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

1. Overview
2. Motivation
3. Gradient Calculation
4. Test Cases
5. Conclusions
Contents

• Overview
• Motivation
• $SU^2$
• CST Transformation
• Results
• Conclusions
Outline

1. Overview
2. Motivation
3. Gradient Calculation
4. Test Cases
5. Conclusions
Motivation

- Increase flexibility of Adjoint Based Optimisation
Motivation

- Increase flexibility of Adjoint Based Optimisation
- Efficient calculation of any parametric sensitivity
$SU^2$ is an open-source CFD/Adjoint optimisation framework\textsuperscript{1}

- Developed at Stanford University

\textsuperscript{1}images taken from http://su2.stanford.edu/
SU² is an open-source CFD/Adjoint optimisation framework

- Developed at Stanford University
- General purpose PDE solution methods

Images taken from http://su2.stanford.edu/
SU² 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, ...)

---

\(^1\) images taken from http://su2.stanford.edu/
$SU^2$ is an open-source CFD/Adjoint optimisation framework\textsuperscript{1}

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

\textsuperscript{1}images taken from http://su2.stanford.edu/
$SU^2$ 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

\(^1\) images taken from \url{http://su2.stanford.edu/}
Native parameterisations in $SU^2$
How to link any parameterisation to $SU^2$?
Outline

1. Overview
2. Motivation
3. Gradient Calculation
4. Test Cases
5. Conclusions
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

\[
\begin{bmatrix}
\frac{\partial f}{\partial A_1} \\
\frac{\partial f}{\partial A_2} \\
\vdots \\
\frac{\partial f}{\partial A_n}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial x_1}{\partial A_1} & \cdots & \frac{\partial x_m}{\partial A_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial x_1}{\partial A_n} & \cdots & \frac{\partial x_m}{\partial A_n}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots \\
\frac{\partial f}{\partial x_m}
\end{bmatrix}
\]

- Gradient
\[\frac{\partial f}{\partial A_i}\]
- Geometric Sensitivities
\[\frac{\partial x_j}{\partial A_i}\]
- Surface Sensitivities
\[\frac{\partial f}{\partial x_j}\]
Gradient Calculation

\[
\begin{bmatrix}
\frac{\partial f}{\partial A_1} \\
\frac{\partial f}{\partial A_2} \\
\vdots \\
\frac{\partial f}{\partial A_n}
\end{bmatrix}
= 
\begin{bmatrix}
\frac{\partial x_1}{\partial A_1} & \ldots & \frac{\partial x_m}{\partial A_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial x_1}{\partial A_n} & \ldots & \frac{\partial x_m}{\partial A_n}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots \\
\frac{\partial f}{\partial x_m}
\end{bmatrix}
\]

- Gradient $\frac{\partial f}{\partial A_i}$
- Geometric Sensitivities $\frac{\partial x_j}{\partial A_i}$
- Surface Sensitivities $\frac{\partial f}{\partial x_j}$
Computed by $SU^2$
Gradient Calculation

\[
\begin{bmatrix}
\frac{\partial f}{\partial A_1} \\
\frac{\partial f}{\partial A_2} \\
\vdots \\
\frac{\partial f}{\partial A_n}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial x_1}{\partial A_1} & \cdots & \frac{\partial x_m}{\partial A_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial x_1}{\partial A_n} & \cdots & \frac{\partial x_m}{\partial A_n}
\end{bmatrix}
\]

- Gradient
- Geometric Sensitivities
- Surface Sensitivities

- \( \frac{\partial f}{\partial A_i} \)
- Geometric Sensitivities
- \( \frac{\partial f}{\partial x_j} \)
Gradient Calculation

Use $SU^2$ native parameterisations
Gradient Calculation

Use $SU^2$ native parameterisations

Or add you own parametric sensitivities:

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

\[
\zeta(\phi) = C^{N_1}_{N_2}(\phi)S(\phi) + \phi \Delta \zeta_{TE}
\]

\[
C^{N_1}_{N_2}(\phi) = \phi^{N_1}(1 - \phi)^{N_2}
\]

Class Function

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

Shape Function
CST Parameterisation

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

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

\[
\frac{\partial x_j}{\partial A_i} - \text{Geometric Sensitivities} \\
\frac{\partial x_j}{\partial A_i} = \left( \frac{\partial x_j}{\partial A_i} n_x + \frac{\partial y_j}{\partial A_i} n_y + \frac{\partial z_j}{\partial A_i} n_z \right)
\]
\[ \frac{\partial f}{\partial A_i} - \text{Gradient} \]

![Drag Gradients Graph]

- Adjoint
- FinDiff
$SU^2$ Optimisation Process

1. Define initial weights
2. Compute geometric sensitivities
3. (re)create surface/mesh
4. Compute flow
5. Compute adjoint
6. Compute surface sensitivity
7. Apply chain rule to compute gradient
8. Predict new weights
Outline

1. Overview
2. Motivation
3. Gradient Calculation
4. Test Cases
5. Conclusions
Inviscid test case: NACA0012 starting aerofoil.

- \( M_\infty = 0.8 \)
- \( \alpha = 1.25^\circ \)
- \( f = \min(C_d) \)
- \( C_l \geq 0.33 \)
- \( C_m > 0.034 \)
- \( nDV = 8 \)
NACA0012 Drag minimization

Inviscid test case: NACA0012 starting aerofoil.
NACA0012 Drag minimization

Inviscid test case: NACA0012 starting aerofoil.

initial aerofoil

final aerofoil
NACA0012 Drag minimization

Inviscid test case: NACA0012 starting aerofoil.
Inviscid test case: NACA0012 starting aerofoil.
Viscous test case: RAE2822 starting aerofoil.

- $M_\infty = 0.729$
- $\alpha = 2.31^\circ$
- $f = \min(C_d)$
- nDV = 8
- SA turbulence model
- $y^+ \leq 5$
Viscous test case: RAE2822 starting aerofoil.
Viscous test case: RAE2822 starting aerofoil.

initial aerofoil
Viscous test case: RAE2822 starting aerofoil.

initial aerofoil  

final aerofoil
Viscous test case: RAE2822 starting aerofoil.
Viscous test case: RAE2822 starting aerofoil.
Outline

1. Overview
2. Motivation
3. Gradient Calculation
4. Test Cases
5. Conclusions
• 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
Thank you for your attention

Questions Welcome
CST Parameterisation

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