Aerofoil Optimisation Using CST Parameterisation in SU^2

P. Hewitt*, S. Marques†

School of Mechanical and Aerospace Engineering

Queen's University Belfast, Belfast, UK, BT9 5AH

This paper describes an implementation of the popular method of Class-Shape Transformation for aerofoil design within SU^2 software framework. To exploit the adjoint based methods for aerodynamic optimisation within the SU^2 , a formulation to obtain geometric sensitivities from the new parameterisation is introduced, enabling the calculation of gradients with respect to new design variables. To assess the accuracy and efficiency of the alternative approach, two transonic optimisation problems are investigated: an inviscid problem with multiple constraints and a viscous problems without any constraints. Results show the new parameterisation obtaining reliable optimums, with similar levels of performance of the software native parameterisations.

Nomenclature

A_i	=	CST weights
c	=	chord length
C_{N2}^{N1}	=	CST class function
C_P	=	pressure coefficient
C_d	=	drag coefficient
C_l	=	lift coefficient
C_m	=	pitching moment coefficient
f	=	function of interest - objective function
$\mathbf{F}^c$	=	convective fluxes
$\mathbf{F}^v$	=	viscous fluxes
g	=	inequality constraint function
h	=	equality constraint function
$K_{i,n}$	=	binomial coefficient
M	=	Mach number
$\mathbf{n}$	=	surface normal vector
$\mathbf{Q}$	=	generic source term
$\mathbf{R}$	=	Residual of governing flow equations
S_ψ	=	CST shape function
$\mathbf{U}$	=	flow equations unknowns
x, z	=	horizontal and vertical directions, respectively
α	=	angle of attack
$\mathbf{\Psi}$, ψ	=	flow adjoint equations unknowns
ϕ	=	normalized aerofoil chordwise position
ζ	=	normalized aerofoil thickness position

*PhD. Student

†Lecturer - s.marques@qub.ac.uk

1. Introduction

Aerofoil optimisation is now ubiquitous in a myriad of industries and is one of the most exhausted topics in aerodynamics. Nevertheless, as increasingly more powerful analysis tools become available, new challenges emerge in an attempt to maximise a given measure or measures of performance. Before the advent of modern computers and numerical methods, conformal mappings[1,2] had already been used in low speed aerofoil design and characteristic methods were developed to design shock free aerofoils.[3] Two distinct approaches can usually be used in aerofoil design: inverse design methods and direct methods. Inverse methods aim to generate an aerofoil shape for a given pressure or velocity distribution, further details can be found in ref.[4] Direct methods are becoming increasingly popular as computer power increases, enabling even the application of stochastic methods and the search of multiple-optimuns;[5,6] an alternative to stochastic methods such as Evolutionary Algorithms or brute force searches, is to use adjoint methods. This approach was made popular in aerofoil optimisation by Jameson and co-workers.[7,8] However, as discussed by Drela,[9] the availability of sophisticated optimisation algorithms and analysis methods is not sufficient to ensure practical optimum geometries can be found. This statement can be described as the main motivator for the work illustrated in this paper. The SU^2 solver is a state-of-the-art open source tool capable of high-fidelity aerodynamic analysis and adjoint based design optimisation, made available from Stanford University. It includes specific tools for aerofoil shape optimisation based on Hicks-Henne functions.[10] This paper will show the implementation of an alternative aerofoil parameterisation, in this case the Class-Shape-Transformation (CST) approach proposed by Kulfan,[11] within the SU^2 framework that fully exploits its adjoint based design methods. The CST implementation will be assessed for robustness and performance, using both inviscid and viscous transonic flows.

2. Flow Solver and Optimisation Framework

The SU^2 code was developed primarily for the purpose of providing an open source tool for use in aerodynamic shape optimisation. Additionally, the framework also has the capability to solve various governing equations such as electrodynamics and chemically reacting flows. The philosophy employed by the developers, as stated in Palacios *et al.*[10] is to provide:

- An open source model
- Portability
- Reuseability and encapsulation
- Performance
- Gradient Availability

With the integration of these characteristics in SU^2, the code allows access to all aspects of the implementation and is ready to be modified and expanded upon. To this effect, the code has been partitioned into several primary modules, each used to handle a particular task so as to minimise interference and promote ease of development. The suite consists of 7 primary C++ modules with the most relevant to this paper being:

- SU2_CFD - Main module consisting of the numerical methods to solve partial differential equations.
- SU2_MDC - Mesh Deformation Code used to produce a deformed mesh using the current set of design variables.
- SU2_GPC - Gradient Projection Code used for computing design gradients.

The execution of these individual modules is managed through Python wrapping scripts, which handle the interaction between the modules and the transfer of data.

A. Flow Solver

The CFD module is able to solve several flow models. In this work, the compressible Euler and RANS equations are used to describe the flow. Following the notation presented in reference[10] the RANS equations can be described as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}^c - \nabla \cdot \mathbf{F}^v - \mathbf{Q} = \mathbf{R} = 0 \tag{1}$$

where $\mathbf{U}$ represents the vector of fluid unknowns, $\mathbf{F}^c$ and $\mathbf{F}^v$ are the convective and viscous fluxes respectively and $\mathbf{Q}$ is a generic source term. By setting $\mathbf{Q}$ and $\mathbf{F}^v$ to zero, the Euler equations are recovered. The equations are discretised in space using the classical Jameson-Schmidt-Turkel (JST) scheme. The solution is marched forward in time using an implicit Euler scheme, until a steady state is reached.

B. Adjoint Solver

The SU^2 suite is able to solve the continuous adjoint Euler/RANS equations. The subsequent sections will describe the relation between the design variables and the computation of gradients with respect to the quatities of interest; as it will be seen, the efficient calculation of these gradients will require the solution of the adjoint equations, which can be described as:[12]

$$-\frac{\partial \mathbf{\Psi}}{\partial t} - \nabla \mathbf{\Psi} \cdot \left[\left(\frac{\partial \mathbf{F}^c}{\partial \mathbf{U}} \right)_i - \mu_{tot} \left(\frac{\partial \mathbf{F}^v}{\partial \mathbf{U}} \right)_i \right] - \nabla \cdot \left[\nabla \mathbf{\Psi} \cdot \mu_{tot} \frac{\partial}{\partial x_j} \left(\frac{\partial \mathbf{F}^v}{\partial \mathbf{U}} \right)_i \right] - \mathbf{\Psi} \frac{\partial \mathbf{Q}}{\partial \mathbf{U}} = 0, \quad i,j = 1,2,3 \tag{2}$$

where: $\mathbf{\Psi} = [\psi_1, \psi_2, \psi_3, \psi_4, \psi_5]$ are the adjoint unknowns. The numerical discretization and solution of equation 2 follows the methods available to solve equation 1 and are described in detailed by Economon.[12]

C. Optimisation problem statement

Numerical aerodynamic optimisation involves the minimisation of a chosen objective function, e.g. drag, through the manipulation of a set of design variables. Within gradient based optimisation methods, the gradients are used to guide the design towards an optimum design. With each new step a new set of design variables is produced, causing a change in the objective function.
A general optimisation problem would take the form seen below,

$$\begin{aligned} \text{Minimise}: \quad & f(\mathbf{x}), \\ \text{Subject to}: \quad & g(\mathbf{x}) \leq 0, \\ & h(\mathbf{x}) = 0 \end{aligned} \tag{3}$$

here $f(\mathbf{x})$ is the objective function to be minimised (maximise), $g(\mathbf{x})$ is the inequality constraint and $h(\mathbf{x})$ represents equality constraints. The optimisation algorithm used in this work is the "Sequential Least Squares Programming" implementation in Scipy.

3. Design Parameterisation

For the process of shape optimisation, SU^2 employs several methods to parameterise aerofoil shapes such as NACA series, cosine bumps and Hicks-Henne bumps for aerofoil deformations and the Free-Form deformation method for 3 dimensional designs. The Hicks-Henne bumps functions involve the linear superposition of several shape functions added to the base aerofoil to deform the surface. The magnitude and therefore influence of each shape function is manipulated by weights applied to each function respectively. These weights in turn are taken to be the design variables within the optimisation process. The parameterisation is considered very effective and ensures the resulting deformed mesh is smooth.[13]

An alternative means of aerofoil parameterisation is the use of Class Shape Transformation method. This method combines an analytical class function with a parametric shape function. This allows the CST to deal with complex non-analytical functions that arise due to high curvature and infinite second order derivative at the leading edge. Additionally the CST can represent a large design space using only a few design variables and can control key parameters of the design such as leading edge radius and trailing edge angle.[11] The shape function is generated through a combination of Bernstein polynomial shape functions. This is

manipulated through the class function which is used to define a certain base design. For completeness, the CST formulation is described next,

$$\zeta(\phi) = C_{N2}^{N1}(\phi)S(\phi) + \phi\Delta\zeta_{TE} \tag{4}$$

where $\zeta = \frac{z}{c}$ and $\phi = \frac{x}{c}$. The trailing edge thickness, given by ζ_{TE}, is zero for sharp trailing edges. The class and shape functions are given by C_{N2}^{N1} and $S(\phi)$, respectively. The class function is defined as,

$$C_{N2}^{N1}(\phi) = \phi^{N1}(1-\phi)^{N2} \tag{5}$$

The choice in the exponent values for $N1$ and $N2$ within this class function can be used to generate a certain family of designs. Rounded nose aerofoils with a sharp trailing edge can be generated using the $N1$ and $N2$ values of 0.5 and 1.0 respectively giving a class function of $C_{1.0}^{0.5} = \sqrt{\phi}(1-\phi)$.
The shape function $S(\phi)$ is obtained through the summation of Bernstein polynomials and is given as,

$$S(\phi) = \sum_{i=0}^{n} A_i S_i \tag{6}$$

where,

$$S_i = K_{i,n}\phi^i(1-\phi)^i \tag{7}$$

and $K_{i,n}$ is the binomial coefficient of order n given by,

$$K_{i,n} = \binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{8}$$

As with the case of the Hicks-Henne functions, a set of weights ($\mathbf{A} = \{A_0, A_1, \ldots, A_n\}$), is employed to control the magnitude of the shape functions. A single weight, A_i, is applied to each of the component shape functions S_i as seen in equation 6. The effect that a change in the set of weights has on the shape of the resulting aerofoil is demonstrated in figure 1 where $\mathbf{A}_{Base}$ is a reference set and $\mathbf{A}_{Def}$ is the modified set of weights.

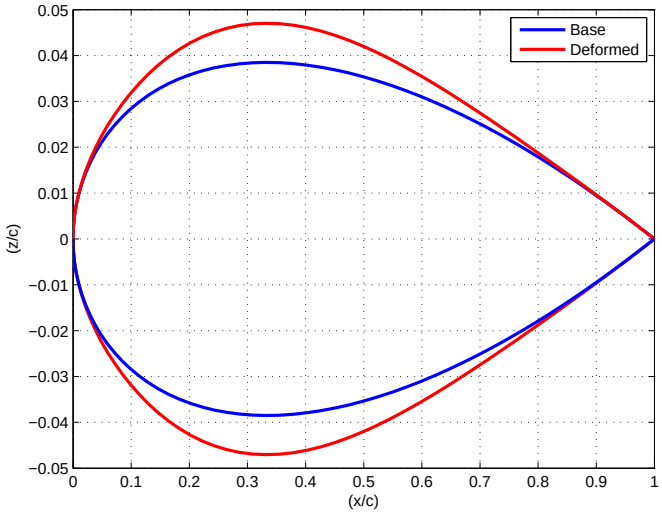


Figure 1. Aerofoils generated using different weight sets: $\mathbf{A}_{Base}$=[1.0, 1.0, 1.0, 1.0, -1.0, -1.0, -1.0, -1.0]; $\mathbf{A}_{Def}$=[1.0, 1.5, 1.0, 1.0, -1.0, -1.5, -1.0, -1.0]

Whereas the Hick-Henne functions are added to an existing surface descritised geometry, the CST parameterisation is not. Hence, a curve fitting is required to generate the initial set of weights that describe the surface in the initial mesh. This is performed within Python by extracting the surface mesh and performing a least squares fit until a set of weighs is found that matches the initial aerofoil surface.

4. Gradient Evaluation

For the optimiser to establish a new search direction it is necessary for the gradient to be evaluated for each design. In this particular case it means evaluating the drag coefficient with respect to the CST weight parameters. Within SU^2 the calculation of the gradient is performed through the use of the chain rule. This is shown in equation 9 and provides flexibility for the use of other parameterisation methods.

$$\underbrace{\begin{bmatrix} \frac{\partial f}{\partial A_1} \\ \frac{\partial f}{\partial A_2} \\ \vdots \\ \frac{\partial f}{\partial A_n} \end{bmatrix}}_{\text{Gradients}} = \underbrace{\begin{bmatrix} \frac{\partial A_1}{\partial x_1} & \cdots & \frac{\partial A_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial A_n}{\partial x_1} & \cdots & \frac{\partial A_n}{\partial x_m} \end{bmatrix}}_{\substack{\text{Geometric} \\ \text{Sensitivities}}} \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_m} \end{bmatrix}}_{\substack{\text{Surface} \\ \text{Sensitivities}}} \tag{9}$$

where n and m are the number of design variables and surface mesh points, respectively; f, represents the current function of interest, be it the objective or constraint functions. The variables x_i represent the normal displacement with respect to the surface for each discrete point on the surface itself. The Jacobian $\frac{\partial \mathbf{A}}{\partial \mathbf{x}}$ is known as the geometric sensitivity matrix or design velocities and measures the influence that each design variable has on the position of each grid point on the surface mesh. The third term represents the surface sensitivities with respect to to a change in the function of interest with a change on the surface grid points. The next sections will explain how these terms are calculated.

A. Surface Sensitivities - Adjoint Method

Within SU^2 the surface sensitivities are calculated through the use of the adjoint method. This is an analytical approach used to calculate the derivatives of a function. The main benefit with choosing this method is that the cost of evaluating the surface sensitivities becomes independent of the number of design variables and so can offer great reduction in computational expense.

The method is applicable when the function of interest, f, depends implicitly on the independent variables of interest, $\mathbf{x}$. The approach can be presented in the case of an aerodynamic problem as follows: the current function of interest, f, depends on both the flow variables, $\mathbf{U}$, and the design variables $\mathbf{x}$ as shown by:

$$f = f\left(\mathbf{x}, \mathbf{U}(\mathbf{x})\right) \tag{10}$$

the relationship between the flow and design variables is obtained through the solution of the governing equations,

$$\mathbf{R} = \mathbf{R}\left(\mathbf{x}, \mathbf{U}(\mathbf{x})\right) = 0 \tag{11}$$

the chain rule for the total derivative we have,

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} \tag{12}$$

and similarly for the governing equations,

$$\frac{d\mathbf{R}}{d\mathbf{x}} = \frac{\partial \mathbf{R}}{\partial \mathbf{x}} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{x}} = 0 \tag{13}$$

The calculation of the term $\frac{\partial \mathbf{U}}{\partial \mathbf{x}}$ has a much higher computational expense than the other partial derivatives as its requires the solution of the governing equations. Rewriting the equation as,

$$\frac{d\mathbf{U}}{d\mathbf{x}} = \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{14}$$

we get an alternative form for the function derivative,

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial f}{\partial \mathbf{U}} \left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^{-1} \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{15}$$

The adjoint method is then introduced the find a vector $\mathbf{\Psi}$, such that,

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right]^T \mathbf{\Psi} = -\left[\frac{\partial f}{\partial \mathbf{U}}\right]^T \tag{16}$$

This can then be substituted into the previous equation to form,

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} - \mathbf{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \tag{17}$$

hence the costly evaluation of $\frac{\partial \mathbf{U}}{\partial \mathbf{x}}$ is no longer required. More so as the adjoint vector is independent of the number of design variables, it is only required to be calculated once for each output function. The resulting surface sensitivities obtained for the are shown in figure 2 for the NACA0012 aerofoil at a Mach number of 0.8 and an angle of attack of 1.25°.

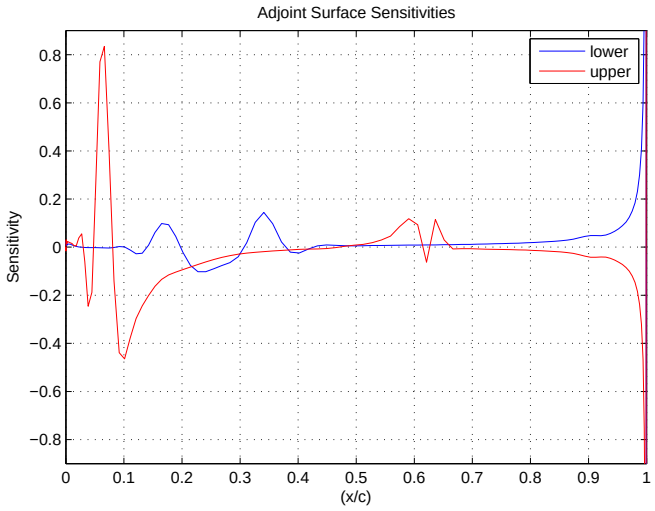


Figure 2. Surface Sensitivities

B. Geometric Sensitivities - CST

The geometric sensitivities, represented by $\frac{\partial \mathbf{A}}{\partial \mathbf{x}}$, map the variables of the parameters to the discrete surface grid points. Within SU^2 this Jacobian is calculated using finite differences, the cost of which is negligible in comparison to the expense of the flow solution. They are calculated by first measuring the displacement of a point on the surface following a perturbation to each of the CST weights. By projecting this displacement along the normal direction to the surface we obtain the design velocity or geometric sensitivity, hence each member i, j of the Jacobian matrix in equation 9 is given by:

$$\left(\frac{d\mathbf{A}}{d\mathbf{x}}\right)_{i,j} = \left(\frac{\partial x_j}{\partial A_i} n_x + \frac{\partial y_j}{\partial A_i} n_y + \frac{\partial z_j}{\partial A_i} n_z\right)^{-1}, \quad i = 1, \ldots, n; \;\; j = 1, \ldots, m \tag{18}$$

The integration of equation 18 into SU^2 is performed through the SU2_GPC module; further modifications were required to input files, and respective parsers, to expand the range of options available and allow selecting this type of parameterisation.

A set of design velocities from a CST parameterisation, using four Bernstein polynomials to represent each surface, are presented in figure 3. The *point ID's* ranging from 0 to 99 correspond to the lower surface and ID's from 101 to 200 relate to the upper surface. Point 100 corresponds to the leading edge of the aerofoil. As previously explained the objective function gradients are computed through the chain rule by relating the geometric and surface sensitivities, equation 9. To validate the gradient values produced using the adjoint based method and CST geometric sensitivities, results were compared against gradients based on finite differences. The results are shown in figure 4, a strong correlation between the two sets is clear, giving confidence in the implemented process.

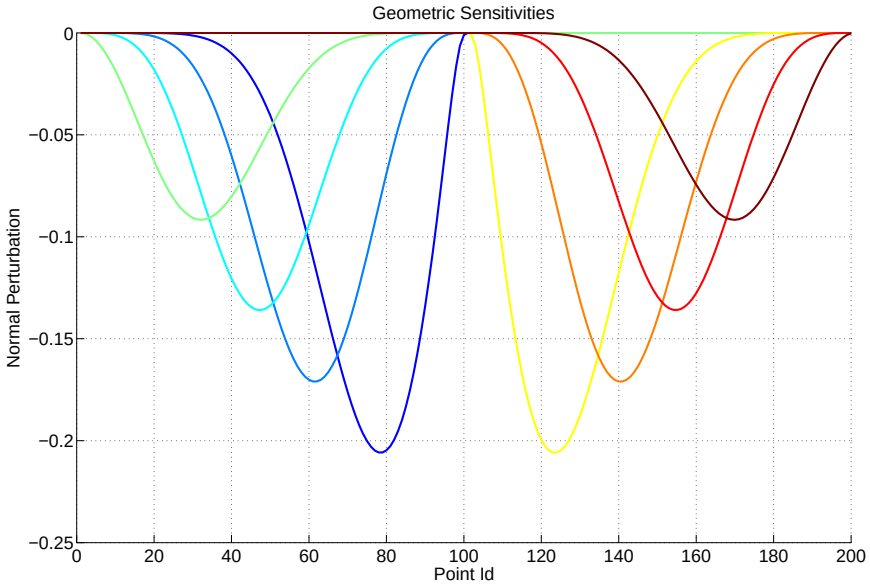


Figure 3. Geometric Sensitivities

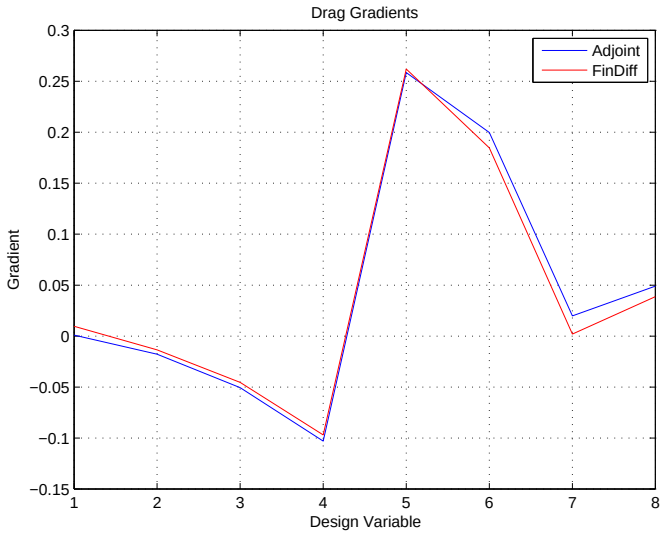


Figure 4. Gradient calculations - adjoint versus finite-differences

5. Results

Two cases are used to evaluate the procedure described in the preceding sections. The first is an inviscid problem where the flow is predicted using the Euler and the respective adjoint equations. Solving the RANS adjoint equations poses additional numerical difficulties, hence a second test case assesses the proposed method in performing a transonic viscous optimisation. Both meshes used to start the optimisation calculations can be obtained from the SU^2 software test cases files[a].

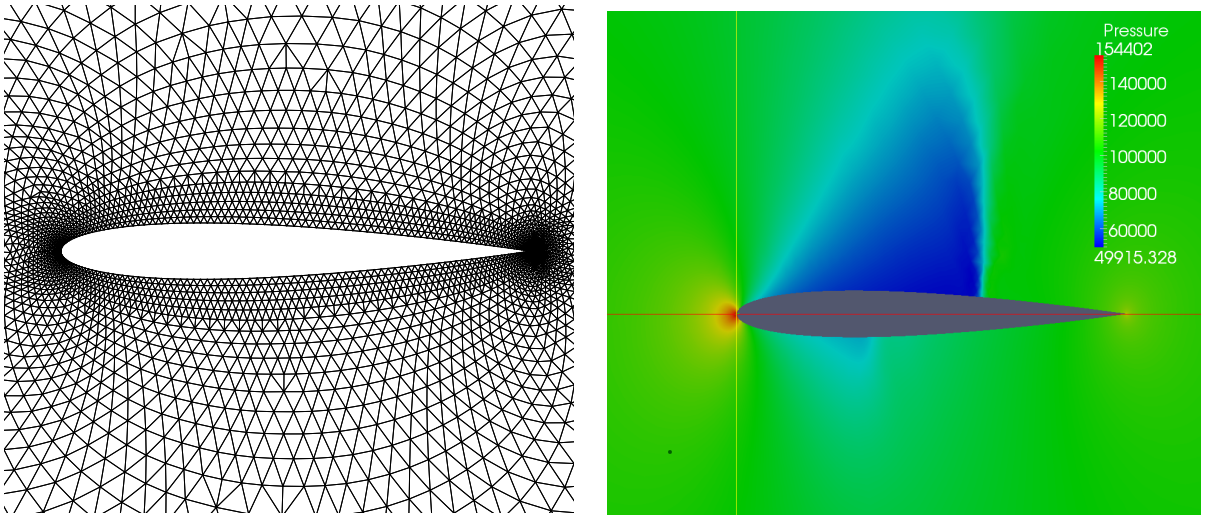
[a]https://github.com/su2code/TestCases.git

A. Inviscid Aerofoil Optimisation

The starting geometry for this test case is the NACA 0012 aerofoil. The flow conditions and optimisation problem are defined as follows:

- Mach Number = 0.8
- Angle of Attack = 1.25°
- Objective Function = $min(C_d)$
- Lift Coefficient Constraint: $C_l > 0.33$
- Pitching Moment Coefficient Constraint: $C_m > 0.034$
- No. of Design Variables = 8

The aeroifoil surface is discretised using 199 points; a detail view of the mesh around the aerofoil is given in figure 5-(a). Under these flow conditions a strong shock-wave forms over the upper surface of the aerofoil, as shown in figure 5-(b). For this case, four design variables were used to control each surface, corresponding to Bernstein polynomials of order 3. Following the optimisation using the CST parameterisation, the shock-



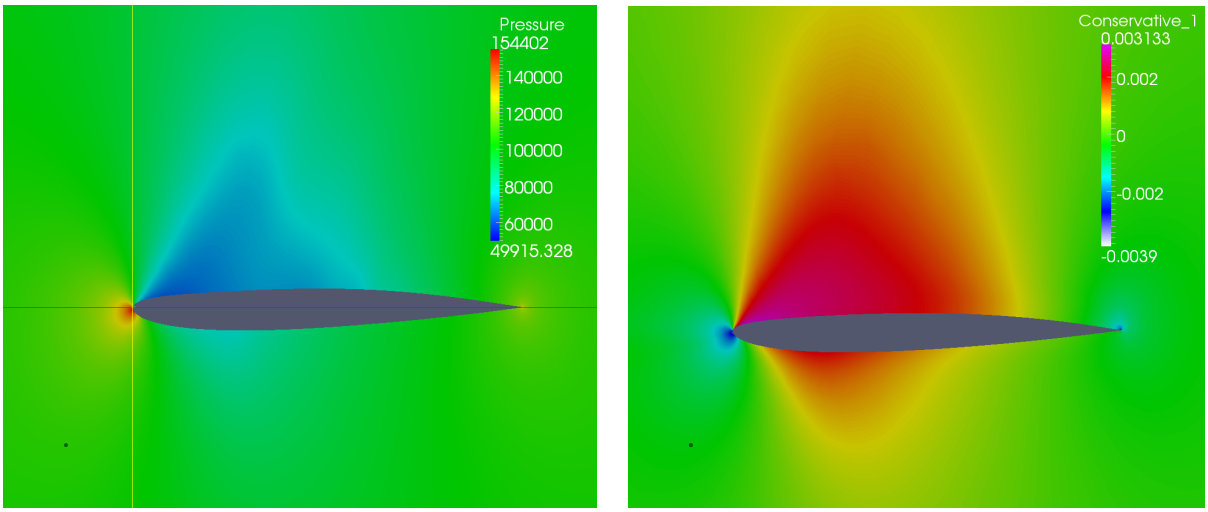
(a) NACA 0012 Surface Mesh Detail (b) NACA 0012 - Pressure Flow Field

Figure 5. Detail of NACA 0012 Surface Mesh and reference flow field, $M_\infty = 0.8$, $\alpha = 1.25°$

wave intensity for the new aerofoil, as shown in figure 6-(a), has been substantially diminished. The resultant field for the Adjoint Variable 1 is also shown in figure 6-(b). The drag reduction is obtained primarily through the reduction in curvature of the upper surface, which reduces the acceleration of the flow over the surface and prevents the formation of a strong shock to decelerate the flow back to free stream conditions. Figure 7 shows the convergence for the C_d from it's initial value of 0.02146 to it's final value of 0.0009915. The same case was repeated using a parameterisation based on Hicks-Henne functions, using 38 design variables. The CST method is seen to perform similarly to that of the Hicks-Henne bumps. For this case, convergence is achieved at an earlier number of cycles with the Hicks-Henne method, whereas a slight improvement on the optimum is obtained through the CST method. More importantly, both approaches converge on a very similar design and C_d .

B. Viscous Aerofoil Optimisation

The second case involved the optimisation of the RAE2822 transonic aerofoil using the RANS level aerodynamics, utilising the Spalart-Allmaras (SA) turbulence model. The computational grid uses an *O-grid* type, quadrilateral mesh, around the aerofoil surface, requiring 192 edges to define the surface. The initial wall spacing is 1.0×10^{-5} chord lengths; the remainder of the domain is discretized using triangles and the whole mesh contains 22842 elements. A detail of the mesh around the aerofoil is shown in figure 8-(a).



(a) Optimised Aerofoil - Pressure Flow Field (b) Optimised Aerofoil - Adjoint Variable 1 Field

Figure 6. Inviscid Optimisation: $M_\infty = 0.8$, $\alpha = 1.25°$

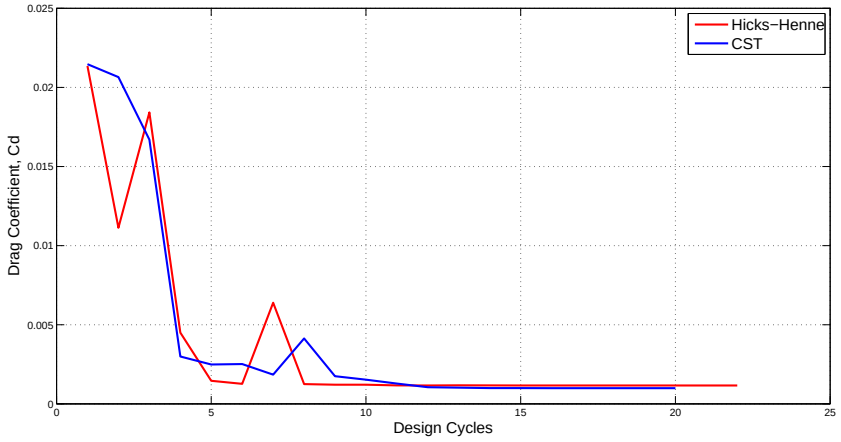
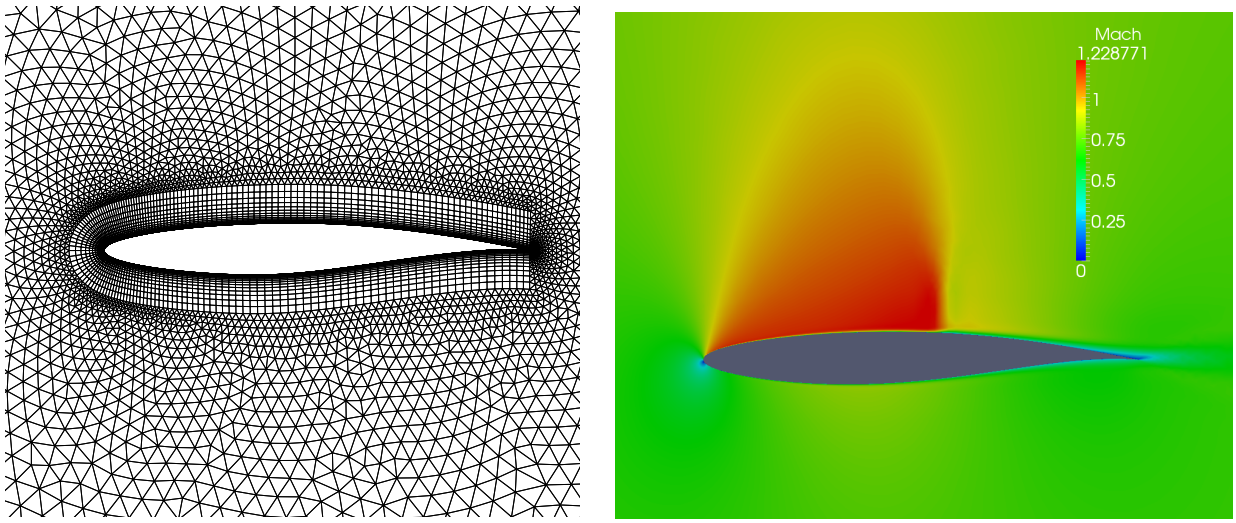


Figure 7. Drag Convergence

- Mach Number = 0.729
- Angle of Attack = 2.31
- Reynolds n. = 6×10^6
- Objective Function = $min(C_d)$
- No. of Design Variables = 8

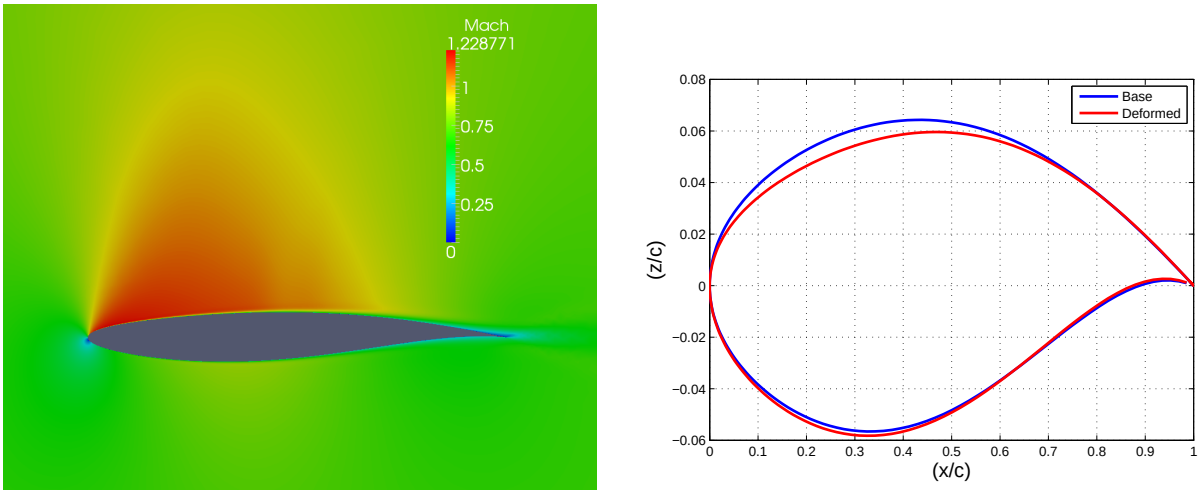
The Mach number flow field corresponding to the RAE2822 aerofoil can be seen in figure 8-(b). At these conditions a strong shock-wave is formed on the upper surface and there is a subsequent thickening of the boundary layer. Using the same number of design variables, the optimisation is able to reduce the drag coefficient from 0.01321 to a final value of about 0.01051. With reference to figure 9, the optimisation resulted in a slightly thinner aerofoil, with the maximum thickness on the upper surface moving rearwards. Again, this limited the flow acceleration on the upper surface, which resulted in a more gradual deceleration and avoided a normal shock as in the original aerofoil. However, this also resulted in the adverse pressure gradient (although milder) extending over a wider region on the upper surface; one consequence of this is

the increase thickness of the boundary layer with respect to the original aerofoil and an increase in viscous drag. The convergence of the optimisation process, shown in figure 10, is very similar to the inviscid case, with the final design reached in less than 15 cycles.



(a) RAE 2822 Surface Mesh Detail (b) RAE 2822 Mach n. Flow Field

Figure 8. RAE 2822 Mesh and initial Flow Field: $M_\infty = 0.729$, $\alpha = 2.31°$



(a) Optimised Aerofoi Mach n. Flow Field (b) Aerofoil Shape Comparison

Figure 9. Aerofoil Viscous Optimisation Solution

6. Conclusions and Outlook

An alternative aerofoil parameterisation based on CST functions was implemented within the SU^2 optimisation suite. The new parameterisation and aerofoil shape generation procedure were integrated with the SU^2 adjoint based optimisation methods, allowing the efficient and robust calculation of gradients with respect to the CST design variables. The alternative parameterisation was tested with two transonic optimisation problems. The first case was performed using Euler level aerodynamics, this was followed by a transonic viscous problem, solving the RANS and respective adjoint equations. In both cases, the optimisation process was able to converge to a new optimum and provide viable design options.

The procedure used here to augment the parameterisations available within SU^2 and expose the link

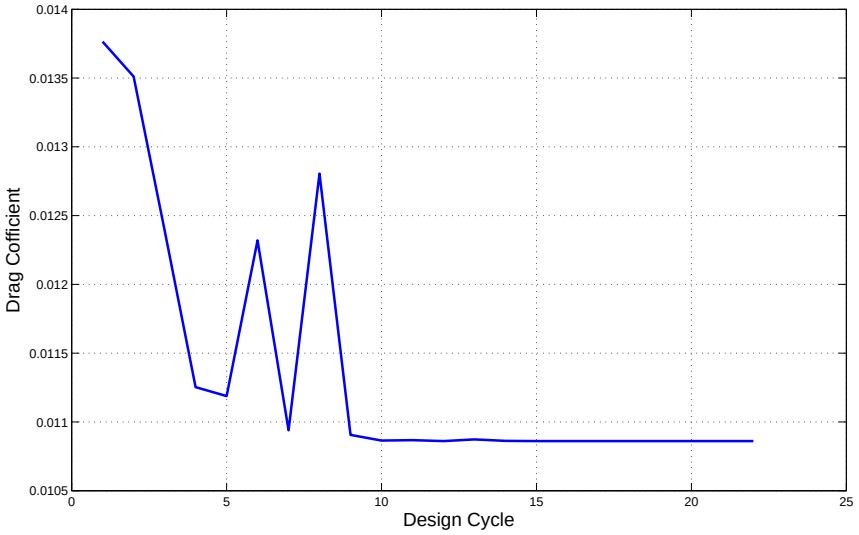


Figure 10. Objective Function Convergence - C_d

between geometric and surface sensitivities, opens the possibility to link the SU^2 solvers with other external alternative parameterisations, namely design velocities obtained directly from CAD models.

7. Acknowledgements

The funding provided for this research from the Engineering and Physical Sciences Research Council (EPSRC) is gratefully acknowledged.

References

[1]Mangler, K. W., "Design of Airfoil Sections," Tech. rep., 1938.

[2]Lighthill, M. J., "A new method of two-dimensional aerodynamics design," *R & M1111, Aeronautical Research Council*, 1945.

[3]Bauer, F., Garabedian, P., Korn, D., , and Jameson, A., *Supercritical Wing Sections I, II, III,*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1972, 1975, 1977, New York.

[4]Selig, M. S. and Maughmer, M. D., "Multipoint inverse airfoil design method based on conformal mapping," *AIAA journal*, Vol. 30, No. 5, 1992, pp. 1162–1170.

[5]Giannakoglou, K., "Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence," *Progress in Aerospace Sciences*, Vol. 38, No. 1, 2002, pp. 43–76.

[6]Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q., "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, Vol. 1, No. 1, 2011, pp. 32–49.

[7]Jameson, A., "Aerodynamic design via control theory," *Journal of scientific computing*, Vol. 3, No. 3, 1988, pp. 233–260.

[8]Jameson, A., Martinelli, L., and Pierce, N., "Optimum aerodynamic design using the Navier–Stokes equations," *Theoretical and computational fluid dynamics*, Vol. 10, No. 1-4, 1998, pp. 213–237.

[9]Drela, M., *Pros and Cons of Airfoil Optimization*, Frontiers of Computational Fluid Dynamics 1998, World Scientific, 1998.

[10]Palacios, F., Colonno, M. R., Aranake, A. C., Campos, A., Copeland, S. R., Economon, T. D., Lonkar, A. K., Lukaczyk, T. W., Taylor, T. W. R., and Alonso, J. J., "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design," No. Paper 2013-0287 in 51st AIAA Aerospace Sciences Meeting and Exhibit, January 2013.

[11]Kulfan, B. M., "Universal parametric geometry representation method," *Journal of Aircraft*, Vol. 45, No. 1, 2008, pp. 142–158.

[12]Economon, T. D., Palacios, F., and Alonso, J. J., "A viscous continuous adjoint approach for the design of rotating engineering applications," *AIAA Paper*, Vol. 2580, 2013, pp. 24–27.

[13]Samareh, J., "Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization ," *AIAA Journal*, 2001.