Analysis of the Adjoint Euler Equations as used for Gradient-based Aerodynamic Shape Optimization

Dylan Jude
Graduate Research Assistant

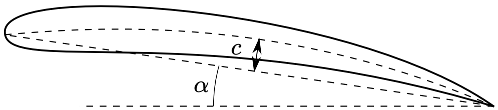
University of Maryland
AMSC 663/664

September 29th, 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 solve the steady Euler equations.

Background



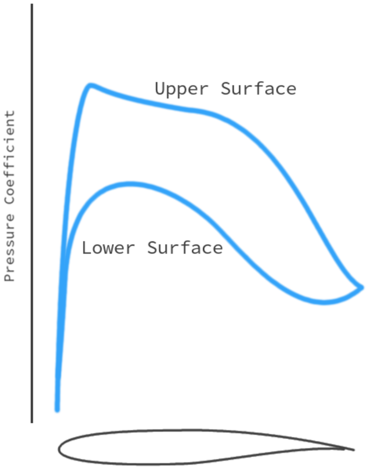
Airfoil Example Problem

Given n design variables $\alpha_1, \alpha_2, \alpha_3 ... \alpha_n$ we can achieve a change in airfoil shape:

Background

Given an airfoil shape and flow solver, we can get a pressure distribution over the airfoil.

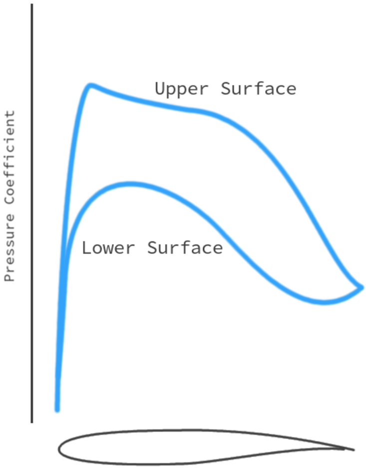
The goal of simple airfoil design could be to achieve an improved **pressure distribution** by altering the **airfoil design variables**.



Background

Given an airfoil shape and flow solver, we can get a pressure distribution over the airfoil.

The goal of simple airfoil design could be to achieve an improved **pressure distribution** by altering the **airfoil design variables**.

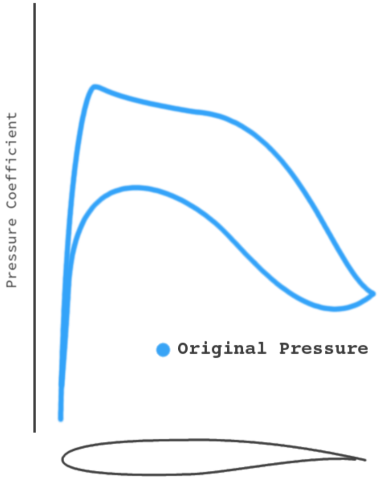


Approach

We want to minimize the cost function I_c in the design process

Mathematically:

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

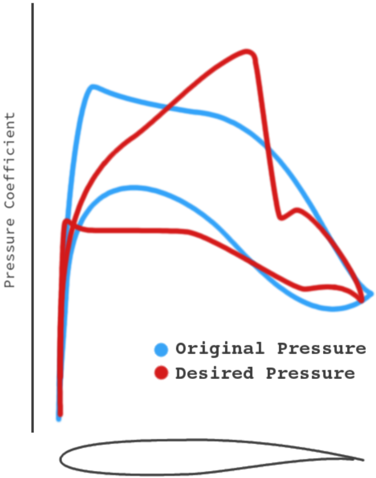


Approach

We want to minimize the cost function I_c in the design process

Mathematically:

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



Approach: CFD

$$I_c = I_c(\text{Fluid Flow Equations})$$

The compressible Navier-Stokes equations in differential form with no source:

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

$$\vec{Q} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ e \end{bmatrix}, \quad \vec{F}_{c,1} = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ (e+p) u_1 \end{bmatrix}, \quad \vec{F}_v = 0$$

(standard use of variables for density, velocity, pressure, and energy)

Approach: Discretization

On a curvilinear grid using a coordinate transformation:

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

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

Where ξ_i is the cartesian grid coordinate, J is the Jacobian of the coordinate transform and V_i is the contravariant velocity.

Approach: Discretization

On a curvilinear grid using a coordinate transformation:

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

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

Where ξ_i is the cartesian grid coordinate, J is the Jacobian of the coordinate transform and V_i is the contravariant velocity.

Approach: Design Process

As an example, using 2 design variables α_1, α_2, the sensitivities would be:

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

Which can be calculated using a brute-force approach:

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

Three CFD flow calculations to find

$$I_c(\alpha_{1,2}), \quad I_c(\alpha_1 + \delta\alpha_1), \quad I_c(\alpha_2 + \delta\alpha_2)$$

For N design variables, $N + 1$ CFD calculations required.

Approach: Design Process

As an example, using 2 design variables α_1, α_2, the sensitivities would be:

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

Which can be calculated using a brute-force approach:

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

Three CFD flow calculations to find

$$I_c(\alpha_{1,2}), \quad I_c(\alpha_1 + \delta\alpha_1), \quad I_c(\alpha_2 + \delta\alpha_2)$$

For N design variables, $N + 1$ CFD calculations required.

Approach: Design Process

As an example, using 2 design variables α_1, α_2, the sensitivities would be:

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

Which can be calculated using a brute-force approach:

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

Three CFD flow calculations to find

$$I_c(\alpha_{1,2}), \quad I_c(\alpha_1 + \delta\alpha_1), \quad I_c(\alpha_2 + \delta\alpha_2)$$

For N design variables, $N + 1$ CFD calculations required.

Approach: Adjoint Equation

For our flow solution q and airfoil geometry $X = X(\alpha_1, ..., \alpha_n)$ our cost function is

$$I_c = I_c(q, X)$$

and a perturbation of the cost function is represented as:

$$\delta I = \frac{\partial I^T}{\partial q}\delta q + \frac{\partial I^T}{\partial X}\delta X$$

A perturbation of the flow residual R is represented as:

$$\delta\left[\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{f}_{c,i}}{\partial \xi_i}\right] = \delta R = \left[\frac{\partial R}{\partial q}\right]\delta q + \left[\frac{\partial R}{\partial X}\right]\delta X = 0$$

Using the method of Lagrange multipliers:

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

If the adjoint equation is satisfied:

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

then

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

In this final equation:

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

the cost function is independent of the flow solution. This means we can calculate all sensitivities

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

from “simply” solving the adjoint equation (same cost as Euler equations)

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

Gradient-based optimization

With the known sensitivities:

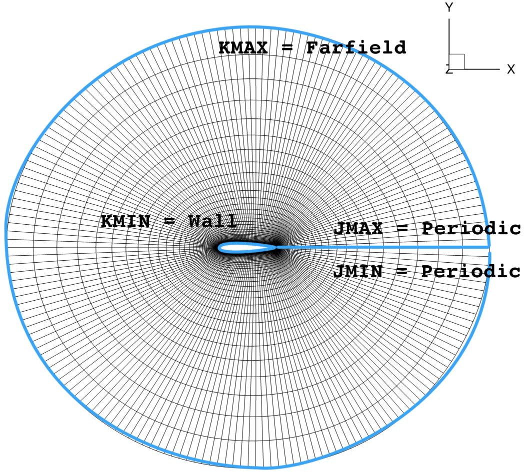
$$\frac{\partial I_c}{\partial \alpha_1}, \quad \frac{\partial I_c}{\partial \alpha_2}$$

Update α in the direction of steepest descent

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

Implementation: System Description

O-grid, 192 x 32 = 6144 points, 4 equations per point



Implementation

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 [HICKS and HENNE(1977)]

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

   - example in appendix

Implementation

4. Adjoint Euler Solver
    - Auto-differentiation with *Tapenade* [Hascoët and Pascual(2004)]

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

    - By-hand discrete solver in C++, based notes from Jameson and Nadarajah [Nadarajah and Jameson(2002)]
    - Possibly parallelized with OpenMP or CUDA (time permitting)

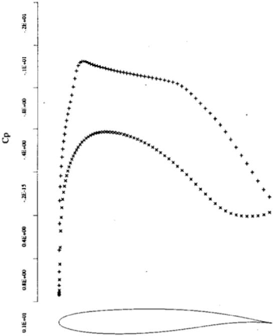
Validation

Sensitivities from solution of the adjoint equation can by also obtained using:

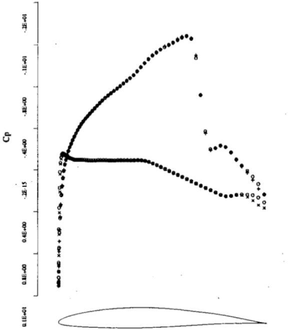
- Auto-differentiation software such as *Tapenade* (previously mentioned)
- Brute-force finite-difference calculations with one CFD-calculation per design variable
  - Using complex variable methods to avoid near-machine-zero round-off errors [Anderson et al.(2001)Anderson, Newman, and Whit]

Testing: Reverse Design

Re-conduct test by Jameson and Nadarajah



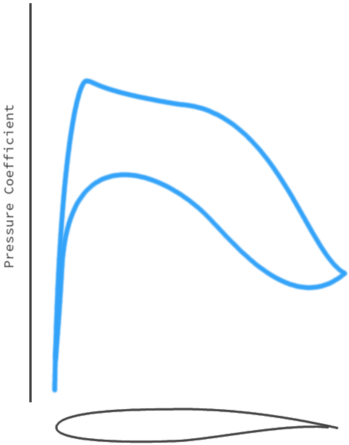
Airfoil 1

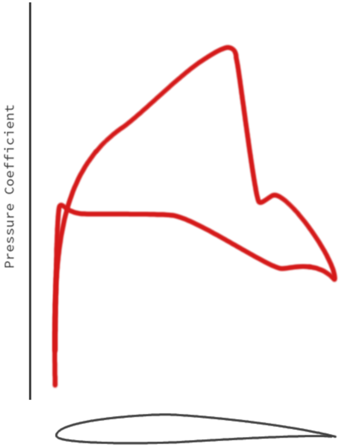


Airfoil 2

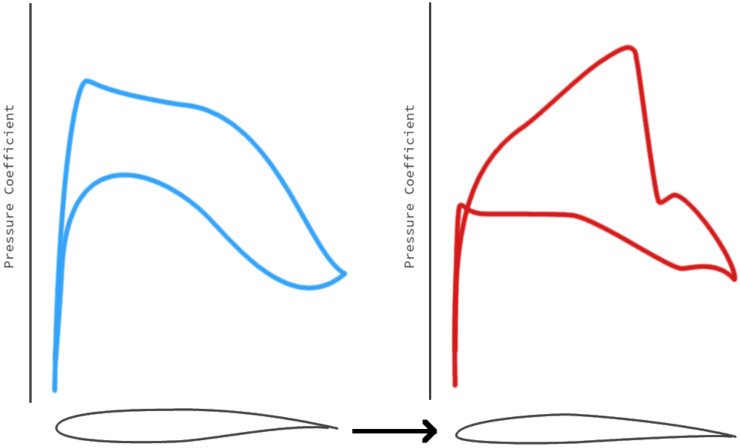
[Nadarajah and Jameson(2002)]

Testing: Reverse Design

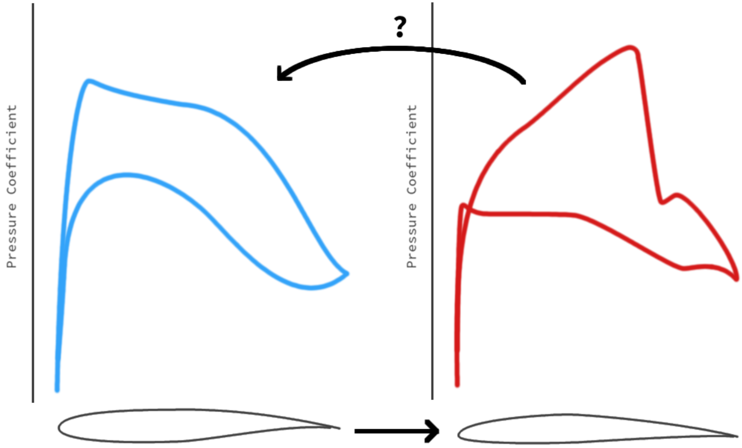




Testing: Reverse Design



Testing: Reverse Design



Schedule I

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
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

Schedule II

3. Phase 3: Implementing Discrete Adjoint Equations
   - Following methodology outlined by Jameson et. al. (2000).
   - estimated time: 4 weeks
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
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

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 January
Implicit routine for discrete adjoint solver.	Early February
Validate discrete adjoint solver against auto-diff and brute-force methods.	Late February
Test discrete adjoint solver with full reverse-design cases.	Mid March

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 I

[HICKS and HENNE(1977)] R. HICKS and P. HENNE.
Wing design by numerical optimization.
Aircraft Design and Technology Meeting. American Institute of Aeronautics and Astronautics, Aug 1977.
doi: 10.2514/6.1977-1247.
URL http://dx.doi.org/10.2514/6.1977-1247.

[Hascoët and Pascual(2004)] Laurent Hascoët and Valérie Pascual.
TAPENADE 2.1 user's guide.
2004.
URL http://www.inria.fr/rrrt/rt-0300.html.

[Nadarajah and Jameson(2002)] Siva Nadarajah and Antony Jameson.
Optimal Control of Unsteady Flows Using a Time Accurate Method.
Multidisciplinary Analysis Optimization Conferences, (June):—-, 2002.
doi: 10.2514/6.2002-5436.
URL http://dx.doi.org/10.2514/6.2002-5436.

[Anderson et al.(2001)Anderson, Newman, and Whit] W Kyle Anderson, James C Newman, and David L Whit.
Sensitivity Analysis for Navier Stokes Equations on Unstructured Meshes Using Complex Variables Introduction.
39(1), 2001.

Appendix: Euler Adjoint Equation Derivation I

Cost Function Variation:

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

Steady Euler equation dependence on δq:

$$\begin{aligned} R &= \frac{\partial f_i}{\partial \xi_i} \quad = 0 \\ \frac{\partial R}{\partial q}\delta q &= \frac{\partial}{\partial \xi_i}\delta f_i = 0 \end{aligned}$$

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

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

integrating by parts

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

Appendix: Euler Adjoint Equation Derivation II

since this is zero, we can add it to the δI equation

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

we then pick ψ to eliminate all dependence on δw. For a cost function of only an integral along the boundary ($P = 0$), the interior integral becomes:

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

using the definition of flux Jacobian $A_i = \partial f_i / \partial q$:

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

Appendix: Hicks Henne Function

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

t_1 locates the maximum of the bump in $0 \leq x \leq 1$
t_2 controles the width of the bump

Appendix: Hicks Henne Function

With 6 bumps, 12 random variables: 3 t_1, 3 a for each the top and bottom of the airfoil, $t2 = 1.0$