Aerodynamic Optimisation Using CAD Parameterisations in SU2

Aerodynamic Optimisation Using CAD Parameterisations in SU^2

4th EASN Flight Physics Workshop
Aachen, Germany

Philip Hewitt
phewitt04@qub.ac.uk

Simão Marques
s.marques@qub.ac.uk

Trevor Robinson
t.robinson@qub.ac.uk

School of Mechanical and Aerospace Engineering
Queen's University Belfast

Outline

Outline

Contents

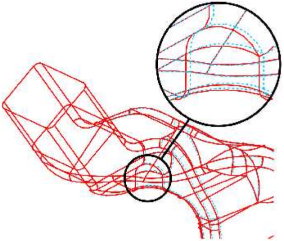
Outline

Motivation

- Increase flexibility of Adjoint Based Optimisation

Motivation

- Increase flexibility of Adjoint Based Optimisation
- Efficient calculation of any parametric sensitivity

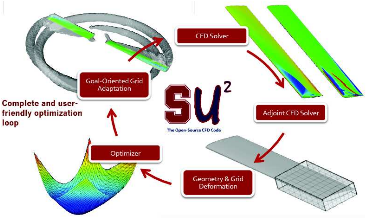


SU^2

SU^2 is an open-source CFD/Adjoint optimisation framework[1]

- Developed at Stanford University

SU2
The Open-Source CFD Code

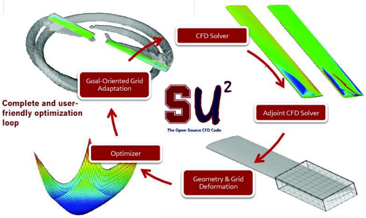


[1] images taken from *http://su2.stanford.edu/*

SU^2

SU^2 is an open-source CFD/Adjoint optimisation framework[1]

- Developed at Stanford University
- General purpose PDE solution methods



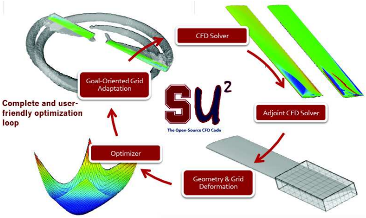
SU2
The Open-Source CFD Code

[1] images taken from *http://su2.stanford.edu/*

SU^2

SU^2 is an open-source CFD/Adjoint optimisation framework[1]

- Developed at Stanford University
- General purpose PDE solution methods
- Range of numerical schemes available (JST, ROE, MG, Euler-Implicit,...)



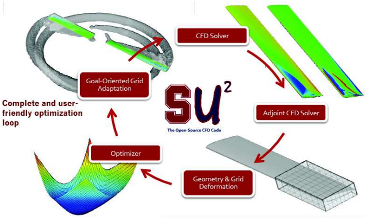
SU2 The Open-Source CFD Code

[1] images taken from *http://su2.stanford.edu/*

SU^2

SU^2 is an open-source CFD/Adjoint optimisation framework[1]

- Developed at Stanford University
- General purpose PDE solution methods
- Range of numerical schemes available (JST, ROE, MG, Euler-Implicit,. . .)
- Mesh deformation/adaptation



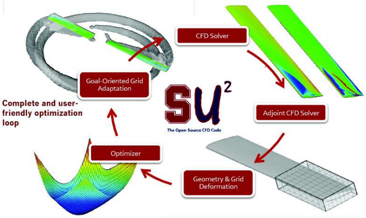
SU2
The Open-Source CFD Code

[1] images taken from *http://su2.stanford.edu/*

SU^2

SU^2 is an open-source CFD/Adjoint optimisation framework[1]

- Developed at Stanford University
- General purpose PDE solution methods
- Range of numerical schemes available (JST, ROE, MG, Euler-Implicit,. . .)
- Mesh deformation/adaptation
- Continuous Adjoint Solver

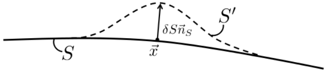


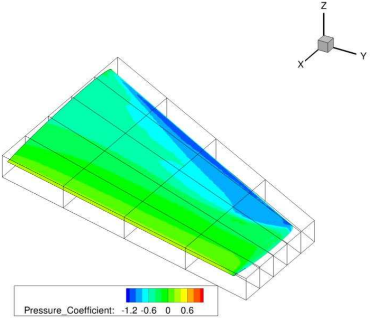
SU2
The Open-Source CFD Code

[1] images taken from *http://su2.stanford.edu/*

SU^2

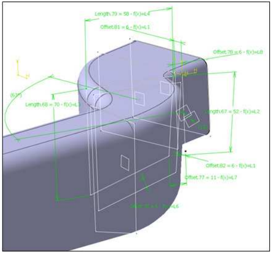
Native parameterisations in SU^2 [2]





[2] images taken from *http://su2.stanford.edu/*

How to link any parameterisation to SU^2 ?



Outline

Adjoint Based Optimisation

- Gradients required for optimisation
- Finite differences not feasible for complex shapes with multiple parameters
- Adjoint method provides an efficient alternative

Gradient Calculation

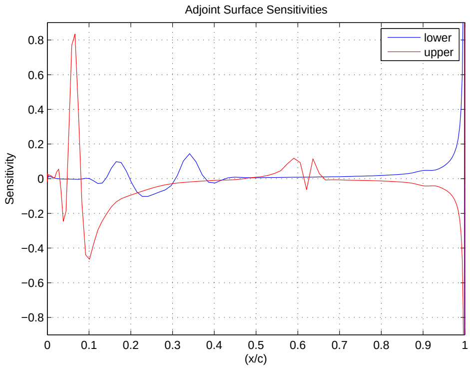
$$\underbrace{\begin{bmatrix} \frac{\partial f}{\partial A_1} \\ \frac{\partial f}{\partial A_2} \\ \vdots \\ \frac{\partial f}{\partial A_n} \end{bmatrix}}_{Gradients} = \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial A_1} & \cdots & \frac{\partial x_m}{\partial A_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial A_n} & \cdots & \frac{\partial x_m}{\partial A_m} \end{bmatrix}}_{Geometric\ Sensitivities} \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}}_{Surface\ Sensitivities}$$

- $\frac{\partial f}{\partial A_i}$ - Gradient
- $\frac{\partial x_j}{\partial A_i}$ - Geometric Sensitivities
- $\frac{\partial f}{\partial x_j}$ - Surface Sensitivities

Gradient Calculation

$$\underbrace{\begin{bmatrix} \frac{\partial f}{\partial A_1} \\ \frac{\partial f}{\partial A_2} \\ \vdots \\ \frac{\partial f}{\partial A_n} \end{bmatrix}}_{Gradients} = \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial A_1} & \cdots & \frac{\partial x_m}{\partial A_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial A_n} & \cdots & \frac{\partial x_m}{\partial A_m} \end{bmatrix}}_{Geometric\ \ Sensitivities} \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}}_{Surface\ \ Sensitivities}$$

- $\frac{\partial f}{\partial A_i}$ - Gradient
- $\frac{\partial x_j}{\partial A_i}$ - Geometric Sensitivities
- $\frac{\partial f}{\partial x_j}$ - Surface Sensitivities



Computed by SU^2

Gradient Calculation

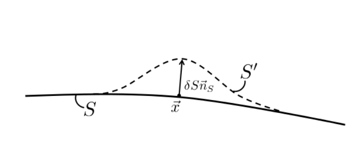
$$\underbrace{\begin{bmatrix} \frac{\partial f}{\partial A_1} \\ \frac{\partial f}{\partial A_2} \\ \vdots \\ \frac{\partial f}{\partial A_n} \end{bmatrix}}_{Gradients} = \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial A_1} & \cdots & \frac{\partial x_m}{\partial A_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_1}{\partial A_n} & \cdots & \frac{\partial x_m}{\partial A_m} \end{bmatrix}}_{Geometric\ \ Sensitivities} \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}}_{Surface\ \ Sensitivities}$$

- $\frac{\partial f}{\partial A_i}$ - Gradient
- $\frac{\partial x_j}{\partial A_i}$ - Geometric Sensitivities
- $\frac{\partial f}{\partial x_j}$ - Surface Sensitivities

SU^2

Gradient Calculation

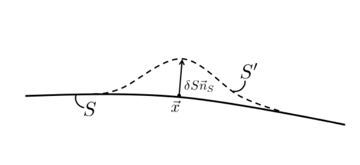
Use SU^2 native parameterisations



SU^2

Gradient Calculation

Use SU^2 native parameterisations



Gradient Calculation

Or add you own parametric sensitivities:

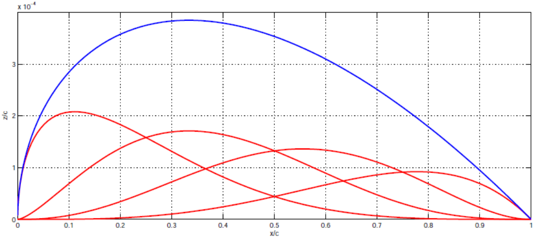
$$\Delta J = \int \phi V_n dS$$

CST Parameterisation

$$\zeta(\phi) = C^{N1}_{N2}(\phi)S(\phi) + \phi\Delta\zeta_{TE}$$

$$C^{N1}_{N2}(\phi) = \phi^{N1}(1-\phi)^{N2} \qquad \text{Class Function}$$

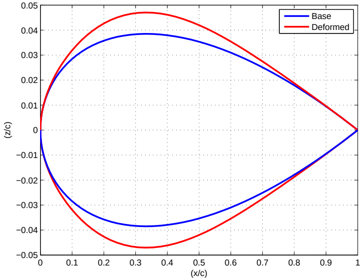
$$S(\phi) = \sum_{i=0}^{n} A_i S_i \qquad \text{Shape Function}$$



CST Parameterisation

The surface is manipulated through the choice of function weights A_i:

$$S(\phi) = \sum_{i=0}^{n} A_i S_i$$

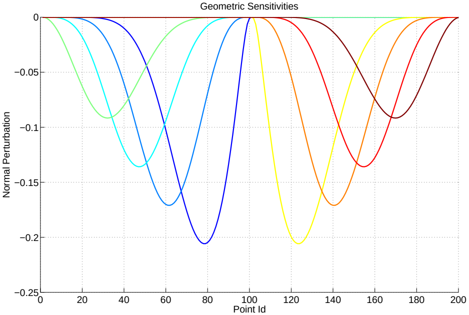


CST Parameterisation

Geometric Sensitivities ($\equiv V_n$)

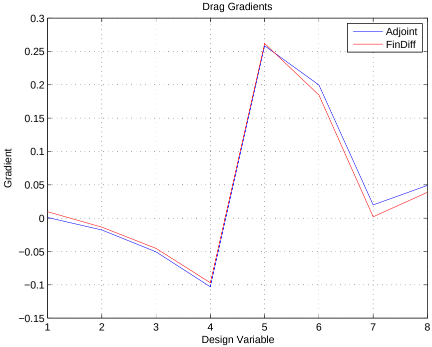
$$\frac{\partial \mathbf{x}_j}{\partial A_i} - \text{Geometric Sensitivities}$$

$$\frac{\partial \mathbf{x}_j}{\partial A_i} = \left(\frac{\partial x_j}{\partial A_i} n_x + \frac{\partial y_j}{\partial A_i} n_y + \frac{\partial z_j}{\partial A_i} n_z \right)$$

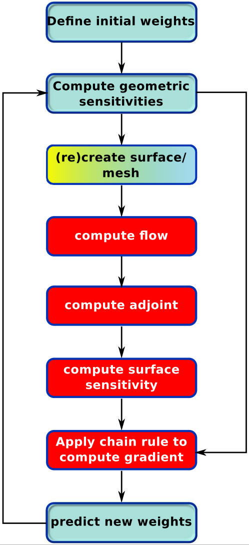


Validation - CST parameterisation

$$\frac{\partial f}{\partial A_i}$$ - Gradient



SU^2 Optimisation Process

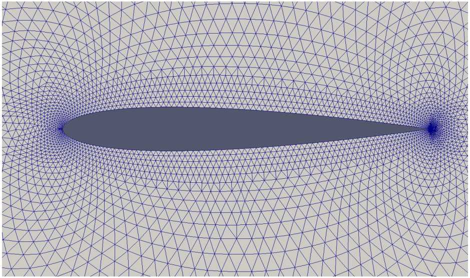


Outline

NACA0012 Drag minimization

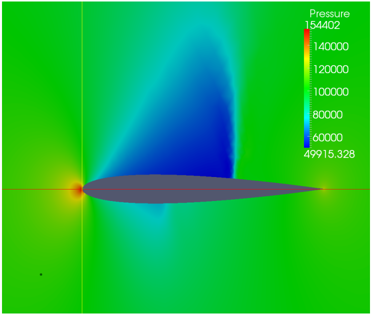
Inviscid test case: NACA0012 starting aerofoil.

- $M_\infty = 0.8$
- $\alpha = 1.25°$
- $f = min(C_d)$
- $C_l \geq 0.33$
- $C_m > 0.034$
- nDV $= 8$



NACA0012 Drag minimization

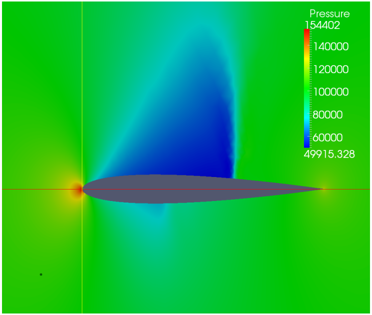
Inviscid test case: NACA0012 starting aerofoil.



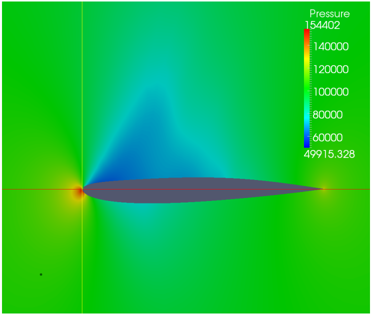
initial aerofoil

NACA0012 Drag minimization

Inviscid test case: NACA0012 starting aerofoil.



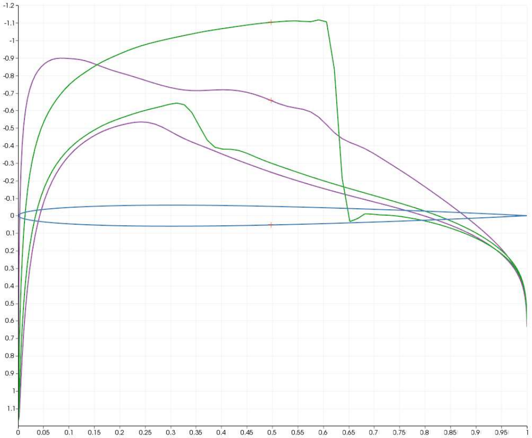
initial aerofoil



final aerofoil

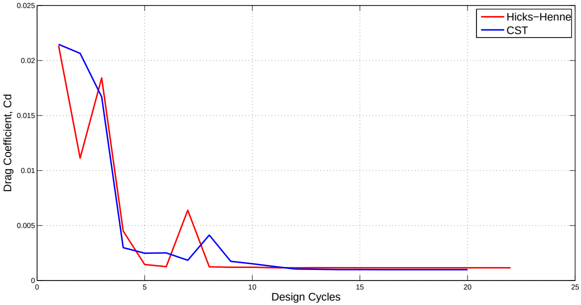
NACA0012 Drag minimization

Inviscid test case: NACA0012 starting aerofoil.



NACA0012 Drag minimization

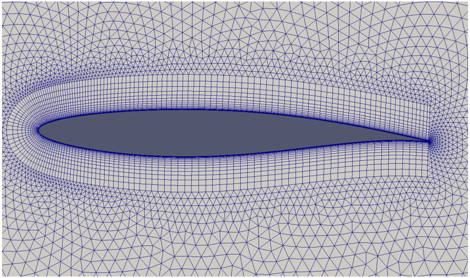
Inviscid test case: NACA0012 starting aerofoil.



RAE2822 Drag minimization

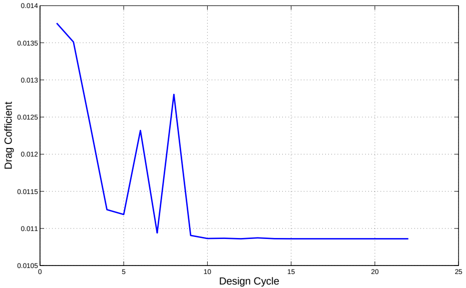
Viscous test case: RAE2822 starting aerofoil.

- $M_\infty = 0.729$
- $\alpha = 2.31^\circ$
- $f = min(C_d)$
- nDV $= 8$
- SA turbulence model
- $y^+ \leq 5$



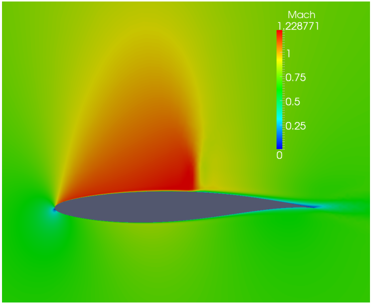
RAE2822 Drag minimization

Viscous test case: RAE2822 starting aerofoil.



RAE2822 Drag minimization

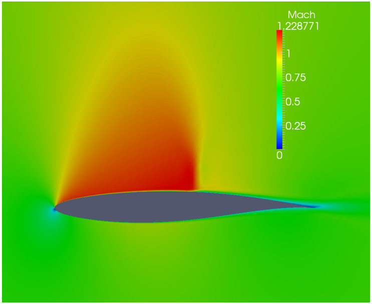
Viscous test case: RAE2822 starting aerofoil.



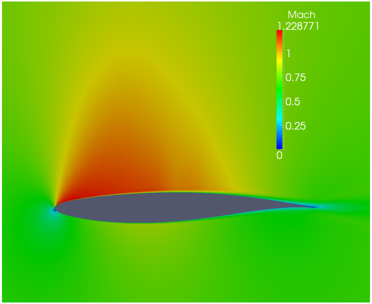
initial aerofoil

RAE2822 Drag minimization

Viscous test case: RAE2822 starting aerofoil.



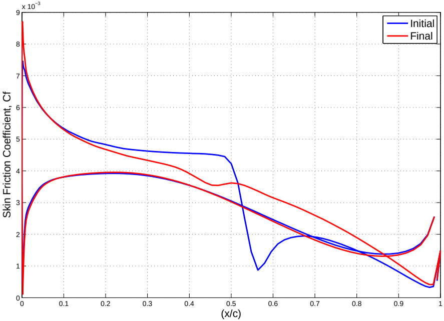
initial aerofoil



final aerofoil

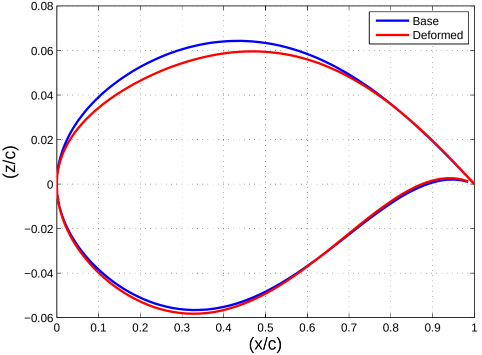
RAE2822 Drag minimization

Viscous test case: RAE2822 starting aerofoil.



RAE2822 Drag minimization

Viscous test case: RAE2822 starting aerofoil.



Outline

RAE2822 Drag minimization

- An alternative parameterisation was introduced into the SU^2 suite
- Model deformation can be performed outside SU^2
- alternative approach does not compromise optimisation efficiency with respect to native parameterisations

Q & A

Thank you for your attention

Questions Welcome

CST Parameterisation

$$
\begin{aligned}
\zeta(\phi) &= C_{N2}^{N1}(\phi)S(\phi) + \phi\Delta\zeta_{TE} \\
C_{N2}^{N1}(\phi) &= \phi^{N1}(1-\phi)^{N2} \\
S(\phi) &= \sum_{i=0}^{n} A_i S_i \\
S_i &= K_{i,n}\phi^{i}(1-\phi)^{i} \\
K_{i,n} &= \binom{n}{i} = \frac{n!}{i!(n-i)!}
\end{aligned}
$$