ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

OPTIMISATION OF A U-BEND USING A CAD-BASED ADJOINT METHOD WITH DIFFERENTIATED CAD KERNEL

Salvatore Auriemma[1], Mladen Banovic[2], Orest Mykhaskiv[3], Herve Legrand[1], Jens-Dominik Müller[3], Tom Verstraete[3] and Andrea Walther[2]

[1]OpenCascade
1 Place des Frères Montgolfier, 78280 Guyancourt, France
e-mail: salvatore.auriemma@opencascade.com

[2] University of Paderborn
Warburger Str. 100, 33098 Paderborn, Germany
e-mail: mladen.banovic@math.uni-paderborn.de

[3] Queen Mary University of London
Mile End Road, London, E14NS, UK
e-mail: o.mykhaskiv@qmul.ac.uk

Keywords: Adjoint Shape Optimisation, CAD Sensitivities, Automatic Differentiation.

Abstract. *In order to optimise the shape of a three-dimensional CAD-based model using the computationally efficient adjoint methods, the calculation of shape sensitivities, the derivatives of the surface position with respect to the design parameters, is required. This sensitivity is usually not available with CAD systems, but can be obtained by applying the Finite Difference method to CAD-system. Finite-Differences or part-analytic differentiation have been proposed to obtain sensitivities, but have their drawbacks. If source code is available, automatic differentiation can provide accurate derivatives without incurring topology changes or requiring hand-differentiation.*

This paper proposes the differentiation of the open-source CAD kernel - OpenCascade Technology (OCCT) with AD software tool ADOL-C (Automatic Differentiation by Overloading in C++). As a case study we consider the optimisation of pressure loss in a U-bend pipe. The geometry of the U-bend is parametrised in OCCT with a number of cross-sections lofted along a guiding path line. The corresponding geometric derivatives are used in CFD optimisation loops with the resulting shape outperforming the initial design.

1 INTRODUCTION

In industrial shape optimisation the computationally efficient gradient-based methods are used extensively. In particular, the adjoint method [1, 2] is the most effective approach as it allows to compute the sensitivities to an arbitrary number of control variables in a single computation. To complete the chain rule of derivatives, we also need the derivative of the shape parametrisation. A wide range of parametrisations have been developed, here it is useful to distinguish between CAD-free and CAD-based methods. CAD-free methods [3] also often referred as mesh- or lattice-based, optimise the positions of computational grid points using either a globally interpolated distortion field from radial basis functions (RBF), an auxiliary grid defining perturbations such as Free-Form deformation (FFD) or lattices of Hicks-Henne bumps. Mesh-based approaches use the surface mesh of the CFD grid [4] to impose a displacement. This design space is actually too rich for the CFD computation, high-frequency oscillations are not damped adequately, and hence require regularisation. A major drawback of all the CAD-free methods is that the optimised shape exists only as a mesh. Importing this shape back to CAD for further analysis or manufacturing is an unsolved challenge and typically incurs significant approximation. As a consequence the quality of the optimum is impaired.

As an alternative, CAD-based methods work with the CAD in the design loop and use CAD parameters or variables as the design variables. The main advantage is that CAD geometry is taken as an input and CAD geometry is produced as an output, however obtaining derivatives is more challenging. Xu et al. [5, 6] propose the NSPCC approach which uses the NURBS control points in the CAD-native boundary representation (BRep) as degrees of freedom. Additional constraints need to be imposed to retain the desired continuity between NURBS patches or to respect thickness, radius or build-space constraints. These constraints are evaluated numerically and the design space is the kernel of the constraint matrix which is computed using SVD. Selecting the SVD cutoff for non-singular modes provides an effective preconditioner of the design space. The NURBS geometry engine is implemented in source and derivatives are obtained by application of automatic differentiation. The main advantage of the NSPCC approach is that it is vendor neutral and only considers the BRep of the standardised STEP file. As a consequence, it is agnostic to any design parametrisation set up in the CAD system.

Robinson et al. [7] use the design parametrisation and internal variables set up in a closed-source CAD system as design variables and obtain derivatives with finite differences. To avoid issues with patch re-numbering and disappearance due to the finite-size displacements, the geometry is projected onto an STL approximation of the surface and the finite-differences of the displacements of grid nodes are evaluated on this STL, which is a computationally expensive process and further affects accuracy of the gradients. However, the method does allow to define a design space with constraints through the CAD parametrisation.

Dannenhoffer and Haimes [8] use the open-source CAD-kernel OpenCascade Technology (OCCT) as a geometric engine. They apply analytic differentiation to known simple shapes such as circles and cylinders defined by origins, radii and axes. Remaining derivatives are evaluated using finite differences.

In this paper we present the differentiation of the entire geometric kernel of OCCT with Automatic Differentiation (AD) by the ADOL-C AD tool [9]. Compared to other CAD-based approaches this results in a number of significant advantages. Similar to [7, 8] the design space is defined by the parametrisation of the CAD system which allows to build in geometric constraints, while in the NSPCC approach [6] these constraints have to be reimposed. As opposed to the finite-difference approaches [7, 8] the geometric sensitivities are exact and not affected by truncation error. Most importantly, there is no finite-size displacement of the geometry during the differentiation, hence this approach is not affected by patch re-numbering or disappearance and a projection onto an intermediate surface is not necessary. This should significantly reduce the computational cost of the method. Finally, automatic differentiation can also be performed in reverse mode [10], which has the potential to dramatically reduce the computational cost of derivative computation as used for the flow solver (cf. Sec. 2.1). While we do not use the reverse-mode differentiation in this paper, the successful demonstration of forward-mode differentiation presented here is a step toward this.

The paper is structured as follows. Sec. 2 presents the governing equations for flow and its adjoint, as well as the assembly of the relevant derivatives. Sec. 3 describes the differentiation of the OCCT CAD system. Sec. 4 shows the U-bend testcase and the definition of the parametrisation, followed by computational results in Sec. 5.

2 CAD-DRIVEN SENSITIVITY

2.1 Primal and adjoint flow equations

To optimise a scalar cost function J which describes the aerodynamic performance of the system of interest, the optimisation problem can be stated as

$$\min_{\alpha} \; J(U(X(\alpha)), X(\alpha), \alpha) \,, \tag{1}$$

$$R(U(X(\alpha)), X(\alpha)) = 0 \,. \tag{2}$$

Equation (2) denotes the system of steady-state Reynolds-Averaged Navier-Stokes equations, where the residual R is driven to zero. R is a function of the state variable U and the mesh coordinates X, both depending on the design parameters α. The objective function J could correspond to drag, lift, total pressure losses, etc. Application of the chain rule to the system yields

$$\frac{dJ}{d\alpha} = \left[\frac{dJ}{dX} + v^T f\right] \frac{\partial X}{\partial \alpha} \,. \tag{3}$$

Here v represents the solution of adjoint equations:

$$\left(\frac{\partial R}{\partial U}\right)^T v = \frac{\partial J}{\partial U} \,, \tag{4}$$

where

$$f = -\frac{\partial R}{\partial X} \,. \tag{5}$$

The algorithm of calculating the surface mesh sensitivities consists of evaluating the volume sensitivities $\frac{dJ}{dX}$ and then projecting them onto design surfaces. This is performed after primal and adjoint CFD runs, followed by projections which use the mesh-perturbation algorithms. For instance, the spring-based or elasticity-based mesh deformation algorithms detailed in [6] could be used. In other words, the mesh perturbation algorithm creates a correspondence between the movements of volume and surface mesh points. Hence, denoting the surface grid points with X_S, the mapping $X_S \to X(X_S)$ allows to rewrite the sensitivity in equation (3) in terms of displacements of the surface grid points

$$\frac{dJ}{d\alpha} = \frac{dJ}{dX_S} \frac{dX_S}{d\alpha} \,, \tag{6}$$

$$\frac{dJ}{dX_S} = \left[\frac{dJ}{dX} + v^T f\right]\frac{\partial X}{\partial X_S} \,. \tag{7}$$

The gradient obtained in (6) is used in an iterative optimisation process to find optimal design parameters α by updating

$$\alpha^{(n+1)} = A(\alpha^{(n)}, \frac{dJ}{d\alpha}(\alpha^{(n)}))\,, \tag{8}$$

were A represents an optimisation algorithm of choice.

2.2 Assembling CAD-based sensitivity

In the previous section we considered a more general nature of the design parameters, from now on we will refer to α as the parameters of a given CAD-model. In this context the two terms in equation (6) correspond to the flow and geometrical (CAD) sensitivities, respectively. These terms, which are composed of derivatives in each of the n surface mesh points, could be calculated independently from each other and afterwards assembled in a global sensitivity by

$$\frac{dJ}{dX_S} = \left[\frac{dJ}{dX_{S,i}}\right]_{i=1,\dots,n}. \tag{9}$$

Similarly, the CAD sensitivity w.r.t. one of the m design parameters $\alpha = (\alpha_1, ..., \alpha_m)$ could be written as:

$$\frac{dX_S}{d\alpha_j} = \left[\frac{dX_{S,i}}{d\alpha_j}\right]^{i=1,\dots,n}_{j=1,\dots,m}. \tag{10}$$

For a 3D CAD-model the mesh points X_S have three components, hence the two matrices listed above have dimensions $3 \times n$. Therefore, in order to calculate the gradient components, one primal and adjoint run of a CFD solver is required, and m computations of the matrix in equation (10). Finally, the m components of the gradient are obtained from the scalar multiplication of two previously written matrices as in equation (6):

$$\frac{dJ}{d\alpha_j} = \sum_{i=1}^{n} \frac{dJ}{dX_{Si}}\frac{dX_{Si}}{d\alpha_j} \,. \tag{11}$$

The process of finding the CAD derivatives (second term) in the last equation will be described in detail in the following sections.

The optimisation algorithm consists of the following steps: for the initial design parameters, the CAD-model and the corresponding mesh are built. Afterwards, at each iteration one solves primal and adjoint flow equations and assembles the gradient as described above. Then, the CAD-model is updated with new parameters and the mesh deformation step is performed for the computational grid to match the updated CAD geometry.

3 AUTOMATIC DIFFERENTIATION OF OPENCASCADE TECHNOLOGY

3.1 Introduction to ADOL-C

ADOL-C is a software tool that facilitates the computation of first and higher derivatives of vector functions that are defined by computer programs written in C/C++. It uses the *operator overloading* concept, which means that it is not generating intermediate source code as it is the case for the source transformation approaches [9]. ADOL-C supports the following modes of differentiation:

- Forward (options: *trace-based* or *traceless*),
- Reverse (*trace-based*),

where *trace* is an internal representation of the function to be differentiated, produced by operator overloading. The difference between *trace-based* and *traceless* mode is that in *traceless* mode the derivative computation is done at the same time as the function evaluation. For computing the derivatives using *trace-based* variants, one has to call driver functions provided by ADOL-C after the trace generation. For the purpose of this article, the *traceless forward differentiation*, which computes first order derivatives in scalar mode and in vector mode, is considered.

3.2 Introduction to OCCT differentiation

A key ingredient for automatic differentiation by overloading is the concept of an *active* variable - which is named `adouble` in ADOL-C. All variables that may be considered as differentiable quantities at some time during the program execution must be of an `active` type. Therefore, the integration of the ADOL-C library into a certain code is done by injection of its specific `adouble` type instead of the native `real` type. This kind of integration is not simple when facing complicated *object-oriented* code like OCCT.

Several possible ways of source code modification were considered for ADOL-C integration, but one was taken as a way to proceed with the full sources - the `typedef` approach. The `typedef` approach is the most intrusive way of integrating ADOL-C into OCCT because of replacing all `double` by `adouble`, by using an existing `typedef` which is named `Standard_Real` in OCCT. It is the fastest way of integration because code modification should be as minimal as possible, while the drawback is about sacrificing memory and efficiency to some extend since all `double` variables, even ones not needed for differentiation, will be `adouble` objects.

Although the idea looks simple, it is not as straightforward as one would expect. The differentiation involved a significant amount of code modification and even after successful compilation, a large number of run-time errors had to be resolved during the testing phase. These issues will not be explained here, but will be set out in detail in a future paper that will be mainly focused on the automatic differentiation of OCCT.

After fixing the parts of the code that were giving the errors, a major part of the full differentiated kernel is working. There are still some run-time errors that have to be resolved, but they are not related to the parts of OCCT executed in the application considered here. Before using the differentiated kernel, AD has been verified against Finite Differences (by a *central difference scheme*) in the required OCCT methods for the U-bend construction.

4 PARAMETRISATION OF THE U-BEND

4.1 Geometry

The U-bend under investigation is a typical internal cooling channel for the turbine blades in turbomachinery applications [11]. The geometry is shown in Figure 1. It consists of a circular U-bend with a hydraulic diameter $D = 0.075m$.

4.2 Parametrisation

The parametrisation is based on a cross-sectional design approach which takes N-curves/N-slices as inputs in order to construct a final surface. Each slice consists of a closed wire com-

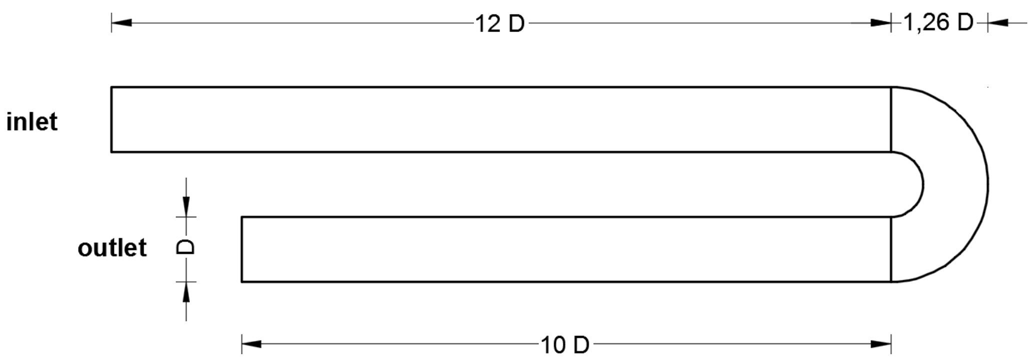


Figure 1: U-bend Dimensions

posed by 4 Bezier curves; every Bezier curve is characterized by 4 control points and shares the first and the last control point with the previous and the following Bezier curve, therefore having in total 12 control points, as shown in Figure 2. This choice allows the section to assume a wide variety of shapes that the optimiser could impose during the optimisation loop.

The plane where the slice lies is constructed in the following way:

- define a B-spline planar curve called *pathline* that drives the U-part of the U-bend in the 3D space.
- Take a point P on the pathline and the vector V tangent to the pathline in P.
- Construct the plane that holds the 12 control points of the section passing through the point P defined as the origin of the plane's (x, y, z) axis, orthogonal to V (whose direction is defined as the z axis) and with y axis orthogonal to the plane where the pathline lies.

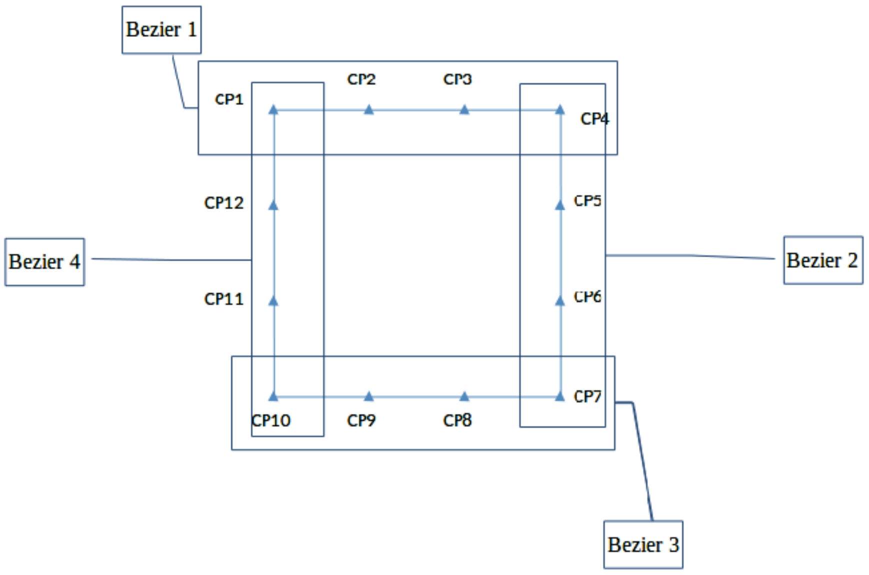


Figure 2: U-bend slice

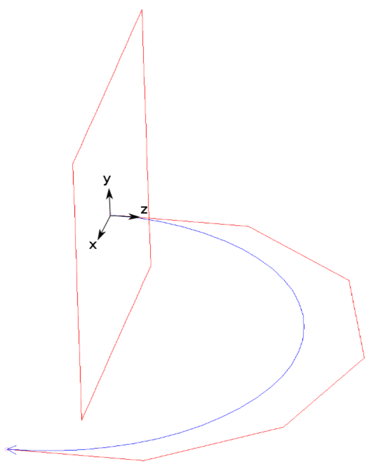


Figure 3: Construction of the first plane

An example of the constructed plane in shown in Figure 3. The slice of the U-part is free to move onto the plane constructed as explained above. Each control point of the section is characterised by a law of evolution along the pathline; in particular, the control point laws are described by B-spline curves in a (ϵ,η,τ) modelling space. These B-spline laws consist of 8 control points whose (ϵ,η) coordinates are the final design parameters of the simulation; the τ coordinates are not considered as design parameters - their values constrain the first and last control point to correspond to the first and the last slice, respectively. The laws are intersected N-times by a plane that is always parallel to plane (ϵ,η) and that passes through a point of coordinates (0,0,P_{length}), where P_{length} is the length of the curve pathline at the point of intersection between the slice and the pathline; this approach in particular is called "Clipping" technique. For each intersection point between the plane and the law, the coordinates of the point of intersection (ϵ_n,η_n) are assigned as coordinates (x_n,y_n) of the control point of the section to which the law is referred. Two examples of the *clipping* technique are shown in Figure 4.
Furthermore to assure the tangency constraint between the U-part and the vertical pipes of the U-bend during the optimisation, for every law the coordinates of the first two and the last two control points are not considered as design parameters of the simulation. In general it would be possible to allow the second and the penultimate law control points to move in τ direction (which means to assign the τ coordinates of these control points as design parameters) but at this stage we preferred to simplify the simulation making the assumption that the optimisation does not consider τ coordinates as design parameters. In a future article an optimisation will be presented that considers all the coordinates of the law control points.

Once that all the control points of a slice assume their position on the plane as described above, it is possible to get the N-slice of the U-bend. The N-slices are then used to obtain the final U-bend B-spline surface. To achieve this result, an Open Cascade cross-sectional design technique [12] that approximates the 4 *NURBS* surfaces (one for every Bezier curve of the section) has been used. Finally the 4 surfaces are merged using an Open Cascade Topology tool.

To complete the U-bend for the test case, the first and the last wire of the U-part are extruded with an Open Cascade tool to get the vertical pipes that corresponds to the inlet and the outlet pipes, as shown in Figure 5.

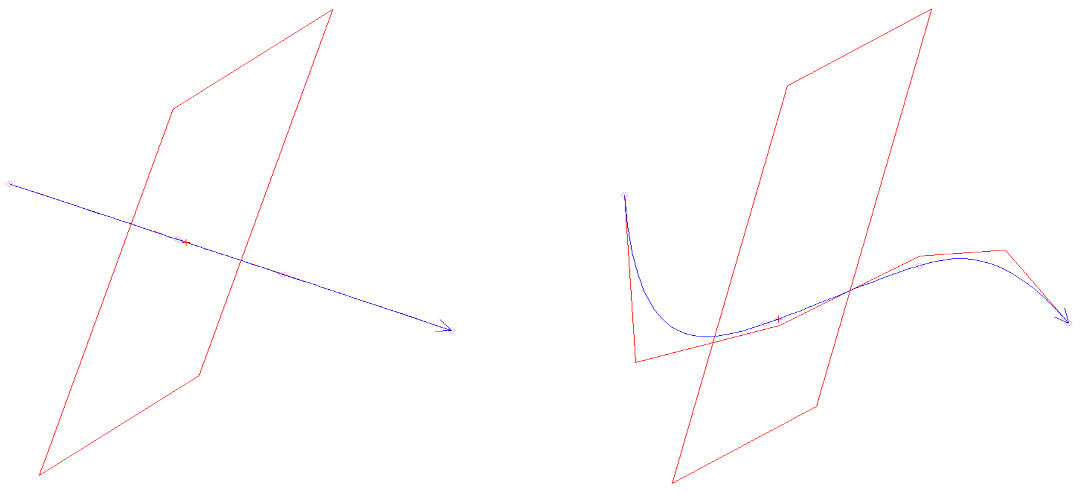

(a) Clipping before optimisation (b) Clipping after optimisation

Figure 4: Clipping before and after optimisation

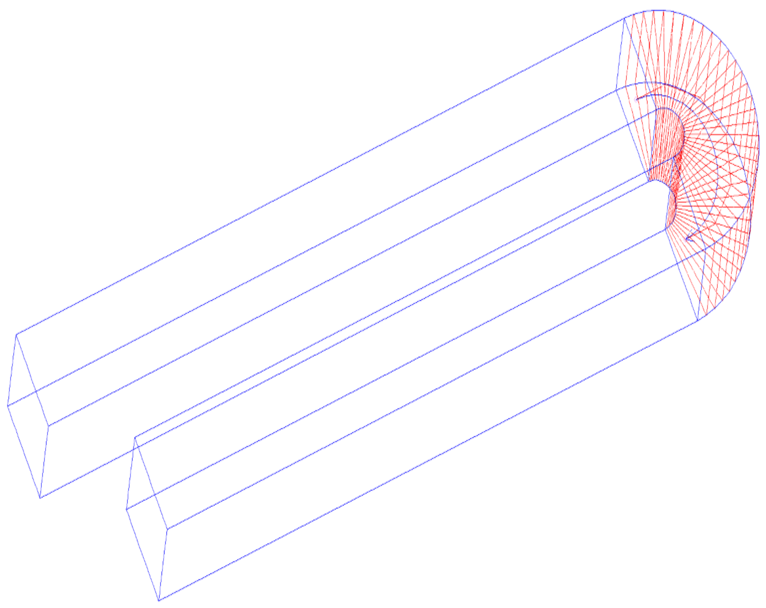


Figure 5: Final U-bend

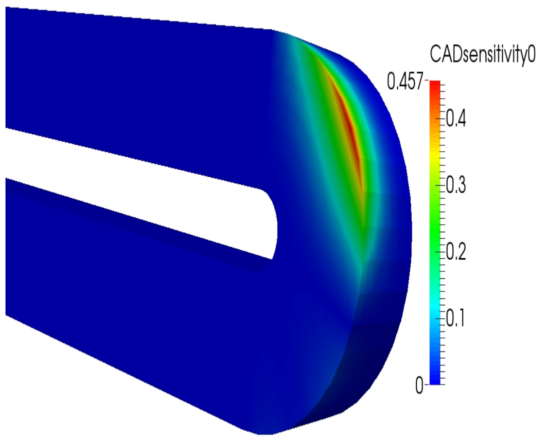
5 OPTIMISATION RESULTS FOR THE U-BEND

The U-bend model described in the previous section will be optimised to reduce total pressure losses between the inlet and outlet:

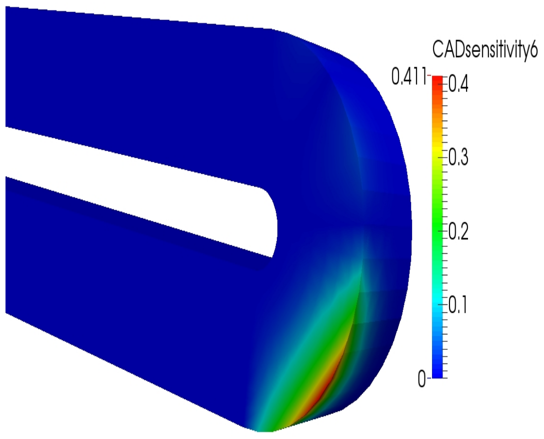
$$J = \frac{\int_{inlet} P(\mathbf{u}\cdot\mathbf{n})dS + \int_{outlet} P(\mathbf{u}\cdot\mathbf{n})dS}{\int_{inlet}(\mathbf{u}\cdot\mathbf{n})dS}\,. \tag{12}$$

Only the U-part is subject to design changes. This testcase was initially proposed in [11]. Since we wanted to point out in particular the strength of CAD-based optimisation approach using the differentiated OCCT, a simplified version of the U-bend was chosen to reduce dramatically the computational cost of the CFD. In comparison with [11] much shorter lengths of outlet and inlet legs were considered. Furthermore, the horizontal outlet and inlet part were not subject to design changes. While longer legs assured fully developed flow at the design part (U-part), shorter legs reduced the size of computational mesh. To further accelerate CFD computations, turbulent flow was replaced with a laminar one. We want to emphasize that the latter does not affect in any way the proposed methodology and more correct physical flow conditions could be applied without any further restrictions.

The CAD-model parametrised in differentiated OCCT depends on 96 design parameters (12 laws of evolution $\times$ 4 control points $\times$ 2 directions), as described in section 4.2. In Figure



(a) Sensitivity Parameter 0



(b) Sensitivity Parameter 6

Figure 6: CAD sensitivities

6 the magnitudes of the CAD sensitivities of two parameters are shown. Since the design parameters correspond to control points positions of the law's B-spline curves, these sensitivities also inherited their local influence feature.

The structured mesh was created to match the given geometry. This could be achieved either by creating a mesh from corresponding CAD-file (STEP, IGES, etc.) or by creating a mesh directly matching the prescribed sizes (feasible in case of non-complex geometries).

For the flow simulation the in-house CFD solver *mgopt* with incompressible setting was used. The solver is based on the vertex-centred Finite Volumes scheme and facilitates a geometric multi-grid method. It includes the differentiated code provided by AD tool Tapenade [13] to implement the discrete adjoint method. To increase stability and robustness of CFD runs the solver makes use of novel implicit time stepping JT-KIRK scheme [14].

The flow is considered to be incompressible with low free stream Mach number $Ma = 0.04566$. Following boundary conditions were imposed: subsonic inlet at the end of the longer leg, and subsonic outlet at the shorter one. On all other boundaries no-slip wall conditions were specified. The air properties at ambient conditions are: density $\rho = 1.204kg/m^3$, viscosity $\mu = 1.813 \times 10^{-5}kg/(sm)$, pressure $P = 101300Pa$ and temperature $T = 293.15K$.

As the optimisation algorithm the Conjugate Gradient method with Armijo-type line search was used. For this particular problem this method has the additional advantage of a direct control over design parameters update, which influences robustness of the mesh perturbation step. The gradient (11) which drives the optimisation was tested with the Finite Differences for few design parameters, showing mutual agreement.

In Figure 7 the decrease of cost function is shown during the course of optimisation. This demonstrates that optimiser components work as expected while the optimised design outperforms the initial one by approximately 4%. The corresponding geometry update is shown in Figure 8.

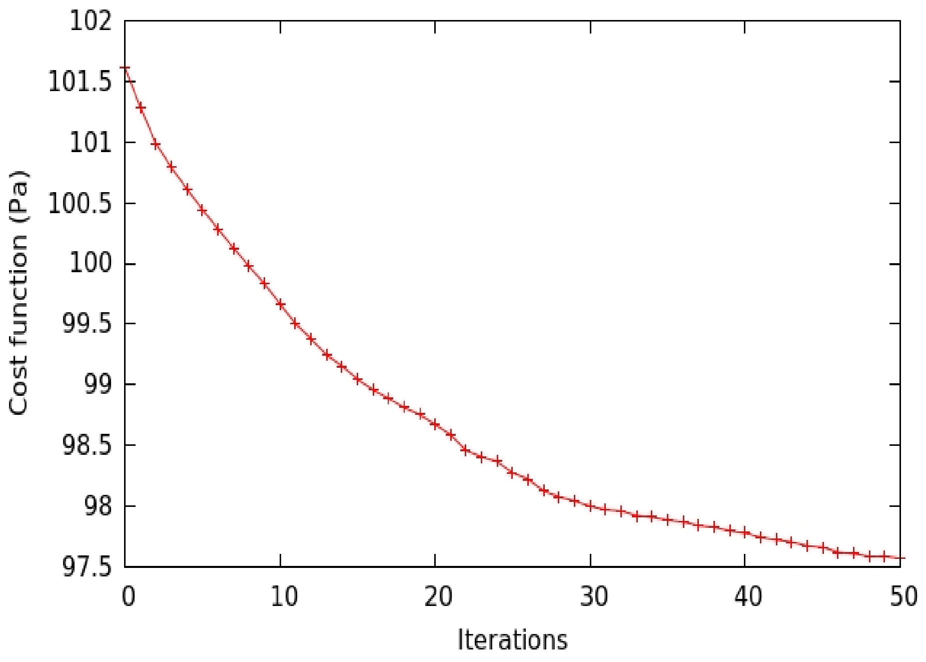
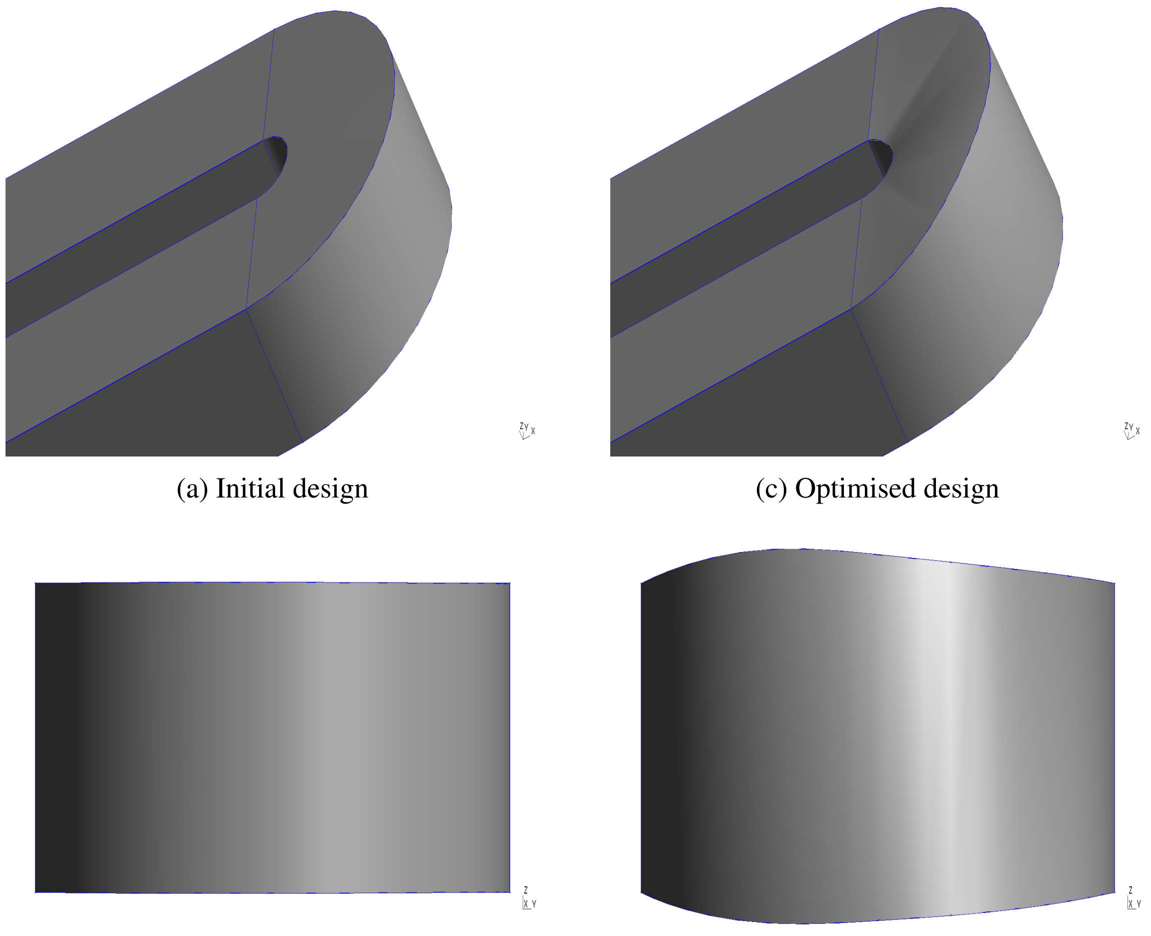


Figure 7: Optimisation loop history

The relatively small improvement can be explained with the case settings. Laminar conditions simplified the flow behaviour in comparison with the turbulent. Fixed inlet and outlet legs also reduced the possible design change. In this setting pressure losses are related mostly to frictional forces and the flow separation occurring after the U-part. In Figure 9 the change in velocities before and after the optimisation are shown. The optimiser managed to suppress the separation bubble and reduce the cost function mainly by increasing the outer U-part hydraulic diameter and a small decrease of the inner circular surface diameter.

6 CONCLUSIONS AND FURTHER WORK

This paper shows that the automatic differentiation of a fully developed CAD system is feasible, as well as its integration into the design loop. The CAD-model was built in OCCT and its derivative information was used in the global gradient assembly driving the optimisation process. Despite relatively small improvements, the advantages of CAD-based optimisation were exploited. First, the output of the optimisation process was the updated CAD-model. Second, the direct use of the parametrisation ensured a seamless process of parameters update without appearance of cross-patches discontinuities. This is an advantage in comparison with other CAD-based approach, when the control points of the surfaces are taken directly as design parameters. Nevertheless, the obtained CAD-software sensitivities were used in the simplification of the testcase mentioned above resulted in relatively small design changes and cost function decrease. The further investigation on this testcase is planned to solve CFD on a finer mesh and for correct flow physics including additional design domains. Together with more advanced optimisation methods the problem could result in more apparent design update.



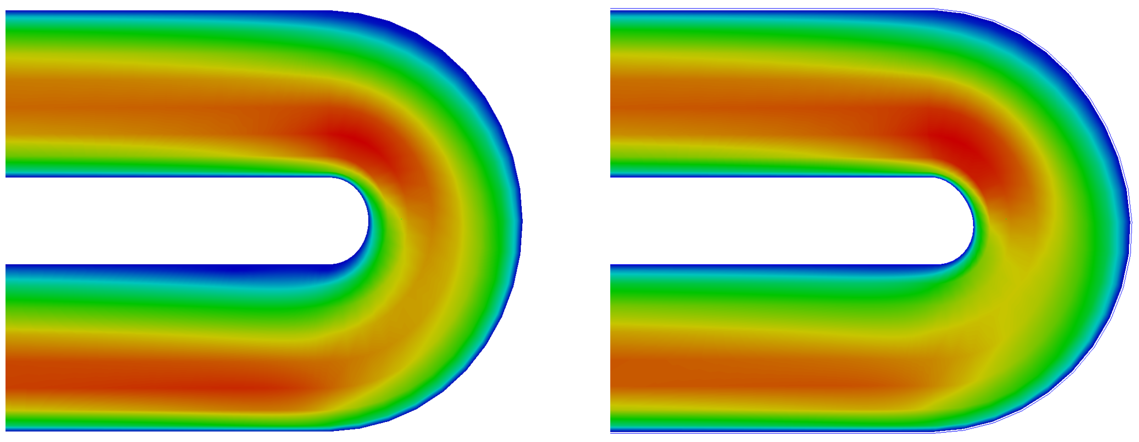
(a) Initial design

(c) Optimised design

(b) Initial outer circular U-part surface

(d) Optimised outer circular U-part surface

Figure 8: Modified Geometry



(a) Initial design

(b) Optimised design

Figure 9: Velocity Magnitude

ACKNOWLEDGMENTS

The authors are very thankful to *Mr. Sergey Slyadnev (OpenCASCADE)* for his support related to the OpenCASCADE Technology, its differentiation and parametrisation.
This research is a part of the IODA project - Industrial Optimal Design using Adjoint CFD. IODA is *Marie Sklodowska-Curie Innovative Training Network* funded by European Commission under Grant Agreement No. 642959.

REFERENCES

[1] A. Jameson, Aerodynamic Design via Control Theory. *JSC*, 3: 233–260, 1988.

[2] M. B. Giles and M. C. Duta and J.-D. Müller and N. A. Pierce, Algorithm Developments for Discrete Adjoint Methods. *AIAA Journal*, 41(2) : 198 –205, 2003.

[3] J.A. Samareh, Aerodynamic shape optimization based on free-form deformation. *AIAA*, 4630: 1–13, 2004.

[4] A. Jameson and J. Vassberg, Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 9(3), 281 – 296, 2000.

[5] S. Xu, J. Wolfram, J.-D. Müller, CAD-based shape optimisation with CFD using a discrete adjoint. *Int. J. Numer. Meth. Fluids*, 74: 153–168, 2013.

[6] S. Xu, D. Radford, M. Meyer, and J.-D. Müller, CAD-Based Adjoint Shape Optimisation of a One-Stage Turbine with Geometric Constraints. ASME Turbo Expo 2015: Turbine Technical Conference and Exposition, 2C: Turbomachinery.

[7] T. Robinson, C. Armstrong, H. Chua, C. Othmer and T. Grahs, Optimizing Parameterized CAD Geometries Using Sensitivities Based on Adjoint Functions. *Computer-Aided Design and Applications,* 9(3), 253-268, 2012.

[8] J. Dannenhoffer and R. Haimes, Design Sensitivity Calculations Directly on CAD-based Geometry. *53rd AIAA Aerospace Sciences Meeting, AIAA SciTech*, AIAA 2015–1370.

[9] A. Griewank and A. Walther. *Getting Started with ADOL-C*, 181 – 202. *Combinatorial scientific computing*, 181 – 202, 2009.

[10] A. Griewank and A. Walther, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. *SIAM*, 2008.

[11] T. Verstraete, F. Coletti, J. Bulle, T. Vanderwielen, and T. Arts, Optimization of a U-Bend for Minimal Pressure Loss in Internal Cooling Channels - part 1: Numerical Method. *ASME Turbo Expo 2011: Power for Land, Sea and Air*, GT2011-46541.

[12] L. Piegl, *On NURBS: a Survey*. Int. IEEE Computer Graphic and Application, 55 - 71, 1991.

[13] L. Hascoët, V. Pascual, Tapenade 2.1 users guide. *Technical Report 0300*, INRIA, 2004.

[14] S. Xu, D. Radford, M. Meyer, J.-D. Müller, Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, 99: 175 – 195, 2015.