Fluid Dynamics and Co-located Conferences
June 24-27, 2013, San Diego, CA
21st AIAA Computational Fluid Dynamics Conference

Automatic Differentiation Adjoint of the Reynolds-Averaged Navier–Stokes Equations with a Turbulence Model

Zhoujie Lyu* and Gaetan K.W. Kenway†
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI

Cody Paige‡
University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada

Joaquim R. R. A. Martins§
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI

This paper presents an approach for the rapid implementation of an adjoint solver for the Reynolds-Averaged Navier–Stokes equations with a Spalart–Allmaras turbulence model. Automatic differentiation is used to construct the partial derivatives required in the adjoint formulation. The resulting adjoint implementation is computationally efficient and highly accurate. The assembly of each partial derivative in the adjoint formulation is discussed. In addition, a coloring acceleration technique is presented to improve the adjoint efficiency. The RANS adjoint is verified with complex-step method using a flow over a bump case. The RANS-based aerodynamic shape optimization of an ONERA M6 wing is also presented to demonstrate the aerodynamic shape optimization capability. The drag coefficient is reduced by 19% when subject to a lift coefficient constraint. The results are compared with Euler-based aerodynamic shape optimization and previous work. Finally, the effects of the frozen-turbulence assumption on the accuracy and computational cost are assessed.

I. Introduction

Recent advances in high performance computing have enabled the deployment of full-scale physics-based numerical simulations and optimization in academia and industry. Computational fluid dynamics (CFD) tools and numerical optimization techniques have been widely adopted to shorten the design cycle times and to explore the design space more effectively. High-fidelity methods enable engineers to perform detailed designs earlier in the design process, allowing them to better understand the design trade-offs and make more informed decisions. In addition, advances in sensitivity analysis via the adjoint method [1] dramatically improve the effectiveness and computational time of aerodynamic shape optimization. However, due to the complexity of the CFD solvers, deployment of the adjoint method in Reynolds-averaged Navier–Stokes (RANS) solvers remains a challenging task.

There are two types of adjoint approaches: continuous and discrete. In the continuous approach, the adjoint method is directly applied to the governing differential equations. Partial derivatives of the objectives and residuals with respect to state variables and design variables are combined via the adjoint variables. The governing equations and the adjoint equation along with the boundary conditions are then discretized to obtain numerical solutions. Several authors have demonstrated aerodynamic shape optimization with the continuous adjoint approach [2, 3, 4]. For the discrete adjoint approach, the adjoint method is applied to the set of discretized flow governing equations instead. The gradient produced by the discrete adjoint is exact in the discrete sense and can thus be verified with great precision using the complex-step method [5]. The discrete adjoint approach is also widely used in aerodynamic shape optimization [6, 7, 8, 9, 10, 11, 12]. The implementation of either continuous or discrete adjoint methods in a complex CFD code remains a challenging time-consuming task. The derivatives involved in the adjoint formulation for the RANS equations are

*PhD Candidate, Department of Aerospace Engineering, University of Michigan, AIAA Student Member
†Postdoc Research Fellow, Department of Aerospace Engineering, University of Michigan, AIAA Student Member
‡Masters Graduate, University of Toronto Institute for Aerospace Studies
§Associate Professor, Department of Aerospace Engineering, University of Michigan, AIAA Associate Fellow

often difficult to derive and require the manipulation of the governing equations. One way to tackle this difficulty is to use automatic differentiation (AD), either by differentiating the entire code [13, 14], or by selectively differentiating the code to compute the partial derivatives required by the adjoint method [15]. In this paper, we extensively use the latter approach to compute the partial derivative terms for a discrete adjoint of the RANS equations. The one-equation Spalart–Allmaras (SA) turbulence model [16] is also differentiated. Simplifications, such as neglecting the turbulent contributions, can be made in the formulation. The accuracy of the frozen-turbulence assumption is assessed. In addition, a coloring acceleration technique is also applied to speed up the construction of partial derivatives in the adjoint. The last section of this paper presents the verification and benchmark of the proposed adjoint implementation with two cases: flow over a bump, and an ONERA M6 wing.

II. Methodology

To develop the RANS adjoint, it is necessary to have a thorough understanding of the governing equations and the flow solvers, such as the size of stencils, the vector of state variables, the call sequences, etc. In this section, we discuss the backgrounds, methods, and tools that are involved in the formulation and implementation of the RANS adjoint.

A. Flow Governing Equations

The RANS equations are a set of conservation laws that relates mass, momentum, and energy in a control volume. To simplify the expressions, the RANS equations, (1) are demonstrated in 2-dimension.

$$\frac{\partial w}{\partial t} + \frac{1}{A} \oint F_i \cdot \hat{n} dl - \frac{1}{A} \oint F_v \cdot \hat{n} dl = 0 \tag{1}$$

The state variable w, inviscid flux F_i and viscous flux F_v are defined as follow.

$$w = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix} \tag{2} \qquad F_{i_1} = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ (E+p)u_1 \end{bmatrix} \tag{3} \qquad F_{v_1} = \begin{bmatrix} 0 \\ \tau_{11} \\ \tau_{12} \\ u_1\tau_{11} + u_2\tau_{12} - q_1 \end{bmatrix} \tag{4}$$

The shear stress and heat flux depends on both the laminar viscosity μ and the turbulent eddy viscosity μ_t, as shown in Equations (5) and (6).

$$\tau_{11} = (\mu + \mu_t)\frac{M_\infty}{Re}\frac{2}{3}(2u_1 - u_2) \tag{5}$$

$$q_1 = -\frac{M_\infty}{Re(\gamma - 1)}(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t})\frac{\partial a^2}{\partial x_1} \tag{6}$$

The laminar viscosity is determined by Sutherland's law. The turbulent eddy viscosity can be updated with turbulence models. In this paper, we use SA turbulence model and it is solved in a segregated fashion to update the turbulent eddy viscosity. When solving for the main flow variables at each iteration, the turbulence variables are frozen, and vice versa.

B. CFD Solver

The flow solver used is Stanford University multiblock (SUmb) [17] solver. SUmb is a finite-volume, cell-centered multiblock solver for the compressible Euler, laminar Navier–Stokes, and RANS equations with steady, unsteady, and time-periodic temporal modes. It provides options for a variety of turbulence models with one, two, or four equations, and options to use adaptive wall functions. Central difference augmented with artificial dissipation is used for the discretization of the inviscid fluxes. Viscous fluxes use central discretization. The main flow is solve with an explicit multi-stage Runge–Kutta method accelerated with geometrical multi-grid, pseudo-time stepping and implicit residual smoothing. The segregated SA turbulence equation is iterated with Diagonally Dominant Alternating Direction Implicit (DDADI) method. Mader *et al.* [15] previously developed a discrete adjoint method for the Euler equations using reverse mode AD. This paper extends the previous adjoint implementation to the steady Euler, laminar

and RANS equations and introduces a simpler forward mode AD approach. This technique requires very little modified to the original code.

C. RANS Automatic Differentiation Adjoint

High fidelity aerodynamic shape optimization with a large number of design variables requires an efficient gradient calculation. Traditional methods, such as finite-difference, are straightforward to implement, but are inefficient for large-scale optimization problems and are subject to subtractive cancellation error. The complex-step method alleviates the errors resulting from subtractive cancellation and can provide machine-accurate gradients, but similar to finite differencing, is inefficient for a large number of design variables. The total number of function evaluation scales with the number of the design variables. Thus, for optimization problems with a large number of design variables, the cost of one complete derivative computation can be on the order of hundreds or thousands flow solutions, which is generally prohibitive when using high-fidelity models. For the adjoint method, cost of the derivative computations is nearly independent of the number of design variables and scales only with the number of functions of interest, which is generally much smaller than the number of design variables.

1. *Adjoint Formulation*

For a CFD solver, the discrete adjoint equations can be expressed as follows.

$$\left[\frac{\partial \mathcal{R}}{\partial w}\right]^T \Psi = -\frac{\partial I}{\partial w}, \tag{7}$$

where $\mathcal{R}$ is the residual of the computation, w is the state variables, I is the function of interest, and Ψ is the adjoint vector. We can see that the adjoint equations do not involve the design variable x. For each function of interest, the adjoint vector only needs to be computed once, and it is valid for all design variables. Once the adjoint vector is computed, the total derivatives can be obtained using Equation (8).

$$\frac{dI}{dx} = \frac{\partial I}{\partial x} - \frac{\partial I}{\partial w}\left[\frac{\partial \mathcal{R}}{\partial w}\right]^{-1}\frac{\partial \mathcal{R}}{\partial x} = \frac{\partial I}{\partial x} + \Psi^T\frac{\partial \mathcal{R}}{\partial x} \tag{8}$$

Partial derivatives in the equations represent an explicit dependence that do not require convergence of the residual. In the case of computational fluid dynamics (CFD), by using the adjoint method, the cost of the total derivatives of any number of design variables can be in the order of or less than the cost of one single CFD solution. Only with the efficient gradient calculation, large-scale engineering optimization problems can be solved within a reasonable time.

2. *Automatic Differentiation Adjoint*

With the adjoint formulation, there is still one challenge — computing the partial derivatives efficiently. One can naïvely use finite difference or complex-step to compute these partial derivatives. However, the prohibitive computational cost from using those methods would completely defeat the purpose of the adjoint method. The partial derivatives can also be derived analytically by hand, but such derivation is non-trivial for a complex CFD solver and typically requires a lengthy development time for the adjoint.

In order to counter those disadvantages, Mader and Martins [15] proposed the ADjoint approach. The main idea is to utilize automatic differentiation techniques to compute partial derivative terms for the adjoint method. The automatic differentiation approach, also known as computational differentiation or algorithmic differentiation, relies on a tool to perform source code transformation on the original solver to create the capability of computing derivatives. The method is based on a systematic application of the chain rule to each line of the source code. There are two modes in automatic differentiation: forward mode and reverse mode. The forward mode propagates the derivatives along the original code execution path. The reverse mode first follows the original code execution path and in the *store-all* approach stores all the intermediate variables. The original code execution path source code is then re-run in the *reverse* execution order and the stored intermediate variables are used in the linearization of line of code.

We use the following example to demonstrate the underlining methodology of forward and reverse mode automatic differentiation.

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1) \tag{9}$$

This function can be written as the sequence of elementary operations on the intermediate variables q_i resulting in the following sequence,

$$\begin{aligned} q_1 &= x_1 \\ q_2 &= x_2 \\ q_3 &= q_1 q_2 \\ q_4 &= \sin(q_1) \\ q_5 &= q_3 + q_4 \end{aligned} \tag{10}$$

This sequence is then used to compute the derivative of Equation (9).

FORWARD MODE Forward mode AD is simpler and more intuitive of the two approaches. In Equation (9), assuming that x_1 and x_2 are independent inputs, the rules of differentiation are applied to the sequence in Equation (10) as follow.

$$\begin{aligned} \Delta q_1 &= \Delta x_1 \\ \Delta q_2 &= \Delta x_2 \\ \Delta q_3 &= \Delta q_1 q_2 + q_1 \Delta q_2 \\ \Delta q_4 &= \cos(q_1) \Delta q_1 \\ \Delta q_5 &= \Delta q_3 + \Delta q_4 \end{aligned} \tag{11}$$

Once the sequence and its corresponding gradients for the function in Equation (9) are known, x_1 and x_2 can be seeded to determine the gradient of the function. Since x_1 and x_2 are assumed to be independent inputs, seeding each input independently means to set the variation to one while the other remains zero such that $\Delta x_1 = [1, 0]$ and $\Delta x_2 = [0, 1]$. Forward mode AD sweeps over the computations in Equation (11) twice, once for each input, and adds the separate derivative evaluations as follows.

$$\begin{aligned} \Delta f(x_1, x_2) &= [f_{x_1}, f_{x_2}] \\ &= \Delta q_{5_{x1}} + \Delta q_{5_{x2}} \\ &= (\Delta q_{3_{x1}} + \Delta q_{4_{x1}}) + (\Delta q_{3_{x2}} + \Delta q_{4_{x2}}) \\ &= ((1)q_2 + q_1(0) + \cos(q_1)(1)) + ((0)q_2 + q_1(1) + \cos(q_1)(0)) \\ &= q_2 + \cos(q_1) + q_1 \\ &= x_1 + x_2 + \cos(x_1) \end{aligned} \tag{12}$$

This is the expected derivative for the original function in Equation (9), and can be written in a more general format by considering the general sequence $q = (q_1, ..., q_n)$. Considering m input variables and p output variables, the sequence becomes $q = (q_1, ...q_m, q_{m-p+1}, ..., q_n)$. For $i > m$, each q_i must have a dependence on some member of the sequence prior to i. If $k < i$, the entry q_i of the sequence must depend explicitly on q_k. The forward mode can then be written as the chain rule summation as shown in [18],

$$\Delta q_i = \sum \frac{\partial q_i}{\partial q_k} \Delta q_k, \tag{13}$$

where $i = m + 1, ..., n$ and $k < i$. The forward mode AD evaluates the gradients of the intermediate variables first such that $\Delta q_1, ..., \Delta q_{i-1}$ are known prior to the evaluation of Δq_i. It is easy to see that the forward mode builds up the derivative information as it progresses forward through the algorithm, producing the derivative information for all of the output variables with respect to a single seeded input variable.

REVERSE MODE The reverse mode, though less intuitive, is dependent only on the number of outputs. With reference to the previous example, it is easier to understand reverse mode AD by examining the partial derivatives of f. Consider the following,

$$\frac{\partial q_5}{\partial q_1} = \frac{\partial q_5}{\partial q_1}\frac{\partial q_1}{\partial q_1} + \frac{\partial q_5}{\partial q_2}\frac{\partial q_2}{\partial q_1} + \frac{\partial q_5}{\partial q_3}\frac{\partial q_3}{\partial q_1} + \frac{\partial q_5}{\partial q_4}\frac{\partial q_4}{\partial q_1}, \tag{14}$$

where q_5 represents the single output f. The reverse mode first runs a forward sweep to determine all of the intermediate values in the sequence. Then, starting with a single output variable, (q_5 in this case), the AD tool steps *backwards* through the algorithm to compute the derivatives in reverse order. The example, and the implementation of the sequence in Equation (14), produces the following final result.

$$\begin{aligned}
\frac{\partial q_5}{\partial q_5} &= 1 \\
\frac{\partial q_5}{\partial q_4} &= 1 \\
\frac{\partial q_5}{\partial q_3} &= 1 \\
\frac{\partial q_5}{\partial q_2} &= \frac{\partial q_5}{\partial q_3}\frac{\partial q_3}{\partial q_2} = (1)(q_1) \\
\frac{\partial q_5}{\partial q_1} &= \frac{\partial q_5}{\partial q_3}\frac{\partial q_3}{\partial q_1} + \frac{\partial q_5}{\partial q_4}\frac{\partial q_4}{\partial q_1} = (1)(q_2) + (1)(\cos(q_1)
\end{aligned} \tag{15}$$

where we have,

$$\begin{aligned}
\frac{\partial q_5}{\partial q_1} &= \frac{\partial f}{\partial x_1} = x_2 + \cos(x_1) \\
\frac{\partial q_5}{\partial q_2} &= \frac{\partial f}{\partial x_2} = x_1.
\end{aligned} \tag{16}$$

The advantage here is that only a single reverse sweep is required to evaluate the derivatives with respect to both x_1 and x_2. Should there be a greater number of inputs, which is typical in an aerodynamic shape optimization problem, a single forward sweep to accumulate the code list as well as a single reverse mode sweep is all that would be necessary to calculate the sensitivities for a single output.

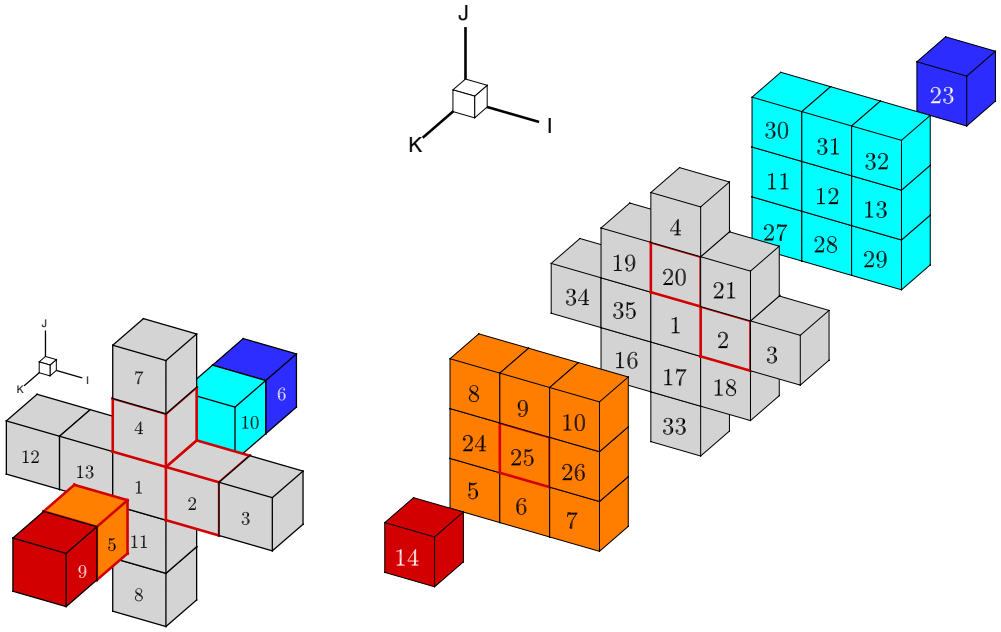
The disadvantage of the reverse mode is that the implementation is more complicated than the forward mode. The reverse mode was used in the original development of the ADjoint method and so is used as a benchmarking tool in the development of the forward mode ADjoint method. To avoid the high computational costs associated with using the forward mode of AD, a coloring method is used to accelerate the computation.

For the adjoint equations (7) and (8), the partial derivatives $\partial\mathcal{R}/\partial w$, $\partial I/\partial w$, $\partial I/\partial x$, and $\partial\mathcal{R}/\partial x$ are computed with forward automatic differentiation. The following sections explain the implementation of each partial derivatives in details. There are various automatic differentiation tools available including ADIC [19], ADIFOR [20], FADBAD++ [21], OpenAD/F [22], and TAPENADE [23]. The work presented in this paper uses TAPENADE to perform the task. TAPENADE is an automatic differentiation engine developed by the Tropics team at Institut National de Recherche en Informatique et Automatique and supports both forward and reverse modes.

3. *Computation of $\partial\mathcal{R}/\partial w$ and $\partial I/\partial w$*

To compute $\partial\mathcal{R}/\partial w$, there are three flux calculations involved: inviscid fluxes, artificial dissipation fluxes, and viscous fluxes. For Euler equations, the combined stencil is the current cell and the 12 adjacent cells in each of the three dimensions — a total of 13 cells, as shown in Figure 1a. The Laminar and RANS equations have much larger stencils due to the nodal averaging procedure used in the viscous fluxes. The RANS stencil is a dense 3x3x3 block around the center cell plus additional 6 adjacent cells in each direction, as shown in Figure 1b. The size and the shape of the stencil is important for the coloring acceleration techniques, which is discussed further in Section E. For RANS equations, the state vector w contains the five main flow state variables and one turbulence variable for the one-equation SA model. Therefore, the residual computation for the SA equation also need to be included in the automatic differentiation.

The contribution of the turbulence to the main flow residual is included via the turbulence variable. The frozen-turbulence assumption can be made by neglecting the turbulence contribution to the main flow. Since $\partial\mathcal{R}/\partial w$ is a square matrix, in principle both forward and reverse modes would require similar number of function calls to form the matrix. However, forward mode is more intuitive and has lower overhead cost and for forming $\partial\mathcal{R}/\partial w$, the forward mode is faster than the reverse mode in practice. $\partial\mathcal{R}/\partial w$ is stored in transpose form in a block compressed sparse row matrix format.



a) Euler flux stencil: 13 cells **b) RANS flux stencil: 33 cells**

Figure 1. Euler and RANS flux Jacobian stencil

Special care must be taken for the computation of $\partial I/\partial w$ with forward mode AD. If the routine to compute I, which we will assume consists of integrated forces or moments on wall boundary, is simply included with the residual evaluation, all cells near the surface that influence the force evaluation on the wall would have to be perturbed independently and the advantage of the graph coloring approach described in Section E would be nullified.

To enable the evaluation of $\partial I/\partial w$ simultaneously with $\partial\mathcal{R}/\partial w$ it is necessary to evaluate individual forces and moments at each cell, not just the overall sum. Stencils for individual force and moment computations are compact. For both Euler and RANS cases this force stencil is smaller than the corresponding residual stencil. For the linear pressure extrapolation wall boundary condition, the Euler force stencil has only two cells: the cell on the surface and the cell above. The RANS force stencil consists of a 3x3 patch on the surface and one layer above, with a total of 18 cells. Both Euler and RANS force stencil can be packed inside the respective flux Jacobian stencils. Once the individual forces are resolved, their contribution to the chosen objective, I, is evaluated and the correct contribution can be added to $\partial I/\partial w$.

4. Computation of $\partial\mathcal{R}/\partial x$ and $\partial I/\partial x$

The calculations of $\partial\mathcal{R}/\partial x$ and $\partial I/\partial x$ depend on the design variables. For aerodynamic shape optimization, we are generally interested in geometric design variables, such as airfoil profile, wing twists, etc, and flow condition design variables, such as Mach number, angle of attack, side-slip angle etc. The partials that involve flow design variables are relatively straight-forward. Each flow design variables are seeded and forward mode AD is used to obtain the residual and objective partial derivatives. No coloring scheme is necessary, since only one pass of the residual routine is needed

for each flow design variables.

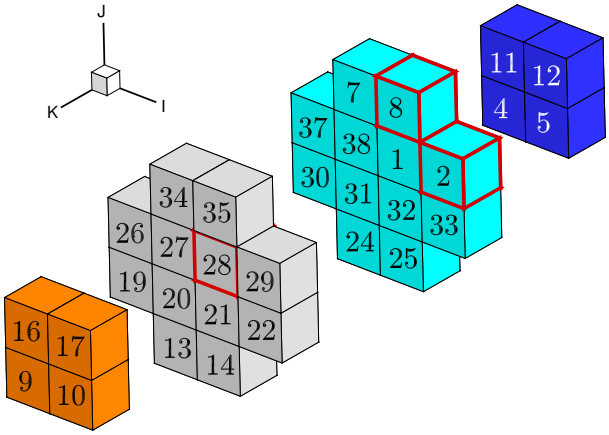


Figure 2. Euler spatial stencil: 32 cells

The partial derivatives for the geometric design variables require careful implementation. In an effort to modularize codes, SUmb does not require the specific information about the geometric design variables. Instead, we calculate $\partial\mathcal{R}/\partial x_{pt}$ and $\partial I/\partial x_{pt}$, where x_{pt} includes all the nodes in the CFD mesh. We use a separate utilities to perform the mesh deformation sensitivity calculation [24] and manipulation of the surface geometry. The surface geometry is manipulated using the free-form deformation (FFD) approach. It is then transform the spatial derivatives into the derivatives with respect to the control points of the FFD volume.

To compute $\partial\mathcal{R}/\partial x_{pt}$ and $\partial I/\partial x_{pt}$, we again use forward mode AD. The choice of forward mode may not be obvious here. The benefit of using forward mode AD is that the same SUmb residual routine can be used in both state and spatial derivatives, resulting a much less demanding implementation and fewer modifications. Similar to the state partial derivatives, both $\partial\mathcal{R}/\partial x_{pt}$ and $\partial I/\partial x_{pt}$ can be computed at the same time. The center of the stencils for the spatial derivatives is a node instead a cell. Figure 2 shows the Euler $\partial\mathcal{R}/\partial x_{pt}$ stencil, with a total of 32 cells. The corresponding RANS stencil is a dense 4x4x4 block containing 64 cells. The Euler stencil for $\partial I/\partial x_{pt}$ is a 4x4 surface patch, while the RANS one includes one addition layer above. As we can see, both spatial force derivatives can also be fitted inside the spatial residual stencils.

D. Adjoint Solution Method

We use a preconditioned GMRES [25] algorithm in PETSc Portable, Extensible Toolkit for Scientific Computation) [26, 27, 28] to solve the adjoint system. PETSc is a suite of data structures and routines for the scalable parallel solution of scientific applications modelled by partial differential equations. The system is preconditioned with restrictive additive Schwartz method and incomplete LU (ILU) factorization on each sub-domain. We found that GMRES with approximate preconditioner produced using a first-order approximation is very effective with Euler adjoint. The RANS adjoint, especially without the frozen-turbulence assumption, is considerably more stiff than the Euler adjoint with the problem more prominent at high Reynolds number. A stronger preconditioner, such as full $\partial\mathcal{R}/\partial w$ Jacobian as preconditioner, may be necessary.

E. Coloring Acceleration Techniques

As previously noted, a naïve approach for computing $\partial\mathcal{R}/\partial w$ and $\partial\mathcal{R}/\partial x$ would require a total of $N_{\text{state}} + 3 \times N_{\text{nodes}}$ evaluations, where N_{state} and N_{nodes} are the total number of cells and nodes respectively. In this approach each column (or row of the transposed Jacobian) is computed one at a time. If we however, exploit the sparsity structure of $\partial\mathcal{R}/\partial w$ and $\partial\mathcal{R}/\partial x$, it is possible to fully populate the Jacobians with far fewer function evaluations.

The general idea is to determine groups of independent columns of the Jacobian. A group of columns is considered independent if no row contains more than one nonzero entry. This allows a group of independent columns to be evaluated simultaneously. The process of determining which columns are independent is known as graph coloring. The determination of an optimal (smallest) set of *colors* for a general graph is quite challenging. For unstructured grids, a greedy coloring scheme can be used resulting is a satisfactory number of colors[29].

For structured grids with regular repeating stencils, the graph coloring problem is substantially simpler [30]. Consider the 13-cell stencil for the Euler residual evaluation shown in Figure 1a. It is clear at least 13 colors will be required. Determining the optimum graph coloring for this case is is equivalent to finding a three dimensional packing sequence that minimizes the unused space between stencils. Fortunately, for this stencil, a perfect packing sequence is possible and precisely 13 colors are required. A three-dimensional view of the stencil packing is shown in Figure 3. For the 33 cell RANS stencil, it is not possible to perfectly pack the stencils. We have, however, found a coloring scheme that requires only 35 colors, shown in Figure 4. To assign the coloring number mathematically to each cell, simple formulae can be derived emplying the remainder function $mod(m, n)$. m is a function determined by the numbered stencil and n is the total number of colors required to populate the matrix. Equations (17) through (19) show the coloring function for the Euler and RANS stencils. These optimal graph colorings reduce the number of forward mode AD perturbations to a fixed constant, independent of the mesh size.

$$C_{\text{Euler, state}}(i,j,k) = \mod(i + 3j + 4k, 13) \tag{17}$$

$$C_{\text{Euler, spatial}}(i,j,k) = \mod(i + 7j + 27k, 38) \tag{18}$$

$$C_{\text{RANS, state}}(i,j,k) = \mod(i + 19j + 11k, 35) \tag{19}$$

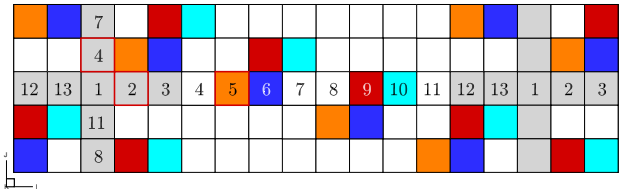


Figure 3. Euler state coloring patterns with 13 colors

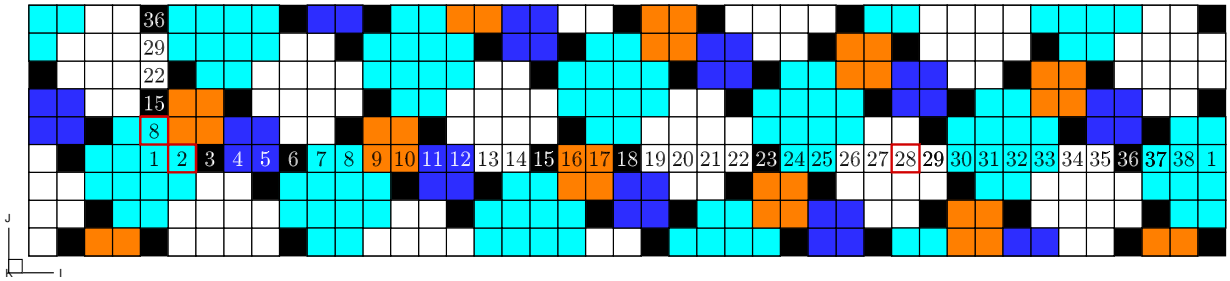


Figure 4. Euler spatial coloring patterns with 38 colors

III. Accuracy Verification

A flow over a bump case is chosen as the test case to verify Euler, Laminar NS, and RANS adjoint solutions. The computational mesh for the test case used is shown in Figure 6. It is a single block mesh with 3 072 cells. The side walls of the channel use symmetry boundary conditions. The inflow and outflow faces and the upper wall is set to

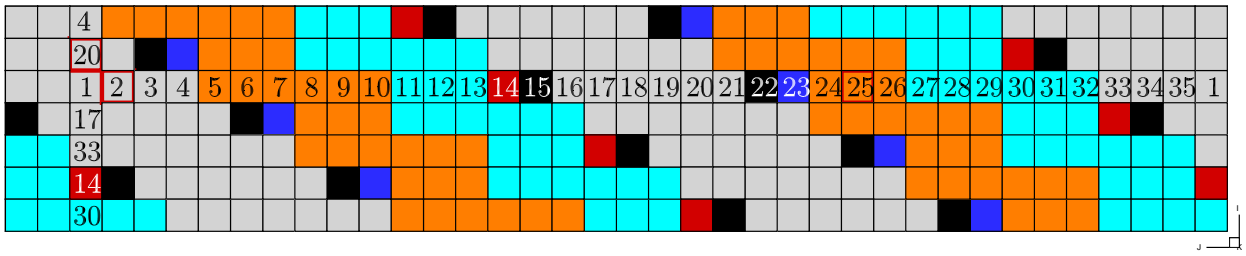


Figure 5. RANS state coloring patterns with 35 colors

far-field conditions. The bottom wall is deformed with a sinusoidal bump to create a reasonable variation in the flow, which has solid wall boundary condition.

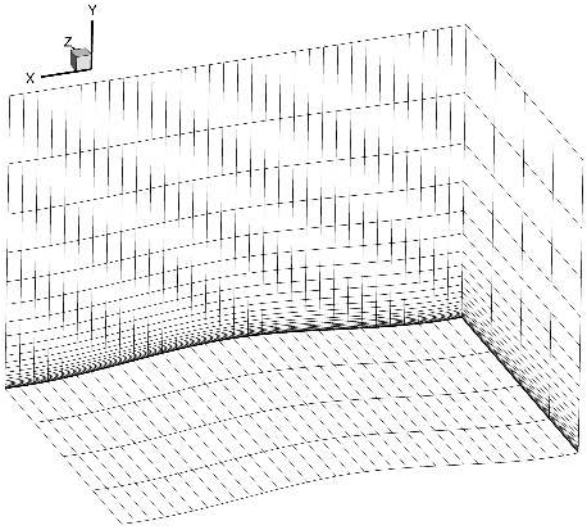


Figure 6. Volume mesh for bump verification case

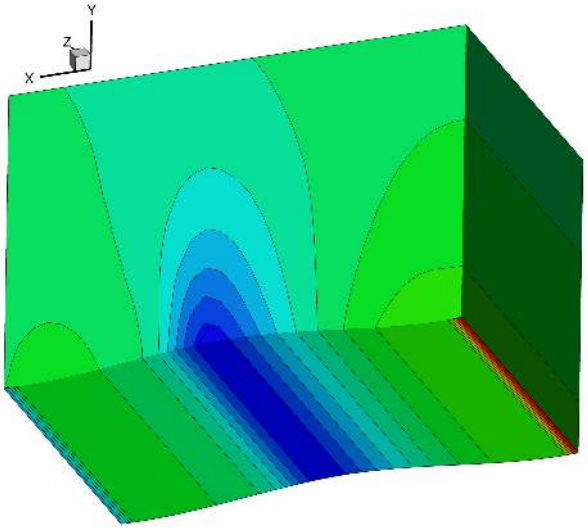


Figure 7. C_p distribution of the RANS solution

Both the flow solution and the adjoint solutions are converged to a tolerance of $\mathcal{O}(10^{-12})$. We verify the adjoint accuracy with complex-step derivative approach given by: [5],

$$\frac{dF}{dx} = \frac{Im[F(x+ih)]}{h}, \tag{20}$$

where h is the complex step length. An imaginary step of $10^{-40}j$ is chosen as the perturbation.

Euler, Laminar NS, and RANS with both frozen-turbulence and full-turbulence are benchmarked against complex-step. We choose Mach number 0.8 and Reynolds number 10 million for the flow condition. Figure 7 shows the C_p distribution of the RANS solution. Two objective functions, C_D and C_L, are used for verification. For the design variables, we choose Mach number to verify the aerodynamic derivatives and a point on the surface to verify the spatial derivatives. The results are summarized in Table 1 to Table 4.

We can see that the resulting derivatives match with complex-step solutions. The full-turbulence aerodynamic derivatives matches significantly better than the frozen-turbulence ones. Due to the complexity of the wall distance

Derivatives	Complex-Step	Adjoint	Difference
dC_D/dM	0.652989**53**	0.652989**64**	1.5E-8
dC_L/dM	1.678545**80**	1.678545**72**	4.9E-9
dC_D/dx	0.152323**769**	0.152323**071**	9.6E-8
dC_L/dx	0.011324**974**	0.011324**975**	4.6E-6

Table 1. Accuracy validations of the Euler adjoint

Derivatives	Complex-Step	Adjoint	Difference
dC_D/dM	0.655985**401**	0.655985**467**	1.0E-7
dC_L/dM	1.819804**777**	1.819804**889**	6.1E-8
dC_D/dx	0.011845**928**	0.011845**836**	7.7E-6
dC_L/dx	0.1453**07150**	0.1453**12443**	3.6E-5

Table 2. Accuracy validations of the Laminar NS adjoint

Derivatives	Complex-Step	Frozen-Turbulence Adjoint	Difference
dC_D/dM	0.673**453841**	0.673**684112**	3.4E-4
dC_L/dM	1.7**67928150**	1.7**72398147**	2.5E-3
dC_D/dx	0.009952**556**	0.009952**686**	1.3E-5
dC_L/dx	0.1**29946365**	0.1**30232663**	2.2E-3

Table 3. Accuracy validations of the frozen-turbulence adjoint

Derivatives	Complex-Step	Full-Turbulence Adjoint	Difference
dC_D/dM	0.6734538**41**	0.6734538**42**	1.1E-9
dC_L/dM	1.7679281**50**	1.7679281**53**	1.4E-9
dC_D/dx	0.0099**52556**	0.0099**49493**	3.1E-4
dC_L/dx	0.129**946365**	0.129**890985**	4.2E-4

Table 4. Accuracy validations of the full-turbulence adjoint

function in SA turbulence model, the wall distance computation is not linearized and is assumed constant in the turbulence model to simplify the automatic differentiation. Therefore, we see that the spatial derivatives have less accuracy than the aerodynamics derivatives for the full-turbulence adjoint.

IV. Results

To demonstrate the effectiveness of the RANS adjoint formulation for aerodynamic shape optimization, an example of lift constrained drag minimization of a transonic wing is presented. The particular test considered is the well-studied ONERA M6 wing [31]. This geometry has been studied by numerous authors [32, 33, 34, 35, 36, 8, 10] due to the simple, well defined geometry and the availability of experimental data.

The optimization problem considered is described below:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & C_D(x) \\ \text{subject to} \quad & C_L \geq C_L^* \\ & V \geq V_0 \\ & t_i \geq 1, \; i = 1, \ldots, 21. \end{aligned}$$

The objective is to reduce the drag coefficient while maintaining a specified lift coefficient, $C_L^* = 0.271$. The lift coefficient is based on a reference area of 0.75296 m^2. Additional geometric constraints in the form of volume and thickness constraints are also used and are described in section B.

A. Verification and Grid Refinement Study

Before optimizations were carried out, a grid refinement study and comparison with experimental data was made. The external flow condition for the experimental data and subsequent optimizations is:

$$M = 0.8395 \qquad Re = 11.72 \times 10^6 \qquad \alpha = 3.06^\circ \tag{21}$$

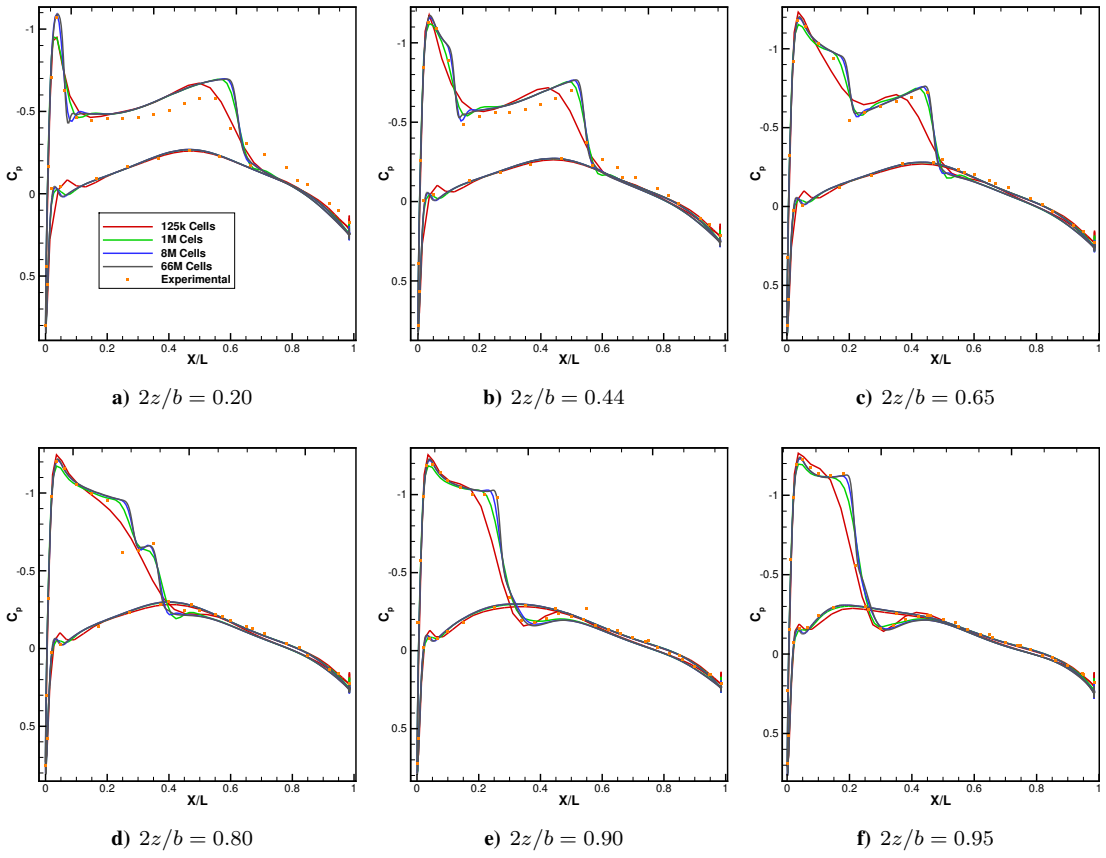
A sequence of 4 uniformly refined grids, labeled L1 through L4, were generated with grid sizes ranging from 129 thousand cells to over 66 million cells. The grids are generated using an in-house 3D hyperbolic mesh generator. The L2, L3 and L4 grids are all computed directly from their respective surface meshes while the L1 grid is obtained from the L2 grid by removing every other mesh node. An additional algebraic C-O topology Euler mesh was also generated for the purposes of comparing optimization results obtained with Euler and RANS analysis methods. The Euler grid has approximately the same number of cells as the L2 RANS mesh to facilitate comparison between the computational cost for roughly equivalent Euler and RANS optimizations. A description of all grids used in this work is given in Table 5. For all grids the far-field surface is located approximately 100 Mean Aerodynamic Chords away from the body.

Table 5. Mesh sizes

Grid	Cells	Surface Cells	Off-wall Cells	Off-wall Spacing	$y^+_{\max}$
RANS L1	129 024	4 032	32	3.0×10^{-6}	1.50
RANS L2	1 032 192	16 128	64	1.5×10^{-6}	0.67
RANS L3	8 257 536	54 512	128	0.75×10^{-6}	0.35
RANS L4	66 060 288	258 048	256	0.375×10^{-6}	0.18
Euler	1 044 480	18 432	40	3.0×10^{-4}	–

The comparison of the experimental data with each of the four RANS grids is given in Figure 8. Overall, the flow solver has fairly accurately predicted the coefficient of pressure at each span-wise section. As expected, the finer grid resolutions do a better job of resolving both the location and strength of the shocks, although there is little discernible difference between the L3 and L4 grids. We believe the discrepancy between the computed and experimental data near the root can be attributed to wind tunnel effects and the splitter plate used in the physical setup that are not modelled

computationally. A second discrepancy appears at the $2z/b = 0.90$ section where it is clear the position of the leading edge shock is displaced rearward as compared with the experimental data. This computational behaviour is however, consistent with other results obtained on highly refined grids [37]. A possible explanation is due to small aeroelastic deformation of the physical model which not present in the computational model.



a) $2z/b = 0.20$ **b)** $2z/b = 0.44$ **c)** $2z/b = 0.65$

d) $2z/b = 0.80$ **e)** $2z/b = 0.90$ **f)** $2z/b = 0.95$

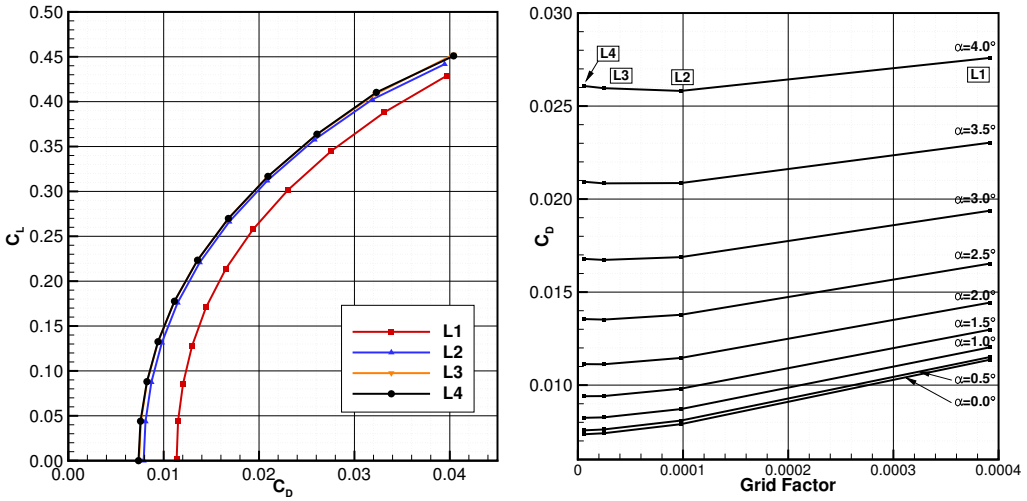
Figure 8. C_p **contours for each grid refinement level compared with experimental data**

Additionally, an angle of attack sweep from 0° to 5° was run for each grid level to generate drag polars at the design Mach number of $M = 0.8395$. The polar is shown in Figure 9a. It is clear that the coarsest grid, L1, is not sufficiently resolved for accurate drag prediction. Conversely, the L3 and L4 grids are nearly indistinguishable from each other except at the higher lift coefficients. While the discrepancy between the L2 and L4 grids is clearly visible it is fairly small and this level of refinement offers significantly computational computational savings compared to the L3 and L4 grids for the purposes of optimization.

Drag convergence curves for various angles of attack are given in Figure 9b. The x-axis scale is given in terms of the *Grid Factor* which is defined as $N_{\text{cell}}^{-2/3}$. In general, the total drag coefficient decreases with increasing grid size. However, between the L3 and L4 grids at higher angles of attack, the trend reverses and the larger grids see a slight increase in drag. The root cause of this behaviour is not known and warrants further investigation.

B. Geometric parametrization, Constraints and Grid Movement

The geometric manipulation of the initial geometry is carried out using the Free Form Deformation (FFD) volume approach [38]. The design variables, x, are used to perturb the control points on a 3-dimensional parametric B-spline



a) Drag polars for four levels of uniform refinement **b) Grid convergence for C_D for various angles of attack.**

Figure 9. Polar and grid convergence for each grid level.

volume which in turn, perturbs the coordinates of the CFD surface mesh embedded parametrically inside. The design variable vector consists of 6 twist values that twist each of the six span-wise planes of control points, and 144 shape variables. Each shape variable perturbs individual coordinates of the FFD in the y (normal) direction. Note that since the root twist is allowed to vary, angle of attack is *not* a variable and the optimizations are carried out a fixed angle of attack of $3.06°$.

To ensure a well-posed optimization problem, several additional geometric constraints are also employed. The internal volume of the wing is constrained to be greater than or equal to its initial value. A total of 21 thickness constraints are used; 10 distributed along the 15% chord line, 10 distributed along the 99% chord line and a single additional constraint near the mid-chord position at the wingtip. The leading edge constraints prevent a sharp leading edge from forming and the trailing edge constraints prevent a reduction in the thickness of the trailing edge.

A view of the initial wing geometry, the FFD volume box and the thickness constraints are given in Figure 10. Note that the distribution of control points on the FFD are not uniform in the chord-wise direction. This clustering around the leading edge was used to ensure the optimizer is given sufficient geometric freedom to eliminate the leading edge shock present on the baseline design. Further, the blunt trailing edge of physical model is retained for the RANS simulations. A sharp trailing edge modification is used for the Euler grid.

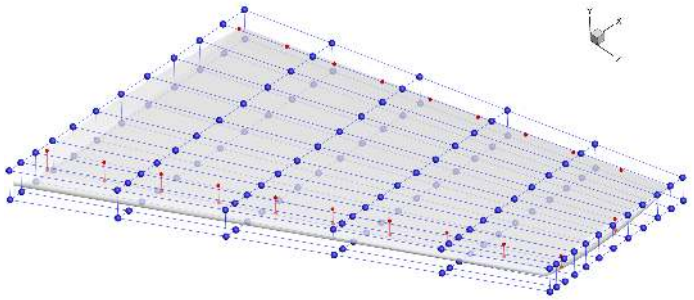
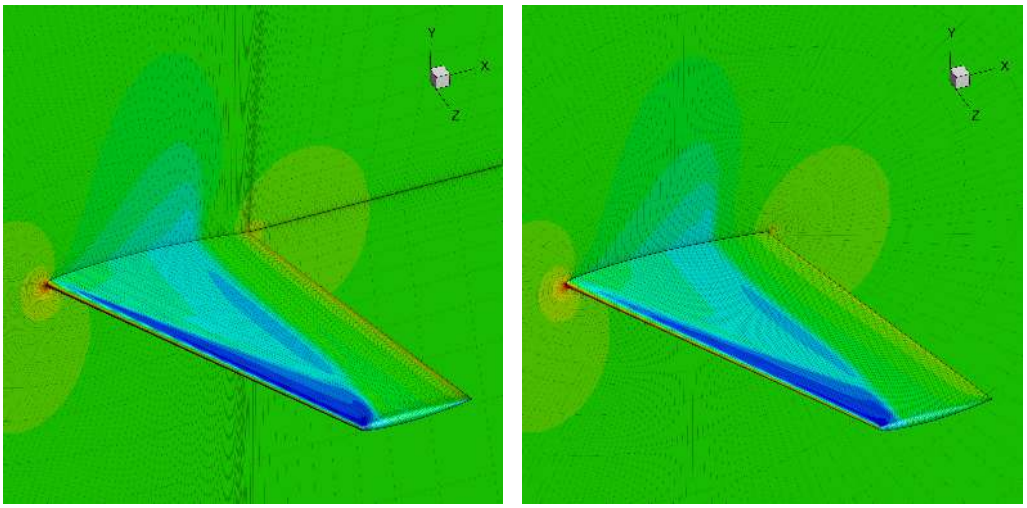


Figure 10. FFD control points (blue spheres) and thickness constraints (red lines).

The grids are deformed using a hybrid linear-elasticity algebraic mesh deformation algorithm previously developed

by the authors [38]. The mesh sensitivities required for the $\psi^T \frac{\partial A}{\partial x}$ computation are computed using Reverse Mode AD and a mesh adjoint equation.

A view of the surface mesh, symmetry plane and flow solution for the Euler and RANS grids are given in Figure 11.



a) Euler grid b) RANS grid

Figure 11. Computational grids used for Euler and RANS analysis. C_p contours are shown for $M = 0.8395$ and $\alpha = 3.06°$.

C. Optimization Algorithm

Due to high computational cost of the CFD solver, it is critical to choose an efficient optimization algorithm that requires a reasonably low number of function calls. Gradient-free methods, such as genetic algorithms, have a higher probability of getting close to the global minimum for cases with multiple local minima. However, slow convergence and a large number of function calls would make gradient-free aerodynamic shape optimization infeasible with current computational resources. Therefore, we use gradient-based optimizers combined with adjoint gradient evaluations to achieve an efficient optimization process. For a large number of design variables, the use of gradient-based optimizers is advantageous. We use a Python-based optimization package, *pyOpt* [39], to interface with CFD and adjoint solvers. We choose a gradient-based optimization algorithm, Sparse Nonlinear OPTimizer (SNOPT) [40], as the optimizer. SNOPT is a sequential quadratic programming (SQP) method, designed for large-scale nonlinear optimization problems with thousands of constraints and design variables. It uses a smooth augmented Lagrangian merit function and the Hessian of the Lagrangian is approximated using a limited-memory quasi-Newton method.

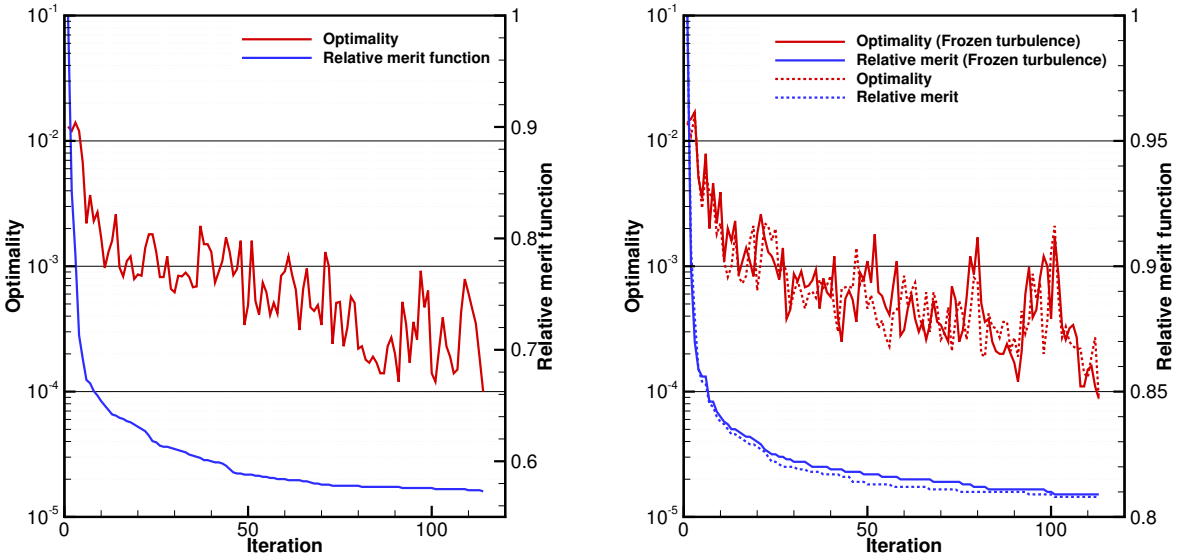
D. Computational Resources

The three optimizations are performed on a massively parallel supercomputer. Different processor counts are chosen for the Euler and RANS optimizations in an effort to keep the wall time of each optimization within a one day turnaround. Due to the lower computational and memory requirements for the Euler analysis, this optimization uses 32 processors while the two RANS optimizations use 88 processors.

E. Optimization Results

Three optimizations are considered: a RANS optimization employing the frozen turbulence assumption for the adjoint, a RANS optimization with the turbulence model linearization and an Euler optimization. An effort is made to compare the computational cost and accuracy of these differing approaches.

Firstly, we examine the convergence history of the optimizations, given in Figure 12. All optimization are converged to an optimality tolerance of 1×10^{-4} and take approximately 112 major iterations to reach this level of convergence.



a) Convergence for Euler optimization **b) Convergence for RANS optimization**

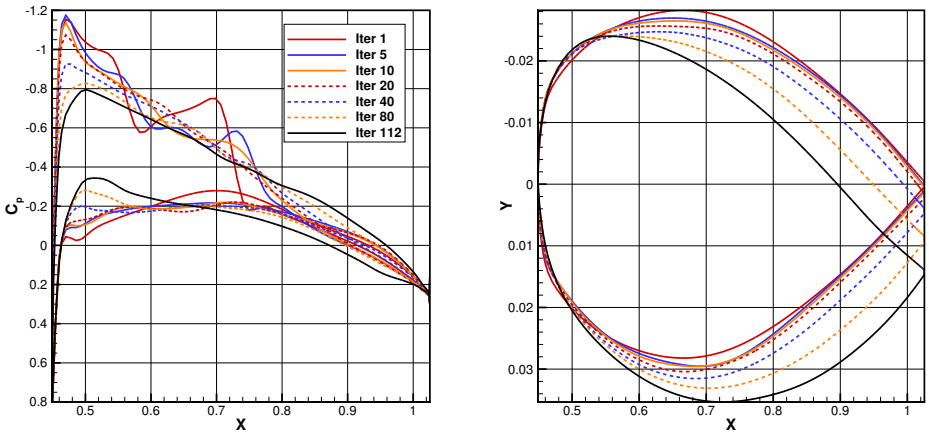
Figure 12. Convergence history of optimality and relative merit function.

Qualitatively, the merit function convergence for each optimization is similar: There is a very rapid decrease in C_D at the beginning of the optimization followed by much slower decreases as the optimization progresses. The first phase of the optimization involves the weakening of the two upper surface shocks. Referring to Figure 13a, by the 10^{th} iteration, the shocks have been entirely smoothed due to shape changes and this is responsible for the majority of the drag reduction. The second phase involves minor adjustments to the shape and modifications to the twist distribution. During this phase, an increase skin friction drag is traded for lower pressure and an overall decrease in the objective function. It is clear from Figure 13b, that the majority of the wing twist present in the optimized design is added towards the end of the optimization, which is use primarily to reduce the induced drag of the wing.

We now examine the cross sectional C_p contours of each of the three optimized designs. The same six span-wise locations as used in the experimental verification are reused. Figure 17b shows the contours for the baseline design, the frozen turbulence RANS optimization and the full RANS optimization. Figure 17a shows the baseline design, the optimized design and the optimized Euler design analyzed using RANS analys. For this last case, the geometric design variables from the Euler optimization were used to perturb the L2 RANS grid and then obtain a solution at C_L^*.

Generally, the C_p contours for the two RANS optimization are similar. However, there are some slight differences, with the full RANS design resulting in somewhat smoother C_p contours. The largest discrepancy is observed on the lower surface near the leading edge.

A breakdown of the pressure and skin friction drag components is given in Table 6. In addition to the drag from the optimization (Optimized (L2)), we also analyze the baseline design and optimized design using the L3 grid. The goal is to verify that the gains made during the optimization are realized on the finer grid. This is indeed the case; The total drag reduction on the L3 grid is nearly identical to that on the L2 grid, justifying our choice of the L2 grid for optimization. For comparison, we also analyze the Euler design using the L2 RANS grid. The FFD approach greatly



a) Evolution of C_p during the optimization

b) Evolution of cross section during the optimization

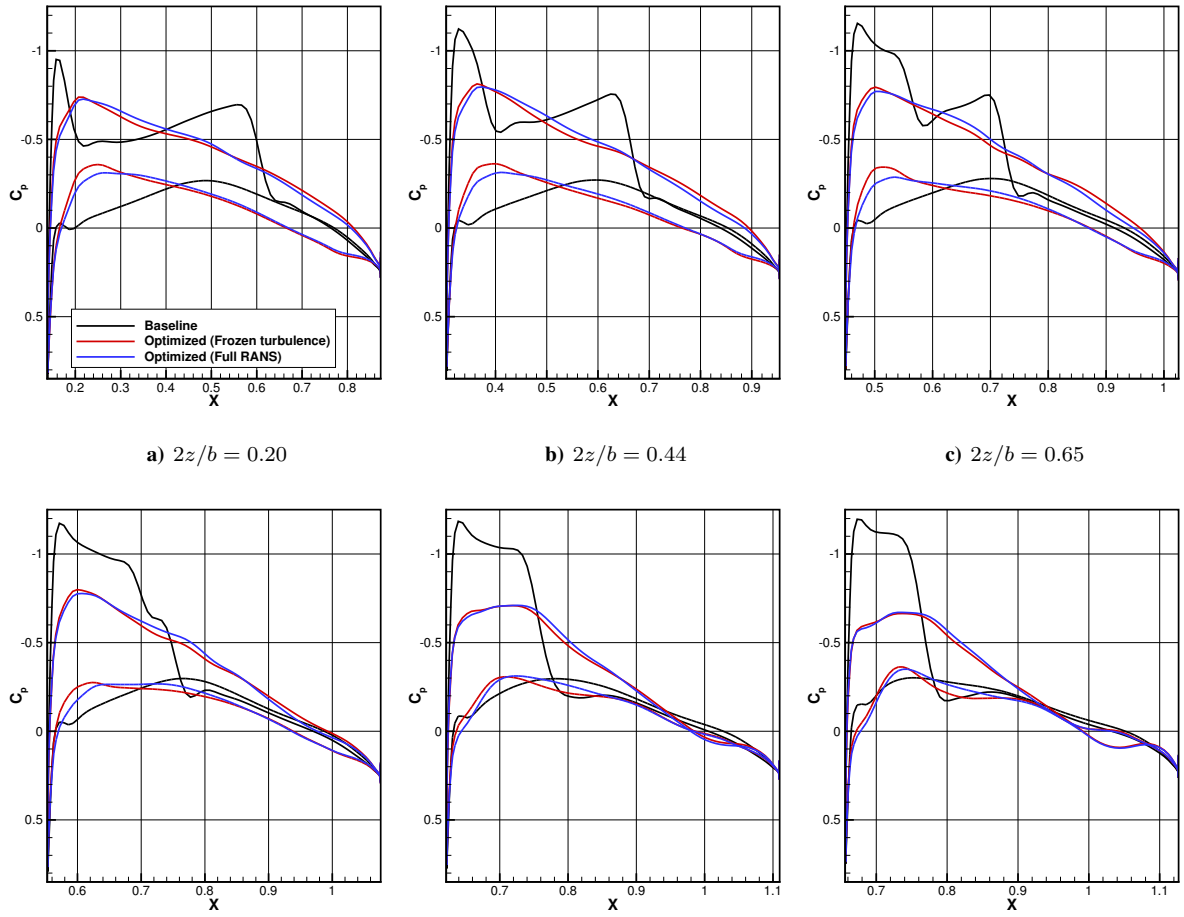
Figure 13. Evolution of C_p and cross section shape for section $2z/b = 0.65$.

facilities this exercise since the geometric design variables operate independently of the underlying mesh or surface topologies. Interestingly, the Euler optimized design shows remarkably good RANS performance, with the total drag coefficient only 3.7 counts higher than the RANS optimized design. Nevertheless, it worthwhile noting that this drag level was obtained by the RANS optimization after only 20 iterations, and corresponds to the initial optimization phase described previously. Most of the improvements small detailed shape and twist changes in the Euler optimization are evidently not realized in the viscous flow case.

Geometry	C_L	$C_{D_{\text{total}}}$	$\Delta C_{D_{\text{total}}}$	$C_{D_{\text{pressure}}}$	$\Delta C_{D_{\text{pressure}}}$	$C_{D_{\text{friction}}}$	$\Delta C_{D_{\text{friction}}}$
Baseline (L2)	0.2710	0.01725	–	0.01199	–	0.00526	–
Optimized (L2)	0.2710	0.01400	−0.00325	0.00847	−0.00343	0.00553	0.00027
Euler Design (L2)	0.2710	0.01437	−0.00288	0.00875	−0.00324	0.00561	0.00035
Baseline (L3)	0.2710	0.01687	–	0.01158	–	0.00529	–
Optimized (L3)	0.2710	0.01364	−0.00323	0.00816	−0.00342	0.00548	0.00019

Table 6. Drag break down for baseline and optimized designs on two mesh levels

A timing beak-down of the various components of each optimization is given in Figure 7. The *Miscellaneous* category accounts for the time required for initial setup time, I/O, geometric manipulation, total sensitivity calculations and the optimization algorithm. Since the Euler optimization used few processors, the *Processor Hours* row indicates the true computational cost of the respective optimization. While, the full RANS simulation converges to a slightly better optimum in the same number iterations, the computation cost of the full RANS optimization is significantly more. Referring to Table 7, the main increase in cost for the full RANS simulation is the increased cost of residual assembly due to the extra state to be perturbed and the additional cost of solving the adjoint system. The increase in the adjoint solving cost is twofold: The matrix-vector products and preconditioner application is more costly due to the larger number of non-zeros (a factor of $(6/5)^2 = 1.44$) and the systems require more GMRES iterations for convergence. This results in the Full RANS optimization requiring approximately 70% more CPU time that the frozen-turbulence assumption optimization. For the remainder of this section, for comparison, purposes we use the frozen turbulence optimization results.



a) $2z/b = 0.20$ **b)** $2z/b = 0.44$ **c)** $2z/b = 0.65$

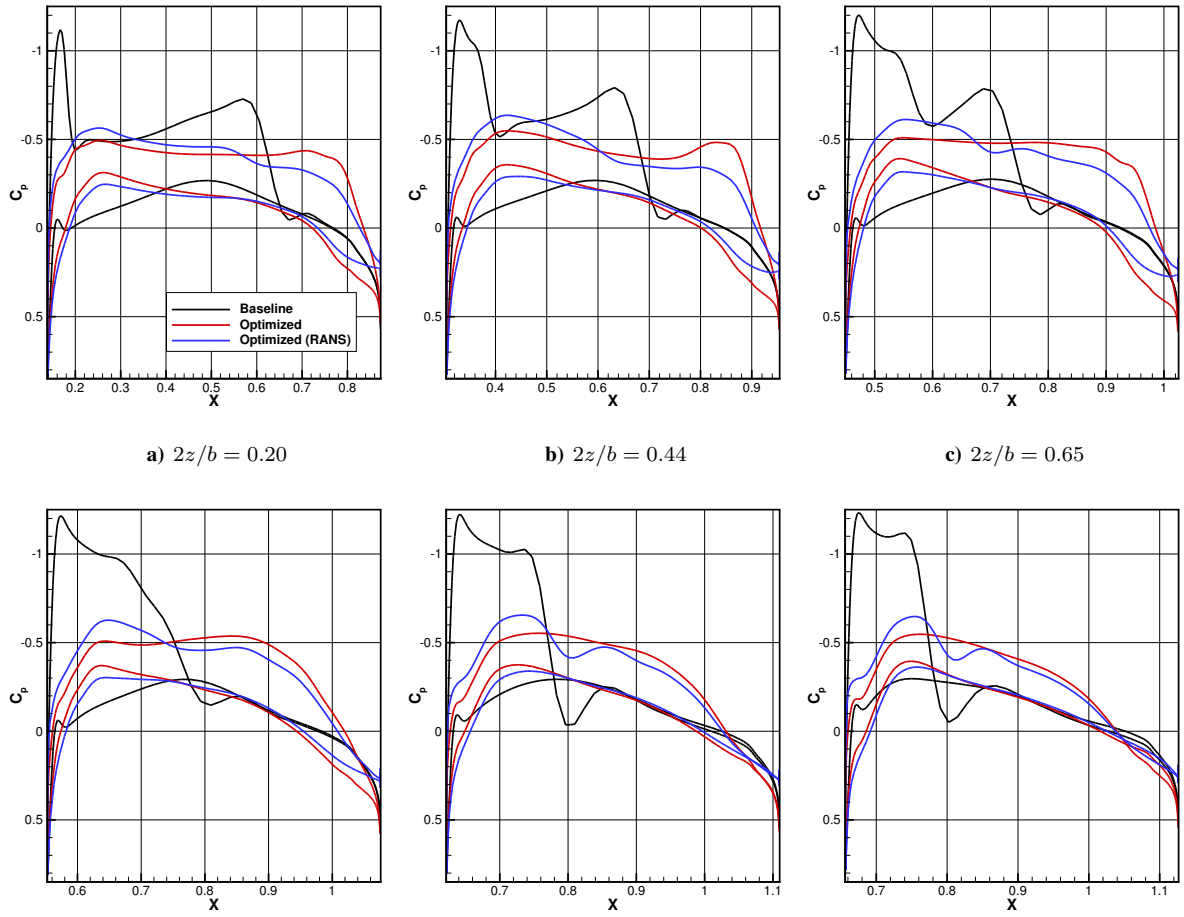
d) $2z/b = 0.80$ **e)** $2z/b = 0.90$ **f)** $2z/b = 0.95$

Figure 14. C_p **contours for RANS optimized designs.**

Component	Euler		RANS (Frozen Turbulence)		RANS	
	Time (h)	Fraction	Time (h)	Fraction	Time (h)	Fraction
Flow solution	1.82	0.167	5.98	0.397	5.43	0.212
Adjoint assembly	0.70	0.065	1.32	0.088	1.73	0.067
Adjoint solution	7.88	0.724	7.37	0.490	17.93	0.701
Miscellaneous	0.49	0.045	0.37	0.025	0.50	0.020
Wall-time Total	10.89	1.000	15.04	1.000	25.59	1.000
Processor Hours	348.5	–	1323.52	–	2251.0	–

Table 7. Timing breakdown for each optimization

We compared our optimization results with previous optimization studies of the ONERA M6 wing, summarized in Table 8. We obtained a large drag reduction of 19.1%. Present work used a relatively large grid size and compar-

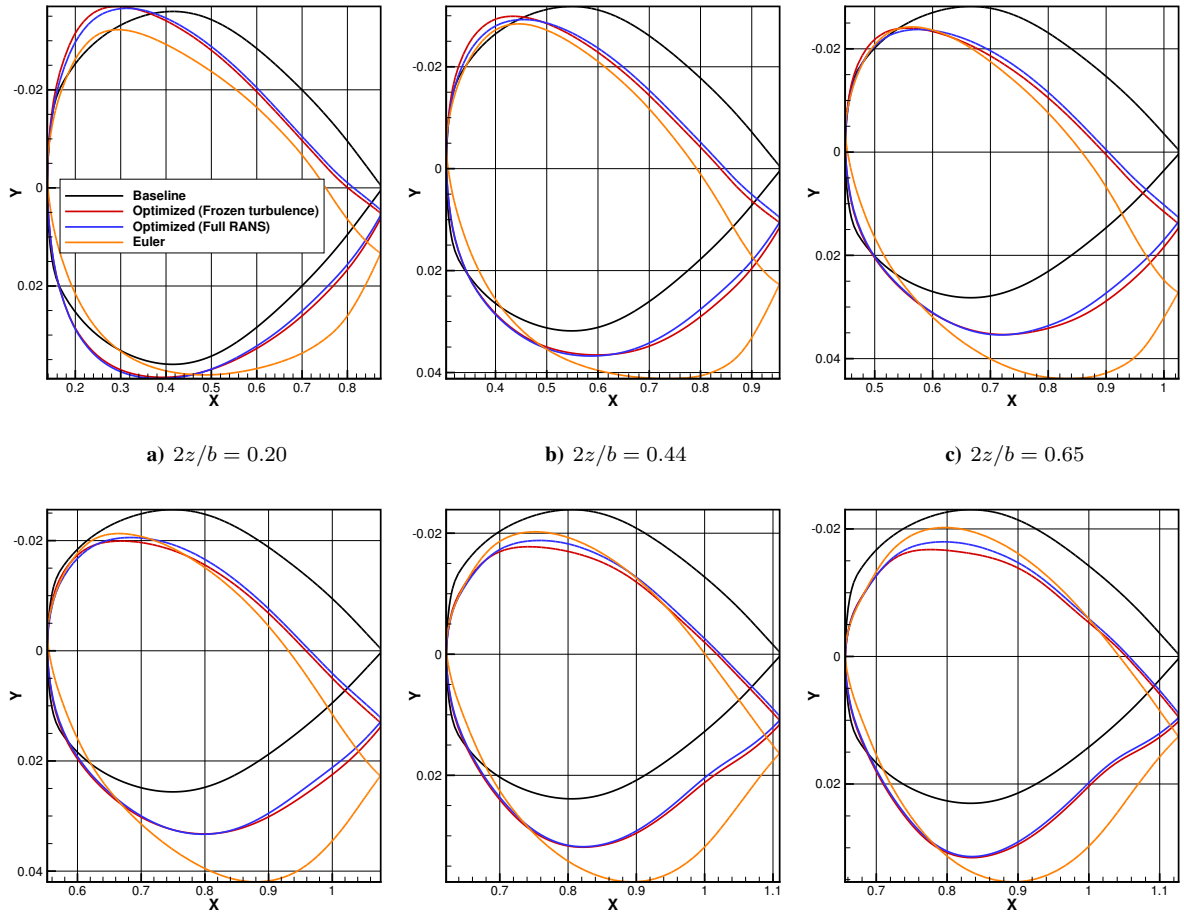


a) $2z/b = 0.20$ **b)** $2z/b = 0.44$ **c)** $2z/b = 0.65$

d) $2z/b = 0.80$ **e)** $2z/b = 0.90$ **f)** $2z/b = 0.95$

Figure 15. C_p **contours for Euler optimized design.**

atively a large number of design variables. Due to the clustering of shape design variables near the leading edge, the optimization was able to completely eliminate the leading edge shock.



a) $2z/b = 0.20$ **b)** $2z/b = 0.44$ **c)** $2z/b = 0.65$

d) $2z/b = 0.80$ **e)** $2z/b = 0.90$ **f)** $2z/b = 0.95$

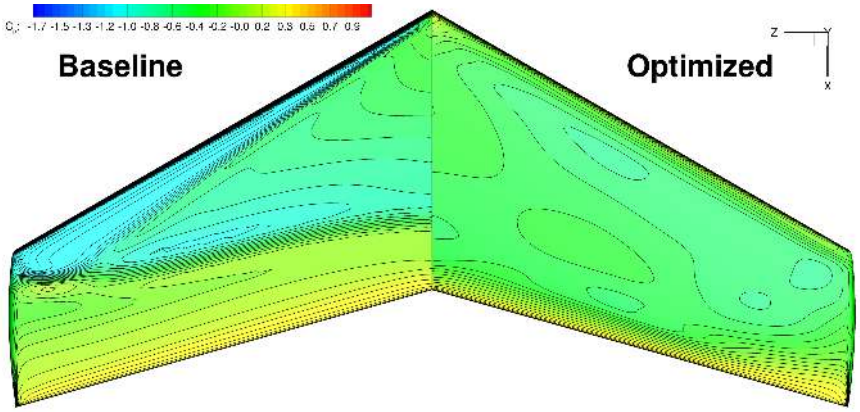
Figure 16. Cross section shapes for RANS and Euler optimized designs.

Origin	Grid Size	N_{DV}	Initial			Optimized			
			C_L	C_D	L/D	C_L	C_D	ΔC_D (%)	L/D
Present Work (L3)	8M	150	0.2710	0.01687	16.06	0.2710	0.01364	−19.1	19.86
Present Work (L2)	1M	150	0.2710	0.01725	15.71	0.2710	0.01400	−18.8	19.36
Osusky and Zingg [10]	2.2M	226	0.2590	–	–	0.2590	–	−17.1	–
Bueno-Orovio et al. [41]	43K	12	–	0.01712	–	–	0.01558	−10.0	–
Le Moigne and Qin [35]	312K	86	0.2697	0.01736	15.54	0.2964	0.01478	−14.9	18.23
Neilson and Anderson [34]	359K	21	0.2530	0.01680	15.06	0.2530	0.01420	−15.5	17.82
Lee et al. [36]	291K	40	0.2622	0.01751	14.97	0.2580	0.01586	−9.4	16.27

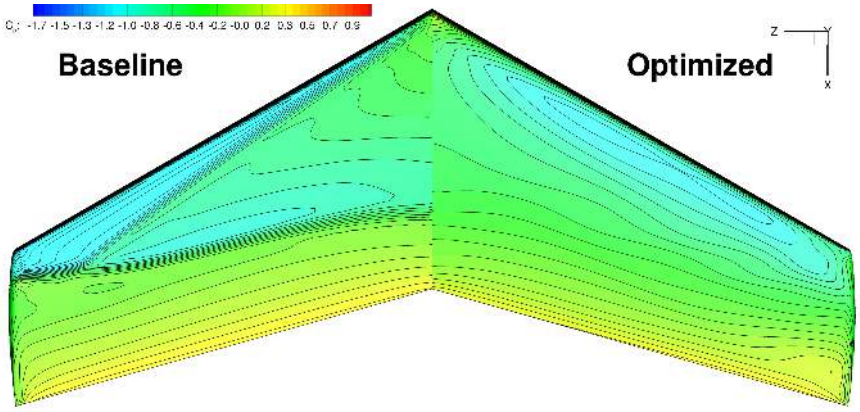
Table 8. Comparison of aerodynamic coefficients with previous work.

The C_p contours for both the Euler and RANS optimized designs are shown in Figure 17. We can see that both Euler and RANS achieved a shock-free solution. The Euler optimized design has a rapid pressure recovery near the TE. The RANS optimized solution, however, has parallel pressure contour lines with nearly constant spacing, indicating a

gradual increase of pressure.



a) Euler optimization

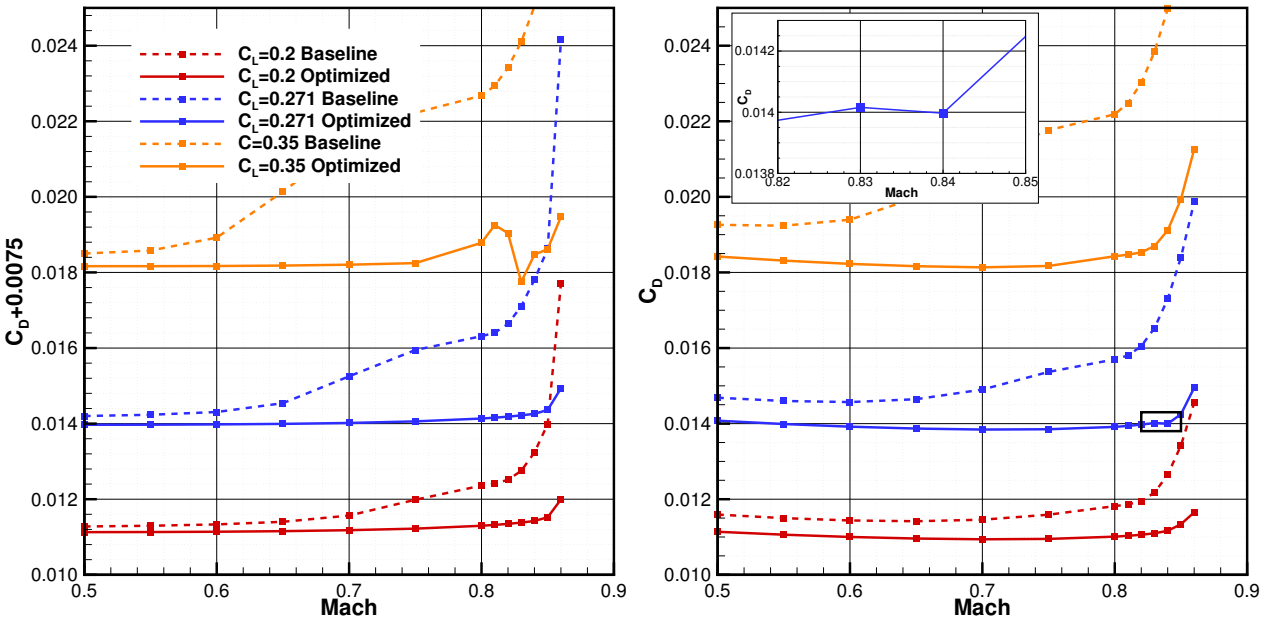


b) RANS optimization

Figure 17. C_p contours for baseline and optimized designs for Euler and RANS

We also investigated the drag divergence at different C_L. Figure 18 shows the drag divergence plot for three different C_L values. Both Euler and RANS optimized design reduced drag over the entire Mach range and the divergence Mach number are increased at all C_L values as compared to the baseline design. The drag coefficient remains nearly constant up to the divergence Mach numbers. At higher C_L, we see a drag pocket at the optimized Mach number for Euler solution. However, the drag dip on the RANS design is not significant. The effect could be due to the relatively low C_L. The drag dip at the optimized Mach number may become more prominent at higher loadings.

The baseline designs have lift distributions that are already reasonably close to elliptic. Both Euler and RANS optimized designs result in lift distributions that are very close to the optimum elliptical distribution, as shown in Figure 19. As a result, lift-induced drags were decreased, contributing to the pressure drag reduction shown in Table 6. The shift in lift distributions were obtained by the change in twist distributions shown in Figure 20. We also see that the Euler optimization tends to change t/c more significantly than the RANS optimization.



a) Euler optimization

b) RANS optimization

Figure 18. Drag divergence curves for three fixed lift coefficients.

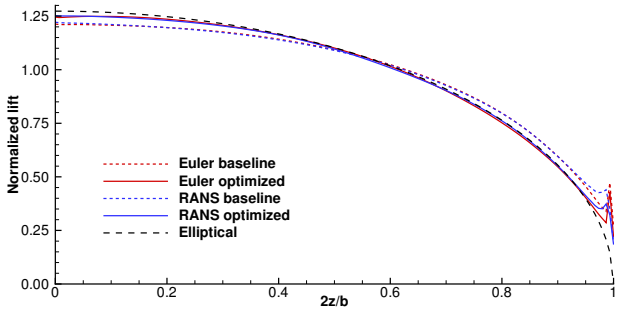


Figure 19. Lift distribution for baseline and optimized designs.

V. Conclusion

We presented an approach for rapid development of the adjoint to the Reynolds-Averaged Navier–Stokes equations with a Spalart–Allmaras turbulence model. Automatic differentiation is used to construct the partial derivatives for the discrete adjoint formulation. The resulting adjoint is computationally efficient and highly accurate. We use an analytic coloring acceleration technique to improve the adjoint assembly efficiency. The resulting RANS adjoint is verified with complex-step method using a flow over a bump test case. The aerodynamic gradients differ by $\mathcal{O}\left(10^{-9}\right)$, and the spatial gradients differ by $\mathcal{O}\left(10^{-4}\right)$ when compared with the complex-step method. A RANS aerodynamic shape optimization of the ONERA M6 wing is presented as a preliminary test case. The results are compared with a design obtained by a comparable Euler optimization. We achieved a drag reduction of 19% as compared the baseline wing. The shocks on the upper surface was completely eliminated and the optimized design improved the drag coefficient at all flight Mach numbers. The drag divergence Mach number of the optimized design is also increased. For the ONERA M6 optimization problem considered, the full RANS adjoint formulation resulted in a slightly better optimized design,

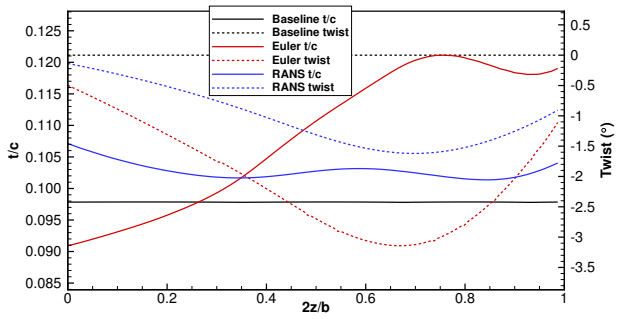


Figure 20. Thickness-to-chord ratio and twist distributions for baseline and optimized designs.

but the optimization was 70% more costly than the frozen turbulence formulation.

VI. Acknowledgments

The authors would like to thank Dr. Charles Mader for his suggestions and the previous implementation of reverse mode Euler ADjoint in SUmb that inspired and guided the work presented in this paper. The computations were performed on three supercomputers: Scinet HPC funded by the Canada Foundation for Innovation, NSERC, the Government of Ontario, Fed Dev Ontario, and the University of Toronto, Flux HPC at the University of Michigan CAEN Advanced Computing Center, and Stampede HPC of the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

References

[1] Jameson, A., "Aerodynamic Design via Control Theory," Vol. 3, No. 3, 1988, pp. 233–260. doi:10.1007/BF01061285.

[2] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, part 1," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60. doi:10.2514/2.2413.

[3] Reuther, J. J., Jameson, A., Alonso, J. J., Rimllnger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, part 2," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 61–74. doi:10.2514/2.2414.

[4] Leoviriyakit, K. and Jameson, A., "Multi-Point Wing Planform Optimization via Control Theory," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005. doi:10.2514/6.2005-450.

[5] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003, pp. 245–262. doi:10.1145/838250.838251.

[6] Nadarajah, S. K., *The Discrete Adjoint Approach to Aerodynamic Shape Optimization*, Ph.D. thesis, Stanford University, 2003.

[7] Nemec, M., Zingg, D. W., and Pulliam, T. H., "Multipoint and Multi-Objective Aerodynamic Shape Optimization," *AIAA journal*, Vol. 42, No. 6, 2004, pp. 1057–1065. doi:10.2514/1.10415.

[8] Hicken, J. E. and Zingg, D. W., "Aerodynamic Optimization Algorithm with Integrated Geometry Parameterization and Mesh Movement," *AIAA journal*, Vol. 48, No. 2, 2010, pp. 400–413. doi:10.2514/1.44033.

[9] Hicken, J. E. and Zingg, D. W., "Induced-Drag Minimization of Nonplanar Geometries Based on the Euler Equations," *AIAA journal*, Vol. 48, No. 11, 2010, pp. 2564–2575. doi:10.2514/1.J050379.

[10] Osusky, L. and Zingg, D., "A Novel Aerodynamic Shape Optimization Approach for Three-Dimensional Turbulent Flows," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012. doi:10.2514/6.2012-58.

[11] Lyu, Z. and Martins, J. R. R. A., "Aerodynamic Shape Optimization of a Blended-Wing-Body Aircraft," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013. doi:10.2514/6.2013-283.

[12] Lyu, Z. and Martins, J. R. R. A., "RANS-based Aerodynamic Shape Optimization of a Blended-Wing-Body Aircraft," *43rd AIAA Fluid Dynamics Conference and Exhibit*, June 2013.

[13] Bischof, C., Corliss, G., Green, L., Griewank, A., Haigler, K., and Newman, P., "Automatic Differentiation of Advanced CFD Codes for Multidisciplinary Design," *Computing Systems in Engineering*, Vol. 3, No. 6, 12 1992, pp. 625–637. doi:10.1016/0956-0521(92)90014-A.

[14] Sherman, L. L., Taylor III, A. C., Green, L. L., Newman, P. A., Hou, G. W., and Korivi, V. M., "First-and Second-order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods," *Journal of Computational Physics*, Vol. 129, No. 2, 1996, pp. 307–331. doi:10.1006/jcph.1996.0252.

[15] Mader, C. A., Martins, J. R. R. A., Alonso, J. J., and van der Weide, E., "ADjoint: An Approach for the Rapid Development of Discrete Adjoint Solvers," *AIAA Journal*, Vol. 46, No. 4, April 2008, pp. 863–873. doi:10.2514/1.29123.

[16] Spalart, P. and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th Aerospace Sciences Meeting and Exhibit*, 1992. doi:10.2514/6.1992-439.

[17] van der Weide, E., Kalitzin, G., Schluter, J., and Alonso, J., "Unsteady Turbomachinery Computations Using Massively Parallel Platforms," *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006. doi:10.2514/6.2006-421.

[18] Rall, L. B. and Corliss, G. F., "An Introduction to Automatic Differentiation," *Computational Differentiation: Techniques, Applications, and Tools*, 1996, pp. 1–17.

[19] Bischof, C., Roh, L., and Mauer-Oats, A., "ADIC: an Extensible Automatic Differentiation Tool for ANSI-C," *Software—Practice & Experience*, Vol. 27, No. 12, 1997, pp. 1427–1456. doi:10.1002/(SICI)1097-024X(199712)27:12<1427::AID-SPE138>3.3.CO;2-H.

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

[21] Bendtsen, C. and Stauning, O., "FADBAD, a Flexible C++ Package for Automatic Differentiation," Tech. rep., Department of Mathematical Modelling, Technical University of Denmark, 1996.

[22] Utke, J., Naumann, U., Fagan, M., Tallent, N., Strout, M., Heimbach, P., Hill, C., and Wunsch, C., "OpenAD/F: A Modular Open-Source Tool for Automatic Differentiation of Fortran Codes," *ACM Trans. Math. Softw.*, Vol. 34, No. 4, July 2008, pp. 18:1–18:36. doi:10.1145/1377596.1377598.

[23] Hascoët, L., "TAPENADE: a Tool for Automatic Differentiation of Programs," *Proceedings of 4th European Congress on Computational Methods, ECCOMAS*, 2004, pp. 1–14.

[24] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, TX*, 2010. doi:10.2514/6.2010-9231.

[25] 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. doi:10.1137/0907058.

[26] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202. doi:10.1007/978-1-4612-1986-6_8.

[27] Balay, S., Brown, J., , Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[28] Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Web page," 2013, http://www.mcs.anl.gov/petsc.

[29] 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. doi:10.2514/1.15830.

[30] Goldfarb, D. and Toint, P. L., "Optimal Estimation of Jacobian and Hessian Matrices that Arise in Finite Difference Calculations," *Mathematics of Computation*, Vol. 43, No. 167, 1984, pp. 69–88. doi:10.2307/2007400.

[31] Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," *Experimental Data Base for Computer Program Assessment*, 1979, pp. B1–1.

[32] Obayashi, S. and Guruswamy, G. P., "Convergence Acceleration of a Navier-Stokes Solver for Efficient Static Aeroelastic Computations," *AIAA Journal*, Vol. 33, No. 6, 2013/05/30 1995, pp. 1134–1141. doi:10.2514/3.12533.

[33] Mani, M., Ladd, J., Cain, A., Bush, R., Mani, M., Ladd, J., Cain, A., and Bush, R., "An Assessment of One- and Two-Equation Turbulence Models for Internal and External Flows," *28th Fluid Dynamics Conference*, 1997. doi:10.2514/6.1997-2010.

[34] 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. doi:10.2514/2.1765.

[35] Moigne, A. L. and Qin, N., "Variable-Fidelity Aerodynamic Optimization for Turbulent Flows Using a Discrete Adjoint Formulation," *AIAA Journal*, Vol. 42, No. 7, 2004, pp. 1281–1292. doi:10.2514/1.2109.

[36] Rho, O., Lee, K., Kim, C., Kim, C., and Lee, B., "Parallelized design optimization for transonic wings using aerodynamic sensitivity analysis," *40th AIAA Aerospace Sciences Meeting & Exhibit*, 2002. doi:10.2514/6.2002-264.

[37] Osusky, M. and Zingg, D., "A parallel Newton-Krylov-Schur flow solver for the Reynolds-Averaged Navier-Stokes equations," *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2012. doi:10.2514/6.2012-442.

[38] Kenway, G. K., Kennedy, G. J., and Martins, J. R. R. A., "A CAD-free Approach to High-Fidelity Aerostructural Optimization," *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, TX*, 2010. doi:10.2514/6.2010-9231.

[39] Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structures and Multidisciplinary Optimization*, Vol. 45, No. 1, 2012, pp. 101–118. doi:10.1007/s00158-011-0666-3.

[40] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM journal on optimization*, Vol. 12, No. 4, 2002, pp. 979–1006. doi:10.1137/S1052623499350013.

[41] Bueno-Orovio, A., Castro, C., Palacios, F., and Zuazua, E., "Continuous Adjoint Approach for the Spalart-Allmaras Model in Aerodynamic Optimization," *AIAA Journal*, Vol. 50, No. 3, 2012, pp. 631–646. doi:10.2514/1.J051307.