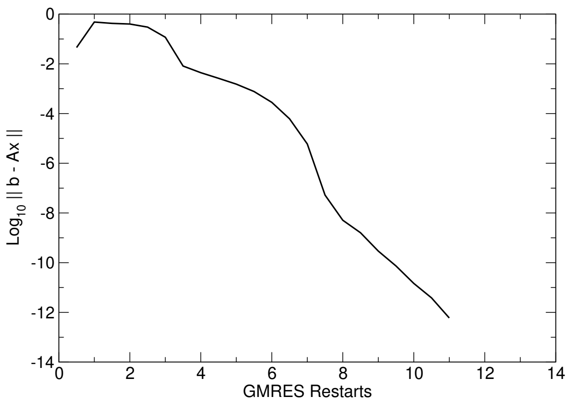
Toolbox	FDOT	DART (original[25])	DART (improved)
Total (GB)	4.61	1.29	1.45
MUMPS (MB)	-	732	95
Normalized	1.0	**0.279**	**0.314**

In the original version of the DART toolbox, the distance-4 greedy graph coloring approach is used to color the computational mesh which results in 95 distinct colors. This means that the number of tape reversal and adjoint evaluations necessary for calculating the flow Jacobian matrix would be only a small fraction of $\mathrm{N_{DOF}}$. As explained earlier, the computational grid for this case has 65,792 nodes with four conservation variables at each node for this inviscid two-dimensional test case. This means that the total number of degrees-of-freedom for this case is 263,168 while the total number of tape reversal and adjoint evaluations would only be $\mathrm{N_{AD}} = 380$ which results in a very significant memory saving of more than 99.8%. In the present work, however, only the first-order Jacobian is formed to obtain the preconditioner matrix even though the adjoint solutions are based on the second-order fluxes of the primal flow solver. As a result, a distance-2 graph coloring is used which leads to only nine colors for the graph of the computational mesh. Additionally, the memory footprint and CPU time breakdowns for the DART approach are presented in Tables (1) and (2) for which the size of the Krylov subspace is set to $m = 10$.

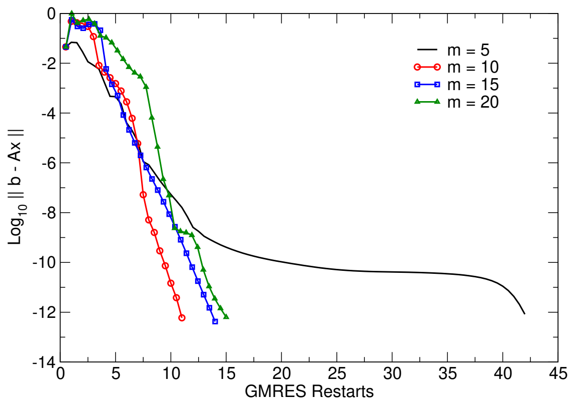
The data shows that both the original and the improved versions of the DART toolbox lead to very small memory footprints due to much shorter expression trees that are recorded into the OP-Stack as was depicted in Figs. (1) and (2). Additionally, the compressed sparse storage format for the Jacobian matrix saves memory by only storing a very small number of non-zero terms. The MUMPS package is also designed to be memory efficient for small to moderately-sized problems and the total time spent for factorization and solution of the linear system is only a small fraction of the total time spent in the primal code. Furthermore, it is important to compare the memory and computational efficiencies of the original and improved versions of the DART toolbox. Tables (1) and (2) show that the ANK-based version requires a higher memory footprint as it needs the formation and storage of the preconditioner matrix. On the other hand, the use of the matrix-free GMRES technique significantly reduces the computational cost of the adjoint solver. Overall,

Table 2. Breakdowns of the wall-clock time for the sensitivity analysis of the transonic NACA 0012 airfoil using different discrete adjoint tools.

Toolbox	**FDOT**		**DART (original[25])**		**DART (improved)**	
Runtime		**unit**		**unit**		**unit**
Total	4.5	(hrs)	**3.88**	(min)	**2.67**	(min)
Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$	-		3.16	(min)	-	-
matrix-free GMRES	-		-	-	1.33	(min)
Preconditioner, $\partial\mathbf{R}/\partial\mathbf{Q}_{\mathrm{PC}}$	-		-	-	52	(s)
Residual Sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$	-		21.5	(s)	21.5	(s)
Vectors, $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$	-		2.1	(s)	2.1	(s)
MUMPS:						
Factorization	-		9.29	(s)	3.12	(s)
Solve	-		2.32	(s)	1.53	(s)



(a) $m = 10$



(b) various numbers of Krylov vectors

Figure 6. Convergence histories of the preconditioned matrix-free GMRES solver for the adjoint equations for different numbers of Krylov vectors included in the subspace: Inviscid transonic flow past NACA 0012 airfoil.

the most expensive part in the original DART toolbox is the formation of the second-order Jacobian matrix based on the distance-4 graph coloring. This requires $\mathrm{N_{AD}} = \mathrm{N_{color}} \times \mathrm{N_{eqn}}$ evaluations of the tape (OP-Stack) for the second-order residual calculation. However, in the matrix-free GMRES approach, this tape only needs to be evaluated $k(m+1)+1$ times, where k is the number of GMRES restarts and m is the size of the Krylov subspace. The GMRES iterations are continued until at least 10 orders of magnitude reduction in the original results has been achieved which, for this inviscid transonic case, can be attained after 11 GMRES restarts with 10 Krylov vectors. Additionally, the convergence history of the GMRES solver is shown in Fig. (6) for this inviscid transonic case involving the NACA 0012 airfoil. As is commonly done in the CFD literature, the effect of the Krylov subspace size is studied next by varying number of Krylov vectors included in the GMRES algorithm. It can be seen that a small number of Krylov vectors can lead to a significantly slower convergence rate or even stability issues while unnecessarily increasing the size of the Krylov vector would only increase the memory footprint with little to no effects on the convergence rate.

The accuracy of the gradient evaluations using the original FDOT toolbox has been studied in previous works.[2,20] However, in order to examine the accuracy of total derivative calculations using the new approach, the sensitivity of the drag coefficient with respect to the free-stream Mach number is considered. As presented in Table 3, the sensitivity values obtained from the original FDOT toolbox[24] and DART toolbox in its original and improved versions are compared to the finite-difference approximations using a second-order central difference scheme. The results clearly demonstrate the accuracy of the gradient evaluations.

Table 3. Comparison of the drag sensitivity calculations w.r.t free-stream Mach number ($\frac{\partial C_D}{\partial M_\infty}$) for the NACA 0012 airfoil case.

Finite-Difference (2nd order)	FDOT Toolbox	DART Toolbox (original[25])	DART Toolbox (improved)
$\mathbf{0.3278671\underline{1019}}$	$\mathbf{0.32786704296927\underline{5}}$	$\mathbf{0.32786704296927\underline{1}}$	$\mathbf{0.327867042969\underline{145}}$

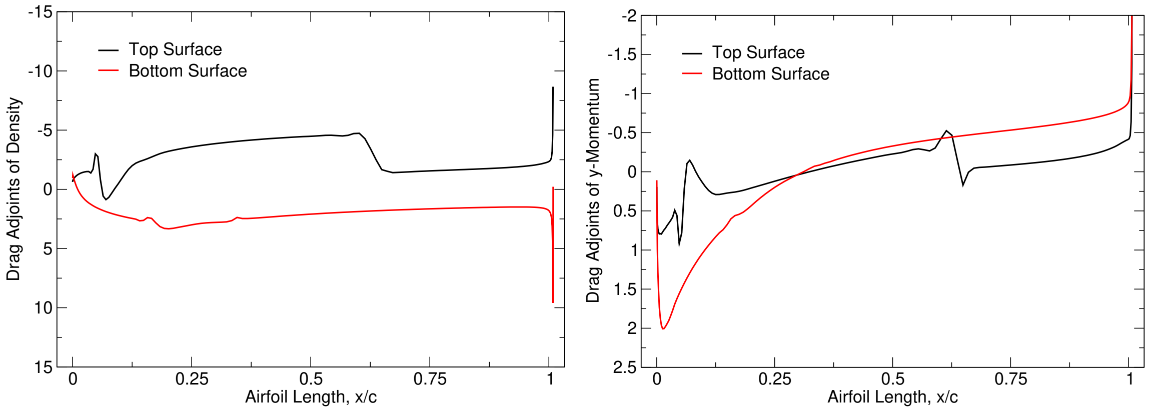


Figure 7. Drag adjoints of density and y-momentum on the surface of the NACA 0012 airfoil subject to inviscid transonic flow.

Finally, the distributions of the drag adjoints in terms of density and y-momentum on the surface of the NACA 0012 airfoil are presented in Fig. (7). Once again, the total drag coefficient is extremely sensitive to the flow solution upstream of the shocks. In particular, the strong shock formation on the suction side of the airfoil is the main contributor to the drag coefficient and eliminating this shock would lead to significant reductions in drag.

B. Transonic Flow Past the ONERA M6 Wing

The second test case focuses on the sensitivity analysis of the transonic flow past the ONERA M6 wing by utilizing the proposed toolbox. The flow structure over this wing involves a supersonic region with a special shock formation known as the lambda shock.[2, 55, 56] The geometry of the wing is based on the symmetric ONERA D airfoil, which has a maximum thickness-to-chord ratio of 10%. The 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[57] with a free-stream Mach number of 0.8395 and an angle of attack of 3.06°. Transonic flow past the ONERA M6 wing has been used in the literature as a standard benchmark test case for validation and verification of many CFD solvers while the unconstrained or lift-constrained drag minimization of this wing has been investigated extensively[58–60] for optimization studies. The computational grid used for this study consists of a rectangular outer boundary that extends about 10 mean chord lengths on each side. The fully unstructured grid has $108,396$ 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 characteristic boundary conditions are used for far-field boundaries.

The drag coefficient is used as the objective function for sensitivity analysis. In the original version of the DART toolbox, the distance-4 greedy graph coloring approach was used to color the computational mesh which resulted in $1,374$ distinct colors providing significant reduction in the number of tape evaluations necessary for calculating the second-order flow Jacobian compared to the $108,396$ nodes of the computational mesh. While this is an impressive reduction in the total number of tape reversal and adjoint evaluations, the improved version of the DART toolbox only needs to calculate and form the first-order Jacobian which will be used for preconditioning the GMRES solver even though the adjoint solutions are based on the second-order fluxes of the primal flow solver. As a result, a distance-2 graph coloring is used which leads to only 38 colors for the graph of the computational mesh. For comparison purposes, the memory footprint and CPU time breakdowns for the DART approach are presented in Tables (4) and (5) along with FDOT and original DART requirements.

Table 4. Peak memory breakdowns for the sensitivity analysis of the transonic ONERA M6 wing using different discrete adjoint tools.

Toolbox	FDOT	DART (original[25])	DART (improved)
Total (GB)	69.3	24.6	40.1
MUMPS (GB)	-	19.3	4.97
Normalized	1.0	**0.355**	**0.579**

It is apparent that the memory footprint is significantly reduced and the total memory requirements of the original and improved versions of the DART approach are still remarkably less than the memory footprint of the original FDOT toolbox. However, compared to the two-dimensional case of the NACA 0012 airfoil studied earlier, the memory reductions are less extreme. This can be associated with the grid pre-processing and metric calculations for a three-dimensional case that are more demanding compared to a two-dimensional case. It is, however, important to note that the overall memory footprint for the improved DART approach is increased since two operation stacks (or tapes) are recorded (one for the first-order and the other for the second-order flux computations) instead of only one (for the second-order flux computations). As discussed before, the operation stack based on the first-order flux calculation is used for evaluating and forming the preconditioner matrix, $\partial \mathbf{R}/\partial \mathbf{Q}_{\text{PC}}$, the other tape is used for matrix-free GMRES calculations. This results in a higher memory footprint for the adjoint evaluation toolbox. However, as was also shown in the previous test case, the use of the ANK approach for solving the adjoint system, results in a significant reduction of the computational cost. Detailed wall-clock time breakdowns and comparisons between the original and improved versions of the DART toolbox are presented in Table (5).

Table 5. Breakdowns of the wall-clock time for the sensitivity analysis of the transonic ONERA M6 wing using different discrete adjoint tools.

Toolbox	**FDOT**		**DART (original[25])**		**DART (improved)**	
Runtime		**unit**		**unit**		**unit**
Total	32.5	(hrs)	**4.85**	(hrs)	**23.2**	(min)
Jacobian, $\partial \mathbf{R}/\partial \mathbf{Q}$	-		4.72	(hrs)	-	-
matrix-free GMRES	-		-	-	18.02	(min)
Preconditioner, $\partial \mathbf{R}/\partial \mathbf{Q}_{\text{PC}}$	-		-	-	3.64	(min)
Residual Sensitivity, $\partial \mathbf{R}/\partial \mathbf{x}$	-		42.4	(s)	42.4	(s)
Vectors, $\partial I/\partial \mathbf{Q}$ and $\partial I/\partial \mathbf{x}$	-		12.3	(s)	12.3	(s)
MUMPS:						
Factorization	-		2.82	(min)	32.1	(s)
Solve	-		12.1	(s)	2.42	(s)

These results once again exhibit the efficiency and robustness of the proposed DART approach compared to the original FDOT automatic differentiation toolbox, which itself is quite efficient. While the total time required for the adjoint solver based on the FDOT toolbox is always a factor of 1 to 3 times of that of the primal code, the original DART approach significantly reduces this computational overhead. The improved version of the DART approach, presented in this work, however, further increases the computational efficiency of adjoint calculations to the points that an adjoint run that used to take more than one day is now executed in less than half an hour. This obviously makes the proposed approach an extremely viable option for gradient-based aerodynamic design optimization in large-scale applications.

The sensitivity or the gradient of the objective function with respect to the free-stream Mach number has also been validated against the finite-difference approximations in order to verify the accuracy of the DART approach in performing the discrete adjoint analysis. These results are presented in Table (6) and they

Table 6. Comparison of the drag sensitivity calculations w.r.t free-stream Mach number ($\frac{\partial C_D}{\partial M_\infty}$) for the ONERA M6 wing case.

Finite-Difference (2nd order)	FDOT Toolbox	DART Toolbox (original[25])	DART Toolbox (improved)
0.0797812$\underline{495}$	**0.079781289687318**$\underline{1}$	**0.079781289687318**$\underline{7}$	**0.079781289689**$\underline{8271}$

show the close agreements of the sensitivity calculations obtained from the original and improved versions of the DART toolbox compared to the finite-difference approximations using a second-order central difference scheme and the result obtained from our original FDOT toolbox.[20]

As explained earlier, the most expensive part in the original DART toolbox is the formation of the second-order Jacobian matrix based on the distance-4 graph coloring which requires $N_{AD} = N_{color} \times N_{eqn}$ evaluations of the tape (OP-Stack). However, in the matrix-free GMRES approach, the number of tape evaluations would only depend on the size of the Krylov subspace and the number GMRES restarts. For the transonic flow past the ONERA M6 wing, the GMRES iterations are continued until 8 orders of magnitude reduction in the original results has been achieved which occurs after 12 GMRES restarts with 10 Krylov vectors included in the subspace. Additionally, the convergence history of the GMRES solver is given in Fig. (8). Also presented is a comparison of convergence histories for the GMRES algorithm with varying number of Krylov vectors. While there is no specific rule of thumb for the required size of the Krylov subspace, it is shown that unnecessarily increasing the number of Krylov vectors would only increase the memory footprint with little to no effects on the convergence rate.

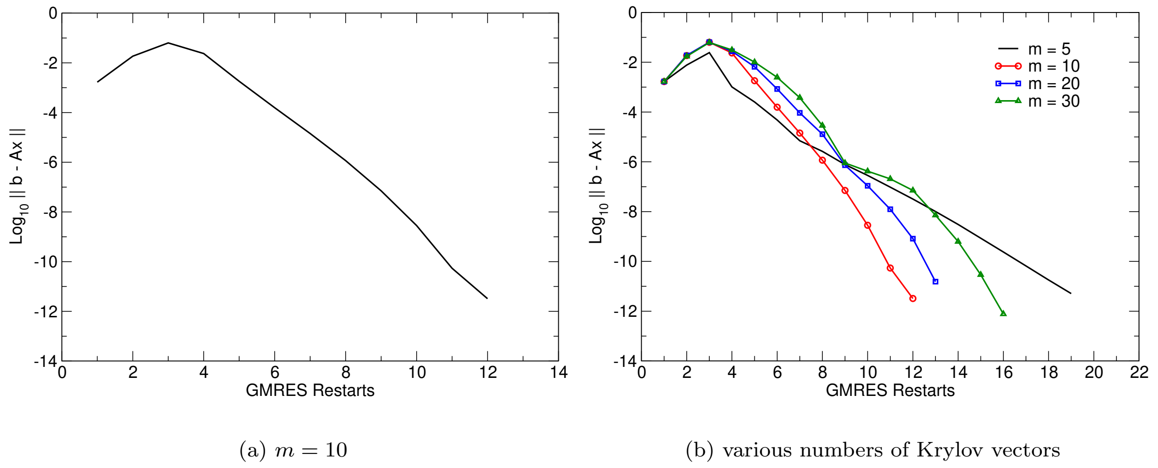
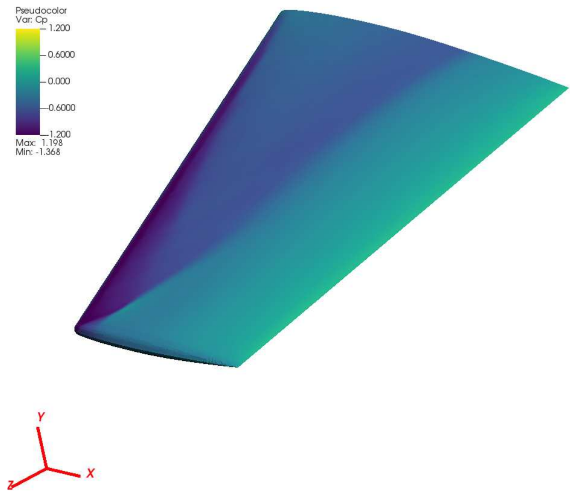
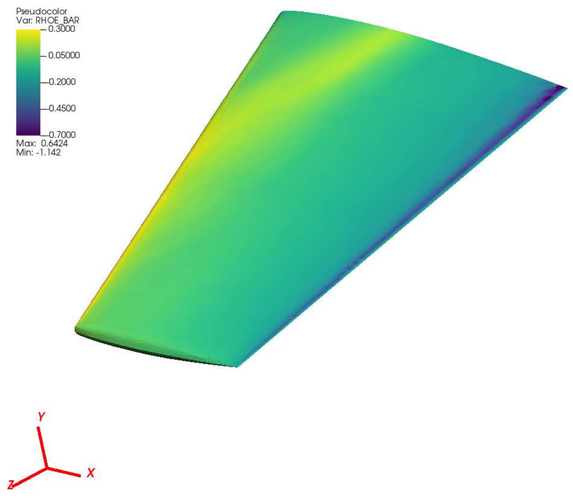


Figure 8. Convergence histories of the preconditioned matrix-free GMRES solver for the adjoint equations for different numbers of Krylov vectors included in the subspace: Inviscid transonic flow past ONERA M6 wing.

Finally, the primal and adjoint fields are presented in Fig. (9) which includes the contours of pressure coefficient on the surface of the wing as well as the drag adjoints of the total energy, $\overline{\rho E}$. The numerical results obtained for the inviscid transonic flow past the ONERA M6 wing are also compared against the experimental data of Schmitt and Charpin.[57] These solutions are reported at 2 different sections along the span of the wing and the distribution of the surface pressure coefficient at each section. As can be seen in Fig. (10), the primal solutions show a good agreement between the UNPAC solver results and the experimental data. Additionally, the adjoints of the total energy on the surface of the wing clearly depict the adjoint lambda shock that forms on the top surface of the ONERA M6 wing as opposed to the actual lambda shock that can be seen in the primal solution.

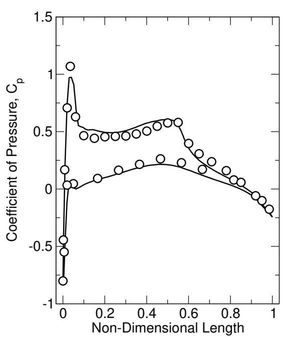


(a) Pressure coefficient contours

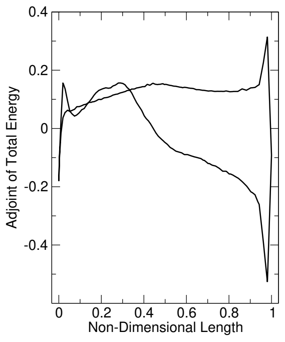


(b) Adjoint of total energy, $\overline{\rho E}$, contours

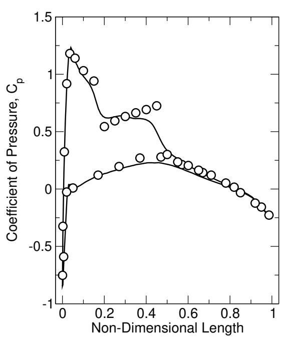
Figure 9. Pressure coefficient as well as the adjoint of total energy contour fields obtained from the primal and adjoint solvers for the inviscid transonic ONERA M6 wing case. The quantity of interest for this sensitivity analysis is the drag coefficient.



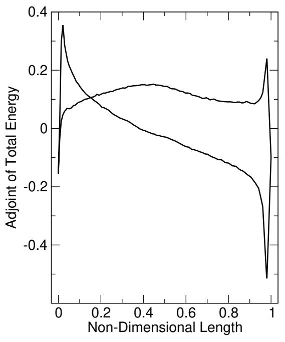
(a) C_p at $\eta = 20\%$



(b) $\overline{\rho E}$ at $\eta = 20\%$



(c) C_p at $\eta = 65\%$



(d) $\overline{\rho E}$ at $\eta = 65\%$

Figure 10. Primal and adjoint solutions at the $\eta = 20\%$ and 65% spanwise locations along the ONERA M6 wing for the inviscid transonic case with the drag coefficient used as the objective function.

C. Lift-Constrained Drag Minimization: RAE 2822 Airfoil

The last test case presented in this work studies the lift-constrained drag minimization of the RAE 2822 airfoil in turbulent transonic flow. 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 and $13,937$ nodes is considered which consists of $4,800$ quadrilateral elements in the near-field region and $18,042$ 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 shown in previous works,[24] 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 with a target lift coefficient of 0.824. This target lift constraint would initially require an angle of attack of 2.9209° to be satisfied for the current primal solver. Here, 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.

Table 7. Peak memory breakdowns for the lift-constrained drag minimization of the RAE 2822 airfoil.

Toolbox	FDOT	DART (original[25])	DART (improved)
Total (MB)	1,722	502	678
MUMPS (MB)	-	218	36
Normalized	1.0	**0.291**	**0.394**

Table 8. Breakdowns of the wall-clock time for the lift-constrained drag minimization of the RAE 2822 airfoil.

Toolbox	**FDOT**		**DART (original[25])**		**DART (improved)**	
Runtime		**unit**		**unit**		**unit**
Total	3.87	(hrs)	**1.84**	(min)	**1.23**	(min)
Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$	-		1.77	(min)	-	-
matrix-free GMRES	-		-	-	31.9	(s)
Preconditioner, $\partial\mathbf{R}/\partial\mathbf{Q}_{\mathrm{PC}}$	-		-	-	31	(s)
Residual Sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$	-		8.4	(s)	8.4	(s)
Vectors, $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$	-		1.2	(s)	1.2	(s)
MUMPS:						
Factorization	-		2.76	(s)	1.22	(s)
Solve	-		0.07	(s)	0.07	(s)

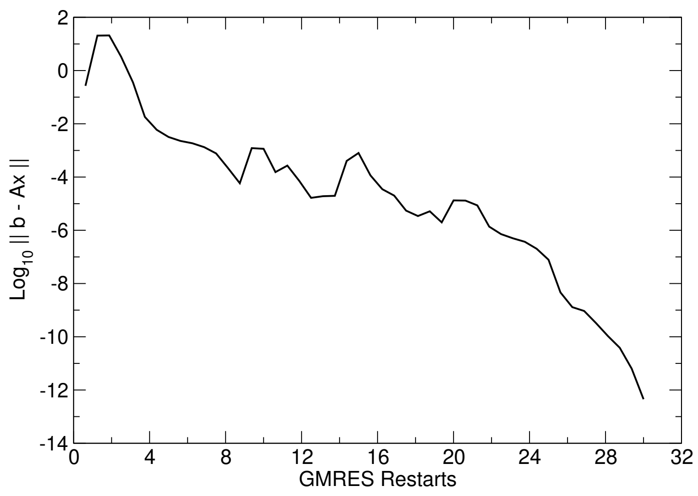
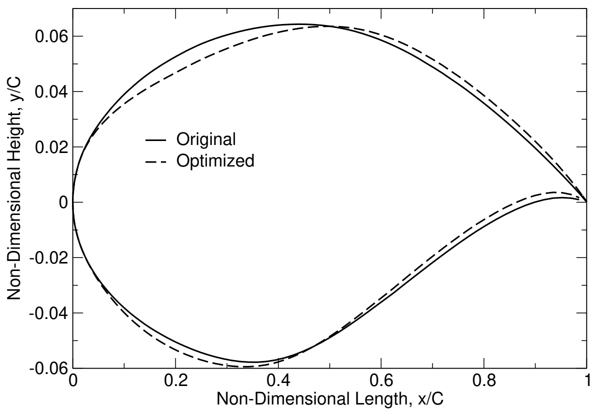
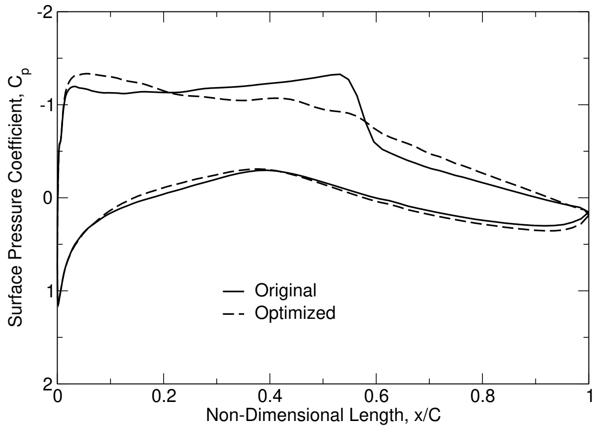


Figure 11. Convergence histories of the preconditioned matrix-free GMRES solver for the adjoint equations for $m = 15$ Krylov vectors included in the subspace: Turbulent transonic flow past RAE 2822 airfoil.

First, the memory footprint and CPU time breakdowns for the DART toolbox in its original and improved versions are presented in Tables (7) and (8). Once again, the original DART toolbox requires the distance-4 greedy graph coloring for the calculation of the second-order Jacobian which results in 142 distinct colors. For the RAE 2822 airfoil case, the computational grid consists of 13,937 nodes which means that the graph coloring algorithm utilized in the original DART toolbox leads to an almost 98.98% reduction in the number of tape reversal and adjoint evaluations during the matrix assembly procedures in DART toolbox. However, the reductions are significantly higher in the improved version since a distance-2 coloring is used for calculating the preconditioner matrix which relies on only 16 colors. Additionally, for this turbulent transonic case which involves the solution of the 2D RANS equations, the size of the Krylov subspace had to be increased to $m = 15$ and it takes 30 restarts for the GMRES solver to converge. Once again, the wall-clock time breakdowns presented in Table (8) show that the computational cost of the GMRES solver is significantly lower compared to the cost of explicitly forming the second-order Jacobian matrix as done in the original DART toolbox.



(a) Airfoil shape



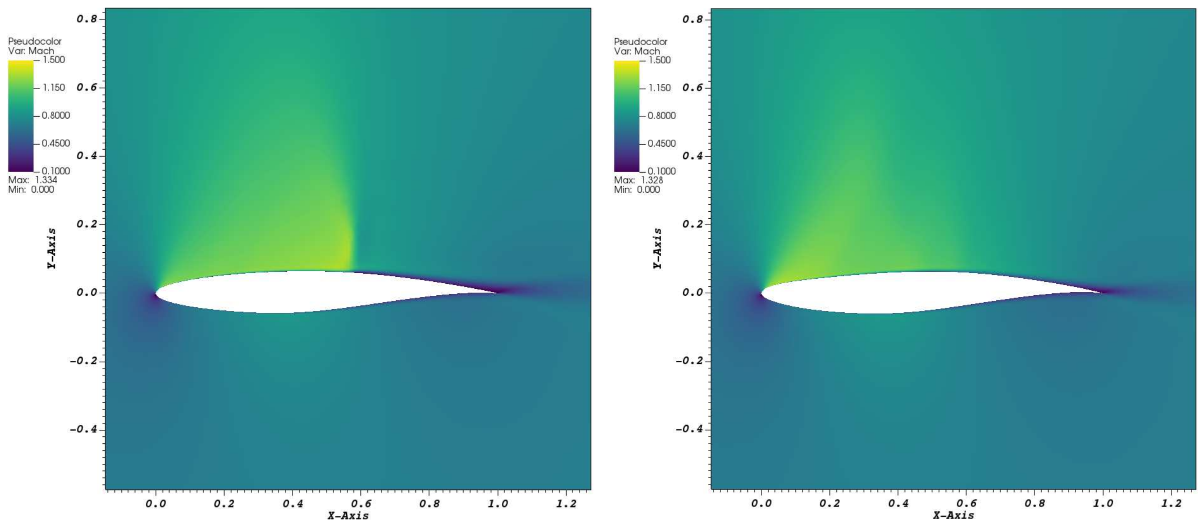
(b) Surface pressure coefficient, C_p

Figure 12. Comparison of RAE 2822 airfoil shape and the surface pressure coefficients for the original and optimized geometries.

Having presented the performance test results for the proposed DART toolbox, we now focus on the aerodynamic design optimization results. For this case, the original geometry with the prescribed flow conditions leads to a drag count of 208 which is reduced to 112 counts (almost 46.1% reduction) after the shape optimization. Since the target lift coefficient is maintained via the equality constraint incorporated as the augmented Lagrangian functional,[24] the drag minimization achieved here will result in an almost 86% increase in the airfoil efficiency.

Next, the airfoil shape and the surface pressure coefficient distributions are compared for the original and optimized geometries. These results are presented in Fig. 12 where the shock is completely eliminated as the shock-boundary-layer interaction is the main contributing factor to the total drag being minimized for this airfoil. The elimination of the shock during the design optimization process can be also shown via the Mach number contour plots for the original and optimized cases presented in Fig. 13. The optimized configuration exhibits a weaker shock-boundary-layer interaction for the deformed RAE 2822 airfoil which results in a reduced drag count for this airfoil at the present flow conditions. Additionally, the FFD box deformation as well as the interior mesh deformation are presented in Fig. 14 demonstrating maximum deformation of the FFD box around the quarter-chord of the airfoil.

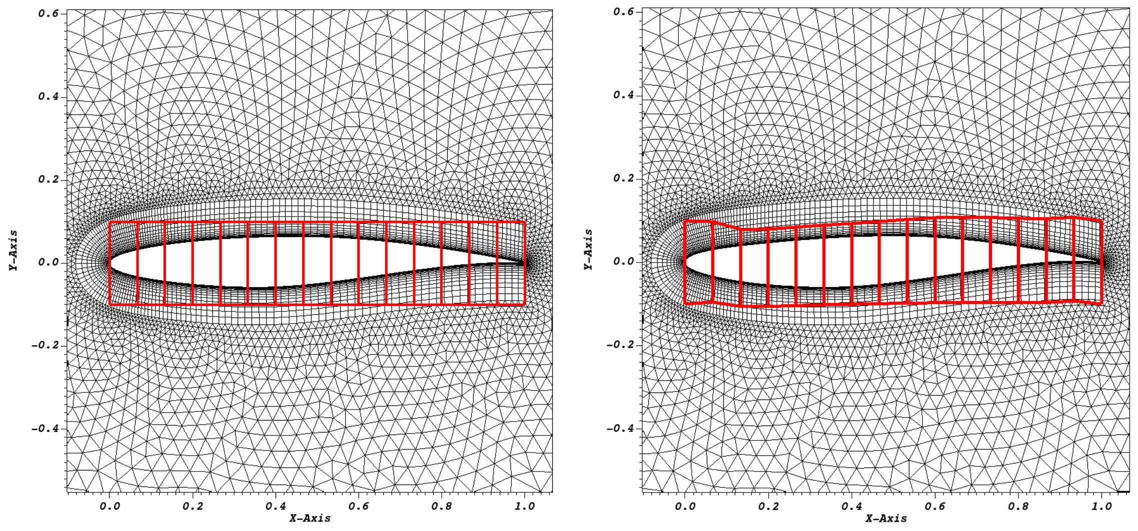
Finally, the adjoint fields are presented in Fig. (15) for density as well as the modified eddy viscosity (also known as the SA model working variable). Once more, it is interesting to note the flow reversal that is exhibited in the adjoint field as well as the triangular adjoint shock structure. This adjoint shock formation is indicative of the region in the flow field where the sensitivities of the drag coefficient with respect to the flow solution are large. Therefore, variations in the flow field around this region would have the most significant effects on the total drag count of the RAE 2822 airfoil. Additionally, the adjoints of the modified eddy viscosity, $\overline{\rho\tilde{\nu}}$, show that the sensitivities of the drag coefficient with respect to the eddy viscosity are understandably large inside the boundary layer as well as upstream of the airfoil.



(a) Original Design

(b) Optimized Design

Figure 13. Contour field of Mach number for the turbulent transonic flow past RAE 2822 airfoil.



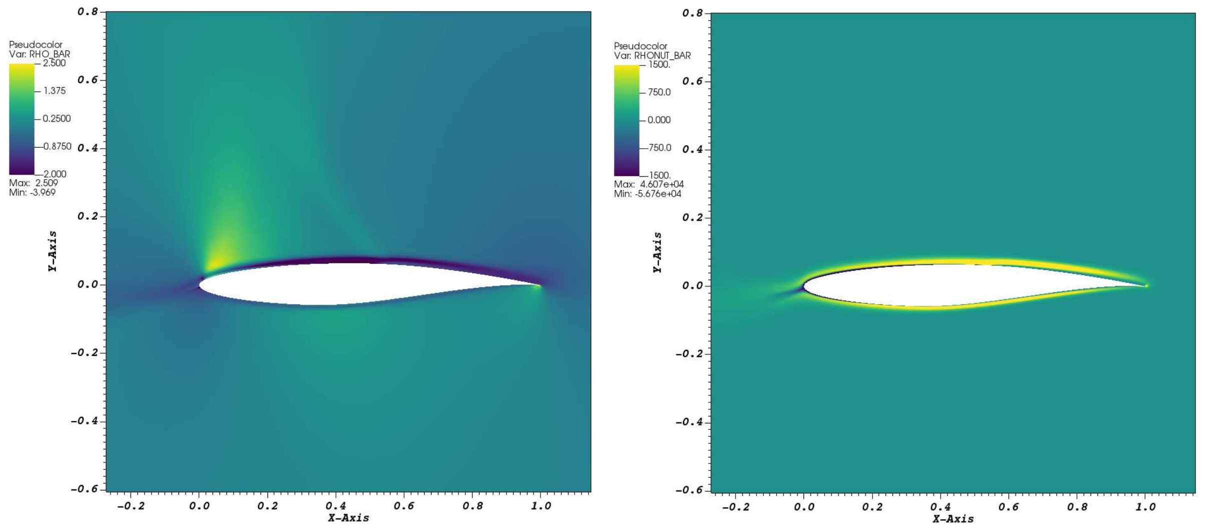
(a) Original Box

(b) Deformed (Optimized) Box

Figure 14. Original and deformed FFD box that parameterizes the RAE 2822 airfoil for the lift-constrained drag minimization problem.

V. Conclusions

The Discrete Adjoint Recursive Technique (DART) toolbox, that was previously introduced by the authors, is improved in this work that enables a memory efficient and robust sensitivity analysis. The DART toolbox relies on the in-house automatic differentiation toolbox, FDOT,[20] that was originally developed by the authors for discrete adjoint sensitivity analysis and gradient calculation of CFD solvers written in modern Fortran programming language based on the operator-overloading technique. The original version of the proposed DART toolbox utilizes a recursive technique for evaluating the second-order Jacobian matrix of



(a) Adjoint of density, $\overline{\rho}$, contours

(b) Adjoint of modified eddy viscosity, $\overline{\rho\tilde{\nu}}$, contours

Figure 15. Adjoint fields of density and modified eddy viscosity for the lift-constrained drag minimization of the RAE 2822 airfoil.

the CFD solver. It also relies on a distance-4 greedy graph coloring of the computational grid that can significantly reduce the computational cost of tape reversal necessary for calculating and forming the Jacobian matrix. However, unlike the original version that uses the MUMPS sparse solver package to directly solve the linear adjoint system, the improved version relies on an iterative Newton-Krylov approach. Therefore, the matrix-free GMRES technique is utilized to solve the adjoint system where the transposed Jacobian-vector products are calculated using a reverse or adjoint evaluation of the operation stack. Therefore, it is no longer necessary to explicitly form the second-order flow Jacobian, thus resulting in an extremely efficient adjoint calculation. The convergence and stability of the GMRES technique is, however, contingent upon a suitable preconditioner matrix. In this work, the first-order flow Jacobian matrix is used for preconditioning the adjoint system, thus resulting in an "Approximate Newton-Krylov (ANK)" solution technique. This Jacobian matrix is calculated in a very efficient and extremely accurate fashion based on the underlying automatically-differentiated CFD solver (using the FDOT toolbox) and by utilizing a distance-2 graph coloring. Ultimately, the preconditioner matrix is factorized directly using the MUMPS linear solver package. The factorized preconditioner is then used as part of the right-preconditioned GMRES algorithm to solve the linear adjoint system. Additionally, the proposed toolbox uses a forward/tangent mode of AD for calculating the non-square matrix, $\partial\mathbf{R}/\partial\mathbf{x}$, that is necessary for calculating the total derivatives of the objective function with respect to the vector of design variables. Finally, the resulting sensitivity information can be directly used for gradient-based design optimization. The developed DART toolbox is ultimately used for sensitivity analysis and drag minimization of various airfoil and wing designs. It must be noted that the computational grids used for cases studied in this work are relatively coarse. However, our numerical results have shown that the memory requirements of the MUMPS solver, that is used for the factorization of the preconditioner matrix, grow exponentially with the grid size. This can create barriers for the simulations on standard servers and clusters that typically have about 200-300 Gbytes of RAM per node. These memory issues are the subject of an ongoing research and will be addressed in a future work that aims at enhancing the memory efficiency of the proposed DART toolbox by utilizing alternative approaches for forming and utilizing preconditioner matrices with a higher computational efficiency and lower memory footprint, while still providing stability to the preconditioned GMRES solver used in the DART toolbox.

VI. Acknowledgments

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

References

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

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

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

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

[5]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.

[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]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.

[9]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.

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

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

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

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

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

[15]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.

[16]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.

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

[18]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.

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

[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, 2019, pp. 159–174.

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

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

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

[24]Djeddi, R. and Ekici, K., "Memory Efficient Adjoint Sensitivity Analysis for Aerodynamic Shape Optimization," AIAA Paper 2020-0885, 2020.

[25]Djeddi, R. and Ekici, K., "Discrete Adjoint Recursive Technique for Aerodynamic Design Optimization," AIAA Paper 2021-0000, 2021.

[26]Saad, Y. and Schultz, M. H., "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, Vol. 7, No. 3, 1986, pp. 856–869.

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

[28]Blazek, J., *Computational Fluid Dynamics: Principles and Applications*, Butterworth-Heinemann, Oxford, UK, 2015.

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

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

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

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

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

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

[35]Zhang, L. and Wang, Z., “A block LU-SGS implicit dual time-stepping algorithm for hybrid dynamic meshes,” *Computers & fluids*, Vol. 33, No. 7, 2004, pp. 891–916.

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

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

[38]Amestoy, P., Duff, I. S., Koster, J., and L’Excellent, J.-Y., “A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling,” *SIAM Journal on Matrix Analysis and Applications*, Vol. 23, No. 1, 2001, pp. 15–41.

[39]Amestoy, P., Buttari, A., L’Excellent, J.-Y., and Mary, T., “Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures,” *ACM Transactions on Mathematical Software*, Vol. 45, 2019, pp. 2:1–2:26.

[40]Yildirim, A., Kenway, G. K., Mader, C. A., and Martins, J. R., “A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations,” *Journal of Computational Physics*, Vol. 397, 2019, pp. 108741.

[41]Ekici, K., *Parallel computing techniques for rotorcraft aerodynamics*, Ph.D. thesis, Purdue University, 2001.

[42]Gebremedhin, A. H., Manne, F., and Pothen, A., “What color is your Jacobian? Graph coloring for computing derivatives,” *SIAM review*, Vol. 47, No. 4, 2005, pp. 629–705.

[43]Gray, J. S., Hearn, T. A., and Naylor, B. A., “Using Graph Coloring To Compute Total Derivatives More Efficiently in OpenMDAO,” AIAA Paper 2019-3108, 2019.

[44]Nielsen, E. J. and Kleb, W. L., “Efficient construction of discrete adjoint operators on unstructured grids using complex variables,” *AIAA journal*, Vol. 44, No. 4, 2006, pp. 827–836.

[45]Kedward, L., Allen, C. B., and Rendall, T., “Comparing Matrix-based and Matrix-free Discrete Adjoint Approaches to the Euler Equations,” AIAA Paper 2020-1294, 2020.

[46]He, P., Mader, C. A., Martins, J. R., and Maki, K. J., “An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM,” *Computers & Fluids*, Vol. 168, 2018, pp. 285–303.

[47]Kenway, G. K., Mader, C. A., He, P., and Martins, J. R., “Effective adjoint approaches for computational fluid dynamics,” *Progress in Aerospace Sciences*, Vol. 110, 2019, pp. 100542.

[48]Erhel, J., Burrage, K., and Pohl, B., “Restarted GMRES preconditioned by deflation,” *Journal of computational and applied mathematics*, Vol. 69, No. 2, 1996, pp. 303–318.

[49]Luo, H., Baum, J. D., and Löhner, R., “A fast, matrix-free implicit method for compressible flows on unstructured grids,” *Journal of Computational Physics*, Vol. 146, No. 2, 1998, pp. 664–690.

[50]Qin, N., Ludlow, D. K., and Shaw, S. T., “A matrix-free preconditioned Newton/GMRES method for unsteady Navier–Stokes solutions,” *International Journal for Numerical Methods in Fluids*, Vol. 33, No. 2, 2000, pp. 223–248.

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

[52]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.

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

[54]Vassberg, J. C. and Jameson, A., “In pursuit of grid convergence for two-dimensional Euler solutions,” *Journal of Aircraft*, Vol. 47, No. 4, 2010, pp. 1152–1166.

[55]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.

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

[57]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.

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

[59]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.

[60]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.