Adjoint-based Shape Optimization of High-lift Airfoils using the NSU2D Unstructured Mesh Solver

Taisuke Nambu[*]
Graduate School of Fundamental Science and Engineering, Waseda University, Shinjuku, Tokyo, 169-8555, JAPAN

Dimitri J. Mavriplis[†] and Karthik Mani[‡]

Department of Mechanical Engineering, University of Wyoming, Laramie, WY, 82071, USA

A high-lift multi-element airfoil is optimized using a discrete adjoint method. The unstructured mesh RANS solver NSU2D is used as the flow and adjoint solver, and LBFGS-B, which is an optimization algorithm based on a quasi-Newton method, is used for driving shape optimization. In order to achieve a larger design space, the mesh deformation solver in NSU2D is modified by re-computing the mesh stiffness matrix during the mesh deformation process. Design variables consist of rigging parameters such as flap and slat gap, overlap and deflections, as well as surface shape parameters. Two kinds of objective functions are optimized, a drag coefficient constrained by a target lift coefficient and a maximum lift coefficient. In the first case, the optimized shape reduces drag by 64 counts, and in the second case, the maximum lift coefficient of the optimized shape is increased from the baseline value of 4.340 to 4.602.

I. Introduction

Shape optimization methods are important tools in the aircraft design process. Many shape optimization techniques for aircraft have been demonstrated in previous works. Among several different kinds of optimization methods, gradient-based optimization has been used in many cases for aircraft design. Generally, the computational cost of gradient-based optimization is lower than non-gradient-based methods such as genetic algorithms (GA)[1]. In particular, the evaluation of aircraft performance objectives such as lift and drag requires significant computational effort due to the high cost of obtaining flow solutions, and thus gradient-based optimization is the only feasible method in many cases. Gradient-based optimization needs an efficient method for computing sensitivities of objective functions with respect to design variables. For example, a finite difference method is the most straightforward approach to compute sensitivities. However, the finite difference method requires one flow field solution for each design variable and is not practical for aircraft shape optimization which generally involves a large number of design variables. Additionally, the magnitude of the perturbation in the finite difference method can significantly affect the accuracy of the computed sensitivities. However, an adjoint method is well suited for computing sensitivities. That is because the adjoint method can compute the gradients of all design variables at a computational cost that is independent of the number of design variables. Additionally, the discrete adjoint method employs direct differention of the discretized governing equations, thus obviating the need to assign an ambiguous perturbation value as required in the finite difference method. Therefore, adjoint-based shape optimization represents a powerful technique for aircraft design optimization.

In past works, many aircraft optimization problems have been performed using the adjoint method[2-8]. In this paper, we focus on optimization of high-lift multi-element airfoils. Although there is significant previous work concerning optimization of high-lift airfoils[9-12], optimization of high-lift systems remains an important concern today. A major difficulty of these optimizations is due to the complex flow fields around high-lift airfoils[13], which often include massive separation regions in the wake, unsteady flow in the cove of the slat and main elements, strong streamline curvature, turbulent transition and so on. Additionally, the maximum lift coefficient is an important parameter for high-lift airfoils, which means that an accurate flow solution at the stall point is required.

[*] Graduate Student, AIAA student Member
[†] Professor, AIAA Associate Fellow.
[‡] Associate Research Scientist, AIAA Member.

Additionally, high-lift optimization problems generally involve complex geometries that require more complicated mesh configurations, which must be deformed at each optimization cycle. A large design space is desirable to obtain good optimization results, but the large deformations associated with large shape changes tend to produce invalid computational meshes. Hence, lack of robustness of the mesh deformation limits the design space.

In this work, NSU2D[14, 15] is used to obtain the flow and adjoint solutions. NSU2D is a two-dimensional flow solver for unstructured computational meshes which has been used for many flow fields including stall conditions[16]. NSU2D also incorporates a discrete adjoint method which computes the sensitivity of selected performance objectives with respect to design variables. The advantage of NSU2D is that it fully linearizes all computational operations including mesh deformation, as opposed to other adjoint solvers that may use finite differences or approximate formulations in some part of the linearization. Hence, NSU2D can compute exact sensitivities and the cost of these computations is independent on the number of design variables. NSU2D includes a mesh deformation capability, but the robustness for large deformations was found to be lacking. Hence, the mesh deformation solver in NSU2D has been modified to increase robustness and enable a larger design space.

The optimization process of the present work is described in Fig. 1. At first, the flow solution about the initial shape is computed, and then the sensitivities of the objective function with respect to design variables $dL/d\boldsymbol{D}$ are computed using the adjoint method at the state given by the flow variables $\boldsymbol{U}$. The new design variables $\boldsymbol{D}_{new}$ are computed by an optimization algorithm using the quasi-Newton method, LBFGS-B[17], and then new mesh coordinates $\boldsymbol{x}_{new}$ are generated using the mesh deformation method. The optimization process is iterated until the objective function is minimized. The design variables consist of the rigging parameters (physical locations) and surface shapes of the high-lift devices. In this work, two kinds of optimizations are conducted:

1. Minimization of drag coefficient at constant lift
2. Improvement of maximum lift coefficient

Obviously, drag minimization for high-lift devices should not compromise generated lift. Therefore, we try to decrease the drag coefficient while keeping the lift coefficient of the initial shape constant in the first case. In the second case, the flow angle is treated as one of the design variables and the lift coefficient is maximized. This case searches for an optimized shape of high-lift devices which has larger maximum lift coefficient.

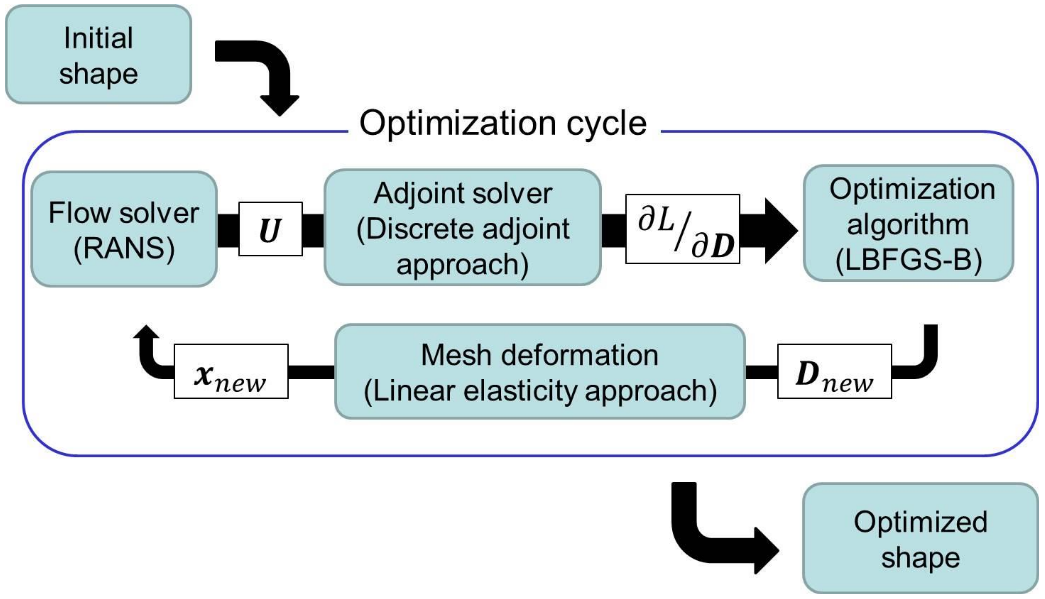

Initial shape
Optimization cycle
Flow solver (RANS)
$\boldsymbol{U}$
Adjoint solver (Discrete adjoint approach)
$\partial L/\partial \boldsymbol{D}$
Optimization algorithm (LBFGS-B)
$\boldsymbol{x}_{new}$
Mesh deformation (Linear elasticity approach)
$\boldsymbol{D}_{new}$
Optimized shape

Figure 1: Procedure of the adjoint-based shape optimization.

II. Flow solver

The NSU2D flow solver is described briefly in this section. In the present computations, the Reynolds-Averaged Navier-Stokes equations (RANS) are solved to compute the turbulent flow.

A. Governing equations

The conservative form of the Navier-Stokes equation can be written as:

$$\frac{\partial \boldsymbol{U}(\boldsymbol{x},t)}{\partial t} + \nabla \cdot \left(\boldsymbol{F}_c(\boldsymbol{U}) + \boldsymbol{F}_v(\boldsymbol{U})\right) = 0 \tag{1}$$

$$\boldsymbol{F}_c^x(\boldsymbol{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ (E_t + p)u \end{pmatrix}, \qquad \boldsymbol{F}_c^y(\boldsymbol{U}) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (E_t + p)v \end{pmatrix} \tag{2}$$

$$\boldsymbol{F}_v^x(\boldsymbol{U}) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + k\frac{\partial T}{\partial x} \end{pmatrix}, \qquad \boldsymbol{F}_v^y(\boldsymbol{U}) = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ u\tau_{yx} + v\tau_{yy} + k\frac{\partial T}{\partial y} \end{pmatrix} \tag{3}$$

where $\boldsymbol{U}$ is the vector of conservative variables, and $\boldsymbol{F}_c(\boldsymbol{U})$ and $\boldsymbol{F}_v(\boldsymbol{U})$ are the convective and viscous fluxes respectively. The viscous stress tensor is represented by τ_{ij} and can be written as

$$\begin{aligned} \tau_{ij} &= 2\mu S_{ij} \\ S_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij} \end{aligned} \tag{4}$$

where μ is the viscous coefficient. In order to include the turbulent effect in the solution, the eddy viscosity approximation is employed. In the approximation, the viscous coefficient and thermal conducive coefficient are modified as:

$$\begin{aligned} \mu &= \mu_L + \mu_T \\ k &= c_p\left(\frac{\mu_L}{Pr_L} + \frac{\mu_T}{Pr_T}\right) \end{aligned} \tag{5}$$

where μ_L and Pr_L are the fluid laminar viscosity and Prandtl number respectively. The variable Pr_T is the turbulent Prandtl number. In the present work, the turbulent eddy viscosity μ_T is computed using the Spalart-Allmaras (SA) model. This model computes the turbulent eddy viscosity by solving the transport equation for an eddy viscosity variable. The details of this model can be found in ref. 18.

Equation (1) is integrated over the control volume V and then the divergence theorem is applied. The integrated form of the flow equation is:

$$\int_{\Omega} \frac{\partial \boldsymbol{U}}{\partial t} dV + \int_{\partial\Omega} (\boldsymbol{F}_c(\boldsymbol{U}) \cdot \boldsymbol{n})\, dS + \int_{\partial\Omega} (\boldsymbol{F}_v(\boldsymbol{U}) \cdot \boldsymbol{n})\, dS = 0 \tag{6}$$

where S is the control surface and $\boldsymbol{n}$ is normal vector of control surface. The present work only considers steady-state problems, so the time derivative term can be ignored. Finally, the residual form of flow equation is written as:

$$\boldsymbol{R}(\boldsymbol{U}) = \int_{\partial\Omega} (\boldsymbol{F}_c(\boldsymbol{U}) \cdot \vec{\boldsymbol{n}})\, dS + \int_{\partial\Omega} (\boldsymbol{F}_v(\boldsymbol{U}) \cdot \vec{\boldsymbol{n}})\, dS = 0 \tag{7}$$

B. Spatial discretization

The flux terms are discretized as follows:

$$\int_{\partial\Omega}(\boldsymbol{F}_c(\boldsymbol{U})\cdot\boldsymbol{n})\,dS+\int_{\partial\Omega}(\boldsymbol{F}_v(\boldsymbol{U})\cdot\boldsymbol{n})\,dS=\sum_{i=1}^{n_{edge}}\left(\boldsymbol{F}_{c_{ei}}^{\perp}(\boldsymbol{U},\boldsymbol{n}_{e_i})+\boldsymbol{F}_{v_{ei}}^{\perp}(\boldsymbol{U},\boldsymbol{n}_{e_i})\right)S_{e_i} \tag{8}$$

where n_{edge} is the edge number of each cell and $\boldsymbol{F}_{c_{ei}}^{\perp}$ and $\boldsymbol{F}_{v_{ei}}^{\perp}$ are convective and viscous fluxes normal to the edges. The convective flux is computed using the central difference scheme with an artificial matrix dissipation term as:

$$\boldsymbol{F}_{c_e}^{\perp}=\frac{1}{2}\left\{\boldsymbol{F}_{c_L}^{\perp}(\boldsymbol{U}_L,\boldsymbol{n}_e)+\boldsymbol{F}_{c_R}^{\perp}(\boldsymbol{U}_R,\boldsymbol{n}_e)+\kappa[\boldsymbol{T}][|\boldsymbol{\lambda}|][\boldsymbol{T}]^{-1}\{(\nabla^2\boldsymbol{U})_L-(\nabla^2\boldsymbol{U})_R\}\right\} \tag{9}$$

$$\boldsymbol{F}_c^{\perp}=\begin{pmatrix}\rho V^{\perp}\\ \rho u V^{\perp}+n_x p\\ \rho v V^{\perp}+n_y p\\ E_t V^{\perp}+pV^{\perp}\end{pmatrix} \tag{10}$$

$$(\nabla^2\boldsymbol{U})_i=\sum_{j=1}^{neighbors}\left(\boldsymbol{U}_j-\boldsymbol{U}_i\right) \tag{11}$$

where velocity normal to the edge $V^{\perp}$ is computed as un_x+vn_y. The variables n_x and n_y represent components of unit edge normal vector. The linearization of the convective flux is decomposed by a diagonal matrix of eigenvalues $[\boldsymbol{\lambda}]$ and a matrix $[\boldsymbol{T}]$ consisting the corresponding eigenvectors ($\partial\boldsymbol{F}_c^{\perp}/\partial\boldsymbol{U}=[\boldsymbol{T}][\boldsymbol{\lambda}][\boldsymbol{T}]^{-1}$). The term consisting of the decomposed flow Jacobian matrix and the undivided Laplacian $\nabla^2\boldsymbol{U}$ is a dissipation term, while the other terms corresponds to a simple central difference of the convective fluxes. The undivided Laplacian operator corresponds to a multidimensional third difference, and thus this formulation results in a second-order accurate matrix dissipation scheme. The parameter κ in the dissipation term controls the amount of artificial dissipation. Further details of the discretization used in the flow solver can be found in ref. 14 and 15.

C. Implicit flow solver

The nonlinear residual eq. (7) is linearized and solved using Newton's method as:

$$\begin{gathered}\left[\frac{\partial\boldsymbol{R}(\boldsymbol{U}^j)}{\partial\boldsymbol{U}^j}\right]\delta\boldsymbol{U}^j=-\boldsymbol{R}(\boldsymbol{U}^j)\\ \boldsymbol{U}^{j+1}=\boldsymbol{U}^j+\delta\boldsymbol{U}^j\end{gathered} \tag{12}$$

where $\delta\boldsymbol{U}$ is the update value of each non-linear iteration step, and $\delta\boldsymbol{U}$ tends to zero when the solution is fully converged. Newton's method displays quadratic convergence if the flow Jacobian $[\partial\boldsymbol{R}(\boldsymbol{U}^j)/\partial\boldsymbol{U}^j]$ is an exact linearization of $\boldsymbol{R}(\boldsymbol{U})$. The linear system is solved using an agglomeration multigrid[19] method and a preconditioned GMRES[20] solver using multigrid as the preconditioner. A simple Jacobi method is used as the smoother in the multigrid algorithm.

D. Convergence check of flow solver

Figure 2 shows the convergence of flow solution. In this computation, the agglomeration multigrid solver is used in the first stage of the calculation, and then the preconditioned GMRES solver is used to obtain the final fully converged solution. Multigrid is conducted using a W-cycle with 5 grid levels, although a small $\mathrm{CFL}=10$ value is used to ensure the robustness for the various cases that may be encountered during the optimization procedure, particularly near stall conditions. The GMRES method uses multigrid as a preconditioner, and is restarted at every

100 steps. The computational mesh is shown in Fig. 3. This mesh has approximately 69,000 nodes and consists of triangular cells. The freestream flow conditions are $M_\infty = 0.2$, $Re = 9 \times 10^6$ and $\alpha = 16$ [deg].

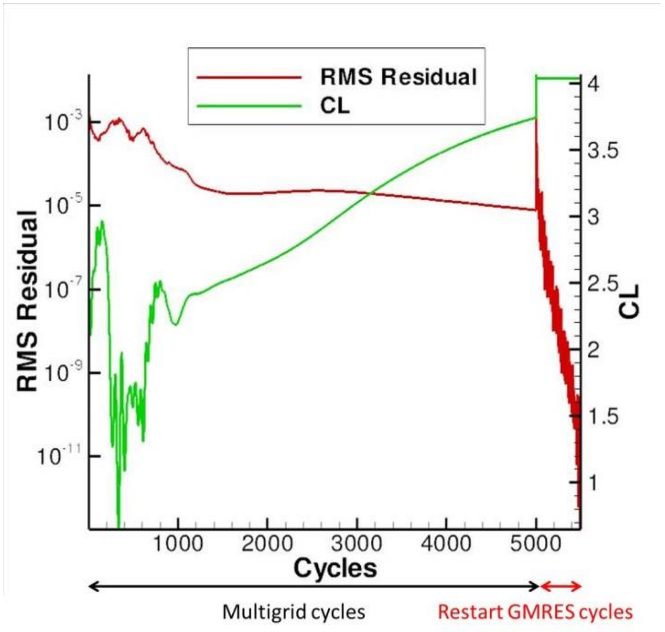
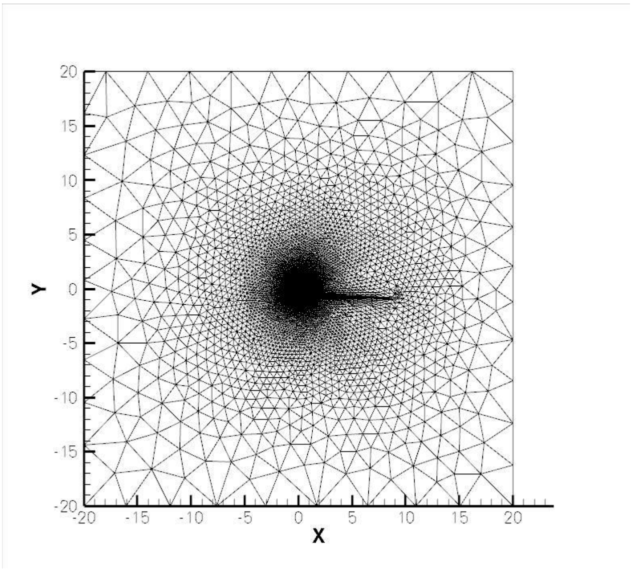
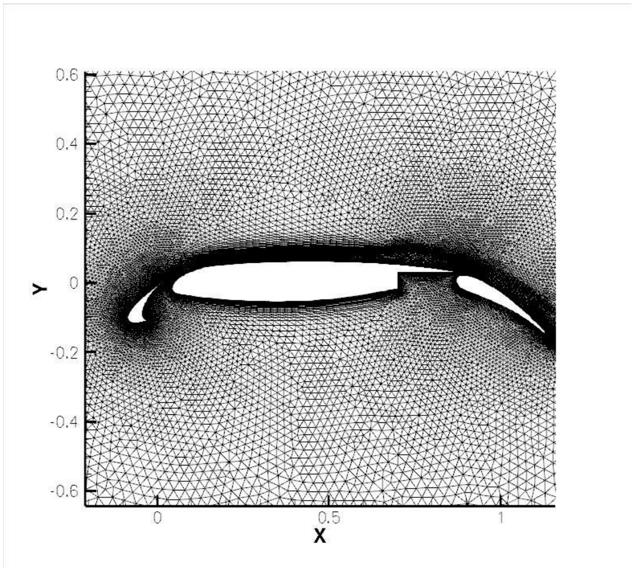


Figure 2: Convergence of the flow solution. The agglomeration multigrid scheme is used in the first stage of the calculation and the final converged solution is obtained using the GMRES solver.



(a) Far field view



(b) Close up view

Figure 3: Computational mesh for three-element high-lift airfoil

III. Mesh deformation method

It is necessary for shape optimization problems to generate a new computational mesh at each optimization cycle. There are two possible mesh modification approaches, mesh deformation and mesh re-generation. However, mesh re-generation, which alters the discretization stencil, is not appropriate in this case because it involves nondifferentiable operations that cannot be linearized for the adjoint process. On the other hand, the mesh deformation approach can compute the new mesh coordinates without altering the node connectivity, which is a differentiable operation, and therefore suitable for the present work. However, this approach lacks robustness for large deformation cases because the mesh topology cannot always be preserved. Obviously, a large design space is preferred for optimization, and thus the robustness of the mesh deformation process is an important consideration for shape optimization problems.

There are two kinds of commonly used mesh deformation approaches, tension spring[21, 22] and linear elasticity[23, 24] approaches. Both approaches are formulated following:

$$\boldsymbol{G}(\boldsymbol{x}) = [\boldsymbol{K}]\delta\boldsymbol{x}_{int} - \delta\boldsymbol{x}_{surf} = 0 \tag{13}$$

where δx_{int} represents the variation of the inner mesh point coordinates and δx_{surf} corresponds to the displacement on the boundary surface. The matrix $[\boldsymbol{K}]$ is a stiffness matrix.

A. Tension spring approach

Tension spring approach is formulated as:

$$\begin{gathered}[\boldsymbol{K}_{ii}] = \sum_{j=1}^{neighbors} l_{ij} \\ [\boldsymbol{K}_{ij}] = -l_{ij} \quad (i \neq j)\end{gathered} \tag{14}$$

where l_{ij} is the inverse of the squared edge length. This approach models the mesh edge as a spring. The spring strength is related to the edge length, and the force balance of adjacent springs is computed at each node.

B. Linear elasticity approach

In the linear elasticity approach, the motion of the computational mesh is assumed to obey the equations of linear elasticity. The linear elasticity equations can be written as:

$$\begin{gathered}\frac{\partial \boldsymbol{\sigma}_{ij}}{\partial \boldsymbol{x}_i} = -\boldsymbol{f}_j \\ \boldsymbol{\sigma} = [\boldsymbol{D}]\boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} = [\boldsymbol{A}]\delta\boldsymbol{x}\end{gathered} \tag{15}$$

where, $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ represent stress and strain. $[\boldsymbol{D}]$ is constitutive matrix and f is an external force. Applying a standard Galerkin method to eq. (15), the equations are formulated as:

$$[\boldsymbol{K}]\delta\boldsymbol{x} = \boldsymbol{F} \tag{16}$$

$$\begin{gathered}[\boldsymbol{K}] = \int_{\partial\Omega} [\boldsymbol{B}]^T [\boldsymbol{D}]\, [\boldsymbol{B}] dS \\ \boldsymbol{F} = -\int_{\partial\Omega} [\boldsymbol{M}]^T \boldsymbol{f} dS \\ [\boldsymbol{B}] = [\boldsymbol{A}][\boldsymbol{M}]\end{gathered} \tag{17}$$

where, $[\boldsymbol{M}]$ is the linear shape functions. The boundary displacements are given by the surface mesh deformation, so the external force $\boldsymbol{f}$ is not required. Therefore, the right-hand of eq. (16) is substituted by the displacement of surface nodes $\delta\boldsymbol{x}_{surf}$, and then eq. (16) becomes of the same form as eq. (13).

The stiffness matrix $[\boldsymbol{K}]$ is a function of the modulus of elasticity E. Cells that have large values of E are displaced as rigid bodies during the mesh deformation process, and therefore large values of E are prescribed in small cells in order to prevent deformation and cross-over in these regions. In the present computation, E is prescribed as proportional to the inverse of the cell volume.

C. Deformation of geometry by bump functions

The baseline geometry is deformed by moving the surface nodes. In order to achieve a smooth surface deformation, Hicks-Henne bump functions[25] are used to control the surface displacement. These functions are written as:

$$\begin{gathered} b_i(x) = a_{max} \cdot sin^{t_b}(\pi x^{m_i}) \\ m_i = ln(0.5)/ln(x_{M_i}) \end{gathered} \tag{18}$$

where x_{M_i} is the location of the bump, and $b_i(x)$ is the displacement length of the each bump location in the direction normal to the x-coordinate. The coefficient a_{max} is the magnitude of the bump at x_{M_i}, and t_b is a parameter that controls the width of the bump. The bump function generates a smooth curve centered at the location of x_{M_i} in the range $0 \leq x \leq 1$. A smooth surface displacement can be achieved by the superposition of several bump functions.

D. Comparing tension spring and linear elasticity approach

In order to examine the quality of the mesh deformation method, the computational mesh of a high-lift airfoil used in the present work (Fig. 3) is deformed using the tension spring and linear elasticity approach. The number of negative volume cells (i.e. crushed cells) in the deformed mesh is counted. Generally, the location of high-lift devices are defined by the gap, overlap and deflection of the slat and flap (see Fig. 4). The initial gaps of the slat and flap are 0.0298 and 0.0132, and these gaps are altered keeping the deflection angle constant. The mesh motion equation is solved iteratively using the line Jacobi method until the residual is reduced by 10 orders of magnitude. The stiffness matrix is computed based on the initial mesh coordinates and remains constant during the iterative process.

Table 1-4 show the number of negative volume cells produced in the deformed mesh. The linear elasticity approach is more robust than the tension spring approach. The tension spring approach causes negative volume cells even for small deformations. An illustration of the crushed cells produced by the tension spring approach is shown in Fig. 5. This figure shows the region near the trailing edge of the slat. The crushed cells show up in the boundary layer region where thin cells are clustered. Meanwhile, the linear elasticity does not crush the thin cells because the small volume cells are assigned high E values and respond as rigid bodies. However, cells in the other areas may be crushed, for example, cells between the slat and main elements as shown in Fig. 6 (b).

In order to improve the robustness of the mesh deformation method, we alter the stiffness matrix evaluation procedure. In the previous cases, the stiffness matrix was taken as a constant with values computed based on the initial shape. However, if large mesh deformation occurs, the stiffness matrix computed using the initial shape may be inappropriate. Therefore, the robustness of mesh deformation approach can be improved if the stiffness matrix is updated during the deformation. In this case, the mesh motion equation becomes a non-linear system. The non-linear mesh deformation equation is formulated as:

$$\begin{gathered} [\boldsymbol{K}(\boldsymbol{x}^j)]\delta \boldsymbol{x}_{int}^j - \frac{\delta \boldsymbol{x}_{surf}}{n} = 0 \\ \boldsymbol{x}^{j+1} = \boldsymbol{x}^j + \delta \boldsymbol{x}_{int}^j \\ (for\, j = 1,2 \cdots n) \end{gathered} \tag{19}$$

where n is a number of non-linear iterations. The displacement of the surface is divided into n steps and the stiffness matrix is updated at each step. Table 5 and 6 show the results of mesh deformation by the linear elasticity approach using the non-linear step approach. The number of non-linear iterations is 5 and 10. Figure 6 (c) shows an example of the deformed mesh produced by the non-linear mesh deformation method. The results show that the robustness of mesh deformation is greatly improved by the non-linear iteration approach. Although invalid cells still show up when slat and flap are moved very close to main element, the robustness is sufficient for the present optimization.

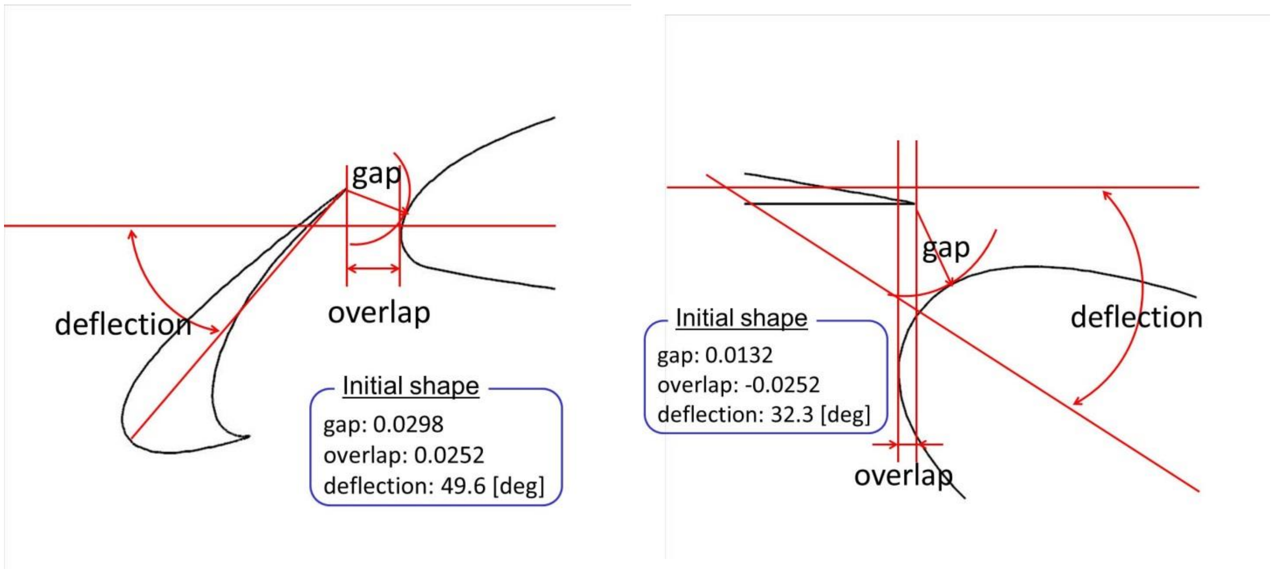


Figure 4: Definitions of gap, overlap and deflection

Table 1: Number of generated negative volume cells (slat, gap increase)

gap	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Tension spring	0	**24**	-	-	-	-	-	-
Linear elasticity	0	0	0	0	**2**	**36**	-	-

Table 2: Number of generated negative volume cells (slat, gap decrease)

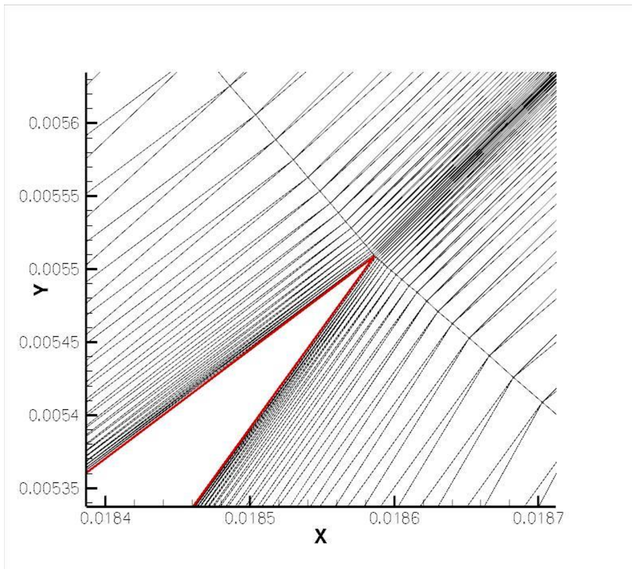
gap	0.025	0.02	0.015	0.01	0.005	0.001
Tension spring	**8**	**26**	-	-	-	-
Linear elasticity	0	0	**2**	**70**	-	-

Table 3: Number of generated negative volume cells (flap, gap increase)

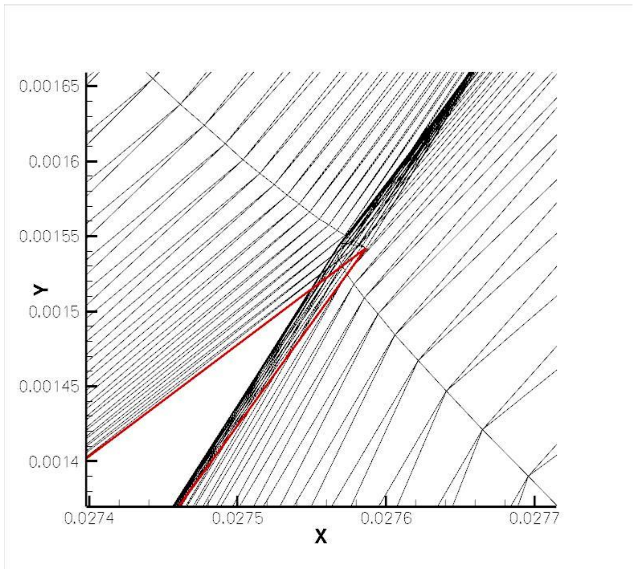
gap	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05
Tension spring	**2**	**44**	-	-	-	-	-	-
Linear elasticity	0	0	0	0	**7**	**11**	-	-

Table 4: Number of generated negative volume cells (flap, gap decrease)

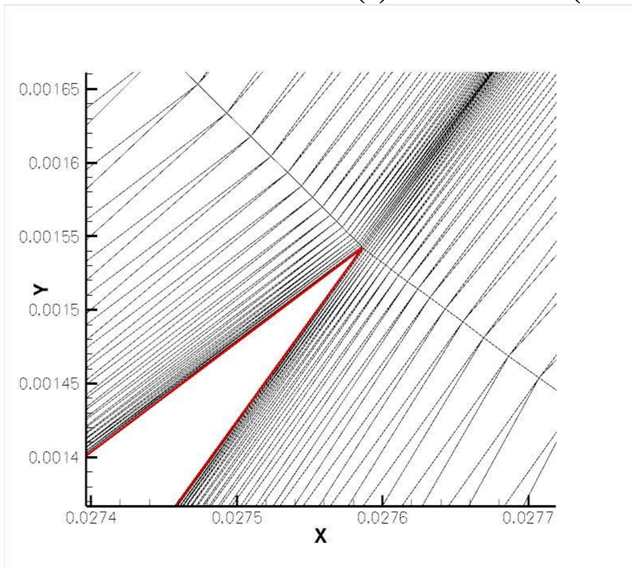
gap	0.01	0.005	0.001
Tension spring	**17**	**102**	-
Linear elasticity	0	**24**	**-**



(a) Initial mesh

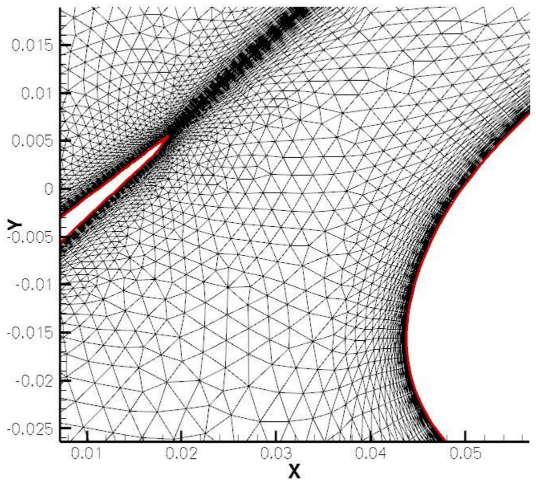


(b) Deformed mesh (tension spring analogy method)

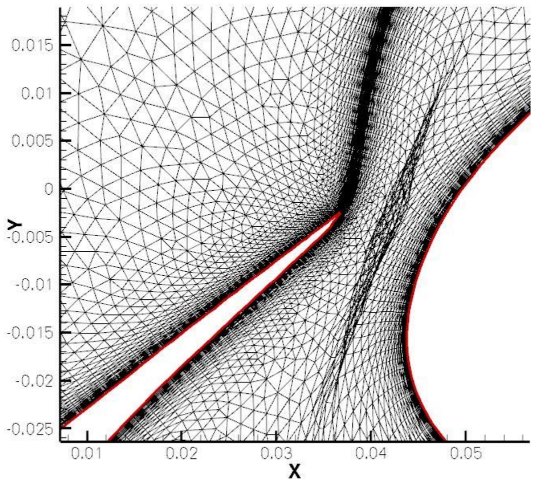


(c) Deformed mesh (linear elasticity method)

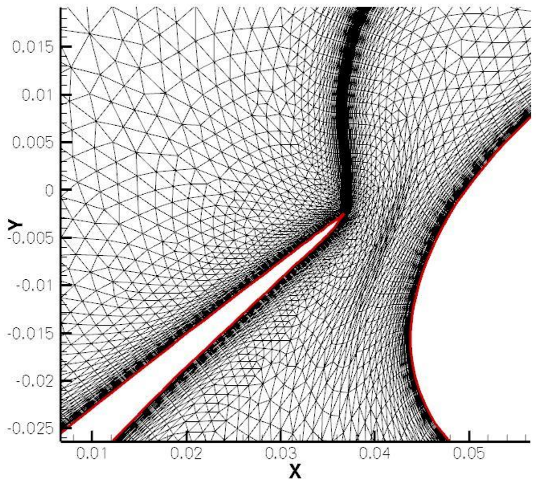
Figure 5: Crushed cells produced by the tension spring method (trailing edge of the slat)



(a) Initial mesh



(b) Deformed mesh (linear elasticity method)



(c) Deformed mesh (linear elasticity method using non-linear stiffness matrix iteration approach)

Figure 6: Crushed cells produced by the linear elasticity method (between the main element and slat)

Table 5: Number of generated negative volume cells (non-linear mesh deformation, slat, gap decrease)

gap	0.025	0.02	0.015	0.01	0.005	0.001
Linear elasticity $(n = 5)$	0	0	0	0	0	**140**
Linear elasticity $(n = 10)$	0	0	0	0	0	**29**

Table 6: Number of generated negative volume cells (non-linear mesh deformation, flap, gap decrease)

gap	0.01	0.005	0.001
Linear elasticity ($n = 5$)	0	0	**18**
Linear elasticity ($n = 10$)	0	0	**3**

IV. Adjoint method

In this section, the discrete adjoint method used in the NSU2D solver is described. In the present work, the objective function L is a scalar function which consists of aerodynamic performance quantities such as lift and drag coefficients. The objective function depends on the design variables $\boldsymbol{D}$, flow variables $\boldsymbol{U}$ and mesh coordinates $\boldsymbol{x}$. The sensitivity of objective function with respect to design variables can be written as:

$$\begin{gathered} L = L(\boldsymbol{D}, \boldsymbol{U}, \boldsymbol{x}) \\ \frac{dL}{d\boldsymbol{D}} = \frac{\partial L}{\partial \boldsymbol{D}} + \frac{\partial L}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{D}} + \frac{\partial L}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}} \end{gathered} \tag{20}$$

This is the forward linearization of objective function. The forward linearization is not efficient for optimizations involving large numbers of design variables. Therefore, the forward linearization is transposed and the adjoint linearization is formulated as:

$$\begin{gathered} \frac{dL}{d\boldsymbol{D}}^T = \frac{\partial L}{\partial \boldsymbol{D}}^T + \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{D}}^T \frac{\partial L}{\partial \boldsymbol{U}}^T + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}}^T \frac{\partial L}{\partial \boldsymbol{x}}^T \\ \frac{dL}{d\boldsymbol{D}}^T = \frac{\partial L}{\partial \boldsymbol{D}}^T + \begin{bmatrix} \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{D}}^T & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \boldsymbol{U}}^T \\ \frac{\partial L}{\partial \boldsymbol{x}}^T \end{bmatrix} \end{gathered} \tag{21}$$

In this equation, the derivatives of the flow and mesh variables with respect to the design variables ($\boldsymbol{\partial U/\partial D}$, $\boldsymbol{\partial x/\partial D}$) are unknowns. These derivatives are determined using the flow and mesh deformation equations. Differentiating the residual form of the flow equation with respect to the design variables:

$$\begin{gathered} \boldsymbol{R} = \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{U}_{bound}, \boldsymbol{x}) \\ \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}_{bound}}\frac{\partial \boldsymbol{U}_{bound}}{\partial \boldsymbol{D}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}} = 0 \end{gathered} \tag{22}$$

where $\boldsymbol{U}_{bound}$ denotes the flow boundary conditions. The residual form of the mesh deformation equation is also differentiated as:

$$\begin{gathered} \boldsymbol{G} = \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{x}_{surf}) \\ \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}_{surf}}\frac{\partial \boldsymbol{x}_{surf}}{\partial \boldsymbol{D}} = 0 \end{gathered} \tag{23}$$

where $\boldsymbol{x}_{surf}$ is the value of the boundary conditions, i.e. displacement of the geometry surface. The derivatives of state variables can be written using the eq. (22) and (23) as:

$$\begin{bmatrix} \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{D}}^T & \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{D}}^T \end{bmatrix} = \begin{bmatrix} -\frac{\partial \boldsymbol{U}_{bound}}{\partial \boldsymbol{D}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}_{bound}}^T & -\frac{\partial \boldsymbol{x}_{surf}}{\partial \boldsymbol{D}}^T \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}_{surf}}^T \end{bmatrix} \begin{bmatrix} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}^T & 0 \\ \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T & \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}}^T \end{bmatrix}^{-1} \tag{24}$$

This equation is substituted into eq. (21), and the sensitivities of objective function can be written as

$$\frac{dL}{d\boldsymbol{D}}^T = \frac{\partial L}{\partial \boldsymbol{D}}^T + \begin{bmatrix} -\frac{\partial \boldsymbol{U}_{bound}}{\partial \boldsymbol{D}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}_{bound}}^T & -\frac{\partial \boldsymbol{x}_{surf}}{\partial \boldsymbol{D}}^T \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}_{surf}}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_{\boldsymbol{U}} \\ \boldsymbol{\Lambda}_{\boldsymbol{x}} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\Lambda}_{\boldsymbol{U}} \\ \boldsymbol{\Lambda}_{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}^T & 0 \\ \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T & \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}}^T \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial L}{\partial \boldsymbol{U}}^T \\ \frac{\partial L}{\partial \boldsymbol{x}}^T \end{bmatrix} \tag{25}$$

where $\boldsymbol{\Lambda}_{\boldsymbol{U}}$, $\boldsymbol{\Lambda}_{\boldsymbol{x}}$ are defined as flow and mesh adjoint variables. Finally, the formulation of the adjoint methods is written as:

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}}\right]^T \boldsymbol{\Lambda}_{\boldsymbol{U}} = \frac{\partial L}{\partial \boldsymbol{U}}^T \tag{26}$$

$$[\boldsymbol{K}]^T \boldsymbol{\Lambda}_{\boldsymbol{x}} = \frac{\partial L}{\partial \boldsymbol{x}}^T - \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\Lambda}_{\boldsymbol{U}} \tag{27}$$

$$\frac{dL}{d\boldsymbol{D}}^T = \frac{\partial L}{\partial \boldsymbol{D}}^T - \left[\frac{\partial \boldsymbol{U}_{bound}}{\partial \boldsymbol{D}}\right]^T \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{U}_{bound}}\right]^T \boldsymbol{\Lambda}_{\boldsymbol{U}} - \left[\frac{\partial \boldsymbol{x}_{surf}}{\partial \boldsymbol{D}}\right]^T \left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}_{surf}}\right]^T \boldsymbol{\Lambda}_{\boldsymbol{x}} \tag{28}$$

Further details of the formulation can be found in ref. 26. Figure 7 and 8 show the convergence of flow and mesh adjoint variables i.e., equations (26) and (27) respectively. The flow adjoint equation is solved by the preconditioned GMRES method using linear multigrid as the preconditioner which is restarted at every 100 steps. The mesh adjoint equation is solved using the line-Jacobi method based on normal lines constructed in the boundary layer region (without multigrid).

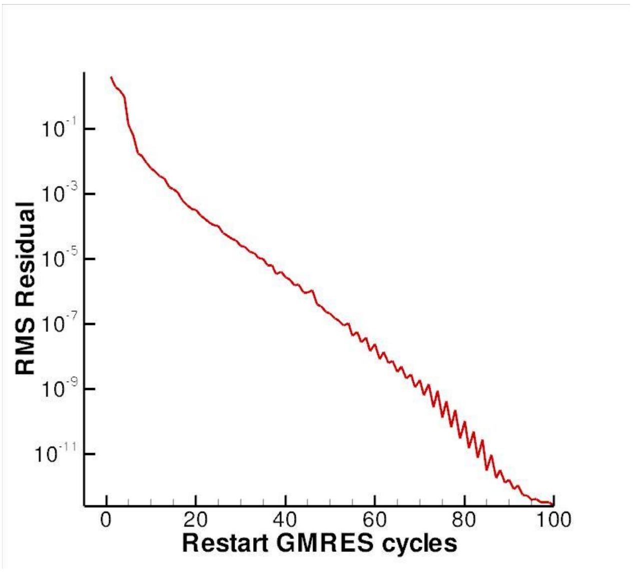


Figure 7: Convergence of the flow adjoint problem computed using the GMRES.

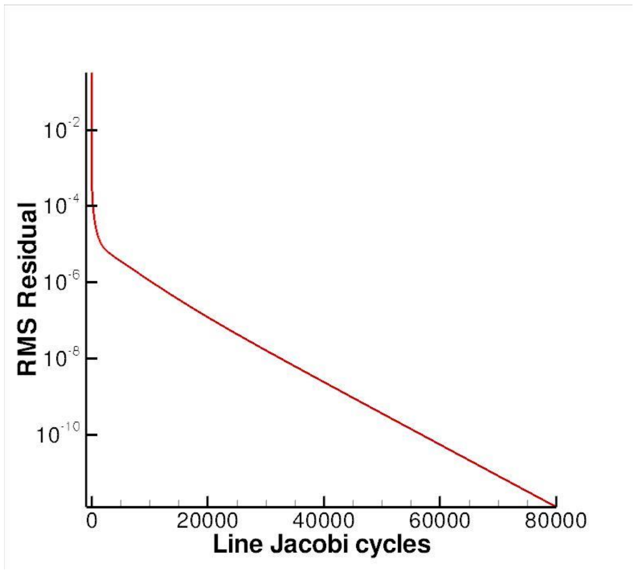


Figure 8: Convergence of the mesh adjoint problem computed using the line-Jacobi method.

V. Optimization algorithm

There are various possible choices for the optimization algorithm in the context of gradient-based optimization. The simplest choice is the steepest-descent method[2, 3]. This method can be written as $\delta \boldsymbol{D} = -\lambda\,(dL/d\boldsymbol{D})$ where $\delta \boldsymbol{D}$ is the correction of the design variables at each optimization cycle. This method is easy to implement, but exhibits poor performance when the number of design parameters is large[30]. Additionally, the optimal value of the coefficient λ must be determined. In the optimization process, each optimization cycle incurs high computational cost. Hence, it is necessary to use a more efficient optimization method and to obtain the optimized result in as few design cycles as possible. In the present work, LBFGS-B[17] is used for the optimization algorithm. LBFGS-B is a quasi-Newton method for bound-constrained optimization. The quasi-Newton method approximates the Hessian matrix of second derivatives using the optimization history, and thus requires only sensitivity and objective function values. The algorithm can set the bounds of the design space, and so it is able to prevent the generation of invalid geometry configurations.

VI. Verification of optimization method

In this section, verification of the optimization method is conducted. At first, the sensitivity computed using the adjoint method is compared with that computed using the finite difference method and the complex step method. Next, a simple optimization, which involves finding the maximum lift coefficient $C_{L\,max}$ using only the flow angle as a design variable, is conducted in order to check the behavior of the optimization process.

A. Comparing adjoint sensitivities with the finite difference sensitivities

The sensitivities computed using the adjoint method are verified using the finite-difference method. The sensitivities are computed using the finite difference method as:

$$\frac{dL}{d\boldsymbol{D}} = \frac{L(\boldsymbol{D}+\varepsilon) - L(\boldsymbol{D})}{\varepsilon} \tag{29}$$

where ε is a small perturbation. The flow solutions with the small perturbation value are computed for each design variable, and then the finite difference sensitivity is computed using the results with and without perturbation. In this verification case, there are 9 design variables. The locations of slat and flap are defined as displacement in the x and y directions and their rotation from the initial location (3 design variables: $\Delta x, \Delta y, \Delta\alpha \times 2$: slat and flap). The rotation centers are located at the midpoints of their chord. The coefficients of specified bump functions are also defined as design variables (a_{max} in eq. (18)). In this verification case, one bump function is located on the main element and one on the flap. Additionally, the flow angle α is treated as a design variable. Table 7 shows the sensitivities of inverse lift coefficient $1/C_L$ computed using the adjoint and finite difference method. The perturbation ε is 1.0×10^{-7}. The flow angle is 16 [deg] and all other design variables are evaluated at $\boldsymbol{D} = 0.0001$. The uniform flow conditions are $M_\infty = 0.2$ and $Re = 9 \times 10^6$. The adjoint sensitivities display good agreement with the finite-difference sensitivities, and these difference are less than 1% in all cases.

Table 7 Comparison of sensitivities computed by the adjoint method with the finite-difference method.

Design variables	Adjoint sensitivities	Finite-difference sensitivities
Δx_{slat}	**-0.4749**22939	**-0.4749**09584
Δy_{slat}	**0.313**688317	**0.313**827521
$\Delta\alpha_{slat}$	**-0.0961**93935	**-0.0961**74877
Δx_{flap}	**-0.238**173739	**-0.238**620514
Δy_{flap}	**0.548**861240	**0.548**008924
$\Delta\alpha_{flap}$	**-0.065**843774	**-0.065**679976
α	**-0.2564**15575	**-0.2564**02551
$a_{\max\_main}$	**0.1856**08371	**0.1856**79962
$a_{\max\ flap}$	**0.0047**48858	**0.0047**15814

B. Comparing with the sensitivities computed by the complex step method

Although the finite difference method can be used to compute sensitivities, the perturbation size affects the accuracy of the computed sensitivities. However, the complex step method can compute sensitivities tohigher accuracy as this method is not sensitive to roundoff error. In this section, the adjoint sensitivities are verified using the complex step method. A derivative of any real function $f(x)$ can be computed by expanding the complex operator $f(x + ih)$ as:

$$f(x + ih) = f(x) + ih\, f'(x) + \cdots \tag{30}$$

$$f'(x) = \frac{Im[f(x + ih)]}{h} \tag{31}$$

where h is the complex perturbation. The derivative is simply the imaginary part of the complex function divided by the perturbation. This method allows setting the perturbation to an arbitrarily small value, and so it is possible to compute the derivative to within machine precision. The complex operator $f(x + ih)$ is computed using a complex version of the NSU2D code, which is developed by redefining all floating point variables as complex. The objective function and the flow conditions are identical to the previous case, except for the flow angle. The flow angle is 0 [deg] and the complex perturbation h is 1.0×10^{-100}. The sensitivities of the flow angle α computed by the adjoint and the complex step method are -1.54729353596978 and -1.54729353596967, and they show almost perfect agreement.

C. Simple optimization case

A simple optimization is conducted to check the behavior of the optimization process. The objective function to be minimized is an inverse lift coefficient $1/C_L$ and the single design variable is the flow angle α. If the optimization process works correctly, the optimized objective function should converge to $1/C_{L\,max}$. The initial flow angle is 0 [deg], and the bound of the design space is taken as $-5 \le \alpha \le 35$ [deg]. The uniform flow conditions are $M_\infty = 0.2$ and $Re = 9 \times 10^6$. Figure 9 shows the optimization history of the design variable (i.e. flow angle). The optimization cycle is run for 30 iterations, and the final values are $\alpha = 21.95$ and $C_L = 4.340$. Figure 10 shows the plots of lift coefficient and flow angle at each optimization cycle. The optimized result is very close to the stall point and indicates that the optimization cycle works properly. The variation of the absolute value of sensitivity is shown in Fig. 11. The final sensitivity is 1 order of magnitude smaller than initial value. However, the ideal value of the final sensitivity should be zero because the optimized point must be a minimum. In this case, the variation of the lift coefficient around the stall point shows a nearly discontinuous variation, making it difficult to drive the sensitivity to zero

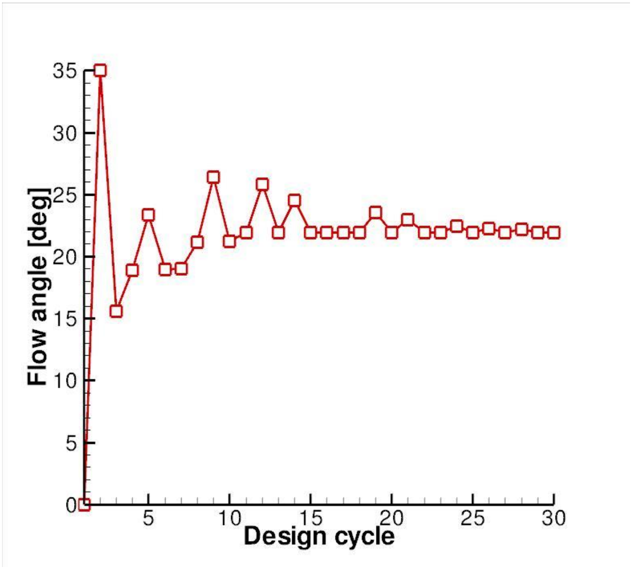


Figure 9: Optimization history of the design variable.

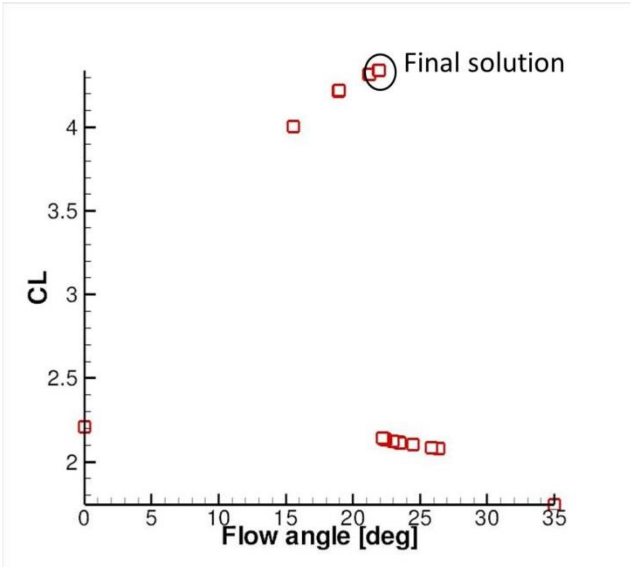


Figure 10: Lift coefficients computed at each optimization cycle.

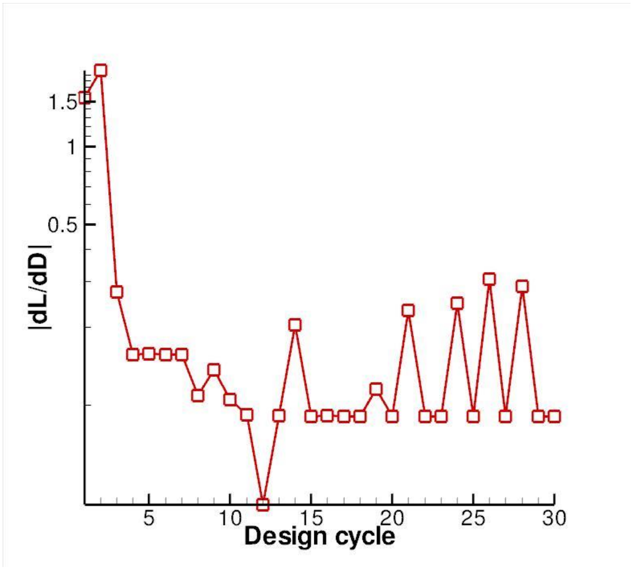


Figure 11: Variation of the sensitivity.

VII. Results

In this work, two types of optimizations are performed. In the first case, the drag coefficient is minimized keeping the lift coefficient constant. In the second case, the flow angle is treated as one of the design variables and the rigging parameters and airfoil shapes are optimized to increase the maximum lift coefficient $C_{L\,max}$.

A. Minimization of drag coefficient at constant lift

A small drag coefficient is desirable for an aircraft while the most important role of high-lift devices is to produce sufficient lift under takeoff and landing conditions. The first optimization case attempts to minimize the drag coefficient while maintaining the lift coefficient of the initial shape. The objective function is written as:

$$L = \left(C_L - C_{L_{target}}\right)^2 - 10(C_D)^2 \tag{32}$$

where $C_{L_{target}}$ is a target value of the lift coefficient, and the weighting factor (10) is used to balance the different magnitudes of lift and drag coefficients. The target lift coefficient is 4.0371. The design variables for this optimization are the location and deflection of the slat and flap $(\Delta x, \Delta y, \Delta\alpha)$. The freestream flow conditions are given as $M_\infty = 0.2$, $Re = 9 \times 10^6$ and $\alpha = 16$ $[deg]$. The bounds of design variables are set as:

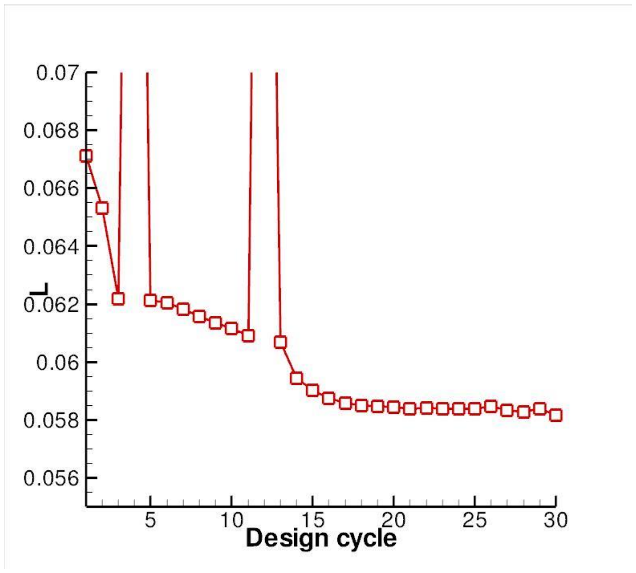
$$\begin{gathered} -0.03 \leq \Delta x_{slat} \leq 0.015 \\ -0.015 \leq \Delta y_{slat} \leq 0.03 \\ -10.0 \leq \Delta\alpha_{slat} \leq 5.0 \text{ [deg]} \\ -0.03 \leq \Delta x_{flap} \leq 0.03 \\ -0.03 \leq \Delta y_{flap} \leq 0.0 \\ -10.0 \leq \Delta\alpha_{flap} \leq 3.0 \text{ [deg]} \end{gathered} \tag{33}$$

Figure 12 shows the optimization history of the objective function and the sensitivities of the objective function with respect to each design variable. Table 8 shows the lift and drag coefficient of the optimized airfoil and the value of the objective function. The optimization succeeds in minimizing the objective function, and the value decreases from 0.067129 to 0.058168. The drag coefficient of the optimized airfoil is approximately 57 drag counts lower than the initial value while the variation of the lift coefficient is small (under 1%). Table 9 shows the final values of the design variables and Fig. 13 shows the optimized configuration.

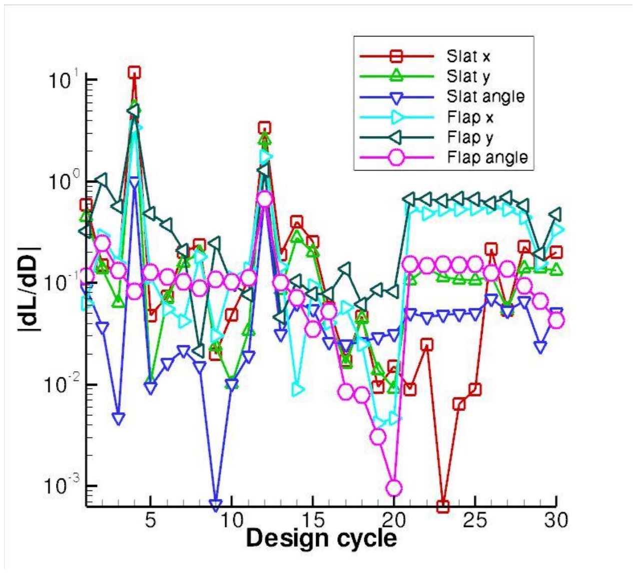
In the previous case, the surface shape of the high-lift devices is fixed and only their locations are changed. In the next case, their surface shape is also deformed using the bump functions. In order to retain the aerodynamic performance at the cruise condition, the deformation is limited to the unexposed area of the airfoils in the stowed configuration. The deformation area is shown in Fig. 14. Ten bump functions are set on the main element and flap upper surfaces only within the first 10% chord, and these are located at equal intervals. The bounds of the bump functions are prescribed as:

$$-0.003 \leq a_{max} \leq 0.003 \tag{34}$$

The initial locations of the slat and flap are the final values of the previous case (Table 9). Figure 15 shows the optimization history of the objective function and the sensitivities of the objective function with respect to each design variable. Table 10 shows the lift and drag coefficient of the optimized airfoil and the value of the objective function. The optimization process decreases the objective function from 0.058168 to 0.057261. In this optimization, the drag coefficient decreases by 7 drag counts. Figure 16 shows the surface deformation compared with the initial shape. Table 11 shows the final values of the design variables and Fig. 17 shows the final design configuration. Table 12 shows the lift and drag coefficient of each element. In these elements, the variations of the slat and main element are relatively large. The performance of the slat is increased while the main element shows lower performance. Figure 18 shows distributions of the pressure coefficient near the gap of the slat and main element. The optimized airfoil has higher pressure on the back side of the slat than the initial one, and mainly this higher pressure affects the performance of the slat and main element.



(a) Objective function



(b) Sensitivities of objective function

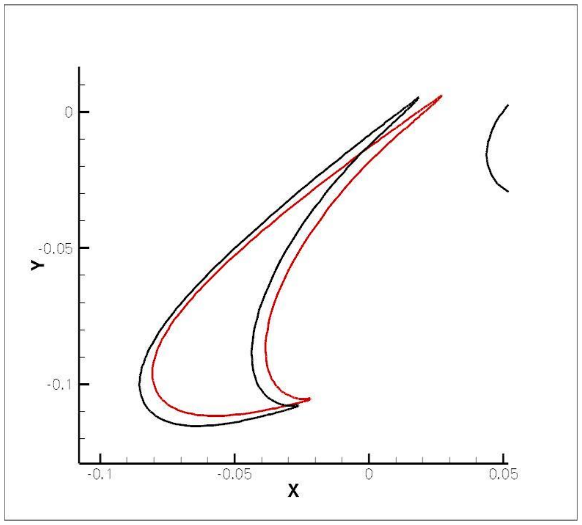
Figure 12: Optimization history for lift-constrained drag minimization problem

Table 8 Lift, drag and objective function values of optimized configuration for lift-constrained drag minimization

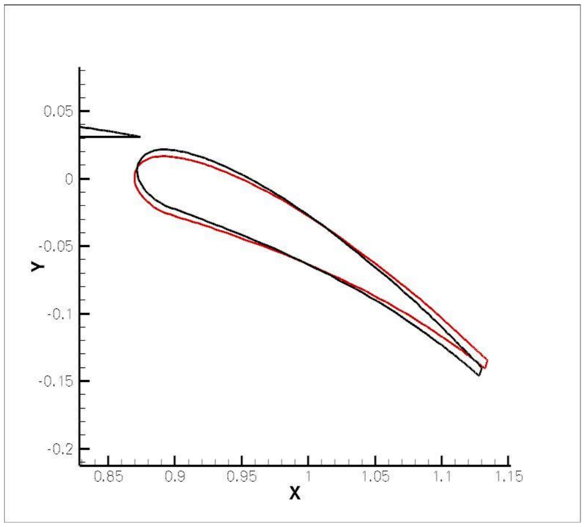
	C_L	C_D	L
Initial airfoil	4.0371	0.081932	0.067129
Optimized airfoil	4.0235	0.076146	0.058168
Δ	-0.0136	-0.005786	-0.008961

Table 9 Final design variables for lift-constrained drag minimization problem

	Δx	Δy	$\Delta\alpha\ [deg]$
Slat	0.0064448	0.0025146	2.0883
Flap	0.0010717	-0.00023812	-2.5162



(a) Slat



(b) Flap

Figure 13: Optimized shape of high-lift devices for lift-constrained drag minimization problem (redline represents the optimized shape and black line represents the initial shape)

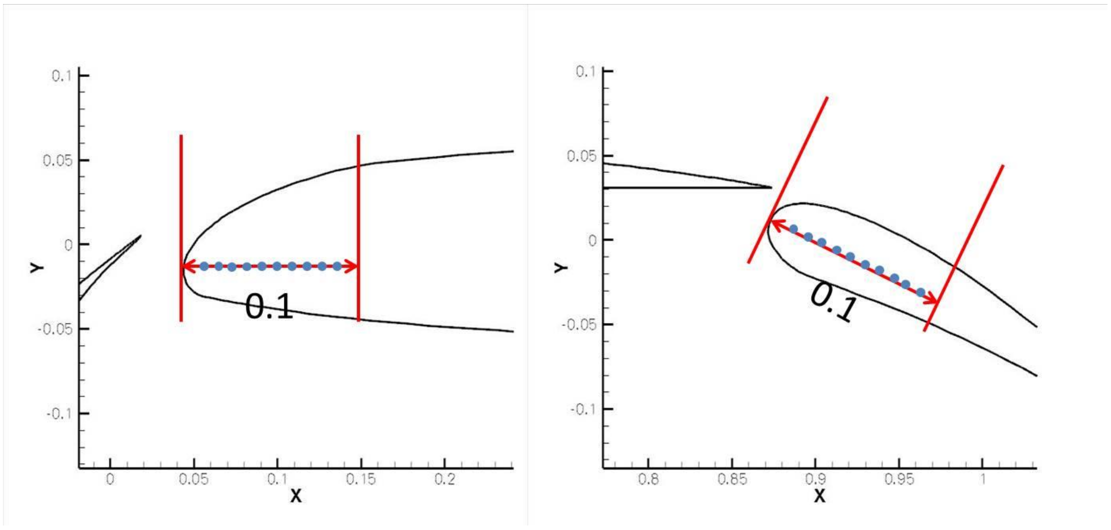
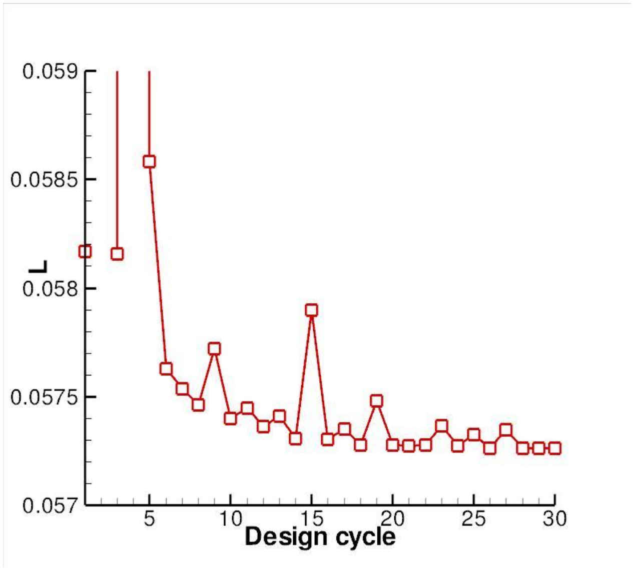
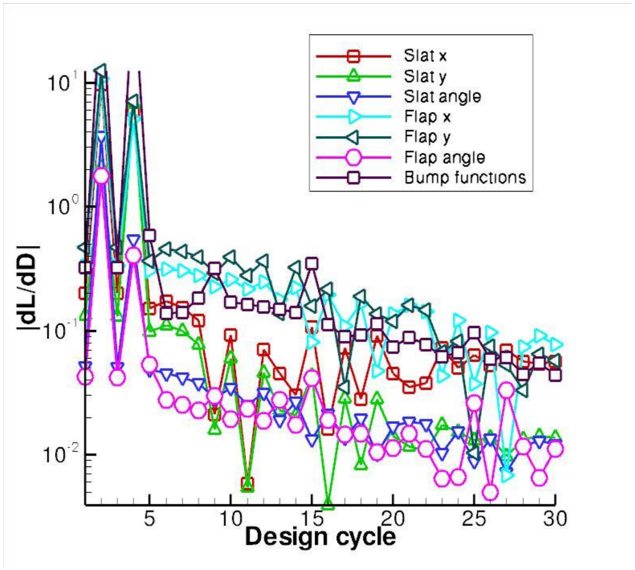


Figure 14: Deformation region of applied bump functions for optimization. Only upper surfaces of main element and flap are deformed.



(a) Objective function

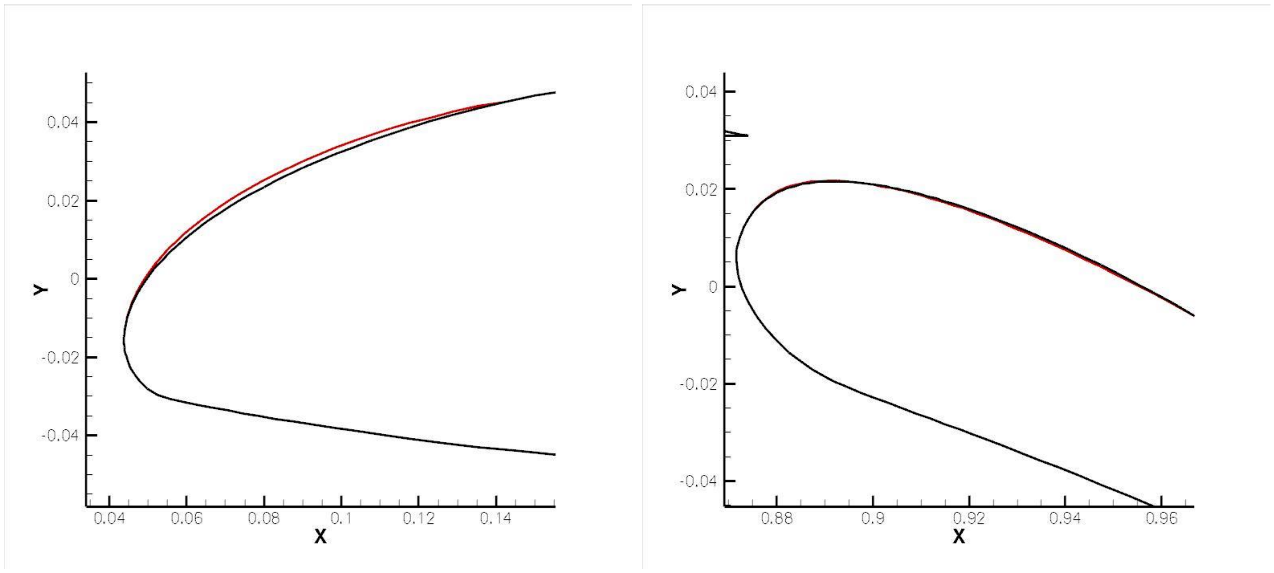


(b) Sensitivities of objective function (sensitivity of bump functions is an average value of all bump functions)

Figure 15: Optimization history for lift-constrained drag case including surface deformation using bump functions

Table 10 Lift, drag and objective values of optimized configuration for lift-constrained drag optimization including surface deformations using bump functions

	C_L	C_D	L
Optimized shape without bump functions	4.0235	0.076146	0.058168
Optimized shape	4.0181	0.075432	0.057261
Δ	-0.0054	-0.000714	-0.000907

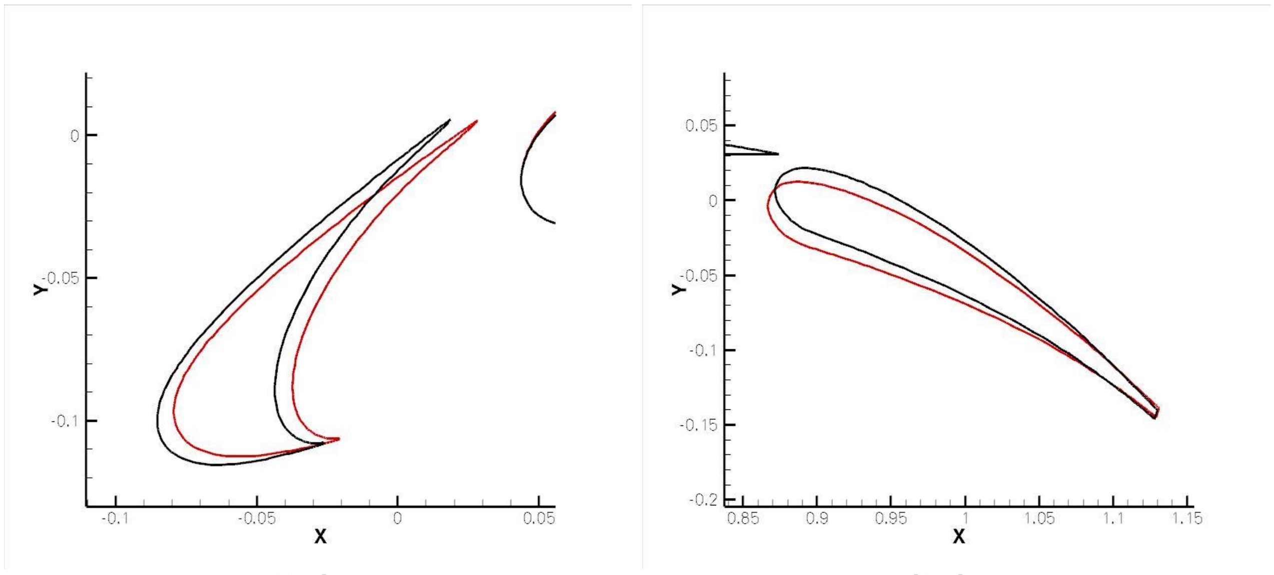


(a) Main element (b) Slat

Figure 16: Surface deformation of main element and flap. The airfoil locations are not changed and only surface deformation is shown. (redline represents the optimized shape and black line represents the initial shape)

Table 11 Final airfoil location design variables for lift-constrained drag optimization including surface deformations using bump functions

	Δx	Δy	$\Delta\alpha\ [deg]$
Slat	0.0076427	0.0016760	2.1141
Flap	-0.0020604	-0.0044401	-2.4980

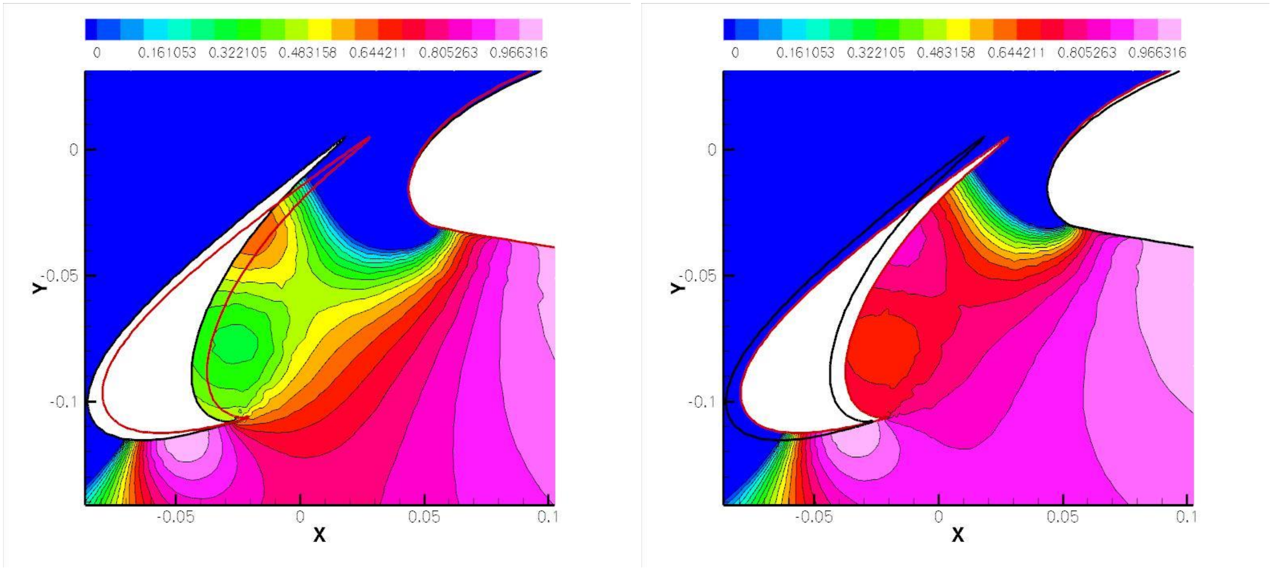


(a) Slat (b) Flap

Figure 17: Optimized shape of high-lift devices for lift-constrained drag optimization including surface deformations using bump functions(redline represents the optimized shape and black line represents the initial shape)

Table 12: Lift and drag coefficients of each element

	Slat	Main element	Flap
C_L (Initial airfoil)	0.59655	3.0421	0.39850
C_L (Optimized airfoil)	0.67690	2.9265	0.41472
C_D (Initial airfoil)	-0.51106	0.26647	0.32653
C_D (Optimized airfoil)	-0.58492	0.34944	0.31092



(a) Initial shape (b) Optimized shape

Figure 18: Distributions of pressure coefficient near the gap of slat.

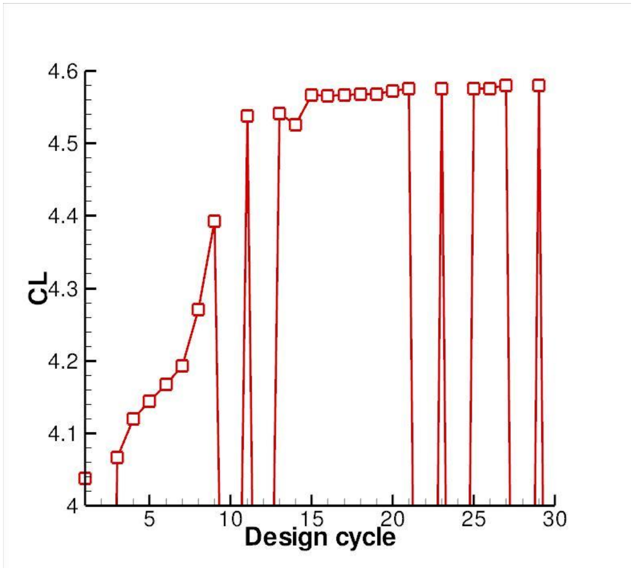
B. Improvement of maximum lift coefficient

One of the most important performance metrics of high-lift devices is the maximum lift coefficient. In this section, the shape is optimized to obtain a higher maximum lift coefficient. The flow angle is treated as one of the design variables and the objective function is defined as the inverse of the lift coefficient $1/C_L$. First, only the locations (rigging parameters) of the high-lift devices are optimized without any surface deformation. The initial flow angle is 16 [deg], and the bound of the design space is:

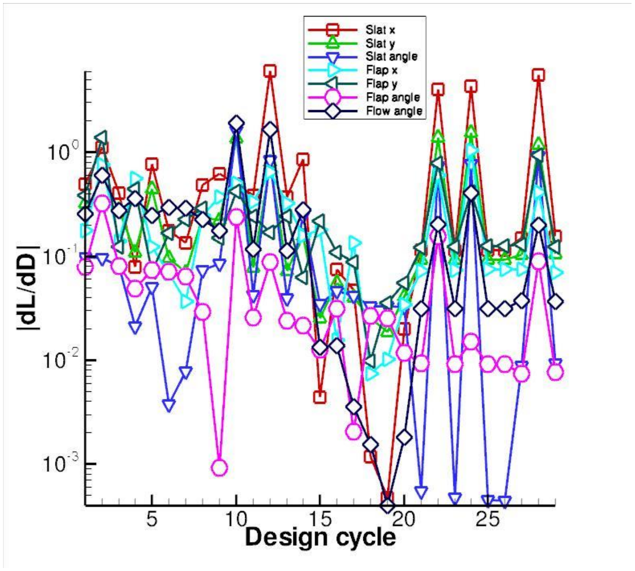
$$15 \le \alpha \le 35\ [deg] \tag{35}$$

The bounds of the other variables are the same as in the previous cases. The flow conditions are $M_\infty = 0.2$ and $Re = 9 \times 10^6$. The optimization histories of lift coefficient and the sensitivities of the objective function with respect to each design variable are shown in Fig. 19. The maximum lift coefficient of the optimized shape is 4.580 and the stall point is 23.30 [deg], while maximum lift coefficient of initial shape is 4.340 and the stall point is 21.95. Table 12 shows the final design variables and Fig. 20 shows the final optimized geometry.

Next, the airfoil shape is deformed by the bump functions. Ten bump functions are placed on the main element and flap as previously described. The initial locations of the flap and slat are the same as the values in Table 12. Figure 21 shows the optimization histories of lift coefficient and the sensitivities of the objective function with respect to each design variable. . The maximum lift coefficient of the optimized shape is 4.602 and the stall point is 23.37 [deg] Figure 22 shows the surface deformation compared with the initial shape. Table 13 shows the final values of design variables and Fig. 23 depicts the optimized geometries. Figure 24 compares the flow fields of the optimized shape and the initial shape around the slat. The flow angle of this figure is 23 [deg], and at this flow condition the initial shape exhibits separation at the leading edge of the main element. Meanwhile, the optimized shape suppresses the separation and achieves a higher stall point with higher $C_{L\,max}$.



(a) Objective function

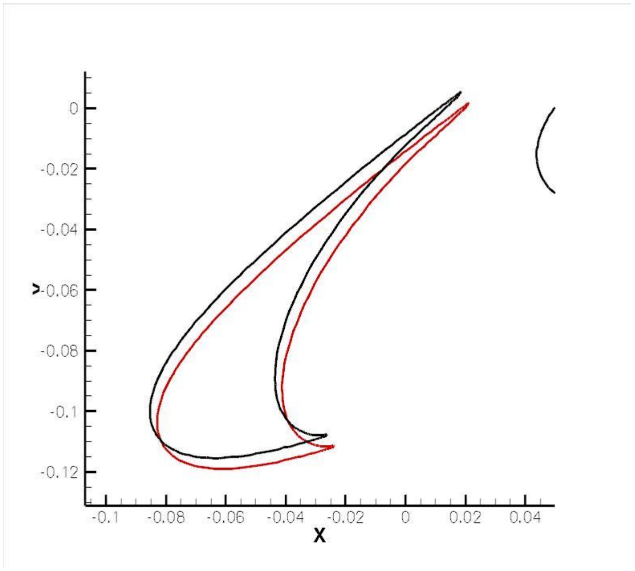


(b) Sensitivities of objective function

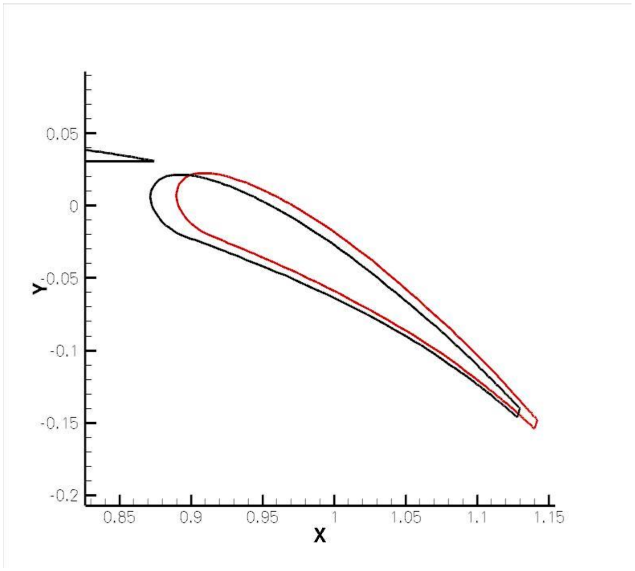
Figure 19: Optimization history of the lift coefficient for C_{Lmax} optimization

Table 12 Final design variables for C_{Lmax} optimization

	Δx	Δy	$\Delta\alpha$ [deg]
Slat	0.0024513	-0.0036617	0.085264
Flap	0.015216	-0.0031194	2.23308
Flow angle [deg]	23.302		

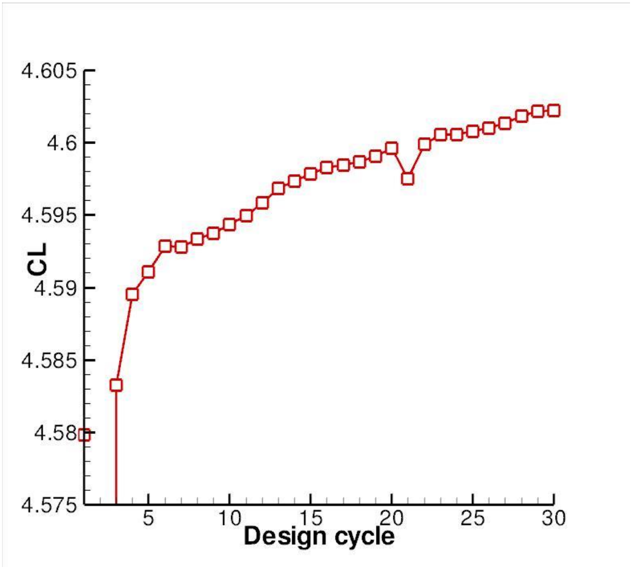


(a) Slat

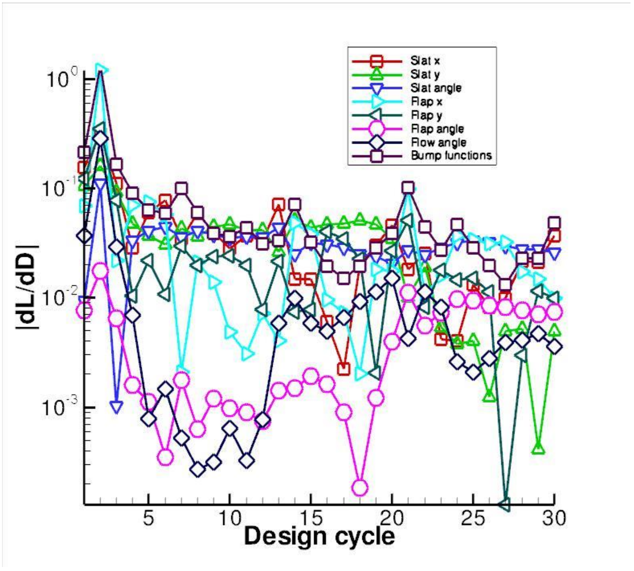


(b) Flap

Figure 20: Optimized shape of high-lift devices for C_{Lmax} optimization case (redline represents the optimized shape and black line represents the initial shape)

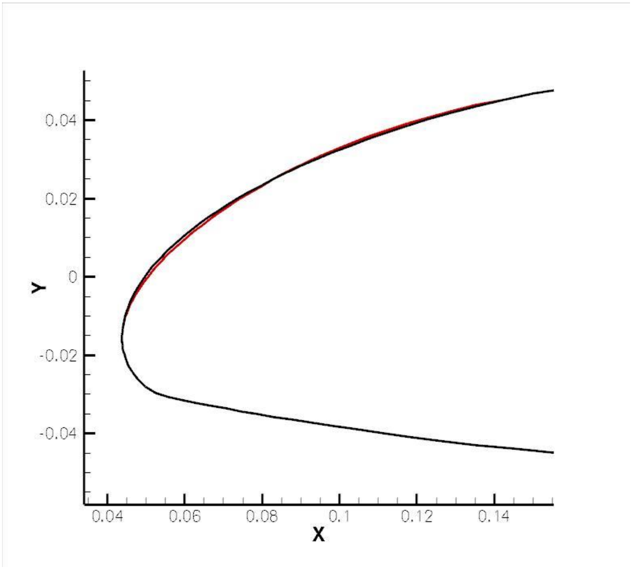


(a) Objective function

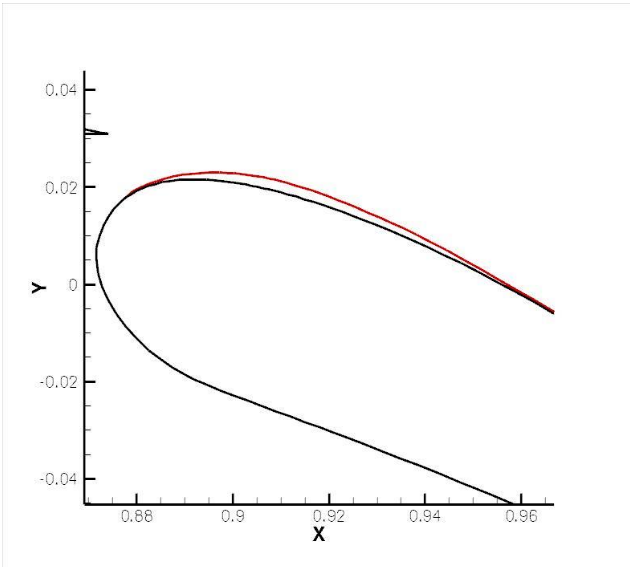


(b) Sensitivities of objective function (sensitivity of bump functions is an average value of all bump functions)

Figure 21: Optimization history of the objective function for C_{Lmax} optimization case with surface deformation using bump functions



(a) Main element

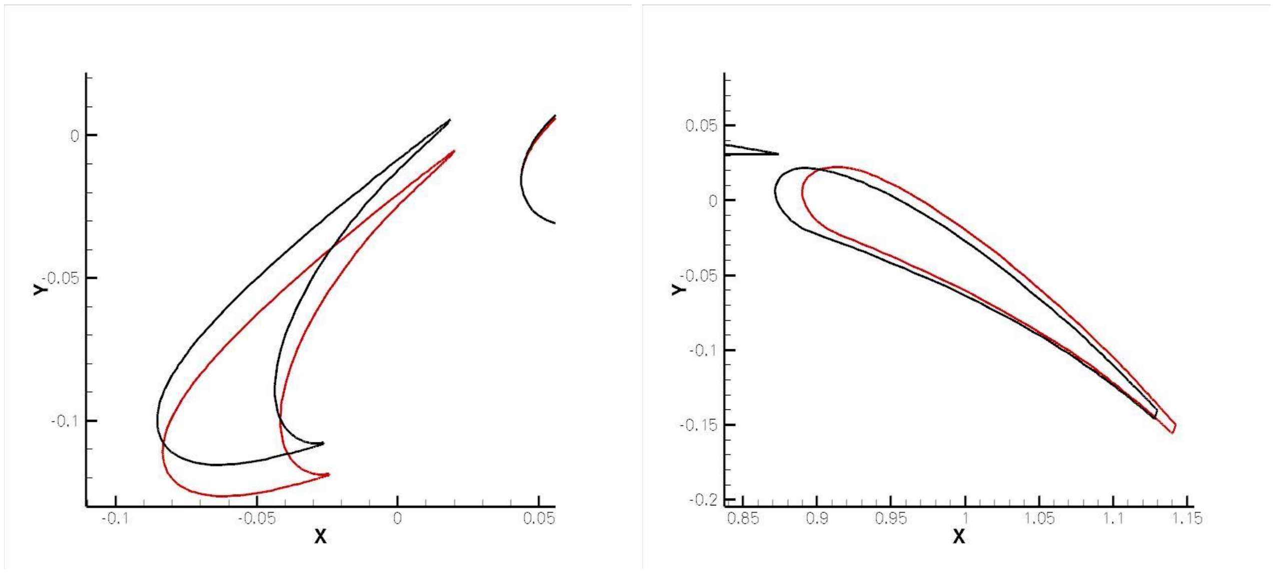


(b) Slat

Figure 22: Surface deformation of main element and flap for C_{Lmax} optimization case using bump functions. The locations are not changed and only surface deformation is shown. (redline represents the optimized shape and black line represents the initial shape)

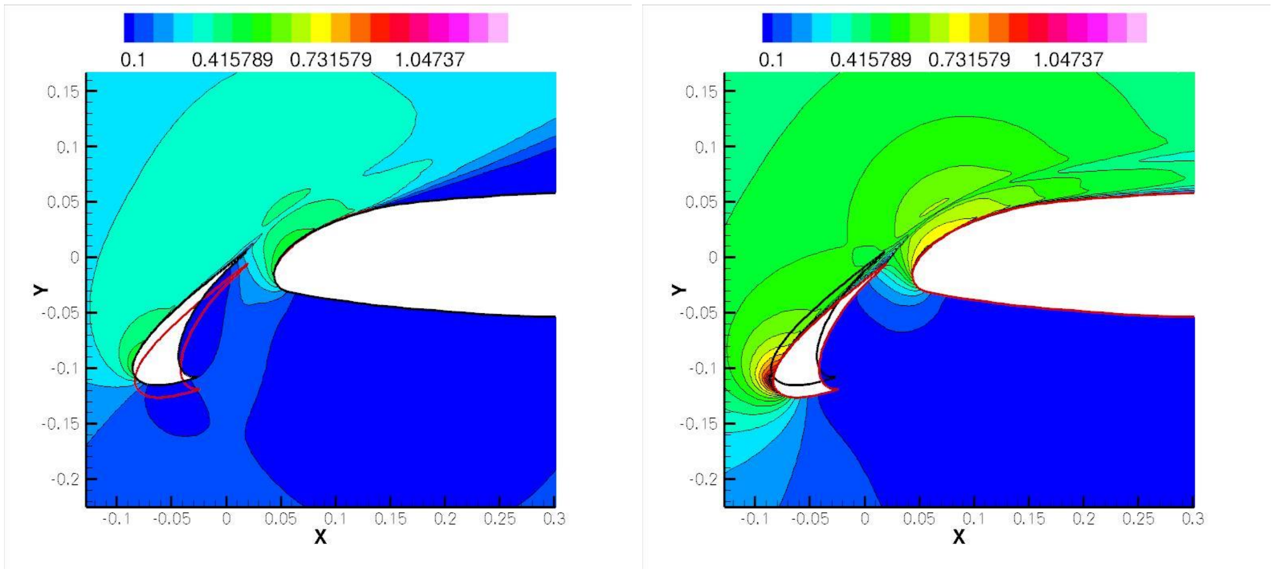
Table 13 Final design variables for C_{Lmax} optimization case (with bump functions)

	Δx	Δy	$\Delta\alpha\ [deg]$
Slat	0.0018141	-0.010908	-0.19197
Flap	0.015412	-0.0048131	2.2518
Flow angle [deg]	23.372		



(a) Slat (b) Flap

Figure 23: Optimized shape of high-lift devices for C_{Lmax} optimization case with surface deformation using bump functions (redline represents the optimized shape and black line represents the initial shape)



(a) Initial shape (b) Optimized shape

Figure 24: Mach number distributions of the flow around the slat ($\alpha = 23$ [deg])

VIII. Conclusions

In the present work, shape optimization of a high-lift airfoil is conducted using the NSU2D solver. A gradient-based optimization method is used and the sensitivity of the objective function with respect to the design variables is computed by the discrete adjoint method. In order to extend the design space, the mesh deformation method is modified by re-computing the stiffness matrix during the deformation process and solving the non-linear system. The design variables for the optimization consist of the location of high-lift devices (i.e. rigging parameters) and the surface shape of the main element and flap upper surface sections that area hidden in the stowed configuration. In the first optimization, the drag coefficient is minimized at constant lift. The optimized airfoil shows a 64 drag count reduction while maintaining almost the same lift coefficient as the initial airfoil. In the next case, the objective function is the inverse of the lift coefficient and flow angle is treated as one of the design variables in order to improve the maximum lift coefficient. In this case, the maximum lift coefficient of the optimized shape is 4.602 while the maximum lift coefficient of the initial shape is 4.340. Future work will investigate more effective optimization strategies, define more sophisticated and flexible shape design parameters than the Hicks-Henne bump functions, and consider multi-point optimization problems

References

[1] Goldberg, D. E. "Genetic Algorithms in Search, Optimization, and Machine Learning" Addison-Wesley Professional, 1989

[2] Jameson, A. "Aerodynamic Shape Optimization using the Adjoint Method" VKI Lecture Series on Aerodynamic Drag Prediction and Reduction, von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium

[3] Jameson, A., Alonso, J. J., Reuther, J. J., Martinelli, L. and Vassberg, J. C., "Aerodynamic shape optimization techniques based on control theory" AIAA Paper 98-2538.

[4] Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Optimization on Unstructured Meshes" AIAA Journal, Vol. 40, No. 6, 2002, pp. 1155–1163.

[5] Elliot, J. and Peraire, J., "Practical three-dimensional aerodynamic design by optimization" AIAA Journal, Vol. 35,No. 9, 1997, pp. 1479–1485.

[6] Soto, R. L. O. and Yang, C., "An Adjoint-Based Design Methodology for CFD Optimization Problems" AIAA-Paper 2003-0299.

[7] Nadarajah, S. and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization" AIAA Paper 2000-0667.

[8] Giles, M., Duta, M., and Muller, J., "Adjoint Code Developments Using Exact Discrete Approach" AIAA Paper 2001-2596.

[9] Kim, S., Alonso, J. J. and Jameson, A. "Design optimization of high-lift configurations using a viscous continuous adjoint method" AIAA Paper 2002-844.

[10] Eyi, S., Lee, K. D., Rogers, S. E. and Kwak, D. "High-lift design optimization using Navier-Stokes equations" Journal of Aircraft, Vol. 33, No. 3, 1996, pp. 499-504.

[11] Wild, J. "Multi-objective constrained optimisation in aerodynamic design of high-lift systems" International Journal of Computational Fluid Dynamics, Vol.22 Issue 3, 2008, pp. 153-168.

[12] Kanazaki, M., Tanaka, K., Jeong, S. and Yamamoto. K. "Multi-Objective Aerodynamic Exploration of Elements' Setting for High-Lift Airfoil Using Kriging Model" Journal of Aircraft, Vol. 44, No. 3, 2007, pp. 858-864.

[13]Rumsey, C. L. and Ying. S. X. "Prediction of high lift: review of present CFD capability" Progress in Aerospace Sciences, Vol. 38, Issue 2, 2002, pp. 145-180.

[14] Valarezo, W. O. and Mavriplis, D. J. "Navier-Stokes applications to high-lift airfoil analysis," Journal of Aircraft, Vol. 32, No. 3, 1995, pp. 618-624.

[15] Mavriplis, D.J., "Viscous Flow Analysis Using a Parallel Unstructured Multigrid Solver" AIAA Journal, Vol.38, No.11, 2000, pp. 2067-2076

[16] Mani, K., Lockwood, B. and Mavriplis, D. J. "Adjoint-Based Unsteady Airfoil Design Optimization with Application to Dynamic Stall" AHS Forum 68, Fort Worth TX, 2012

[17] Byrd, R. H., Lu, P. and Nocedal, J. "A Limited Memory Algorithm for Bound Constrained Optimization" SIAM Journal on Scientific and Statistical Computing, Vol. 16, 5, 1995, pp. 1190-1208.

[18] Spalart, P. R. and Allmaras, S. R. "A One-Equation Turbulence Model for Aerodynamic Flows" AIAA Paper 92-0439

[19] Mavriplis, D. "Multigrid Techniques for Unstructured Meshes," Notes prepared for 26th Computational Fluid Dynamics Lecture Series Program of the von Karman Institute of Fluid Dynamics, Rhode St Genese, Belgium, 1995.

[20] Saad Y. and Schultz, M. H. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems" SIAM J. Sci. Stat. Comput., 7, 1986, 856-869.

[21] Batina, J.T. "Unsteady Euler airfoil solutions using unstructured dynamic meshes" AIAA Journal, Vol. 28, No.8, 1990, pp. 1381–1388.

[22] Venkatakrishnan, V. and Mavriplis, D. J. "Implicit method for the computation of unsteady flows on unstructured grids" Journal of Computational Physics, Vol.127, Issue 2, 1996, 380–397.

[23] Baker, T. and Cavallo, P. A. "Dynamic adaptation for deforming tetrahedral meshes" AIAA Paper 1999-3253.

[24] Nielsen, E. J. and Anderson, W. K. "Recent improvements in aerodynamic optimization on unstructured meshes" AIAA Journal, Vol. 40, No.6, 2002, pp. 1155–1163.

[25] Hicks, R. and Henne, P. "Wing Design by Numerical Optimization" Journal of Aircraft , Vol. 15, No. 7, 1978, pp. 407-412.

[26] Mavriplis, D.J. "A Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes" AIAA Paper 2006-0050.

[27] Dwight, R. and Brezillon, J. "Effect of Various Approximations of the Discrete Adjoint on Gradient-Based Optimization" AIAA Paper 2006-0690

[28] Mavriplis, D. J. "Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes" AIAA Journal, Vol. 44, No.1, 2006, pp. 42-50.

[29] Mavriplis, D. J. "Discrete Adjoint-Based Approach for Optimization Problems on Three-Dimensional Unstructured Meshes" AIAA Journal, Vol. 45-4, 2007, pp. 741-750.

[30] Mani, K. and Mavriplis, D. J. "Unsteady Discrete Adjoint Formulation for Two-Dimensional Flow Problems with Deforming Meshes" AIAA Journal, Vol. 46, No. 6, 2008, pp. 1351-1364.