AMSC663/664 Project Proposal: Analysis of the Adjoint Euler Equations as used for Gradient-based Aerodynamic Shape Optimization

Dylan Jude

October 6, 2016

Abstract

Adjoint methods are often used in gradient-based optimization because they allow for a significant reduction of computational cost for problems with many design variables. The proposed project focuses on the use of adjoint methods for two-dimensional airfoil shape optimization using Computational Fluid Dynamics to model the Euler equations.

1 Background

Airfoil shape optimization is the process by which the shape of an airfoil is modified with the objective of improving the aerodynamics of that airfoil. Certain aerodynamic properties of airfoils could be, for example, lift, drag, or pressure distribution. In control theory, aerodynamic properties can be formalized mathematically as a cost function. Semantically this is introduced with the goal of minimizing the "cost" of an airfoil shape, and therefore minimizing the cost function.

The same way we can mathematically define a cost function, we can also define design variables which control the shape of the given airfoil. As an example, figure 1 shows an airfoil whose general shape can be altered using two variables α and c.

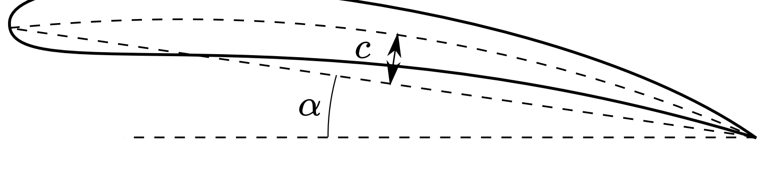

Figure 1: Example Airfoil Design Variables

1.1 Choosing a Cost Function

Assuming we already have an airfoil shape and corresponding two-dimensional mesh, Computational Fluid Dynamics (CFD) can be used to solve the Euler equations. The Euler equations are presented in the following section however initially can be simplified to a black box. From the airfoil, we obtain the flow solution, and from the flow solution we can obtain a pressure at every point on the airfoil. A simple cost function typically chosen in airfoil shape optimization compares the current pressure distribution to a desired pressure distribution. This is illustrated in figure 2. The x-axis follows the chord of the airfoil and therefore the two lines of each color represent the pressure at that location along the airfoil on the top and bottom surfaces.

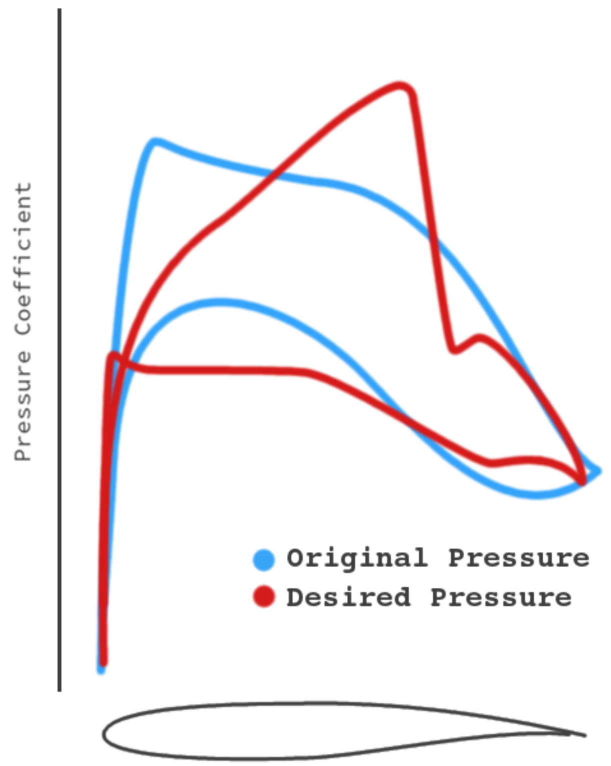

Figure 2: Comparison of a pressure distribution with the desired distribution [1]

A mathematical formulation of a cost function for this comparison could be:

$$I_c(\alpha) = \oint_{airfoil} (P - P_d)^2 ds \tag{1}$$

where α is a set of design variables used to obtain the airfoil shape. For an airfoil defined by a set of N discrete points, for convenience we can force the X-coordinates of each airfoil to be the same so that we can simplify the cost function:

$$I_c(\alpha) = \sum_{i=0}^{N} (P_i - P_{d,i})^2 \tag{2}$$

1.2 Finding Sensitivities

For an example problem with two design variables α_1 and α_2, the sensitivity of the cost function I_c to these design variables is

$$\frac{\partial I_c}{\partial \alpha_1}, \quad \frac{\partial I_c}{\partial \alpha_2} \tag{3}$$

Each of these partial derivatives could be approximated using a finite-difference, or "brute-force" approach where for each variable

$$\frac{\partial I_c}{\partial \alpha_1} = \frac{I_c(\alpha_1 + \delta\alpha_1) - I_c(\alpha_1)}{\delta\alpha_1} \tag{4}$$

For two design variables, this requires three CFD calculations for $I_c(\alpha_{1,2})$, $I_c(\alpha_1 + \delta\alpha_1)$, and $I_c(\alpha_2 + \delta\alpha_2)$. Especially for complex, 3-dimensional flow problems, obtaining the solution using CFD can take on the order of hours or days. Using brute-force finite differences to find the sensitivities of many design variables would therefore be a long and painstaking process.

The goal of using adjoint methods, as presented in the following section, is to eliminate the dependence of the cost function I_c on the flow solution so that all design variable sensitivities can be solved at once.

2 Approach

Since the adjoint Euler equations are derived from the Euler equations, this section will start with an overview of the Euler equations as solved by an in-house CFD solver. This overview will be followed by a brief derivation of the Adjoint Euler equations for the interior domain and boundary.

2.1 Euler Equations

The Euler equations are a subset of the compressible Navier-Stokes equations for inviscid flow. In two dimensions, these equations consist of four equations: one for the conservation of mass, two for the conservation of momentum in x and y, and one for the conservation of energy. In the following description of the flow equations, standard usage of flow variables are used. ρ is the fluid density, $\vec{u}$ is the fluid velocity composed of $u_1 \text{and} u_2$, E is total energy (internal and kinetic), and p is pressure.

2.1.1 Conservation of mass

Over a volume Ω, the conservation of mass in integral form is

$$\frac{d}{dt}\int_{\Omega}\rho d\Omega = 0 \tag{5}$$

which using the Reynolds Transport Theorem can be re-written as

$$\int_{\Omega}\left[\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\vec{u})\right]d\Omega = 0 \tag{6}$$

2.1.2 Conservation of Momentum

The time rate of change of momentum in volume Ω in the direction x_i is

$$\frac{d}{dt}\int_{\Omega}\rho u_i d\Omega = \int_{S} F_i dS$$

where F_i represents the stresses acting on the surface S of the domain. Again the Reynolds Transport Theorem can be used to simplify the equation to

$$\frac{d}{dt}\int_{\Omega}\rho u_i d\Omega = \int_{\Omega}\left[\rho\frac{Du_i}{Dt}\right]d\Omega \tag{7}$$

where $\frac{D}{Dt}$ is the material derivative, sometimes called the convective derivative.

For inviscid flow, there are no viscous stresses and the only force acting on the surface of the domain is pressure p. Since pressure acts inward,

$$\int_{S} F_i dS = \int_{S} -p\delta_{ki}n_k dS$$

Using Gauss' theorem, the surface integral is converted to a volume integral

$$\int_{S} -p\delta_{ki}n_k dS = \int_{\Omega} -\frac{\partial}{\partial x_k}(p\delta_{ki})d\Omega$$

and combining with equation (7), we obtain the integral form of the momentum equation:

$$\int_{\Omega}\left[\rho\frac{Du_i}{Dt} + \frac{\partial p}{\partial x_i} = 0\right]d\Omega \tag{8}$$

2.1.3 Conservation of Energy

The conservation of energy in a control volume without body forces and without a heat source is related to the pressure-work done on the surface of the control volume and the rate of heat loss through the surface.

$$\frac{d}{dt}\int_{\Omega} \rho E d\Omega = \int_{S} p\delta_{ki}u_k n_k dS - \int_{S} -q_k n_k dS \tag{9}$$

In the above equation, q_k is defined by the Fourier law of heat conduction as $q_k = -\kappa(\partial T/\partial x_k)$. Again using the Reynolds Transport theorem, Gauss' theorem, and the definition of the material derivative, the energy equation can be simplified to:

$$\int_{\Omega} \left[\rho \frac{D}{Dt}(E) + \frac{\partial}{\partial x_k}(-p_{ki}u_i + q_k)\right] d\Omega = 0 \tag{10}$$

2.1.4 Euler Equations

The Euler equations for the conservation of mass, momentum, and energy can be written in conservative form as

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}_{c,i}}{\partial x_i} = 0 \quad \text{in domain } \Omega, \quad i = 1,2 \tag{11}$$

$$\vec{Q} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho E \end{bmatrix}, \quad \vec{F}_{c,1} = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ (\rho E + p)u_1 \end{bmatrix}, \quad \vec{F}_{c,2} = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ (\rho E + p)u_2 \end{bmatrix} \tag{12}$$

To close the equations, the pressure is defined by the equation of state

$$p = \rho(\gamma - 1)\left[E - \frac{1}{2}||\vec{u}||^2\right] \tag{13}$$

where γ is the ratio of specific heats. Using a transformation to a Cartesian grid of coordinates ξ_i, the Euler equations can be written as

$$\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{f}_{c,i}}{\partial \xi_i} = 0 \tag{14}$$

$$\vec{q} = J^{-1}\begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ e \end{bmatrix}, \quad \vec{f}_{c,1} = J^{-1}\begin{bmatrix} \rho V_1 \\ \rho u_1 V_1 + \xi_{1,1}p \\ \rho u_2 V_1 + \xi_{1,2}p \\ (e + p)V_1 \end{bmatrix} \tag{15}$$

where J is the Jacobian of the coordinate transformation and V_i is the contravariant velocity in the ξ_i direction:

$$V_i = u_1\xi_{i,1} + u_2\xi_{i,2} \tag{16}$$

the discretization and mapping between Cartesian and curvilinear domains is covered in detail in the work of J. Blazek [2].

2.1.5 Boundary Conditions

For a "O" mesh topology, the 2D grid defined by coordinates `j` and `k`, illustrated in figure 3. The `jmin` and jmax boundaries are periodic boundaries and the far-field can be approximated by a Dirichlet boundary by setting free-stream conditions. At the airfoil wall, the flow tangency condition must be satisfied:

$$(\vec{u} \cdot \vec{n}_{wall}) = 0 \tag{17}$$

where $\vec{n}_{wall}$ is the outward pointing wall-normal vector.

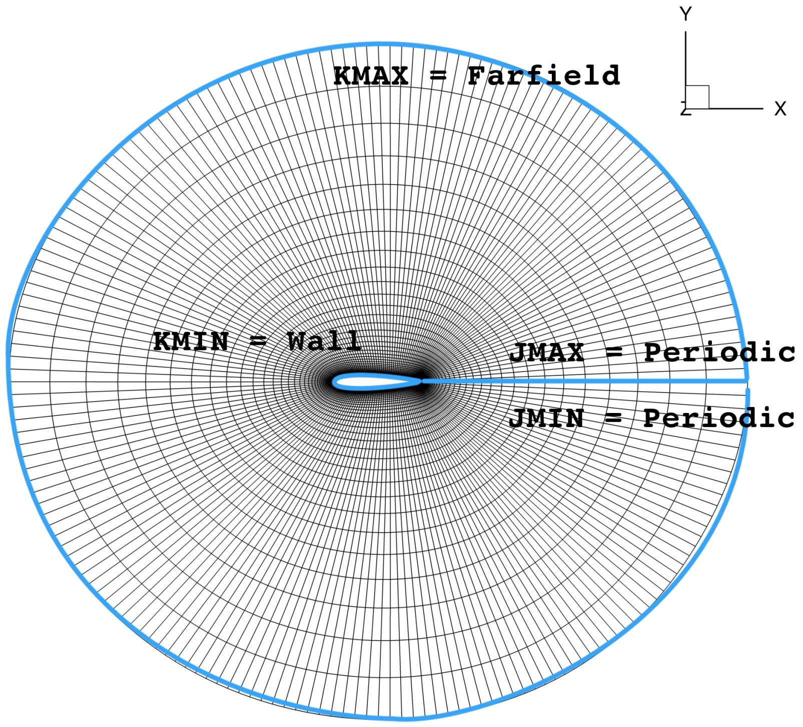

Figure 3: O-Mesh Topology

2.2 Adjoint Euler Equations

The Adjoint to a set of equations is usually defined in one of two ways. The first uses a linear algebra approach to define the problem and the other uses the method of Lagrange variables. Since both methods are equivalent and previous studies in computational aerodynamic design tend to prefer the Lagrangian multiplier approach [3], this section will also motivate the use of adjoint methods using Lagrangian multipliers.

2.2.1 General Derivation

As presented in the previous section, we can define a cost function to minimize during the design process. This cost function can be a defined over the whole domain and/or over the boundary of the domain. The cost function is a function of both the flow solution q and geometry X and can be written as

$$\delta I = \int_{Boundary} \delta M(q, X) dB + \int_{Domain} \delta P(q, X) dD \tag{18}$$

For simple problems, X could be the vector of design variables however more generally it represents the geometry of the grid. In CFD, the grid geometry consists of the cell volumes, face areas, and face vectors. These metrics appear directly in the flow equations as ξ_i, shown in equation (16), and J in equation (15).

Equation (18) can be further broken down into parts dependent on q and X:

$$\begin{aligned} \delta I &= \delta I_q + \delta I_X \\ &= \int_{Boundary} \left[\frac{\partial M}{\partial q} \delta q + \frac{\partial M}{\partial X} \delta X \right] dD + \int_{Domain} \left[\frac{\partial P}{\partial q} \delta q + \frac{\partial P}{\partial X} \delta X \right] dB \end{aligned}$$

Now recalling the steady Euler equations, we can define the residual R and its dependence on q and X as:

$$\begin{aligned} R &= \left[\frac{\partial f_i}{\partial \xi_i} \right] &&= 0 \\ \partial R &= \left[\frac{\partial R}{\partial q} \right] \delta q + \left[\frac{\partial R}{\partial X} \right] \delta X &&= 0 \end{aligned}$$

Since this is equal to 0, we can add this to equation (18) with a Lagrangian multiplier ψ:

$$\delta I = \delta I_q + \delta I_X - \psi(\delta R_q + \delta R_X)$$

To eliminate dependence on q we focus on choosing ψ so that

$$\delta I_q + \psi(\delta R_q) = 0$$

$$\begin{aligned} R = \frac{\partial f_i}{\partial \xi_i} \qquad &= 0 \\ \frac{\partial R}{\partial q}\delta q = \frac{\partial}{\partial q}\left[\frac{\partial}{\partial \xi_i}\delta f_i\right] &= 0 \end{aligned}$$

As an integral over the whole domain, introducing the Lagrange multiplier ψ as the weak form variable:

$$\int_D \frac{\partial}{\partial \xi_i}\delta f_i = \int_D \psi^T \frac{\partial}{\partial \xi_i}\delta f_i = 0$$

integrating by parts

$$\int_B \left[n_i \psi^T \delta f_i\right] dB - \int_D \left[\frac{\partial \psi}{\partial \xi_i}\delta f_i\right] dD = 0$$

since this is zero, we can add it to the δI equation. ψ is then a Lagrangian multiplier for the optimization of I with constraint equation $R = 0$.

$$\begin{aligned} \delta I = &\int_B \delta M(q, X) dB + \int_D \delta P(q, X) dD \\ &+ \int_B \left[n_i \psi^T \delta f_i\right] dB - \int_D \left[\frac{\partial \psi}{\partial \xi_i}\delta f_i\right] dD \end{aligned}$$

we then pick ψ to eliminate all dependence on δq.

2.2.2 Interior Equations

Out cost function, as previously presented in equation (2), involves only an integral of pressure over the surface of the airfoil. Since the surface of the airfoil is along the boundary, the P for this case is 0. The interior equation then becomes:

$$\begin{aligned} -\int_D \left[\frac{\partial \psi}{\partial \xi_i}\frac{\partial f_i}{\partial q}\right] dD &= 0 \\ \frac{\partial \psi}{\partial \xi_i}\frac{\partial f_i}{\partial q} &= 0 \end{aligned}$$

using the definition of flux Jacobian $A_i = \partial f_i / \partial q$, the adjoint residual is

$$[A_i]^T \frac{\partial \psi}{\partial \xi_i} = 0 \tag{19}$$

This form looks very similar to the original Euler equation residual, which was

$$\frac{\partial f}{\partial \xi_i} = [A_i]^T \frac{\partial q}{\partial \xi_i} = 0$$

2.3 Gradient-based optimization

From the solution to the adjoint Euler equations, we are left with solving for the variation of the cost function with the geometry X but holding q constant:

$$\delta I = \left\{ \frac{\partial I^T}{\partial X} - \psi^T \left[\frac{\partial R}{\partial X} \right] \right\} \delta X$$

This equation depends on grid geometry X, which as shown in section (2.2) is not typically a simple array of the design variables. Instead the above sensitivities are found with respect to grid metrics X, and the variation of X with the design variables α_i can be found through brute-force finite-difference grid generation. Re-generating meshes for every design variable α is typically fast compared to a flow calculation, especially in 2D cases considered for this project.

Once the sensitivities are computed using this approach, each design variable can be altered in the direction of steepest descent. Though there are other optimization methods that would likely result in faster convergence to the minimum of the cost function, using the method of steepest descent is the simplest method and has shown to work well for simple airfoil optimization [4].

$$\alpha_i^{n+1} = \alpha_i^n - \lambda \frac{\partial I}{\partial \alpha_i} \tag{20}$$

2.4 Hicks-Henne Bump Functions

The entire airfoil design process relies upon a chosen set of design variables to alter the airfoil shape. The design variables need to be defined by continuous functions in order for gradient-based optimization to work well. One such method of parameterizing airfoil perturbations was presented by Hicks and Henne in 1977, and is commonly referred to as "Hicks-Henne Bump Functions"[5]. These bump functions are sinesoidal perturbations applied at different locations along the airfoil. A commonly used for is

$$b(x) = a \left[sin \left(\pi x^{\frac{log(0.5)}{log(t_1)}} \right) \right]^{t_2}, \quad \text{for } 0 \leq x \leq 1 \tag{21}$$

In this bump equation, t_1 locates the maximum of the bump in $0 \leq x \leq 1$, t_2 controls the width of the bump, and a controls the bump amplitude. Each bump has three design variables. Figure 4 shows an example of random perturbations made to t_1 and a on 6 bumps while keeping t_2 constant. From the original shape (dotted line), this example with 6 bump function, a total of 12 variables, were able to significantly alter the shape of an airfoil.

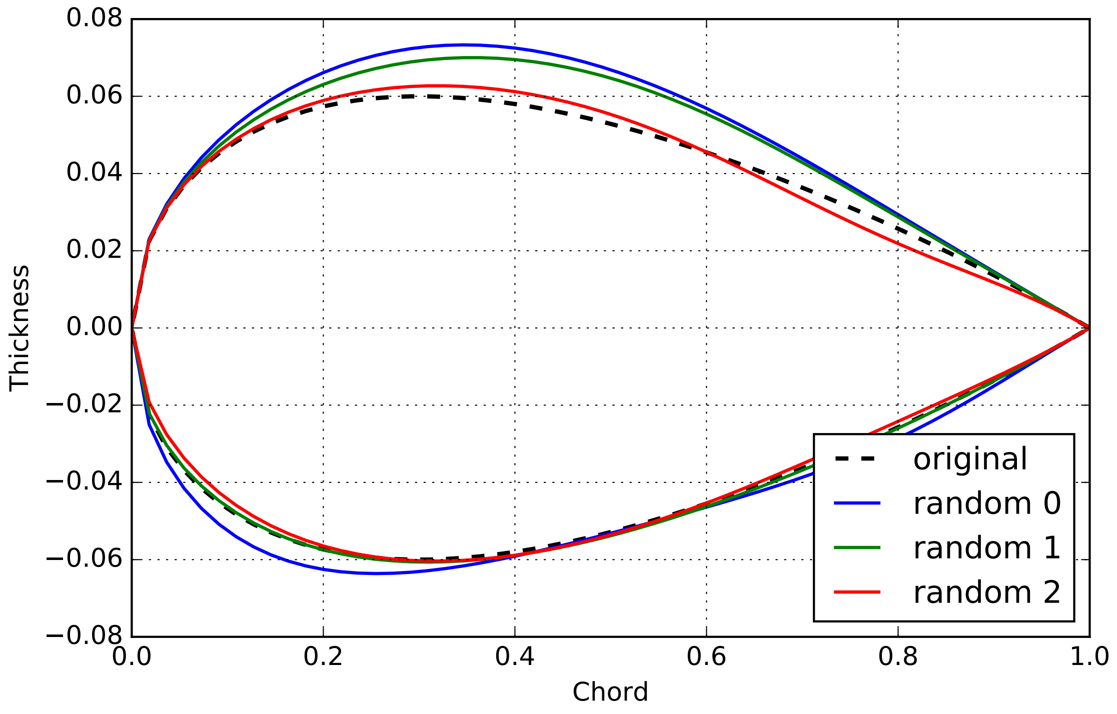

Figure 4: Hicks-Henne Bump functions with random variable perturbations

3 Implementation

3.1 System Description

The adjoint Euler equation solver will be applied to 2D compressible, inviscid airfoil optimization problems. An example airfoil mesh is illustrated in figure 3, however a real mesh would extend much further out from the airfoil surface. The initial chosen airfoil mesh will have 192 x 32 points for a total of 6144. Subsequent tests can be made on finer meshes depending on results.

3.2 Auto-differentiation

Solving the adjoint equation is essentially a differentiation of the flow residual R, many research groups have shown successful implementations of adjoint methods using auto-differentiation [6]. These methods tend to be much less efficient than by-hand adjoint solvers however have shown to produce accurate results [7].

As a first pass at implementing adjoint methods, it is convenient to rely upon auto-differentiation software such as *Tapenade* [8] to quickly develop an adjoint solver. This can be compared with sensitivities from finite-difference ("brute-force") gradients.

3.3 System Description

The full system is composed of the following components:

1. Euler Equation Solver (baseline available in-house, in C++)
2. Grid-generator (baseline available in-house, in C++)
3. Method of changing airfoil shape
    - Hicks-Henne Bump Function [5]
4. Adjoint Euler Solver
    - Auto-differentiation with *Tapenade* [8]

$$\psi^T \left[\frac{\partial R}{\partial q}\right] = \frac{\partial I^T}{\partial q}$$

    - By-hand discrete solver in C++, based notes from Jameson and Nadarajah [9]
    - Possibly parallelized with OpenMP or CUDA (time permitting)

4 Validation

Sensitivities from the discrete adjoint equations should match very well with finite-difference gradients. The validation is outlined as follows:

1. Use the cost function from equation (2) and a single bump function with 2 variables
2. Run the flow solver 3 times to find the
    - initial solution
    - sensitivity to the first variable, equation (4)

- sensitivity to the second variable, equation (4)

3. Run the auto-differentiated adjoint solver and obtain new sensitivities
4. Run the by-hand adjoint solver and obtain new sensitivities
5. Sensitivities should match for all methods

By adding more Hicks-Henne bumps, the validation can be extended to multiple variables to also compare computational costs between each method of finding variable sensitivities.

5 Testing: Reverse Design

A common case in testing adjoint methods is “reverse-design” using the same target-pressure cost function presented in equation (2). Nadarajah and Jameson [4] conducted this test between two airfoils, for both of which the airfoil shapes and pressure solutions are known. The two distributions and shapes are shown in figure 5.

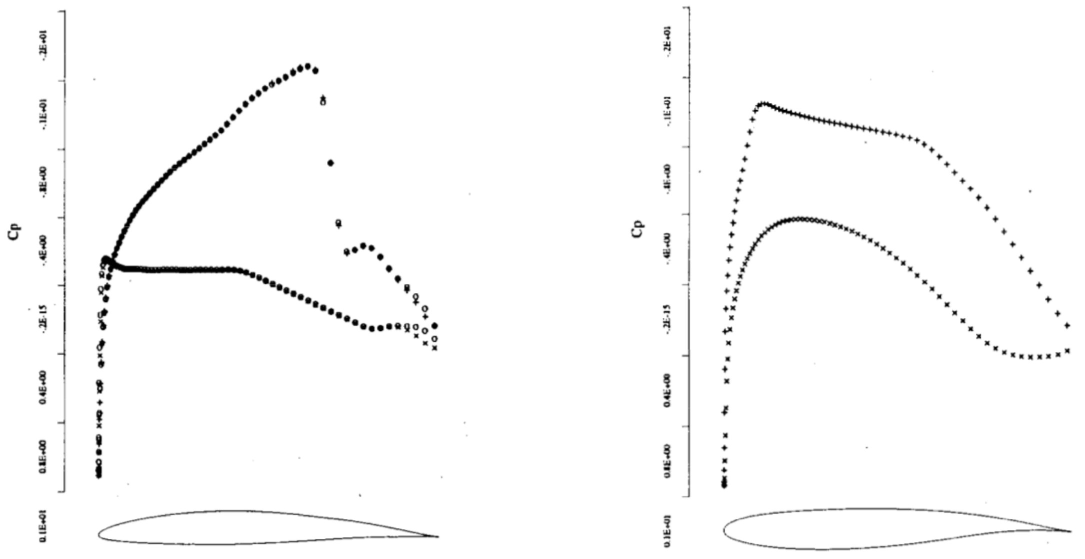

Figure 5: Comparison between two airfoil shapes and pressure distributions [4]

Starting with the shape of airfoil 2 but using the target pressure of airfoil 1, the design process should produce an airfoil matching the target pressure of airfoil 1 and consequently also matching the shape of airfoil 1. This procedure can be initially done for simple cases where the design-parameters are known for two airfoils and then extended to hopefully compare with results from Nadarajah and Jameson [4].

6 Schedule

6.1 Phase 1: Software Preparation

- Altering an existing 2D CFD solver
  - for future addition of adjoint solver
  - for auto-diff software compatibility
  - for output of simple pressure distribution
  - estimated time: 2 weeks
- Altering an existing mesh-generator
  - to automate mesh generation from airfoil shape using shell scripts
  - to allow a Hicks-Henne bump function for airfoil perturbation
  - estimated time: 2 weeks

6.2 Phase 2: Auto-differentiation and Finite-Difference

- Brute-force finite difference method for sensitivities
- Implement complex-variable method for sensitivities
- Apply auto-differentiation, validated by brute-force finite-difference methods
- estimated time: 4 weeks

6.3 Phase 3: Implementing Discrete Adjoint Equations

- Following methodology outlined by Jameson et. al. (2000).
- estimated time: 4 weeks

6.4 Phase 4: Validation

- Validation of discrete adjoint sensitivities compared to auto-differentiation and finite-difference results.
- Validation applied at a number of airfoil configurations: subsonic and transonic.
- estimated time: 2 weeks

6.5 Phase 5: Testing

- Set two airfoil configurations with known geometries and solutions, test a reverse-design cycle.
- Repeat for subsonic, transonic conditions
- estimated time: 3 weeks

7 Milestones

Functioning airfoil perturbation function in combination with mesh generation and 2D Euler Solver.	Late October
Functioning brute-force method for sensitivity of Pressure cost function to airfoil perturbation variables.	Early November
Auto-differentiation of Euler CFD solver.	Late November
Validate auto-diff and brute-force method for simple reverse-design perturbations.	Mid December
Hand-coded explicit discrete adjoint solver.	Mid December
Implicit routine for discrete adjoint solver.	Late January
Validate discrete adjoint solver against auto-diff and brute-force methods.	Mid February
Test discrete adjoint solver with full reverse-design cases.	Mid March

8 Deliverables

1. Airfoil perturbation and grid-generation code.
2. Auto-differentiated Euler CFD code.
3. Results for auto-diff and finite-difference tests on simple reverse-design perturbation problem.
4. Discrete adjoint solver code
5. Results for adjoint code validation with finite-difference and auto-diff tests
6. Results for a full reverse-design cycle test
7. Report on achievements and results

References

[1] S. Nadarajah, *The Discrete Adjoint Approach to Aerodynamic Shape Optimization.* PhD thesis, Stanford University Department of Aeronautics and Astronautics, 2003.

[2] J. Blazek, *Computational Fluid Dynamics: Principles and Applications (Second Edition).* Oxford: Elsevier Science, second edition ed., 2005.

[3] M. Giles and N. Pierce, "An Introduntion to the Adjoint Approach to Design," *Flow, Turbulance and Combustion*, vol. 65, no. 3, pp. 393–415, 2000.

[4] S. Nadarajah and A. Jameson, "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization," *Aerospace Sciences Meetings*, 2000.

[5] R. HICKS and P. HENNE, *Wing design by numerical optimization.* Aircraft Design and Technology Meeting, American Institute of Aeronautics and Astronautics, Aug 1977.

[6] M. B. Giles, D. P. Ghate, and M. C. Duta, "Using Automatic Differentiation for Adjoint CFD Code Development," *Post SAROD Workshop*, 2005.

[7] J.-D. Müller and P. Cusdin, "On the performance of discrete adjoint CFD codes using automatic differentiation," *International Journal for Numerical Methods in Fluids*, vol. 47, no. 8-9, pp. 939–945, 2005.

[8] L. Hascoët and V. Pascual, "TAPENADE 2.1 user's guide," 2004.

[9] S. Nadarajah and A. Jameson, "Optimal Control of Unsteady Flows Using a Time Accurate Method," *Multidisciplinary Analysis Optimization Conferences*, no. June, pp. —-, 2002.