Discrete Adjoint Recursive Technique for Aerodynamic Design Optimization

Reza Djeddi* and Kivanc Ekici†

University of Tennessee, Knoxville, Tennessee 37996

A robust and automated sensitivity analysis toolbox based on the discrete adjoint approach is presented. 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 operator-overloading to automatically differentiate Fortran codes while utilizing advanced techniques for computational and memory efficiency without sacrificing fully-automated features. The present work aims at extending the AD toolbox using a recursive approach for evaluating partial derivatives. The resulting partial derivatives are required for solving the governing equations arising from the application of the discrete adjoint method to the primal CFD solver. This novel toolbox which we call Discrete Adjoint based on a Recursive Technique (DART) uses greedy graph coloring for efficiently evaluating the transpose of the Jacobian matrix using the adjoint mode of FDOT. Additionally, the forward mode of FDOT is used to evaluate the dense matrix that includes the sensitivities of the residual vector with respect to the design variables. Ultimately, the resulting adjoint equations are solved using the MUMPS direct sparse solver package. Computed sensitivity information can then be used for gradient-based design optimization. The DART toolbox developed in this work is eventually 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, it is desirable to use gradient-based algorithms in the framework of aerodynamic design optimization. In this approach, however, derivatives (sensitivities) of a cost function with respect to all design variables (that can number in the hundreds to thousands) are required. While "optimum" configurations can be determined after a small number of optimization cycles using the gradient-based approach, the cost and complexity associated with the gradient evaluations can be overwhelming. These issues are even more pronounced in high-fidelity CFD solvers and in the framework of a multidisciplinary design optimization process. Therefore, a fast and efficient sensitivity analysis tool would be a leap forward that can greatly reduce the time and cost of the CFD-based design and topology optimization.

The most important and challenging part of the gradient-based design optimization approach is the calculation of the gradient information. Traditionally, finite difference (FD) has been used for this purpose. Appropriate step-size dilemma associated with this approximative technique has 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. For problems with

*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.

hundreds of design variables this approach can be prohibitively costly. Also, the FD or CS techniques only provide "approximations" to the desired derivatives with certain accuracy. As an alternative, the algorithmic or automatic differentiation (AD) based on the chain rule results in analytically evaluated derivative values without any truncation error.

Automatic differentiation has two basic modes of operation: 1) forward/tangent, and 2) reverse/adjoint corresponding to a *top-down* or a *bottom-up* strategy of accumulating partial derivatives, respectively. It is important to note that similar to the FD approach the computational cost of the tangent AD is proportional to the number of independent variables. On the other had, 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] Both methods have been extensively used for aerodynamic design optimization in the past two decades.[3–5]

The adjoint method can be classified as 1) discrete[6,7] or 2) continuous[8,9] 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] 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 discrete adjoint 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 Fast automatic Differentiation toolbox based on Operator-overloading Technique (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, a novel tool for gradient calculation is developed that relies on a recursive technique. The Discrete Adjoint based on the Recursive Technique (DART) toolbox solves for the adjoint flow field using a direct approach that forms the sparse transposed Jacobian matrix by taking advantage of operator-overloading. The developed toolbox also uses an effective graph coloring technique that significantly reduces the number of adjoint tape evaluations. 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 several sensitivity analysis and design optimization test cases involving standard airfoils and wings.

II. Governing Equations

This section introduces the governing equations for the primal (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

For flow field calculations, three-dimensional Unsteady Reynolds-Averaged Navier-Stokes (URANS) equations augmented with the Spalart-Allmaras turbulence model[25] 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:

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

Here, $\tilde{\nu}$ is the viscosity-like working variable of the Spalart-Allmaras turbulence model. Additionally, the source vector, $\mathbf{S}$, has all zero terms except for the last equation where the source term is described by the Spalart-Allmaras turbulence model.[25] 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.[2,26,27] 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 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 first developed by Roe.[28] Additionally, limiter functions of Barth and Jespersen[29] and Venkatakrishnan[30] are used for higher-order solution reconstruction at the face center. As is customary, second-order central averaging scheme is used for the calculation of the viscous fluxes. The gradients of the flow variables are calculated at the grid nodes using a Green-Gauss method. Furthermore, the solver is parallelized using the message passing interface (MPI) tools with a non-overlapping domain decomposition,[31] and METIS software package[32] 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[33] 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 $I(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$, defined as

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

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 Lagrangian 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 realized 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 have 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. These gradients are calculated by an "adjoint evaluation" process whose computational cost is only a small multiple of that of the primal solver.[34,35] It must be noted that in the discrete analysis, the derivatives are propagated in the reverse direction, thus require the complete convergence 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 requires minimal changes to an already existing primal solver to obtain the adjoint version of that solver. One of the modifications that is necessary is the declaration of the flow solution variables, $\mathbf{Q}$, 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 to automatically differentiate 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 previous section, the derivation process for the discrete adjoint flow equations was presented. In summary, the numerical procedure used for calculating the total derivatives consists 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 $\partial\mathbf{R}/\partial\mathbf{Q}$ (AD Process 1) and $\partial\mathbf{R}/\partial\mathbf{x}$ (AD Process 2) matrices.
2. Calculate the partial derivatives required to determine $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$ vectors (AD Process 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.

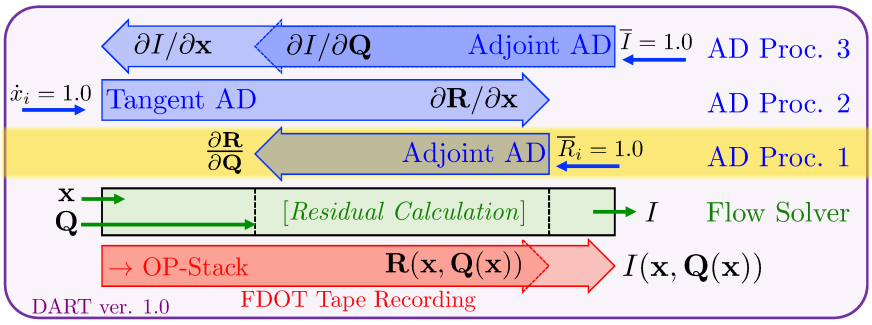


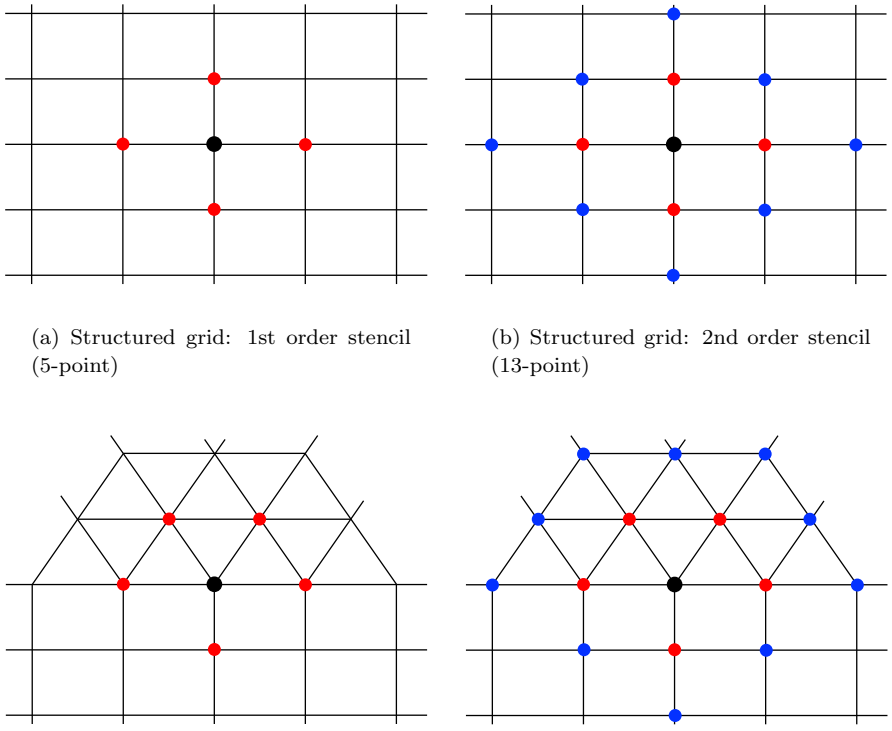
Figure 1. Linearization of the primal code with the FDOT toolbox recording the expression tree into the OP-Stack. Note the adjoint and tangent modes of automatic differentiation that are used for calculating the necessary partial derivatives for the DART approach.

The linearization process for the CFD solver involves running a slightly "modified" version of the primal code as shown in Fig. (1). 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 is followed while the expression tree is recorded with the beginning and end of the iterative section marked via a "checkpointing" function.[20,24] However, for the proposed approach presented in this work, 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, $\mathbf{R}(\mathbf{x}, \mathbf{Q}(\mathbf{x}))$, as well as 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, 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.

B. Jacobian Matrix, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, via an Adjoint/Reverse AD Approach

The most important aspect of the DART approach is the calculation of the Jacobian which can be viewed as a sparse, non symmetric square matrix. The fact that the Jacobian matrix is square limits the benefits of using either a forward/tangent or reverse/adjoint mode of automatic differentiation. On the other hand, 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, the sparsity pattern must be taken into account to minimize the number of partial derivative calculations and storage for a highly sparse Jacobian matrix.



(a) Structured grid: 1st order stencil (5-point)

(b) Structured grid: 2nd order stencil (13-point)

(c) Hybrid grid: 1st order stencil

(d) Hybrid grid: 2nd order stencil

Figure 2. Compact node stencils for the 1st order and 2nd order schemes. Depicted here are examples on typical structured as well as hybrid grids where the distance-1 and distance-2 neighbors for the center node (black) are shown in red and blue, respectively.

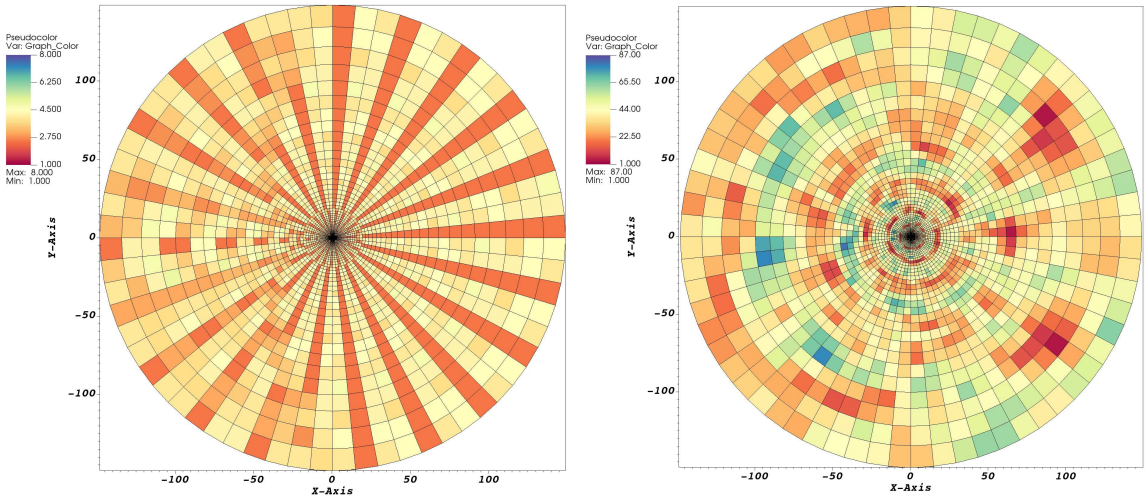
The vertex-based, mixed-element, unstructured/hybrid grid approach utilized in the primal solver (UNPAC) relies on first-order and second-order stencils as shown for typical grids in Fig. (2). In the simplest form used for the first-order accurate Euler solutions, the stencil only includes direct neighbors 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.[36] 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 graph 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.

As such, in the proposed DART toolbox, 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,[37] complex-step,[38,39] as well as the forward mode of AD.[40,41] Interestingly, the use of graph coloring significantly reduces the number of residual evaluations to a very small number, i.e., N_{color}.



(a) 1st order scheme: 8 colors

(b) 2nd order scheme: 87 colors

Figure 3. Results of distance-2 and distance-4 greedy graph coloring for the 64×65 structured grid around the NACA 0012 airfoil.

Going back to the compact stencils presented earlier, it can be easily understood that 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. It must be noted that 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[40,41] 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.[39] In the present work, a greedy distance-2 or distance-4 graph coloring approach is utilized for first-order and second-order stencils. As an example, the graph coloring is applied to an O-typed structured grid around the NACA 0012 airfoil and the results are shown in Fig. (3). For this case, the structured grid has 4,160 nodes while the distance-2 and distance-4 graph coloring techniques lead to 8 and 87 colors, respectively. This results in a very significant reduction in the number of tape reversal and adjoint evaluations for calculating

the sparse Jacobian matrix. It is worth noting that the number of graph colors is typically on the order of $\mathcal{O}(100)$ to $\mathcal{O}(1000)$ for the two- and three-dimensional test cases studied with the present DART approach based on a greedy graph coloring technique.

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 depicted in Fig. (4) 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.

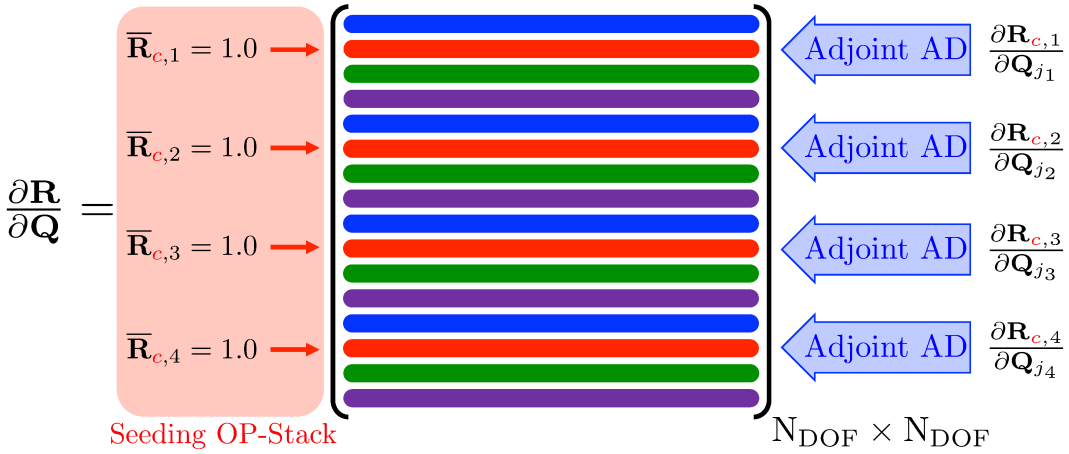


Figure 4. The adjoint mode of AD used in conjunction with a graph coloring approach for efficient evaluation of the Jacobian matrix. For this example, it is assumed that there are only 4 colors in the partitioned matrix with rows of the same color being seeded and evaluated simultaneously.

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

$$\mathrm{N_{AD}} = \mathrm{N_{color}} \times \mathrm{N_{eqn}} \quad \ll \quad \mathrm{N_{DOF}} = \mathrm{N_{node}} \times \mathrm{N_{eqn}} \tag{14}$$

As can be seen, the use of graph coloring can lead to a very efficient approach for calculating the Jacobian matrix based on the adjoint mode of AD. Once again, it must be noted that in the present work, the OP-Stack is recorded using the FDOT toolbox developed previously by the authors which uses an operator-overloading technique for automatic differentiation.

C. 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 Jacobian matrix 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 $\mathrm{N_{DOF}} \times \mathrm{N_{dv}}$ where $\mathrm{N_{dv}}$ 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., $N_{dv} \ll N_{DOF}$, 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 is 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 results in the desired $\partial \mathbf{R}/\partial \mathbf{x}_i$ terms. This process is also shown in Fig. (5). 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.

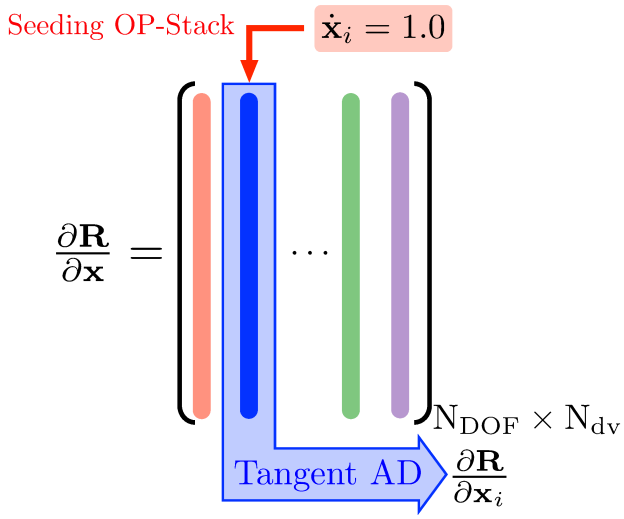


Figure 5. The tangent mode of AD used for efficient evaluation of the $\partial \mathbf{R}/\partial \mathbf{x}$ matrix.

D. 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 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. Having the recorded OP-Stack and as described in the previous section, the two important ingredients of the DART approach, which are the Jacobian and the residual sensitivity matrices, will be calculated using a hybrid adjoint and forward modes of AD. The additional steps performed in the DART toolbox that will lead to the calculation of the total derivatives are described next.

1. Evaluating Objective-Function-Based Vectors

Having the $\partial\mathbf{R}/\partial\mathbf{Q}$ and $\partial\mathbf{R}/\partial\mathbf{x}$ matrices evaluated, it is now time to calculate the two remaining parts 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.[42] 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 (see Fig. [1]).

2. Sparse Linear Solver

As discussed in Section B, in addition to evaluating the partial derivatives, it is necessary to solve the adjoint Eq. (12) to determine the adjoint flow field. To achieve this goal, the system of linear equations may be solved using either direct or iterative methods. The main advantage of using direct methods is that the exact solutions to the linear system can be obtained with a fixed number of operations. However, direct methods can become prohibitively costly for large-scale problems. On the other hand, iterative methods are typically advantageous for large-scale problems as they have no restrictions on the size or the structure of the linear system. Primarily, these techniques rely on a linear fixed-point iteration method where the adjoint solution is incrementally improved from an initial guess via a particular iterative scheme such as Euler or Runge-Kutta. Additionally, Kenway et al.[41] have used a Krylov-subspace generalized minimal residual (GMRES) method to solve the primal and adjoint equations. While providing a fast convergence behavior, GMRES method necessitates a strong preconditioning matrix as the exact Jacobian matrix, $[\partial\mathbf{R}/\partial\mathbf{Q}]^T$, is typically ill-conditioned.

With the advances in computational science and parallel computing, highly efficient direct solver packages have been developed that effectively address the memory issues of the conventional direct methods. Therefore, in this work, we rely on the MUltifrontal Massively Parallel Solver (MUMPS) package[43,44] to solve the adjoint Eq. (12) where the transposed Jacobian matrix is stored in the compressed sparse format. Our numerical studies have shown that the MUMPS solver is a very efficient and accurate direct sparse linear solver with a very low memory footprint and an extremely fast lower and upper factorization. It must be noted that the DART toolbox can be altered very easily to rely on a preconditioned GMRES iterative method which is the focus of an ongoing research.

3. Total Derivative Calculation

As shown earlier, the adjoint Eq. (12) 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

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 this approach enables us to perform the factorization on the $[\partial \mathbf{R}/\partial \mathbf{Q}]^T$ matrix only once. The factorized transposed Jacobian is then used with different right-hand-side vectors to obtain the corresponding adjoint solutions.

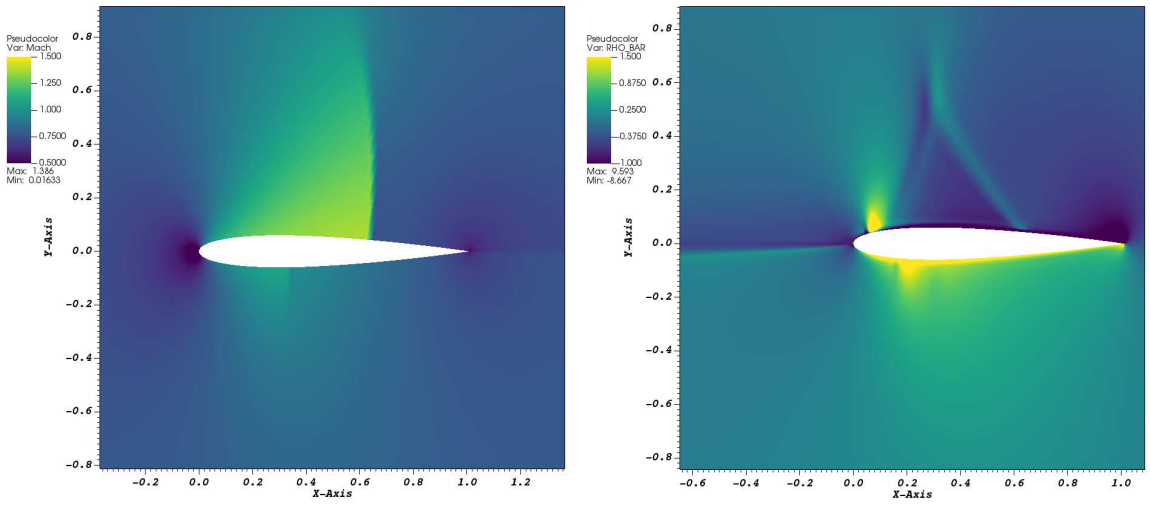
Ultimately, having the adjoint vector, we will perform a matrix-vector multiplication as well as a final vector addition, according to Eq. (13), to evaluate the final total derivative vector. Once again, this process will be repeated for each quantity of interest (objective function and/or the optimization 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[45] and the SLSQP[46] 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 proposed DART toolbox in discrete adjoint-based gradient calculations in addition to conducting a detailed memory footprint and computational effort analysis. Finally, the lift-constrained drag minimization of the RAE 2822 airfoil with turbulent transonic flow is studied.

A. Transonic Flow Past 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[47] 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.



(a) Absolute Mach number contours

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

Figure 6. 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.

First, the primal and adjoint flow fields are presented in Fig. (6) which includes the contours of Mach number for the primal solution as well as the drag adjoints of the total energy. As can be seen, the flow reversal which is the hallmark of the adjoint method is evident in Fig. (6(b)) where 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.

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

Toolbox	FDOT	DART
Total (GB)	4.61	1.29
MUMPS (MB)	-	732
Normalized	1.0	**0.279**

For this case, the distance-4 greedy graph coloring approach leads to 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 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 $N_{AD} = 380$ which results in a very significant saving of more than 99.8%. Additionally, the memory footprint and CPU time breakdowns for the DART approach are presented in Tables (1) and (2).

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**	
Runtime		**unit**		**unit**
Total	**4.5**	(hrs)	**3.88**	(min)
Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$	-		3.16	(min)
Residual Sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$	-		21.5	(s)
Vectors, $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$	-		2.1	(s)
MUMPS:				
Factorization	-		9.29	(s)
Solve, Eq. (12)	-		2.32	(s)

As can be seen, the DART approach leads to a very small memory footprint due to a much shorter expression tree that is recorded into the OP-Stack as was shown in Fig. (1). 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 very memory efficient and the total time spent for factorization and solving the linear system (Eq. [12]) is only a very small fraction of the total time spent in the primal code to obtain the fully converged flow solution.

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 DART approach, the sensitivity of the drag coefficient with respect to the free-stream Mach number is considered. As shown in Table 3, the sensitivity values obtained from the original FDOT toolbox[24] and the present technique using the DART toolbox are compared to the finite-difference approximations using a first-order backward difference as well as a second-order central difference scheme. The presented results show the accuracy of the gradient evaluations using the proposed DART approach.

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, it can be seen that the total drag coefficient is extremely sensitive to the flow solution upstream of the shocks. In particular, the strong shock formation

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 (1st order)	Finite-Difference (2nd order)	FDOT Toolbox	DART Toolbox
0.3278320019	**0.327867**101989	**0.327867042969275**	**0.327867042969271**

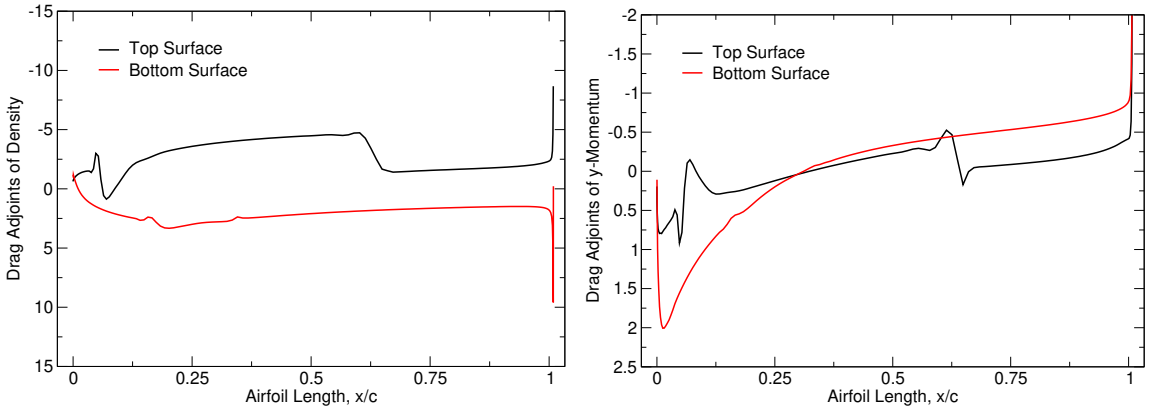


Figure 7. Drag adjoints of density and y-momentum on the surface of the NACA 0012 airfoil subject to inviscid transonic flow.

on the suction side of the airfoil is the main contributor to the drag coefficient and eliminating this shock would lead to significant reductions of the drag coefficient which would be the ultimate goal of a typical "drag minimization" problem.

B. Transonic Flow Past the ONERA M6 Wing

The second test case presented in this work focuse on the sensitivity analysis of the transonic flow past the ONERA M6 wing by utilizing our proposed DART toolbox. The aerodynamics of this wing involve a region of supersonic flow including a special shock formation known as the lambda shock.[2,48,49] The geometry of the transonic M6 wing is based on the symmetric airfoil sections of type ONERA D which have a maximum thickness-to-chord ratio of 10%. The M6 wing has a sweep angle of 30 degrees at the leading edge and an aspect ratio of 3.8 and is tapered with a ratio of 0.562. The flow conditions are set according to the experiments carried out by Schmitt and Charpin[50] with a free-stream Mach number of 0.8395 and an angle of attack of 3.06°. It must be noted that the transonic flow past the ONERA M6 wing has been used in the literature as a standard benchmark test case for the purpose of validation and verification of the CFD solvers while the unconstrained or lift-constrained drag minimization of this wing has also been studied extensively.[51–53] 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 is made of 108,396 nodes and 582,752 tetrahedral elements with 38,756 triangular elements on the surface of the wing. Here, a symmetry boundary condition is used on the root-plane and far-field boundary conditions are used for the rest of the outer boundaries.

For the purpose of sensitivity analysis using our newly developed discrete adjoint toolbox, the drag coefficient is selected as the objective function. The distance-4 greedy graph coloring approach is utilized to color the computation grid which results in 1361 distinct colors. Compared to the number of grid nodes for the computational mesh used herein, the distance-4 graph coloring leads to an almost two orders of magnitude reduction of the tape evaluations in order to calculate the Jacobian matrix. It is worth noting that a distance-2 coloring, on the other hand, leads to only 36 distinct colors for this same computational grid. However, the DART toolbox developed in this work that utilizes a direct approach for obtaining the adjoint flow field relies on the second order flux Jacobian matrix which necessitates the use of a distance-4

greedy graph coloring. The memory footprint and CPU time breakdowns for the DART approach compared to the original FDOT toolbox are presented in Tables (4) and (5) for this test case.

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

Toolbox	FDOT	DART
Total (GB)	69.3	24.6
MUMPS (GB)	-	19.3
Normalized	1.0	**0.355**

As can be seen, the memory footprint is significantly reduced and the total memory requirements of the DART approach, including the internal memory used by MUMPS for factorizing the transposed Jacobian matrix, is 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. Additionally, the wall-clock time breakdowns presented in Table (5) 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 1-3x of that of the primal code, the new DART approach significantly reduces this computational overhead. Therefore, the wall-clock time spent by the adjoint solver using the DART toolbox, is consistently less than that of the primal solver which makes the proposed approach an extremely viable option for gradient-based aerodynamic design optimization in large-scale applications.

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**	
Runtime		**unit**		**unit**
Total	**32.5**	(hrs)	**4.85**	(hrs)
Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$	-		4.72	(hrs)
Residual Sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$	-		42.4	(s)
Vectors, $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$	-		12.3	(s)
MUMPS:				
Factorization	-		2.82	(min)
Solve, Eq. (12)	-		12.1	(s)

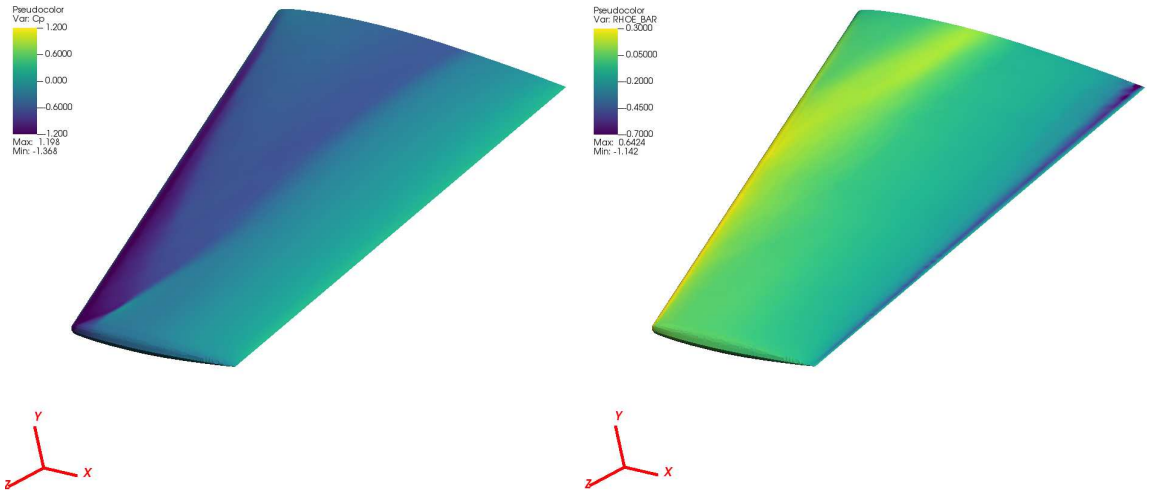
The sensitivity or the gradient of the objective function with respect to the free-stream Mach number has also been validated against the finite-difference approximations in order to verify the accuracy of the DART approach in performing the discrete adjoint analysis. These results are presented in Table (6) and they show the close agreements of the sensitivity calculations compared to the finite-differences and the result obtained from our original FDOT toolbox.[20] Finally, the primal and adjoint flow fields are presented in Fig. (8) which includes the contours of pressure coefficient on the surface of the wing as well as the drag adjoints of the total energy, $\overline{\rho E}$.

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 (1st order)	Finite-Difference (2nd order)	FDOT Toolbox	DART Toolbox
0.0796491138	**0.079691**203612	**0.0796914808069531**5	**0.0796914808069532**9

The numerical results obtained for the inviscid transonic flow past the ONERA M6 wing are compared

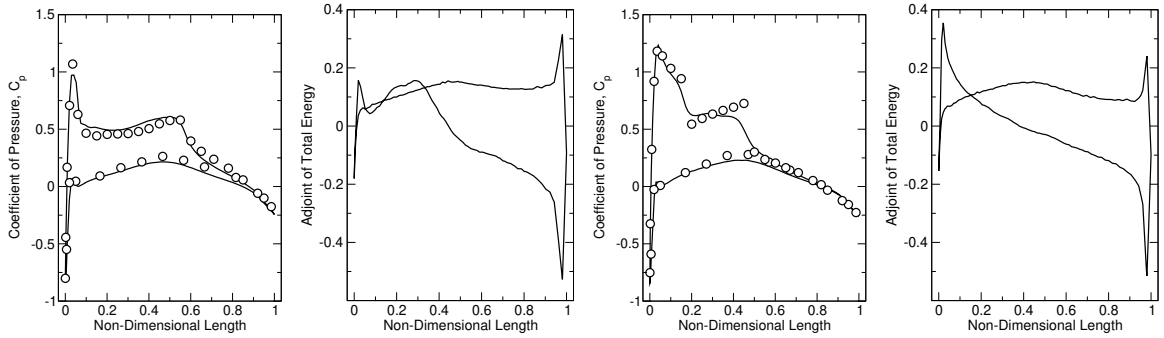
against the experimental data of Schmitt and Charpin.[50] These solutions are reported at 2 different sections along the span of the wing and the distribution of the surface pressure coefficient at each section. As can be seen in Fig. (9), the primal solutions show a good agreement between the UNPAC 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 8. 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 9. 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.

It must be noted that the computational grid used for this case is very coarse and the same test case must be studied using a significantly finer mesh tailored for capturing the turbulent boundary layer of this wing in order to obtain better agreements with the experimental data. Additionally, the finer mesh would result in a smoother adjoint flow field that can better identify the surface sensitivities for the purpose of drag minimization. However, our numerical results have shown that the memory requirements of the MUMPS solver, that is used for the factorization and the inversion of the sparse Jacobian 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. 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 an approximate Newton-Krylov (ANK), preconditioned GMRES solver instead of a direct solver such as the MUMPS package.

C. Lift-Constrained Drag Minimization: RAE 2822 Airfoil

Having presented sensitivity analysis results, we now shift our focus to 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. 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 reaches 29.

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

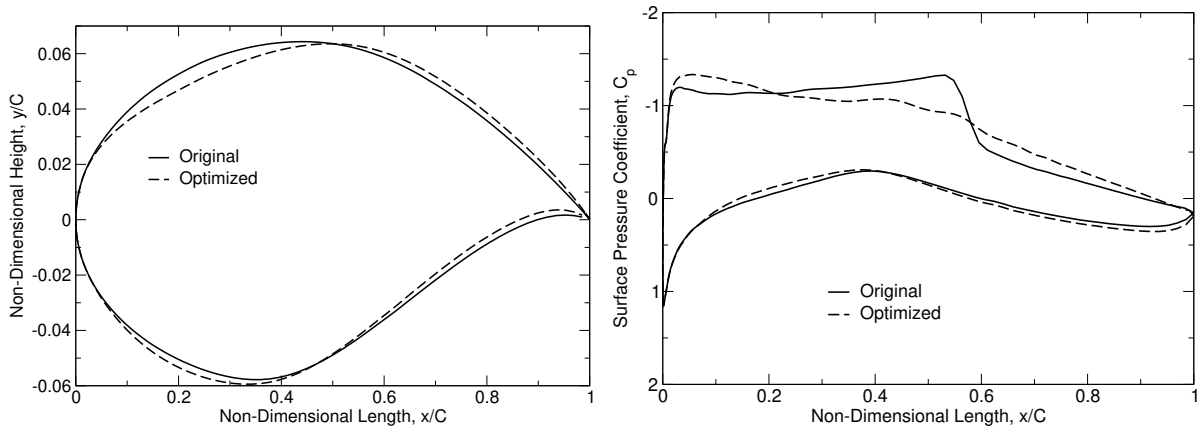
Toolbox	FDOT	DART
Total (MB)	1,722	502
MUMPS (MB)	-	218
Normalized	1.0	**0.291**

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

Toolbox	**FDOT**		**DART**	
Runtime		**unit**		**unit**
Total	**3.87**	(hrs)	**1.84**	(min)
Jacobian, $\partial\mathbf{R}/\partial\mathbf{Q}$	-		1.77	(min)
Residual Sensitivity, $\partial\mathbf{R}/\partial\mathbf{x}$	-		8.4	(s)
Vectors, $\partial I/\partial\mathbf{Q}$ and $\partial I/\partial\mathbf{x}$	-		1.2	(s)
MUMPS:				
Factorization	-		2.76	(s)
Solve, Eq. (12)	-		0.07	(s)

First, the memory footprint and CPU time breakdowns for the DART approach are presented in Tables (7) and (8). For this case, the distance-4 greedy graph coloring approach leads to only 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 here leads to an almost 98.98% reduction in the number of tape reversal and adjoint evaluations during the matrix assembly procedures in DART toolbox. Once again, it can be seen that the proposed DART approach is highly memory efficient and the direct solution method using the MUMPS package turns out to be extremely fast.

Having presented the performance test results for the proposed DART toolbox, we can now focus our attention to 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%

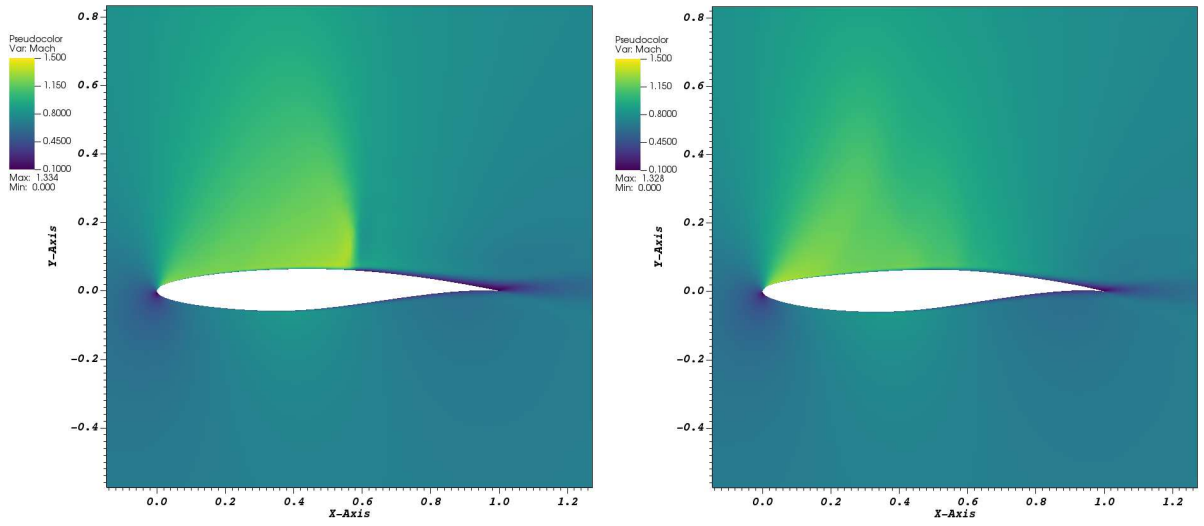


(a) Airfoil shape

(b) Surface pressure coefficient, C_p

Figure 10. Comparison of RAE 2822 airfoil shape and the surface pressure coefficients for the original and optimized geometries.

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 shown in Fig. 10.

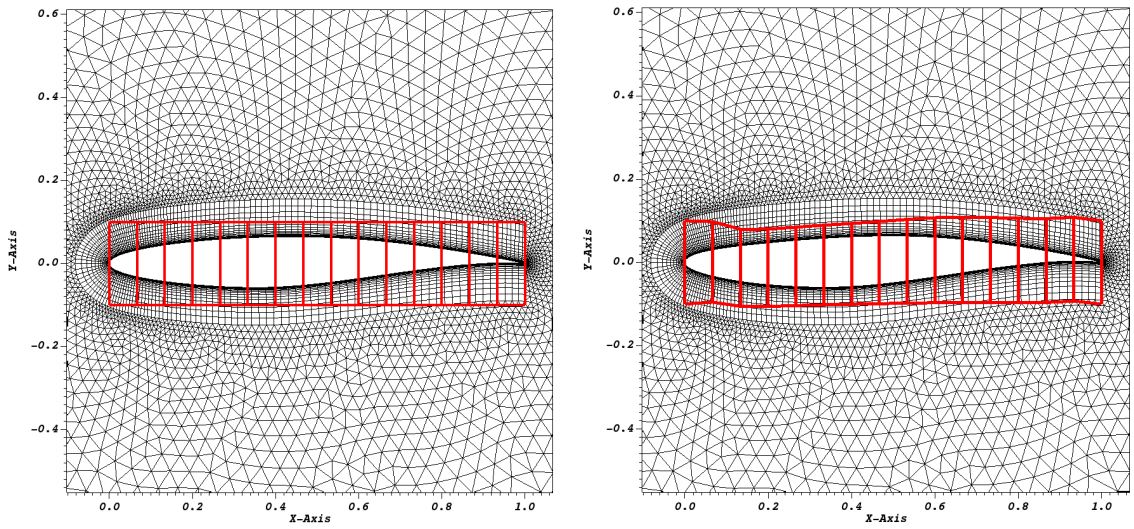


(a) Original Design

(b) Optimized Design

Figure 11. Contour field of Mach number for the turbulent transonic flow past RAE 2822 airfoil.

As can be seen in Fig. (10), 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. These results are shown in Fig. 11 and exhibit 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

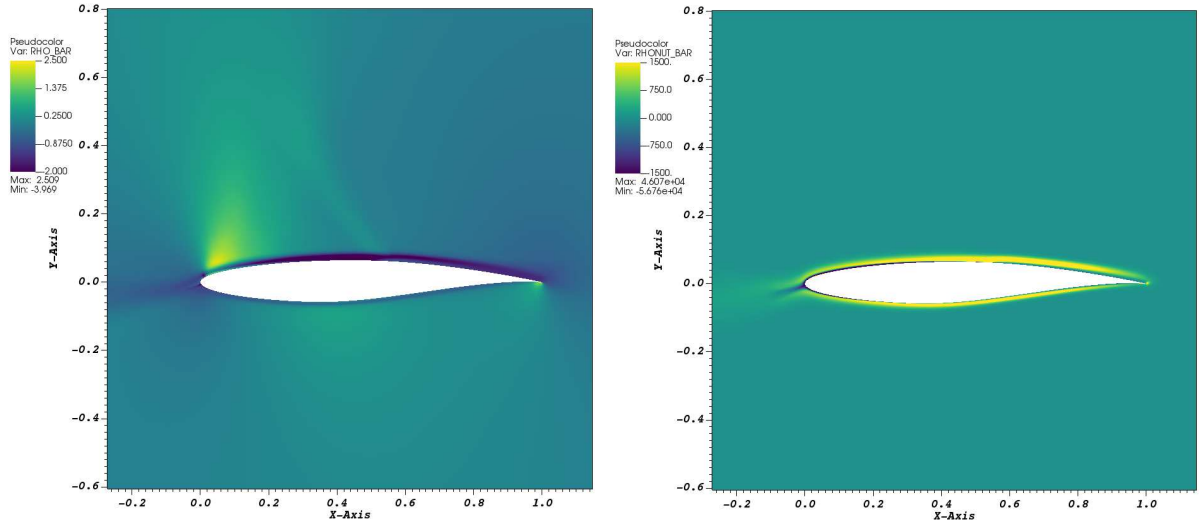


(a) Original Box

(b) Deformed (Optimized) Box

Figure 12. Original and deformed FFD box that parameterizes the RAE 2822 airfoil for the lift-constrained drag minimization problem.

are presented in Fig. 12 for this case. It is shown that this particular optimization leads to the maximum deformation of the FFD box around the quarter-chord of the airfoil.



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

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

Figure 13. Adjoint fields of density and modified eddy viscosity for the lift-constrained drag minimization of the RAE 2822 airfoil.

Finally, the adjoint flow fields are presented for the density as well as the modified eddy viscosity (also known as the SA model working variable) and the results are presented in Fig. (13). Once again, 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 can, once again, identify the region in the flow field upstream of the shock 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 the contamination region upstream of the airfoil.

V. Conclusions

A memory efficient and robust sensitivity analysis toolbox is developed based on the discrete adjoint approach. At its core, the proposed techniques, called Discrete Adjoint Recursive Technique (DART), relies on the in-house automatic differentiation toolbox, FDOT,[20] originally developed by the authors. The FDOT toolbox uses the operator-overloading mode of automatic differentiation for discrete adjoint sensitivity analysis and gradient calculation of CFD solvers written in modern Fortran programming language. The proposed DART toolbox utilizes a recursive technique for evaluating the Jacobian matrix of the CFD solver while taking advantage of a distance-4 greedy graph coloring of the computational grid, thus enabling the DART toolbox to efficiently calculate all necessary partial derivatives with very few AD tape evaluations. The linear adjoint system is then solved using the MUMPS sparse solver package that carries out a direct factorization of the transposed Jacobian matrix. 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.

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

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

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

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

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

[30]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.

[31]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.

[32]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.

[33]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.

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

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

[36]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.

[37]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.

[38]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.

[39]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.

[40]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.

[41]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.

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

[43]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.

[44]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.

[45]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.

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

[47]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.

[48]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.

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

[50]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.

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

[52]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.

[53]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.