10th ANKARA INTERNATIONAL AEROSPACE CONFERENCE **AIAC-2019-205**
18-20 September 2019 - METU, Ankara TURKEY

DEVELOPMENT OF A 2D DISRETE TURBULENT ADJOINT SOLVER USING AUTOMATIC DIFFERENTIATION

Halil Kaya[1] and İsmail H. Tuncer[2]
Middle East Technical University
Ankara, Turkey

Hakan Tiftikçi[3]
Turkish Aerospace Industries Inc.
Ankara, Turkey

ABSTRACT

This paper presents an implementation of discrete adjoint solver using an automatic differentiation tool for a two-dimensional Reynolds Averaged Navier-Stokes (RANS) finite volume solver. Moreover, automatic differentiation tool is also utilized calculating flux Jacobian required by implicit finite volume solver. The developed RANS adjoint solver is computationally efficient and accurate. In the present implementation, the time required for the calculation of adjoint variables is comparable to the time required for one iteration of the flow solver. The RANS adjoint is verified with a brute force method. Finally, the aerodynamic shape optimization capability of the developed RANS adjoint solver is demonstrated by optimizing a RAE 2822 airfoil in terms of drag at constant lift coefficient.

INTRODUCTION

Aerodynamic shape optimization studies generally depend on computational fluid dynamics (CFD) analyses that enable to asses a large number of alternative design configurations within a relatively short time. However, when there are numerous design variables, gradient-based methods also require numerous CFD analyses to calculate sensitivities of objective functions with respect to design variables. Gradient-free optimization methods will also require many analyses to search a wide range and a large dimensional design space. To address this issue, adjoint solvers provide a solution. An adjoint solver accomplishes the remarkable feat of calculating the sensitivities with respect to a large number of design variables simultaneously via single computation. Thus, computational cost of an adjoint solver is independent of the number of design variables. Therefore, the contribution of incorporating an adjoint capability to a solver is remarkable. To incorporate an adjoint capability to a solver, partial derivatives of residuals should be computed. However, in many cases, they are prohibitively complex to derive, and the coding process is time-consuming and error-prone. Automatic differentiation (AD) tools offer a robust method to overcome this issue. To make use of automatic differentiation to calculate the partial derivatives of the residuals allows handling arbitrary complexity without difficulty.

[1] PhD. Candidate, Aerospace Engineering Department, Email: e134851@metu.edu.tr
[2] Professor, Aerospace Engineering Department, Email: ismail.h.tuncer@ae.metu.edu.tr
[3] Guidance, Navigation and Control Senior Specialist Engineer, UAV Group, Email: htiftikci@tai.com.tr

In this paper, it is intended to incorporate into a two-dimensional in-house finite volume RANS solver, a discrete adjoint [Giles *et al.*, 2003] solver capability by utilizing an automatic differentiation tool.

METHOD

Governing Equations

The two-dimensional Navier-Stokes equations can be stated as follows (1).

$$\frac{\partial}{\partial t}\int_{\Omega} \boldsymbol{W} d\Omega + \oint_{\partial\Omega} (\boldsymbol{F_c} - \boldsymbol{F_v}).\boldsymbol{n}\, ds = 0 \quad (1)$$

The conservative state variable vector $\boldsymbol{W}$ and the inviscid flux vector $\boldsymbol{F_c}$ are defined as follows (2), (3).

$$\boldsymbol{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad \boldsymbol{F_c} = f\boldsymbol{i} + g\boldsymbol{j} \quad (2)$$

$$\boldsymbol{f} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho(E+p)u \end{bmatrix}, \boldsymbol{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho(E+p)v \end{bmatrix} \quad (3)$$

The static pressure p is calculated by the ideal gas assumption (4).

$$p = (\gamma - 1)\rho\left[E - \frac{u^2 + v^2}{2}\right] \quad (4)$$

The viscous flux vector $\boldsymbol{F_v}$ is defined as follows

$$\boldsymbol{F_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{yx} \\ \Theta_x \end{bmatrix}\boldsymbol{i} + \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \Theta_y \end{bmatrix}\boldsymbol{j} \quad (5)$$

where $\boldsymbol{\tau}$ is the stress tensor defined as follows.

$$\tau_{ij} = \lambda\, \theta\delta_{ij} + 2\mu\epsilon_{ij} \quad (6)$$

θ is the volumetric dilatation rate (7), and $\boldsymbol{\epsilon}$ is the rate of angular deformation (8).

$$\theta = \boldsymbol{\nabla}\cdot\mathbf{U} \quad (7)$$

$$\epsilon_{ij} = \frac{1}{2}(\mathrm{U}_{i,j} + \mathrm{U}_{j,i}) \quad (8)$$

μ is the dynamic viscosity, and λ is the second viscosity. The value of the second viscosity is $-(2/3)\mu$ as suggested by Stokes. $\boldsymbol{\Theta}$ stands for the work done by the viscous stresses and the heat transfer.

$$\Theta_i = U_i\tau_{ij} + k\frac{\partial T}{\partial x_i} \quad (9)$$

In the flow solver, both shear stress and heat flux depend on both laminar viscosity and the turbulent eddy viscosity. The turbulent eddy viscosity is calculated by the Spalart-Allmaras (SA) one-equation turbulence model [Spalart and Allmaras, 1992].

Two-Dimensional Finite Volume Solver

In the present study, it is intended to incorporate an adjoint solver capability into an in-house two-dimensional finite volume RANS solver. The solver is an unstructured finite volume solver. It utilizes a cell-centered scheme. It has the capability of solving compressible Euler, laminar, and RANS equations. The solver computes the inviscid fluxes using Roe's flux-difference splitting. For a second-order accuracy, the gradients are calculated by using the Green-Gauss approach. The viscous terms are computed through a second-order central difference scheme. Time integration is performed by an Euler implicit scheme. The flux Jacobian required by an implicit scheme is evaluated through the routine that is automatically generated by the source-code transformation AD tool Tapenade [Hascoet and Pascual, 2004]. To increase the convergence rate, a local time-stepping scheme is employed. The closure of the viscous RANS equations is achieved using SA turbulence model. Turbulence working variable is advected through a first-order upwind method.

As pointed out, an implicit scheme is utilized for time integration. The scheme requires to solve a large, sparse, non-symmetric matrix (10) at each pseudo-time step. To solve the linear system, Intel MKL PARDISO solver is employed.

$$\left(\frac{\boldsymbol{\Omega_i}}{\boldsymbol{\Delta t_i}} + \left(\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}}\right)_{\boldsymbol{i}}\right) \boldsymbol{\Delta W_i^n} = -\boldsymbol{R_i^n} \qquad 10$$

In the solver, cfl number is increased as the solution proceeds. By employing a direct solver, the best possible preconditioner, that is the inverse of the matrix, is used, so that we can increase cfl number up to 10,000. Thanks to that, convergence is generally achieved around by 100-200 iterations.

Adjoint Method

The aim of the adjoint solver is to compute gradients of objective functions ($\boldsymbol{I}$) with respect to a computational grid ($\boldsymbol{x}$), which is generally dependent to design variables ($\boldsymbol{\alpha}$).

$$\frac{\boldsymbol{di(W,x)}}{\boldsymbol{dx}} = \frac{\boldsymbol{\partial I}}{\boldsymbol{\partial x}} - \frac{\boldsymbol{\partial I}}{\boldsymbol{\partial W}} \frac{\boldsymbol{dW}}{\boldsymbol{dx}}, \quad \boldsymbol{R(W,x)} = \boldsymbol{0} \qquad 11$$

Since governing equations ($\boldsymbol{R}$) must always be satisfied, the total derivative of the residuals with respect to the computational grid must also be zero. Thus,

$$\frac{\boldsymbol{dr}}{\boldsymbol{dx}} = \frac{\boldsymbol{\partial R}}{\boldsymbol{\partial x}} + \frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}} \frac{\boldsymbol{dW}}{\boldsymbol{dx}} = \boldsymbol{0} \quad \Rightarrow \quad \frac{\boldsymbol{dW}}{\boldsymbol{dx}} = -\left[\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}}\right]^{\boldsymbol{-1}} \frac{\boldsymbol{\partial W}}{\boldsymbol{\partial x}} \qquad 12$$

Substituting this result into the total derivatives of the cost function**.**

$$\frac{\boldsymbol{dI}}{\boldsymbol{dx}} = \frac{\boldsymbol{\partial I}}{\boldsymbol{\partial x}} - \underbrace{\frac{\boldsymbol{\partial I}}{\boldsymbol{\partial W}} \overbrace{\left[\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}}\right]^{\boldsymbol{-1}}}^{-d\boldsymbol{W}}}_{\boldsymbol{\lambda}} \frac{\boldsymbol{\partial R}}{\boldsymbol{\partial x}} \qquad 13$$

There are two ways to solve the linear system (13), depending on which right-hand side is chosen. In the first method, which is the direct method, the total derivative $\boldsymbol{dW/dx}$ is computed directly as it is depicted below (14).

$$-\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}} \frac{\boldsymbol{dW}}{\boldsymbol{dx}} = \frac{\boldsymbol{\partial R}}{\boldsymbol{\partial x}} \qquad 14$$

The alternative of the direct method is the adjoint method. In this method, a matrix called the adjoint matrix, $\boldsymbol{\lambda}$, is computed by solving the following system of equations (15).

$$\left[\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial W}}\right]^{\boldsymbol{T}} \boldsymbol{\lambda} = -\left[\frac{\boldsymbol{\partial I}}{\boldsymbol{\partial W}}\right]^{\boldsymbol{T}} \qquad 15$$

In this case, one has to solve a linear system for each cost function, so the computational cost scales with the number of the objective function, rather than the number of design variables as it is in the direct method. Moreover, typically, in aerodynamic shape optimization problems, there are a few objective functions like $\boldsymbol{CL, CD}$. However, it can be hundreds or thousands of design variables that define a shape.

Automatic Differentiation Software Tool

As formulated in the preceding section, an adjoint solution requires partial derivatives of residual terms and objective functions with respect to design variables and state variables. Hence, residual terms and objective functions need to be exactly differentiated. This difficult task that includes exact differentiation of Roe scheme, turbulence model, etc., is easily realized by automatic differentiation tools. Many researchers, such as [Nemili et al., 2013], [Thomas and Dowell, 2019], [Albring, Sagebaum and Gauger, 2015], [Lyu, Kenway, Paige, and Martins, 2013], [Mader, Martins, Alonso and Van der Weide, 2008] took advantage of automatic differentiation tools to develop an adjoint solver.

Automatic differentiation (AD), also called algorithmic differentiation or computational differentiation, is a technique that systematically applies the chain rule to computer programs. Since a computer program always carries out a series of elementary and arithmetic operations and functions, by performing the chain rule on these operations, derivatives of any routine, no matter how complicated, can be computed automatically. Moreover, it is as accurate as an analytic method. In the present study, Tapenade AD tool that uses source transformation technique is utilized to differentiate the residual calculating routines. By using source transformation technique, Tapenade provides a new routine that computes the desired derivatives.

To get detailed information on AD tools, one may refer to [Margossian, 2019], [Martins and Hwang, 2013]

Adjoint Solver

As formulated in the “Adjoint Method” section, by combining the objective function and residual terms with adjoint variables, the necessity for sensitivity coefficients($\boldsymbol{dW/dx}$) is bypassed. However, the manipulation of the equation is produced a set of adjoint variables ($\boldsymbol{\lambda}$) to solve.

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right]^{\boldsymbol{T}} \boldsymbol{\lambda} = -\left[\frac{\partial \boldsymbol{I}}{\partial \boldsymbol{W}}\right]^{\boldsymbol{T}} \qquad \text{Rep. 16}$$

As for required by the equation (15), the matrices ($\boldsymbol{\partial R/\partial W}$) and ($\boldsymbol{\partial I/\partial W}$) should be computed. As mentioned before, the AD tool Tapenade is utilized to calculate the Jacobians required for the adjoint solver. By considering this fact, in the flow solver, a set of routines that calculates residual and objective function values of a cell were created. The top-level routines are located in a loop that turns over all cells in the domain. Moreover, the residual and the objective function values of each cell only depend on a relatively small number of nearest-neighbor and next nearest-neighbor cells. The modified residual and objective function routines accept conservative variables of a cell, its neighbor cells and next nearest-neighbor cells as inputs and give residuals and objective function values of a cell as outputs. When Tapenade is called to differentiate these routines with respect to state variables, Tapenade systematically applies the chain rule line by line and generates the differentiated code for the computation of the vectors ($\Delta\boldsymbol{r}, \Delta\boldsymbol{i}$).

$$\Delta \boldsymbol{r} = \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{W}}\right] \Delta \boldsymbol{w}, \qquad \Delta \boldsymbol{i} = \left[\frac{\partial \boldsymbol{I}}{\partial \boldsymbol{W}}\right] \Delta \boldsymbol{w} \qquad 16$$

$\Delta\boldsymbol{r}, \Delta\boldsymbol{i}$ are changes in residual and objective value vectors of a cell, with respect to a change in a conservative variable of a dependent cell. So, to construct the Jacobian matrix of ($\boldsymbol{\partial R/\partial W}$) and ($\boldsymbol{\partial I/\partial W}$), the differentiated routines are also located in a loop that turns over

all cells in the domain, and at each cycle, an inner loop that turns over each conservative variables of each dependent variables is performed. After constructing the Jacobian matrices, the adjoint equation needs to be solved. It is in the same size as the linear system at 10, and it is solved in the same manner by employing Intel MKL PARDISO solver. Thus, the time required to solve the adjoint system is comparable to the time required for one iteration of the flow solver. Once the adjoint variables were computed, by the following equation (17) sensitivity of objective functions with respect to computational grid can be found.

$$\frac{\boldsymbol{dI}}{\boldsymbol{dx}} = \frac{\boldsymbol{\partial I}}{\boldsymbol{\partial x}} - \boldsymbol{\lambda}\frac{\boldsymbol{\partial R}}{\boldsymbol{\partial x}} \qquad 17$$

As for required by the equation (17), the matrices $(\boldsymbol{\partial R}/\boldsymbol{\partial x})$ and $(\boldsymbol{\partial I}/\boldsymbol{\partial x})$ should also be computed. They are calculated in the same manner as $(\boldsymbol{\partial R}/\boldsymbol{\partial W})$ and $(\boldsymbol{\partial I}/\boldsymbol{\partial W})$. The only difference is geometrical properties of cells are not dependent variables in the routines that are differentiated to compute $(\boldsymbol{\partial R}/\boldsymbol{\partial W})$ and $(\boldsymbol{\partial I}/\boldsymbol{\partial W})$, but they are dependent variables in the routines that are differentiated to compute $(\boldsymbol{\partial R}/\boldsymbol{\partial x})$ and $(\boldsymbol{\partial I}/\boldsymbol{\partial x})$. Hence, the differentiated routines accept grid coordinates of a cell, neighboring cells, and next nearest-neighbor cells. By the same way, they are located in a loop that turns over all cells in the domain, and at each cycle, an inner loop that turns over each node of each dependent cells is performed to construct Jacobian matrices.

In a shape optimization study, the shape is generally parametrized, and the optimization process is driven by using these shape parameters (or design variables). The following equation (18) specifies sensitivity of objective function with respect to design variables.

$$\frac{\boldsymbol{dI}}{\boldsymbol{d\alpha}} = \frac{\boldsymbol{dI}}{\boldsymbol{dx}}\frac{\boldsymbol{dx}}{\boldsymbol{d\alpha}} \qquad 18$$

By employing the adjoint solver, $\boldsymbol{dI}/\boldsymbol{dx}$ value is obtained. However, to perform an optimization study, $\boldsymbol{dx}/\boldsymbol{d\alpha}$ is also required. In the present study, $\boldsymbol{dx}/\boldsymbol{d\alpha}$ is computed by a brute-force method. In the study, the airfoil is parameterized with b-spline curves using 15 control points as design variables. Moreover, computational grid is generated by an in-house hyperbolic mesh generator. To compute $\boldsymbol{dx}/\boldsymbol{d\alpha}$**,** each design variable is perturbed by $1e^{-6}$ chord length, and the computational grid is regenerated. Finally, the deviation in grid coordinates divided by the perturbation magnitude, so $\boldsymbol{dx}/\boldsymbol{d\alpha}$ is obtained.

RESULTS AND DISCUSSION

To demonstrate the capability of the developed RANS adjoint solver, the drag coefficient of RAE 2822 airfoil is minimized at a constant lift coefficient. Before the optimization study, validation studies for the flow solver and the adjoint solver are performed. To validate the flow solver, the 6^{th} test case of the well-known AGARD report [Cook *et al.*, 1979] is analyzed by the solver. Then, the results of the solver are compared with the experimental results. Corrected flow conditions of the test case are given in the NASA technical report [Slater *et al.*, 2000]. According to the corrected flow conditions, Mach number is 0.729, the angle of attack is 2.34^0, and Reynolds number is 6.5E+06.

The analysis is performed by using a second-order Roe scheme, and the Spalart-Allmaras turbulence model. As a computational grid, a C-type with the size of 799*179 grid is generated (Figure 1). The height of the first layer is 10^{-6} unit, which ensures $y^+<1$.

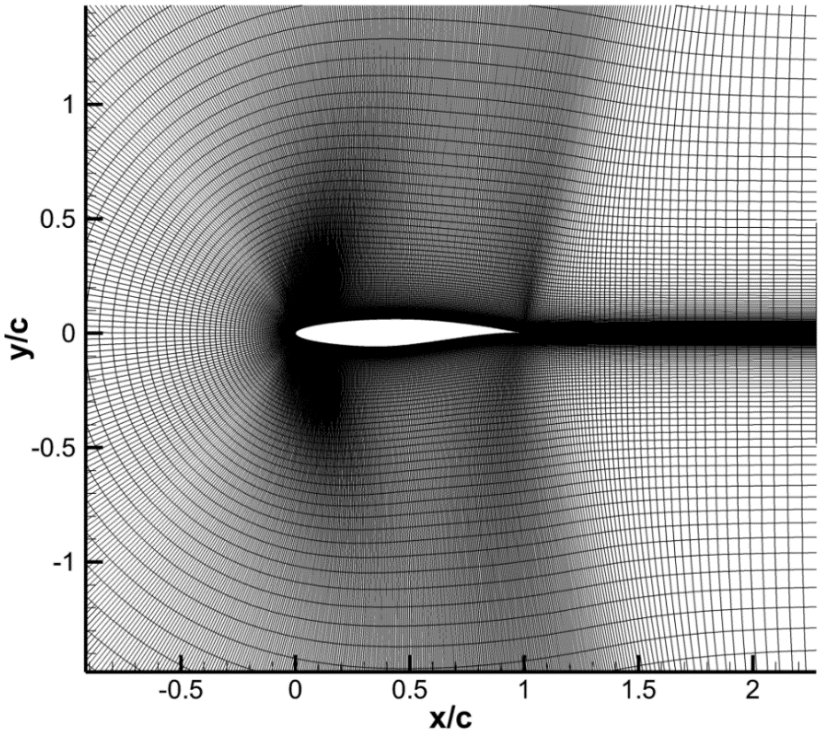

Figure 1 Computational grid of RAE 2822 airfoil

To validate the results, the calculated pressure distribution on the airfoil is compared with the experimental results (Figure 2).

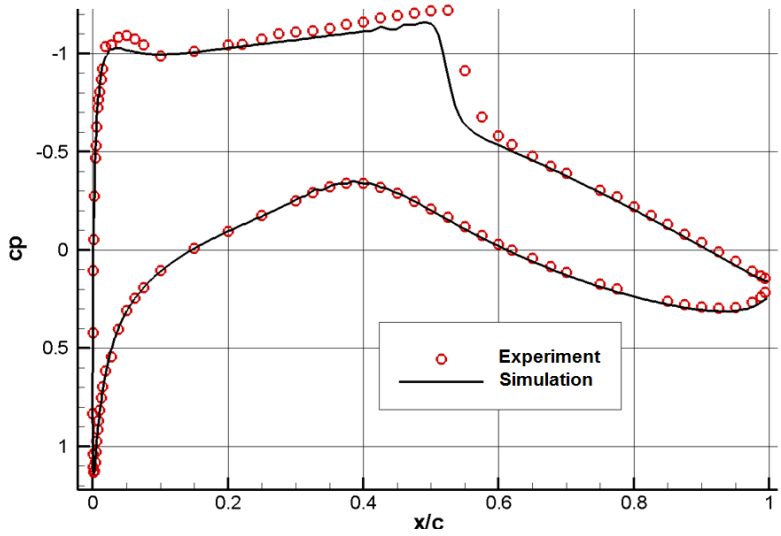

Figure 2 Comparison of pressure distributions

According to the compared results, the pressure distributions of the analysis and the experiment agree well. The discrepancy at the upper leading edge region is considered due to the transition trip located at 3% chord in the experiment. Moreover, the solver estimates the shock location around 3% chord close to upstream. Such a discrepancy at the shock location generally depends on the turbulence model. However, the accuracy in the prediction of pressure distribution is encouraging.

To validate the adjoint solver, firstly, RAE 2822 airfoil is parameterized with b-spline curves using 15 control points. The locations of the control points are determined by minimizing the deviation between the original and the approximated airfoil (Figure 3).

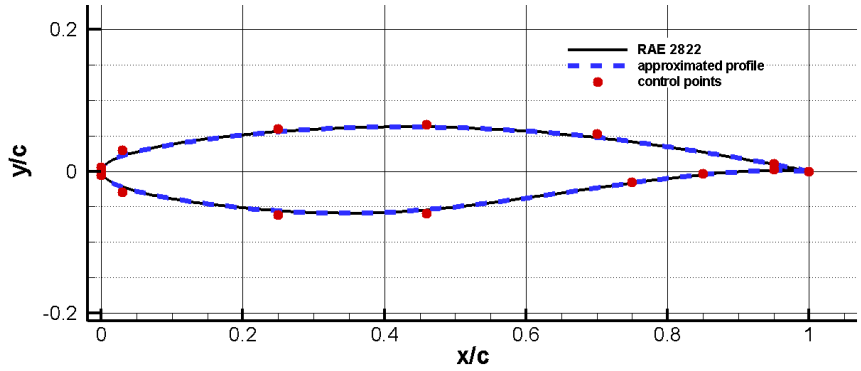

Figure 3 The comparison of the approximate and the original RAE 2822 airfoil

The adjoint solver is validated at the same flow condition with the validation case of the solver. To validate the adjoint solver, the sensitivities of the objective function with respect to the y coordinates of the control points calculated by the adjoint solver and a brute force method are compared. The control points at the trailing edge and leading edge are kept constant, so the optimization study is conducted by using 13 design variables. To keep the lift coefficient constant, a penalty function is introduced. The objective function is defined as follows (18).

$$I = cd + 2 * (cl - cl^{initial})^2 \qquad 19$$

To calculate the sensitivity of the objective function by brute force method, all the control points are moved one by one in y-coordinate in 0.001 unit, then converged flow solutions are obtained for each deformed airfoil, and the gradients are computed. Since the sensitivity obtained by brute force method includes the sensitivity related to second and higher-order derivatives, to make a better comparison, the adjoint solver is performed for deformed airfoil as well. Then, the comparison is conducted through the mean value of the derivatives computed by the adjoint solver for the original and deformed airfoils. The result of the comparison is tabulated below (Table 1). The ordering of the control points is in a counter-clockwise direction starting from the point that is the closest point to the trailing edge in a counter-clockwise direction.

Table 1 Comparison of the gradients computed by the adjoint solver and brute force method

Control point	Adjoint (Δ=0.000)	Adjoint (Δ=0.001)	Adjoint (Mean)	Brute Force (Δ=0,001)	Difference	Difference (%)
1	0.096	0.108	0.102	0.100	0.003	-2.6
2	-0.172	-0.108	-0.140	-0.137	-0.003	-2.2
3	-0.151	-0.097	-0.124	-0.128	0.004	3.3
4	0.691	0.773	0.732	0.731	0.001	-0.2
5	0.478	0.574	0.526	0.524	0.002	-0.3
6	-0.170	-0.153	-0.160	-0.161	0.001	0.9
7	-0.001	-0.015	-0.008	-0.005	-0.004	-42.2
8	-0.010	-0.007	-0.009	-0.010	0.001	16.1
9	0.017	0.018	0.017	0.017	0.001	-4.4
10	0.068	0.079	0.073	0.074	-0.001	1.2
11	0.133	0.155	0.144	0.144	0.001	-0.5
12	0.215	0.267	0.241	0.239	0.002	-0.8
13	0.727	1.320	1.023	1.026	-0.003	0.3

The maximum difference between the computed sensitivity by the brute force method and the adjoint solver is below 0.005. The difference, in terms of percentage, is below 5%, except

the control points 7 and 8. The reason of having large deviations, in terms of percentage, at the control points 7 and 8 is that the norms of the gradients at the points are close to zero, so the deviations in percentage are large. As it is mentioned, the brute force method does not give exact derivatives, so the difference between the adjoint solver and the brute force method is considered as acceptable.

Finally, an optimization study to minimize the drag coefficient at constant lift coefficient (19) is performed. In the study, the selected optimization method is the quasi-newton method. The adjoint solver provides the exact derivatives required by the optimization method. The convergence history of the optimization study is depicted in Figure 4.

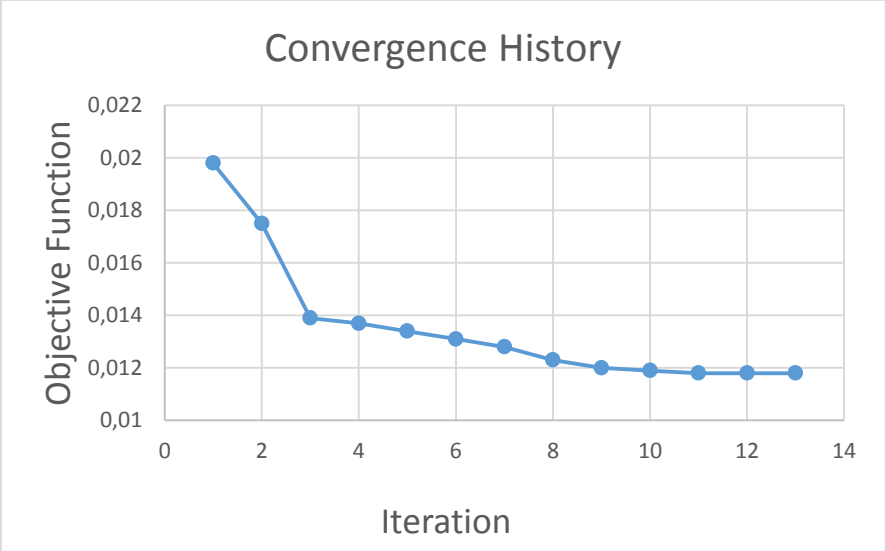

Figure 4 The convergence history of the optimization study

The optimized airfoil and pressure distribution on the airfoil given in Figure 5. At the analyzed flow conditions, there is strong shock on the original airfoil. The strong shock wave replaced by two weak shocks on the optimized airfoil. The objective function value of the optimized airfoil in the same flow condition is 40% lower than RAE 2822 airfoil.

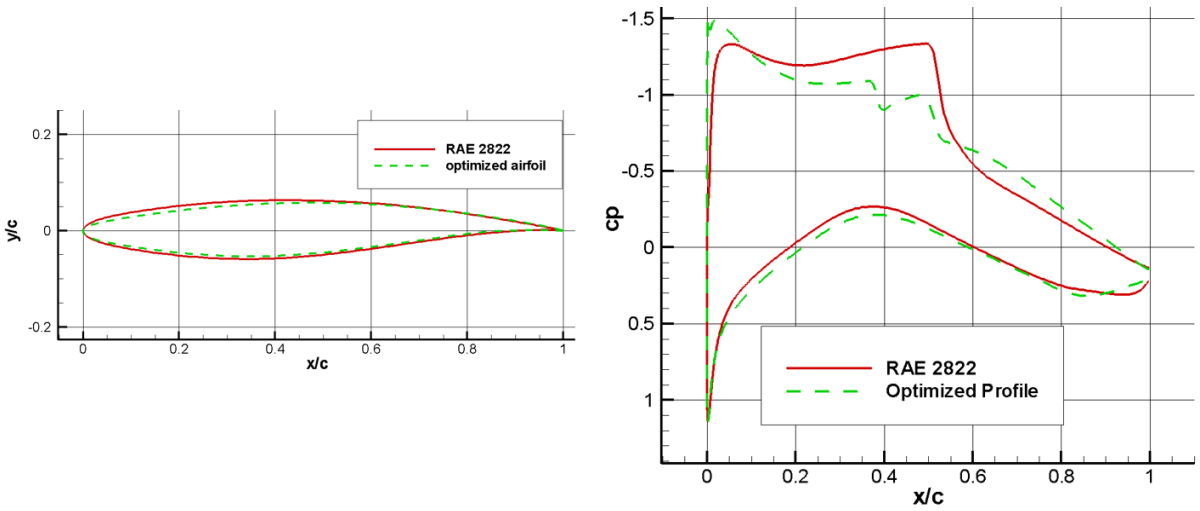

Figure 5 The comparison of the original airfoil and the optimized airfoil

CONCLUSION

In the present study, a turbulent discrete adjoint solver is incorporated into a two-dimensional RANS solver. An automatic differentiation tool that employs a source-code transformation

method automatically generates the routines which compute required partial derivatives in the adjoint solver. The developed RANS adjoint solver is computationally efficient and accurate. In the comparison of the gradients computed by the adjoint solver with the gradients computed by the brute force method. The differences in the norms of gradients are below 5%. Moreover, an optimization study using quasi-newton method is performed by employing the gradients computed by the adjoint solver. The study results in 40% reduction in drag coefficient at constant lift coefficient. Hence, it is experienced through the implementation that automatic differentiation tools are powerful and robust tools that allow rapid development of an adjoint solver in a discrete sense. It catalyzes the development process and substantially shortens the development time.

References

Albring, T., Sagebaum, M. and Gauger, N. R.. (2016) *Efficient Aerodynamic Design using the Discrete Adjoint Method in SU2.* AIAA Paper 2016-3518, 2016.

Cook, P.H., M.A. McDonald, M.C.P. Firmin (1979) *Aerofoil RAE 2822 – Pressure Distributions, and Boundary Layer and Wake Measurements*. Experimental Data Base for Computer Program Assesment, AGARD Report AR, 138, 1979.

Giles M. B., Duta M. C., Müller J.-D., and Pierce N. A. (2003) *Algorithm Developments for Discrete Adjoint Methods*. AIAA Journal, 41(2):198–205, 2003.

Griewank A. (2000) *Evaluating Derivatives* SIAM, Philadelphia, 2000.

Hascoet L. and Pascual V. (2004) *Tapenade 2.1 user's guide*. Technical report 300, INRIA, 2004.

Lyu, Z., Kenway, G. K., Paige, C., and Martins, J. R. R. A.. (2013) *Automatic Differentiation Adjoint of the Reynolds-averaged Navier–Stokes Equations with a Turbulence Model*. 21st AIAA Computational Fluid Dynamics Conference, AIAA Paper 2013-2581, 2013.

Mader, C. A., Martins J.R.R.A., Alonso J.J., and van der Weide, E.. (2008) *ADjoint: an approach for the rapid development of discrete adjoint solvers.* AIAA Journal, 46(4):863-873, 2008.

Margossian C. C. (2019) *A Review of Automatic Differentiation and its Efficient Implementation.* Retrieved from: https://arxiv.org/pdf/1811.05031.pdf, Mar 2019.

Martins, J. R. R. A. and Hwang, J. T.. (2013) *Review and unification of methods for computing derivatives of multidisciplinary computational models*. AIAA Journal, 51(11):2582-2599, 2013.

Nemili, A., Özkaya, E., Gauger, N. R., Kramer F., Höll, T. and Thiele, F.. (2013) *Optimal design of active flow control for a complex high-lift configurations.* AIAA Paper 2013-2585, 2013.

Slater, J.W., Dudek, J.C., and Tatum K.E. (2000) *The NPARC Alliance Verification and Validation Archive*. NASA Technical Report 2000-209946, 2000.

Spalart P.R., Allmaras S.R. (1992) *A One Equation Turbulence Model For Aerodynamic Flows*. In: 30th Aerospace Sciences Meeting and Exhibit. Reno, NV, USA, 6-9, Jun 1992.

Thomas, J. P., and Dowell, E. H.. (2019) *Discrete Adjoint Approach for Nonlinear Unsteady Aeroelastic Design Optimization*. AIAA Journal, 2019.