Discrete adjoint CFD solver for first and second derivatives via Automatic Differentiation

Faidon Christakopoulos & Jens-Dominik Müller

School of Engineering and Materials Science
Queen Mary, University of London

Conference on Industrial Design Optimisation for Fluid Flow
Munich, 28^{th} March, 2012

Introduction

Motivation for adjoint second derivatives :

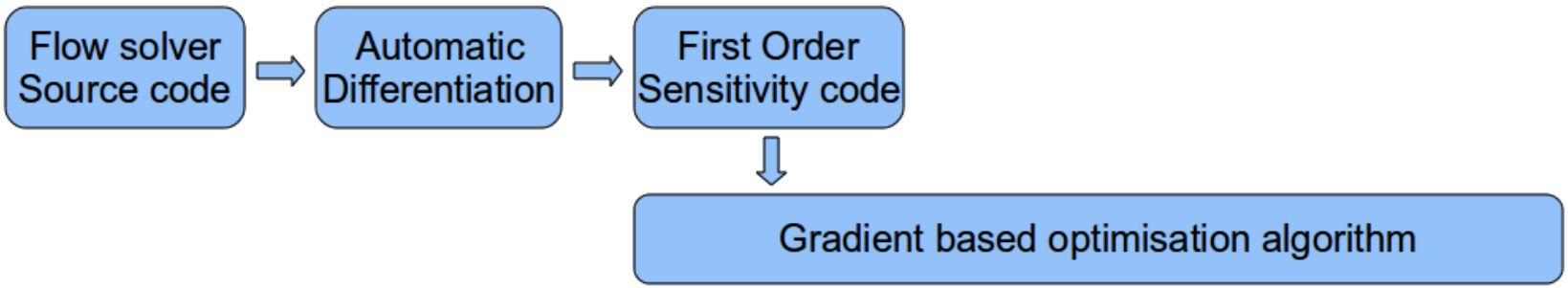
- Uncertainty quantification
- Robust optimisation
- Design convergence acceleration

Why Automatic Differentiation?

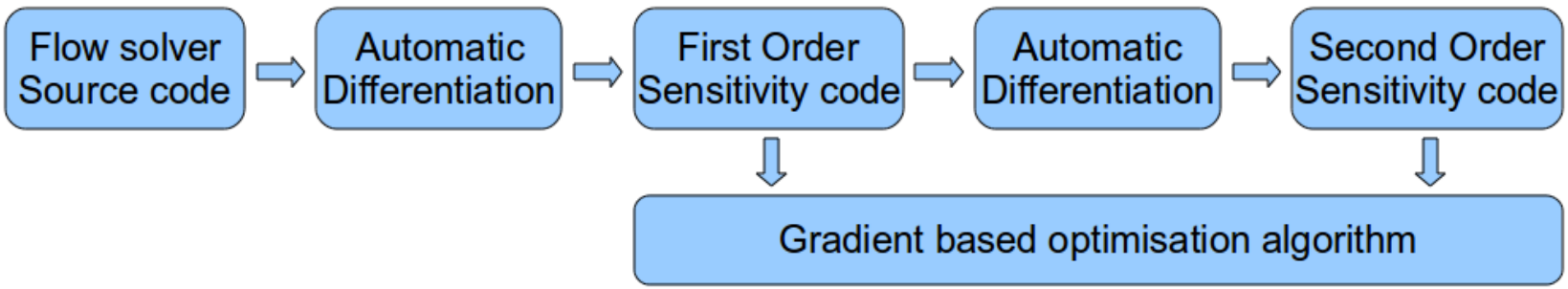
- Automation of the sensitivity code derivation
- Exact transposition matrices
- Direct translation of flow solver changes to the sensitivity solver

Introduction

First order adjoint via AD :



Second order adjoint via AD :

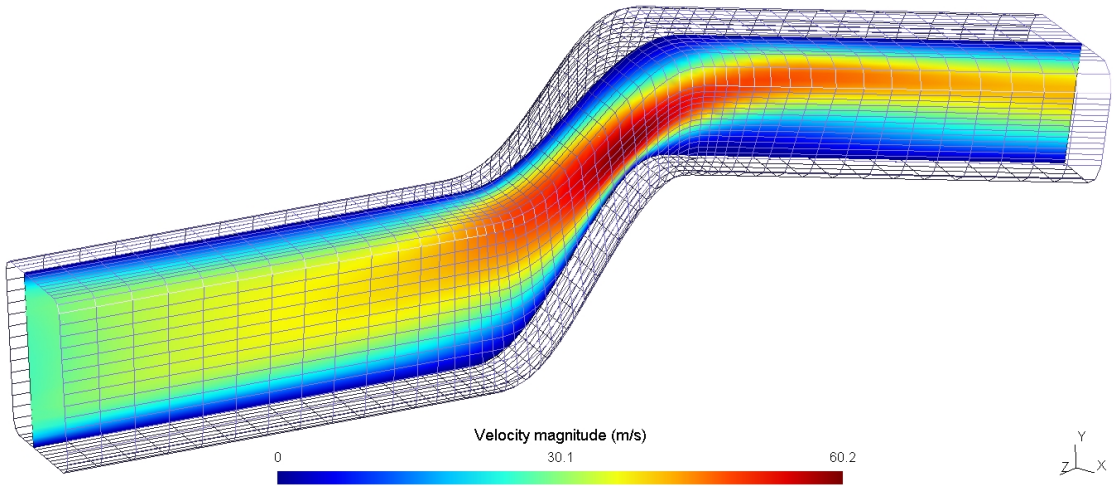


The flow solver

Basic characteristics

- 2D/3D vertex centered
- Euler & Navier-Stokes
- Compressible, including low speed/viscous preconditing
- Hybrib elements (up to hexahedral)
- Spalart-Allmaras turbulence model
- Multigrid

Flow: S-Bend duct



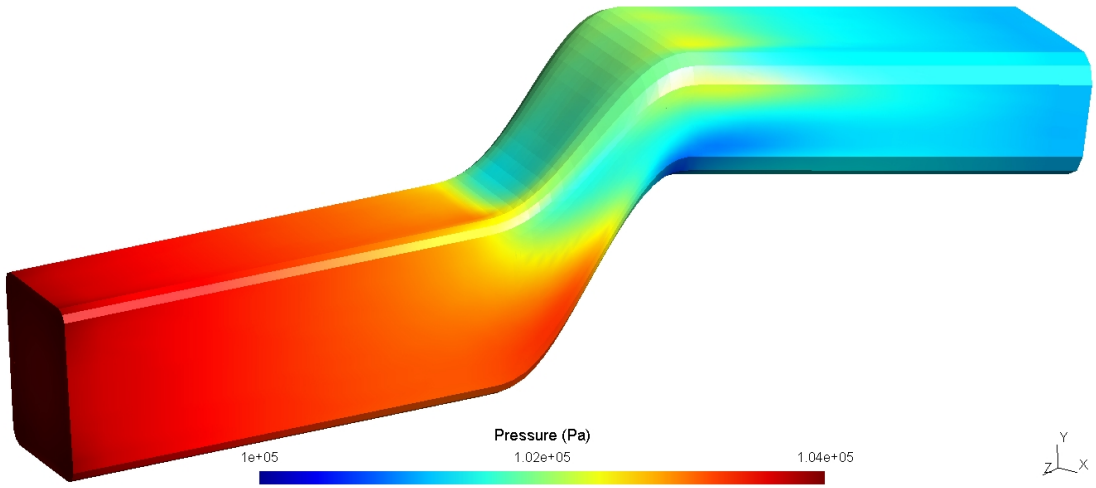


Figure: Flow through an S-Bend (Geometry source VW).

Flow: Formula One front wing

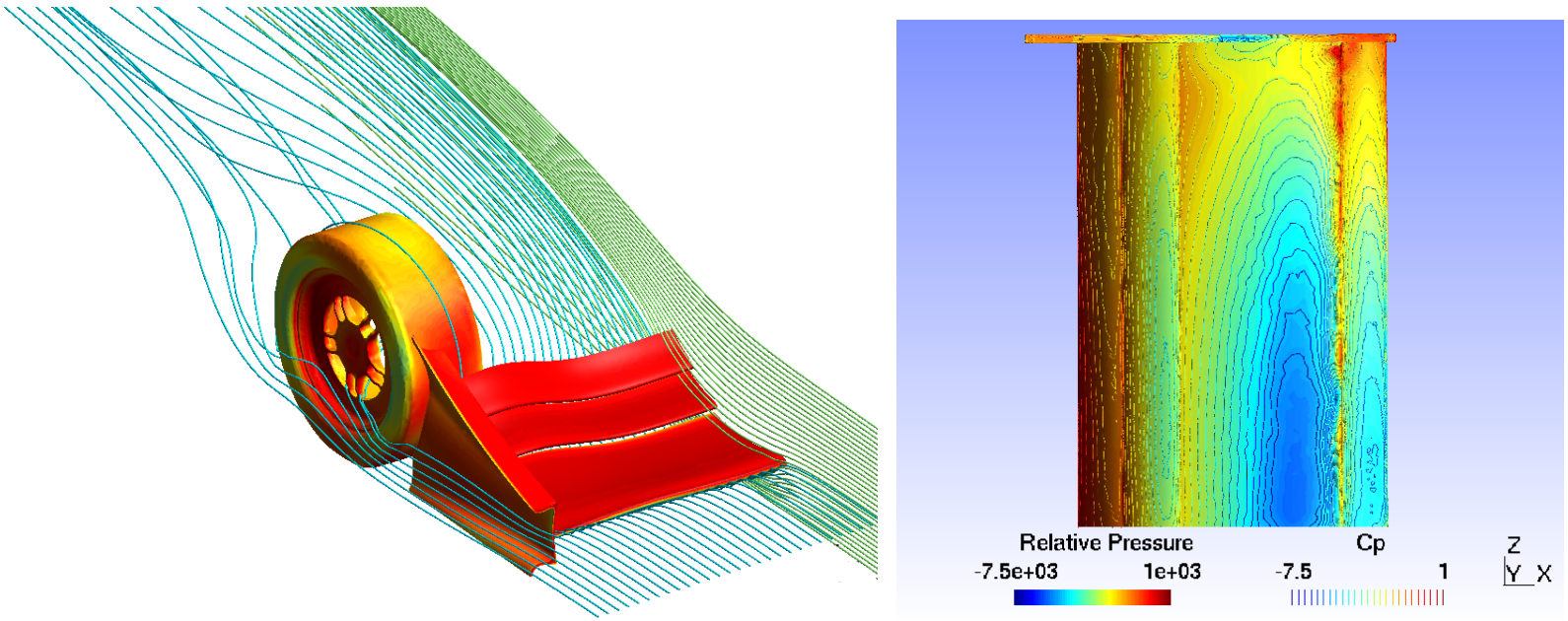


Figure: Pressure and velocity vectors over a Formula front wing.

Flow: Ahmed body

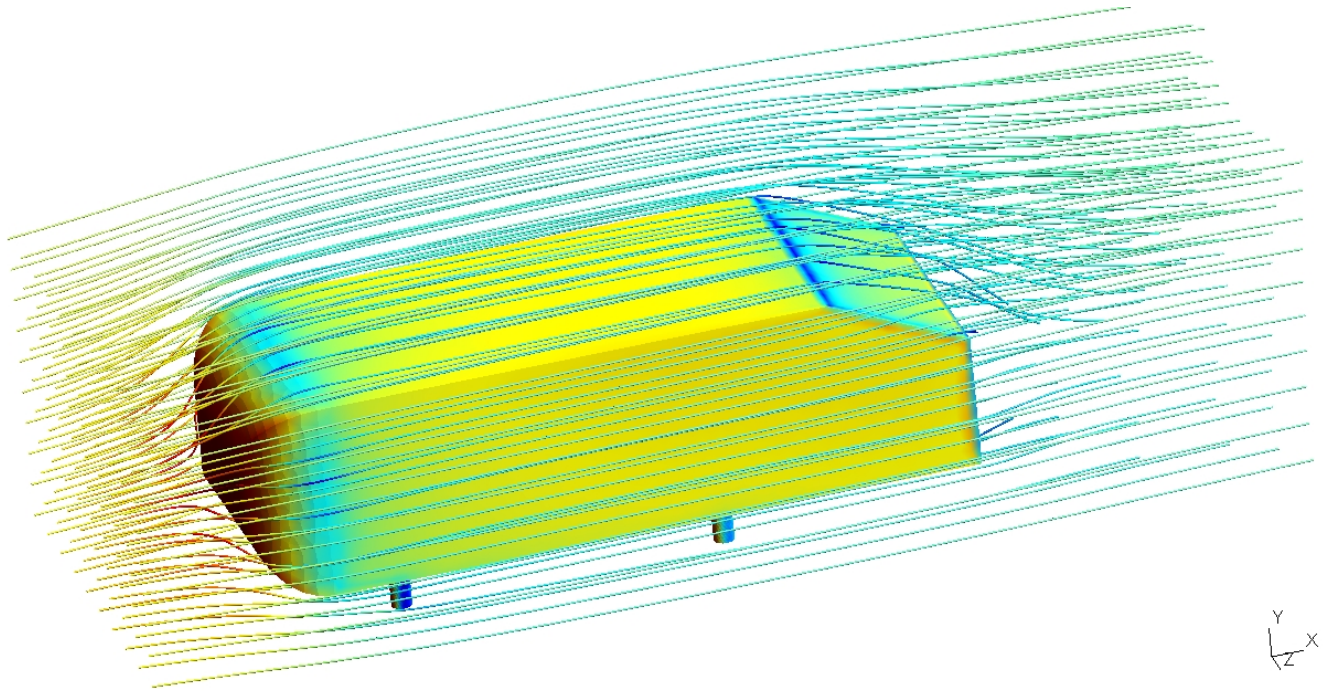


Figure: Total pressure and streamlines over the Ahmed Body.

The adjoint solver

Steps to the adjoint via Automatic Differentiation :

- Make the flow source code AD-able
- Preprocess source code
- Apply Automatic Differentiation
- Assemple AD and source code correctly together
- Compile

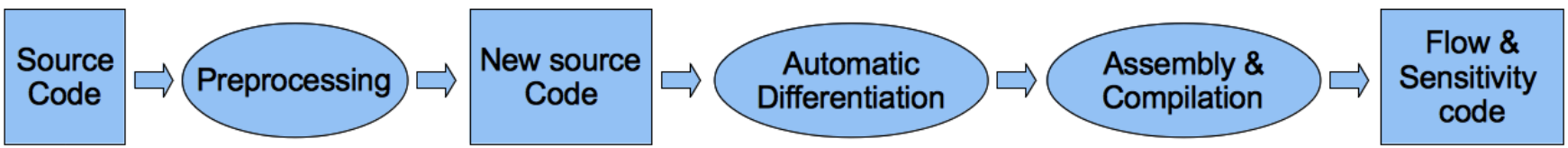


Figure: Steps for sensitivity code via AD.

Adjoint performance acceleration :: Assembled time stepping

Primal time-stepping loop ("→" input, "←" output) :

```
call init_flow ( ←Q )
call metrics ( →X, ←Nrm )
do nIter = 1,mIt
 call residual ( →Q, ←R, →Nrm )
 call update ( →R, ←Q )
end do
call cost_fun ( →X, →Q, ← cost )
```

Altered adjoint time-stepping loop :

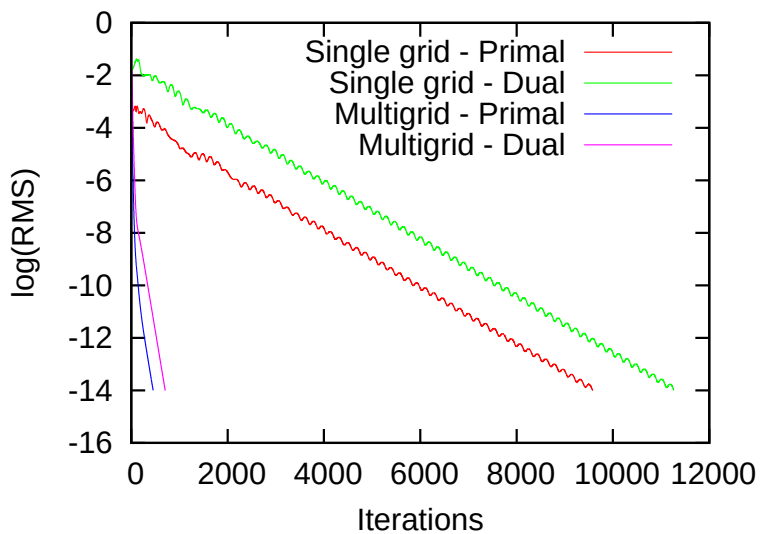
```
do nIter = 1,mIt
 call residual_b ( Q, ← R_b, R, → Q_b, Nrm )
 R_b = R_b + g
 call update ( ← Q_b, → R_b )
end do
call residual_nrm_b ( Q, ← Q_b, R, Nrm, → Nrm_b )
call metrics_b ( →X, ← X_b, →Nrm, → Nrm_b )
```

Adjoint performance acceleration :: Assembled time stepping

Advantages of the altered adjoint time-stepping loop :

- Free choise of a "hot start" for $\overline{\texttt{Q}}$, suitable for one-shot.
- Same acceleration mechanisms of the primal ($\mathbf{A}$ and $\mathbf{A}^T$ share the same eigenvalues).
- Runtime decrease (calculation of $\overline{\texttt{Nrm}}$ only once).

	Primal	Altered Adjoint	"Brute force" adjoint
Time-step	1.0	2.6337738	4.2138729



Adjoint optimisation

One-shot

Converge the KKT system simultaneously :

$$\begin{cases} \frac{\partial Q}{\partial t} = -\mathbf{R}(Q, \alpha) \\ \frac{\partial v}{\partial t} = g - \mathbf{A}^T v \\ \frac{\partial \alpha}{\partial t} = -\frac{\partial J}{\partial \alpha} - v^T f \end{cases}$$

Dynamic convergence criterion :

$$gRMS = \frac{|\nabla J|}{C}$$

Advantage : Design convergence in less evaluations.

Design: Formula front wing

Cost function : **Downforce <u>constrained</u> with drag**

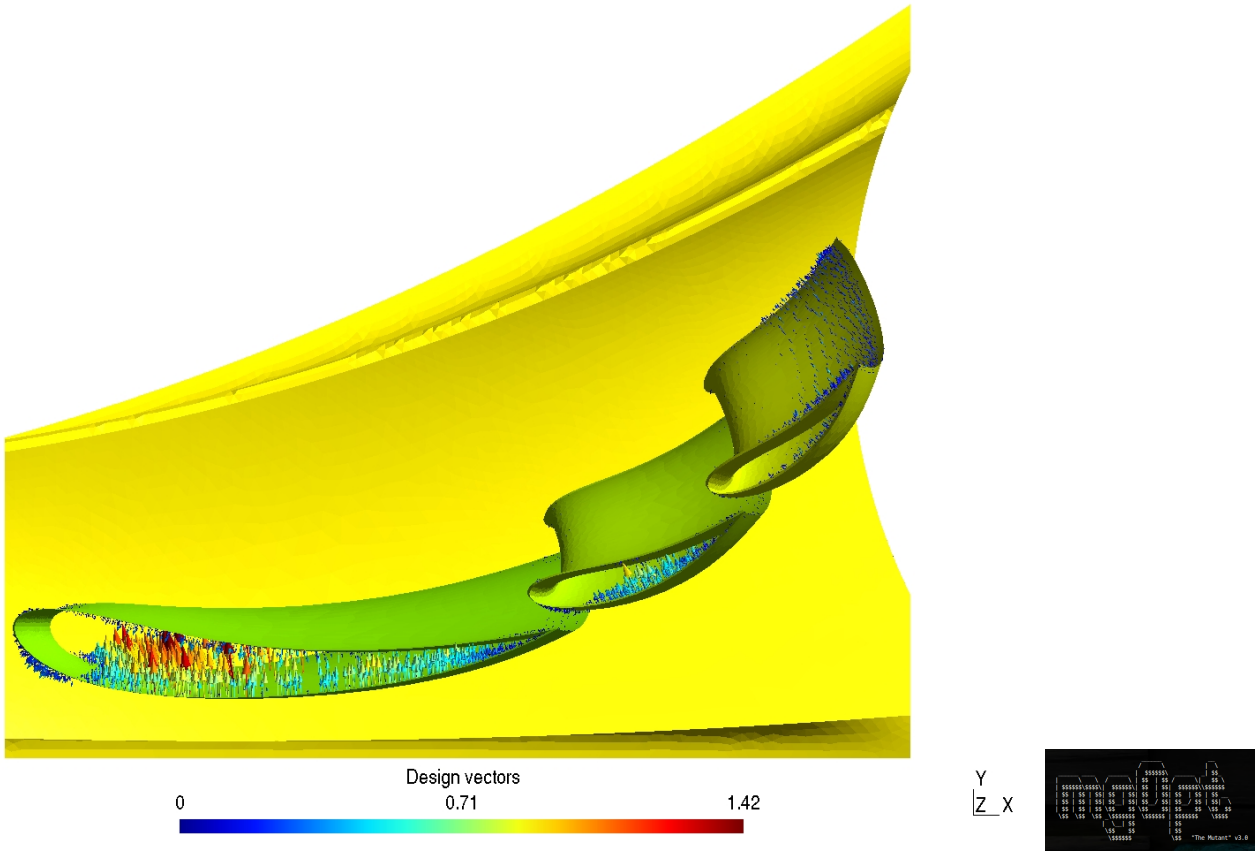
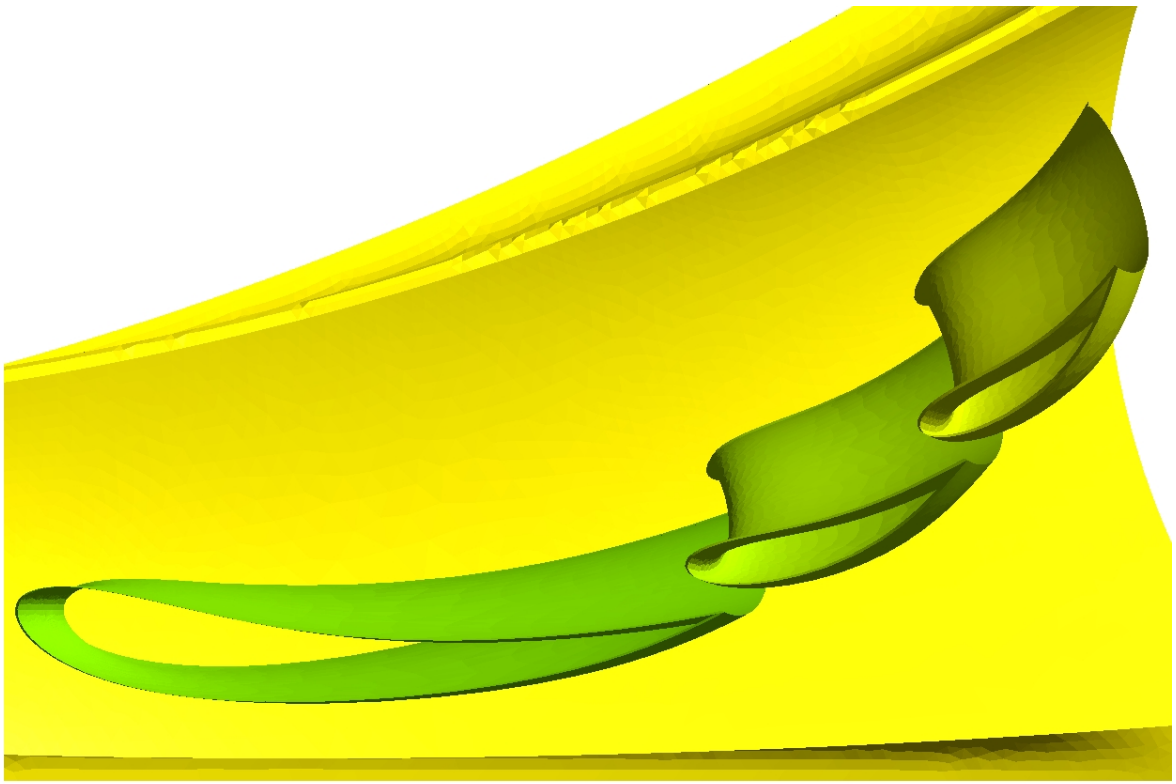


Figure: Design vectors on Formula front wing.

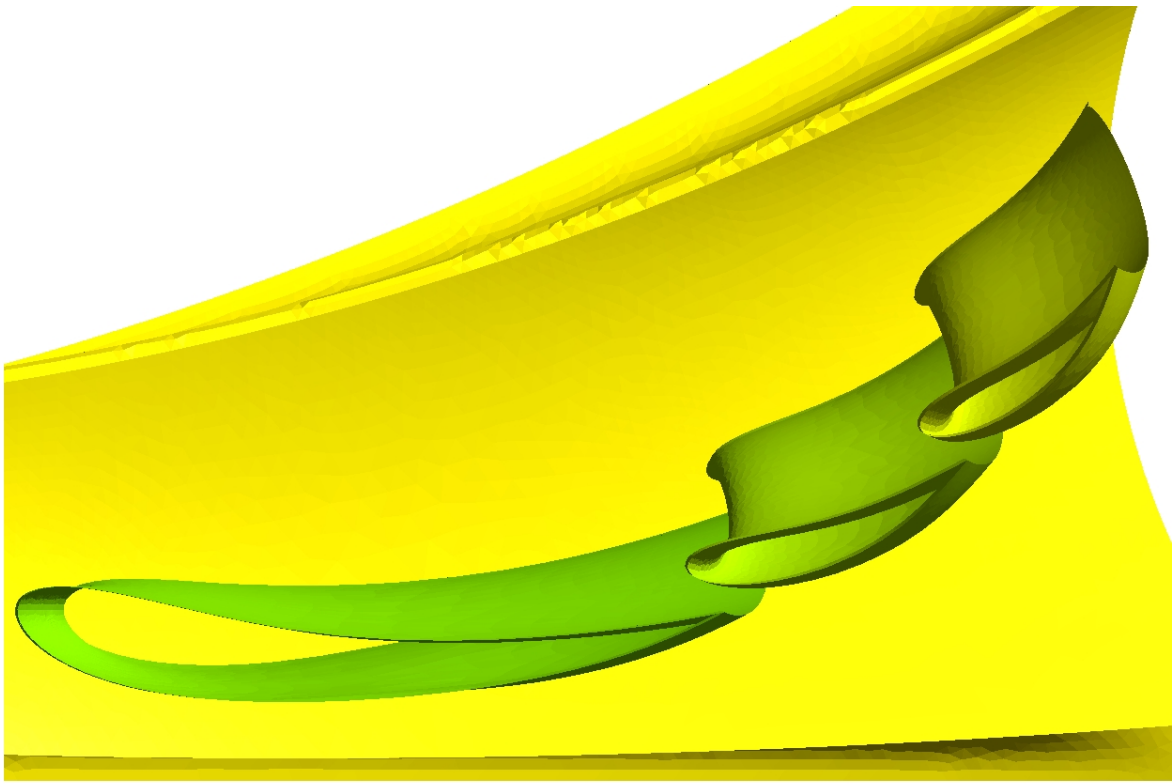
Design: Formula Front Wing

Step 1



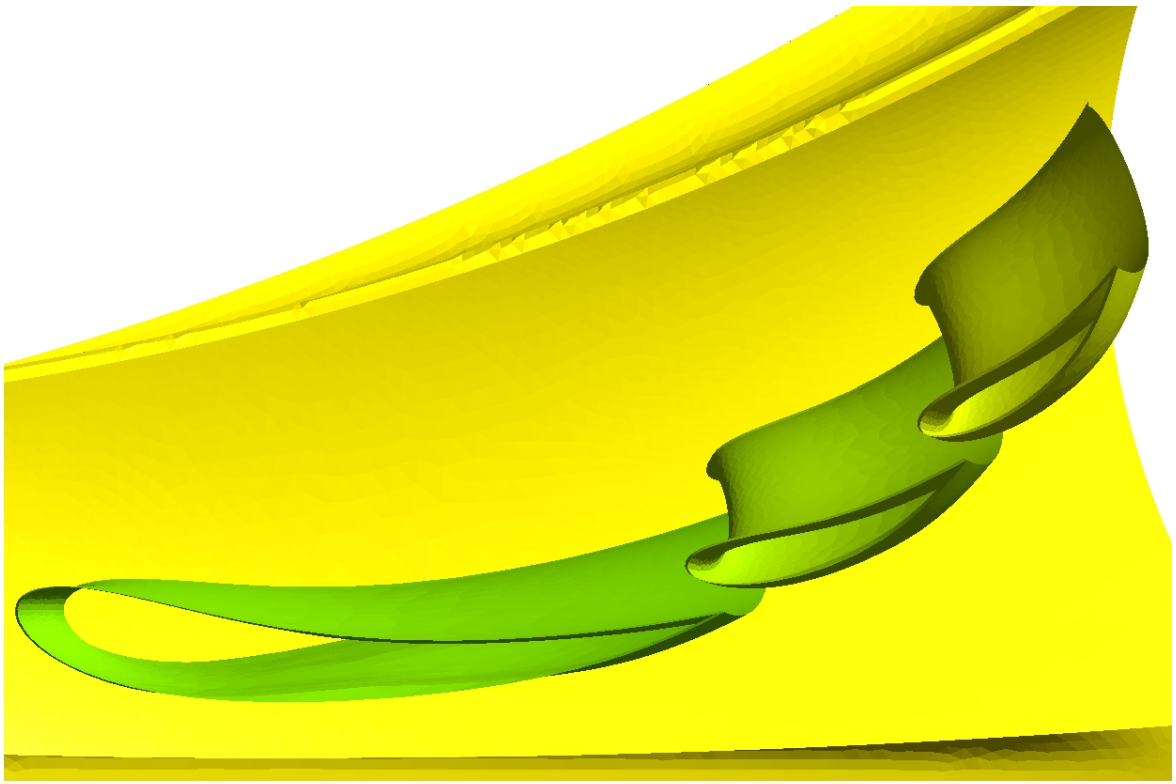
Design: Formula Front Wing

Step 2



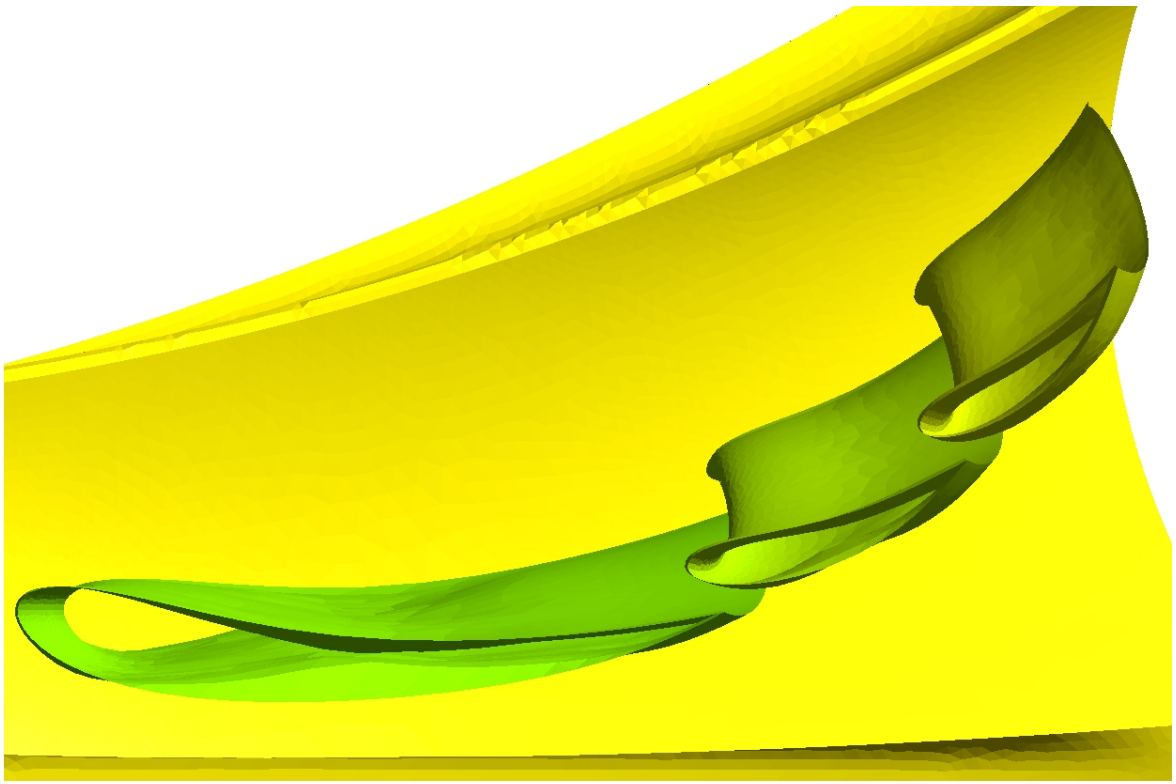
Design: Formula Front Wing

Step 3



Design: Formula Front Wing

Step 4



Design: S-Bend duct

Cost function : **Pressure drop**

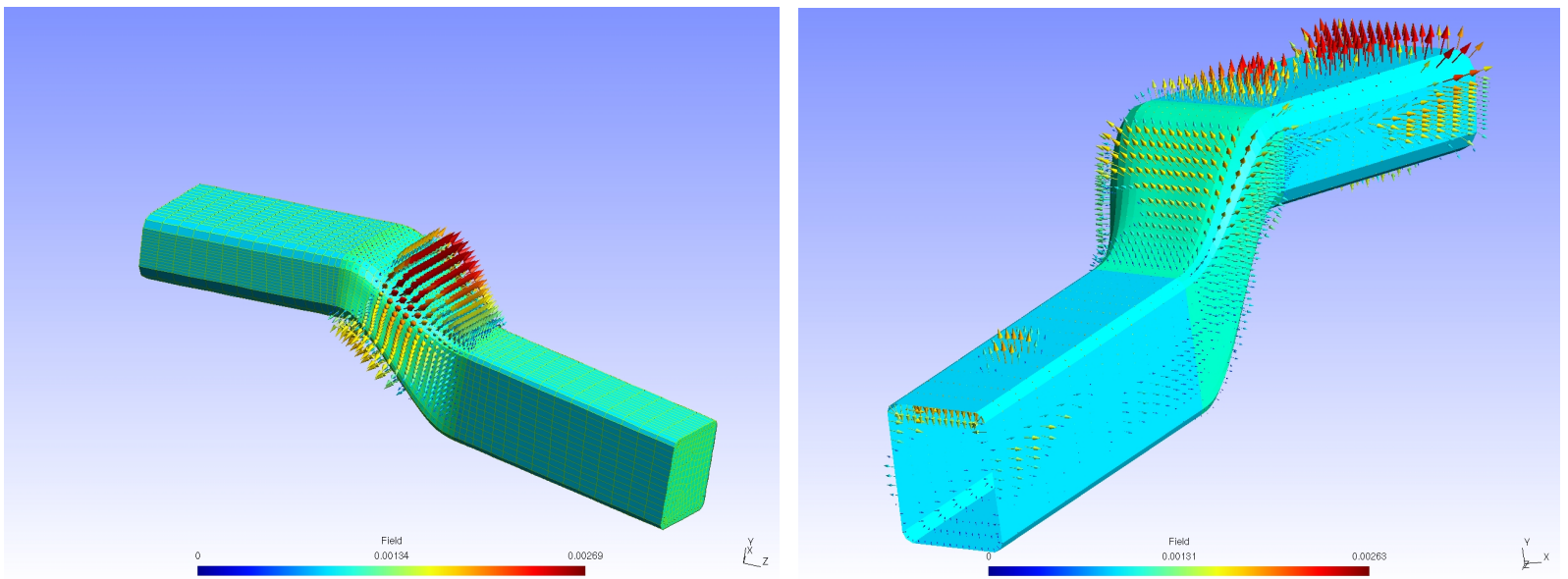


Figure: Design vectors on an S-Bend duct.

Entire S-Bend optimisation

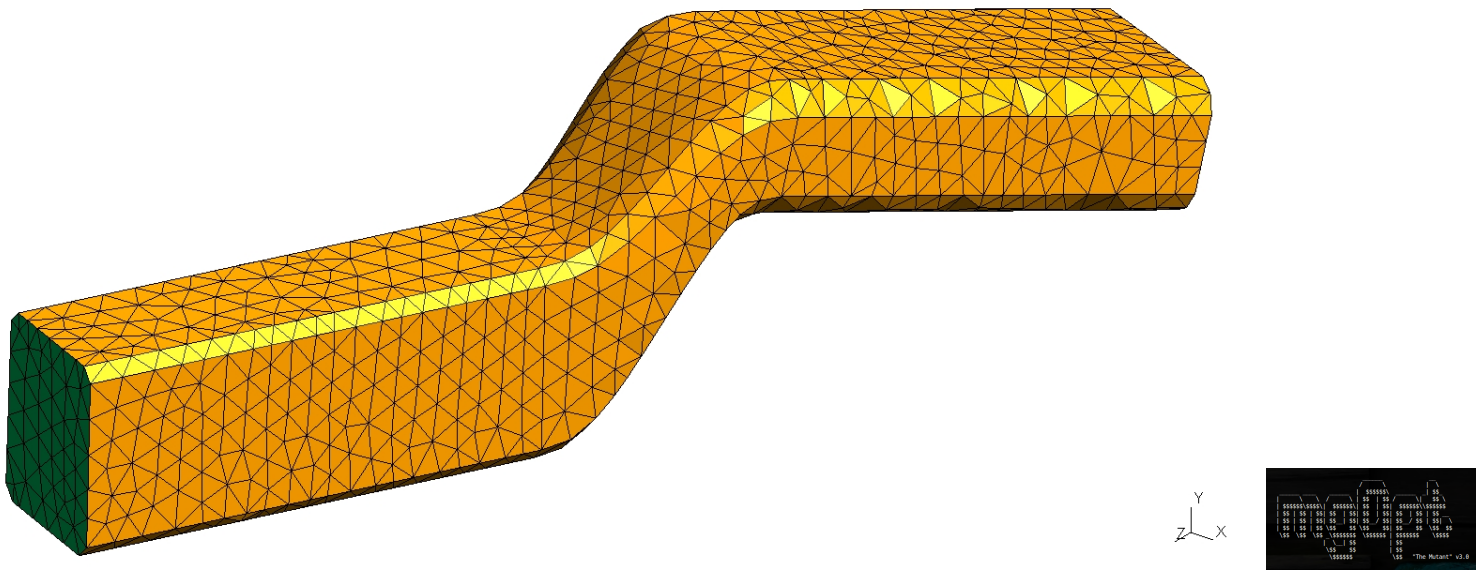


Figure: Optimisation of the entire S-Bend, Step 1

Entire S-Bend optimisation

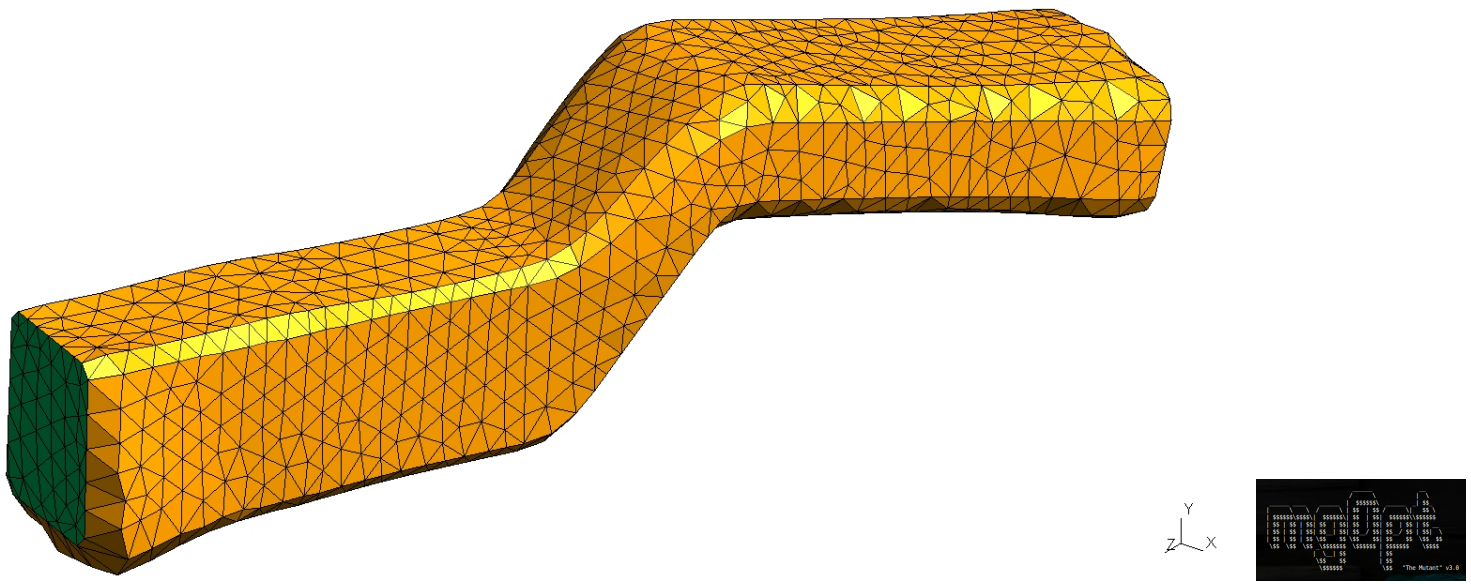


Figure: Optimisation of the entire S-Bend, Step 2

Entire S-Bend optimisation

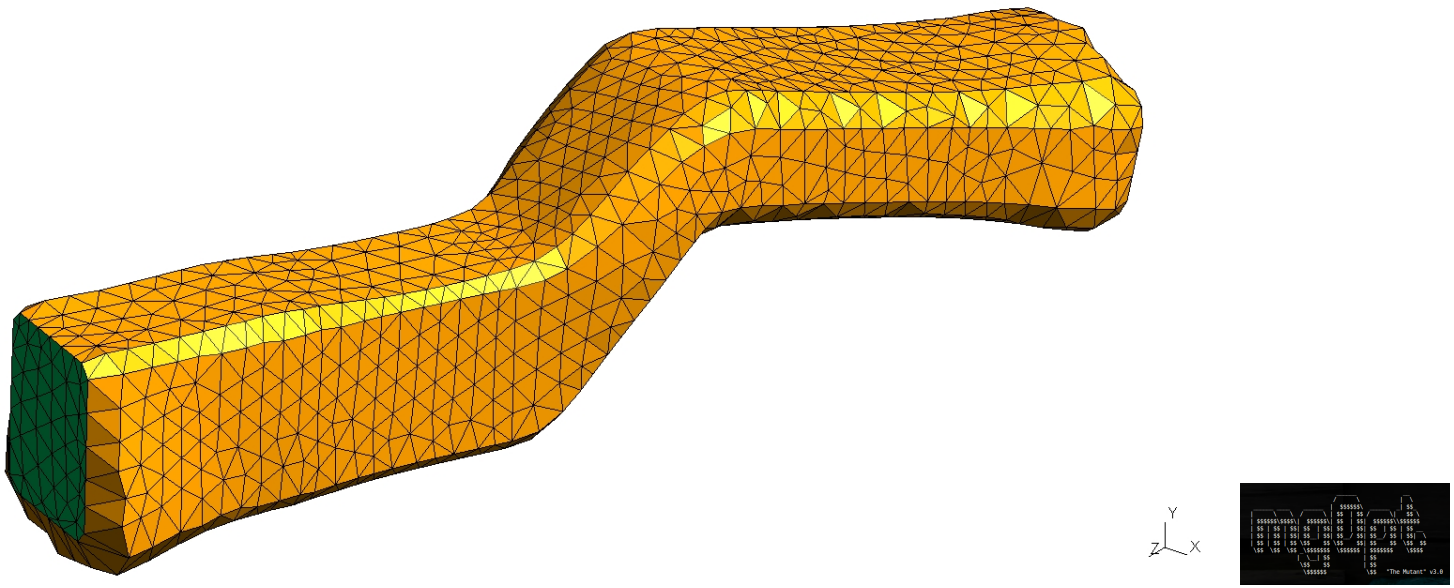


Figure: Optimisation of the entire S-Bend, Step 3

Entire S-Bend optimisation

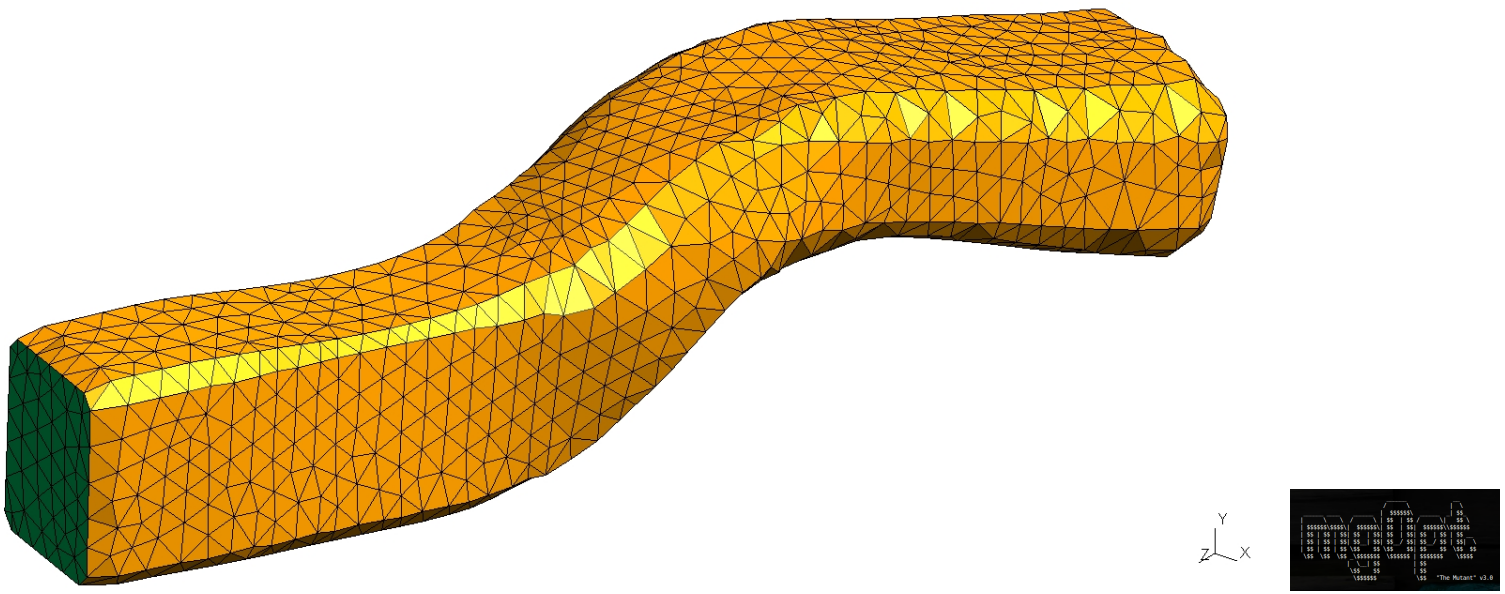


Figure: Optimisation of the entire S-Bend, Step 4

Entire S-Bend optimisation

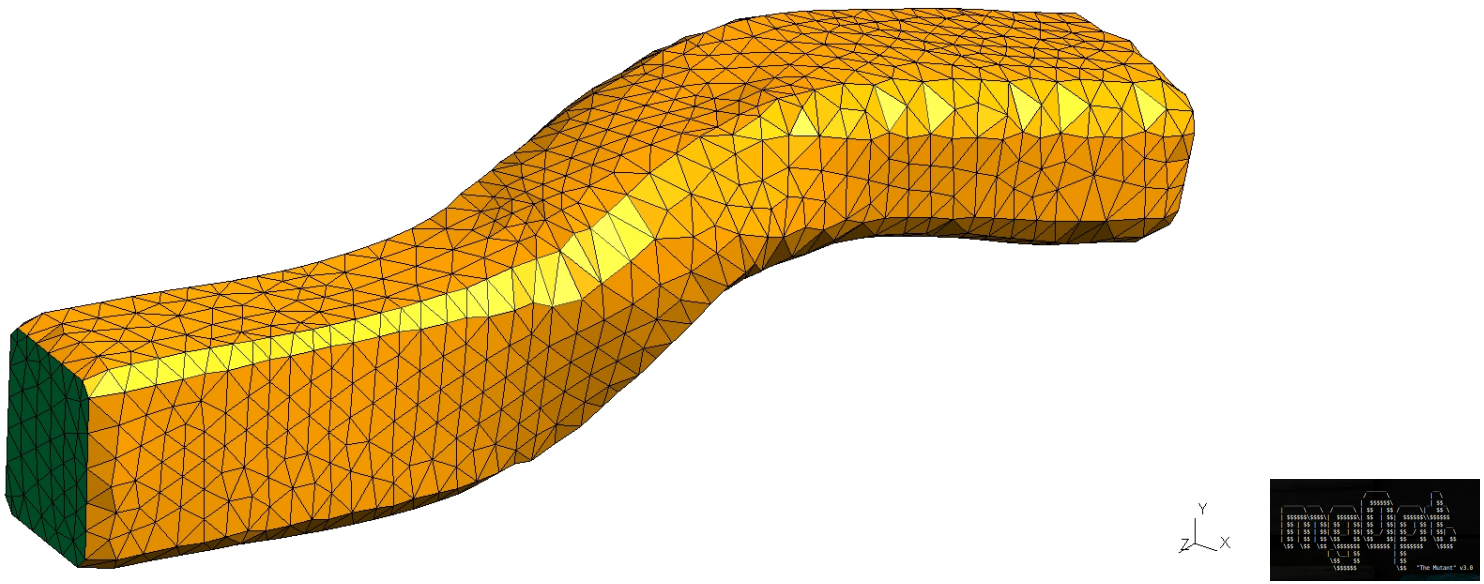


Figure: Optimisation of the entire S-Bend, Step 5

Entire S-Bend optimisation

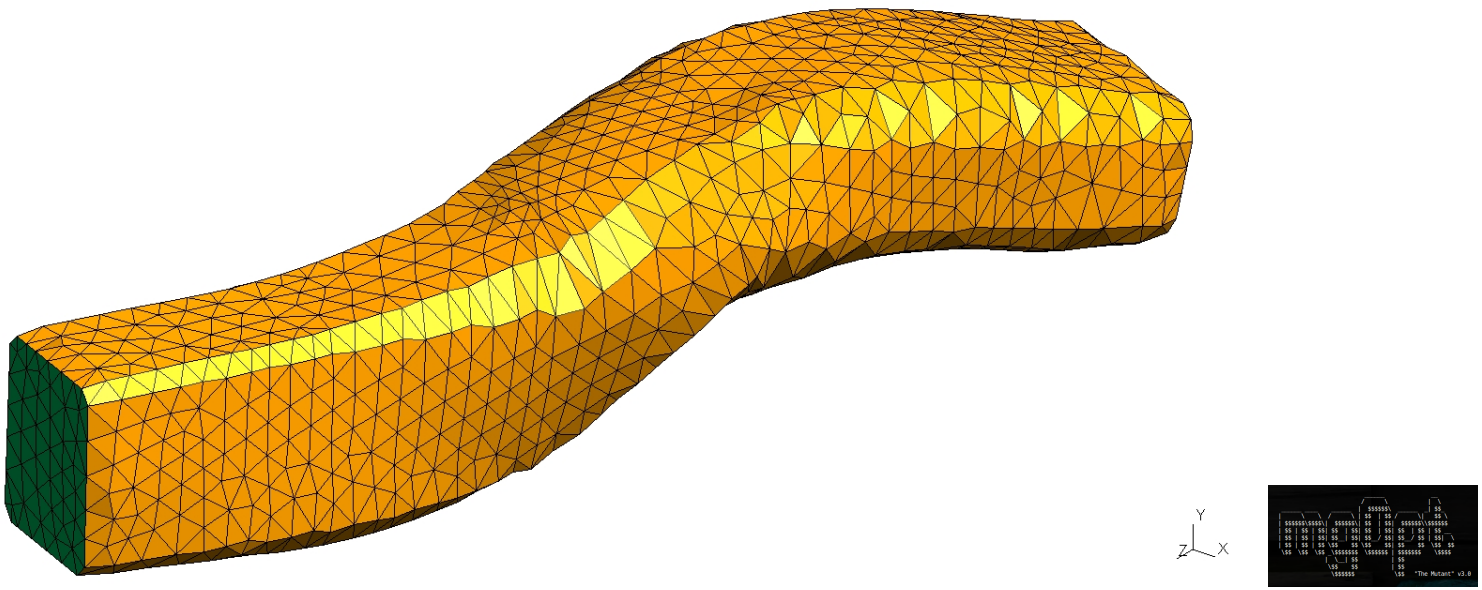


Figure: Optimisation of the entire S-Bend, Step 6

Entire S-Bend optimisation

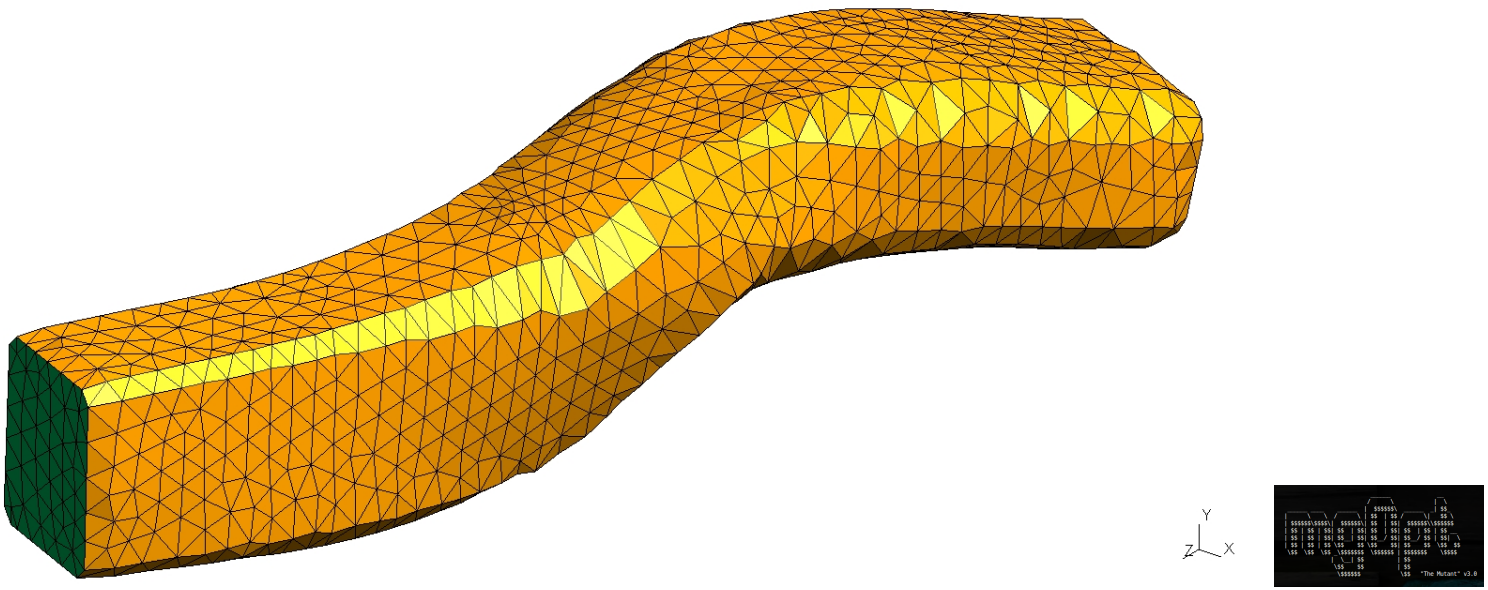


Figure: Optimisation of the entire S-Bend, Step 7

Entire S-Bend optimisation

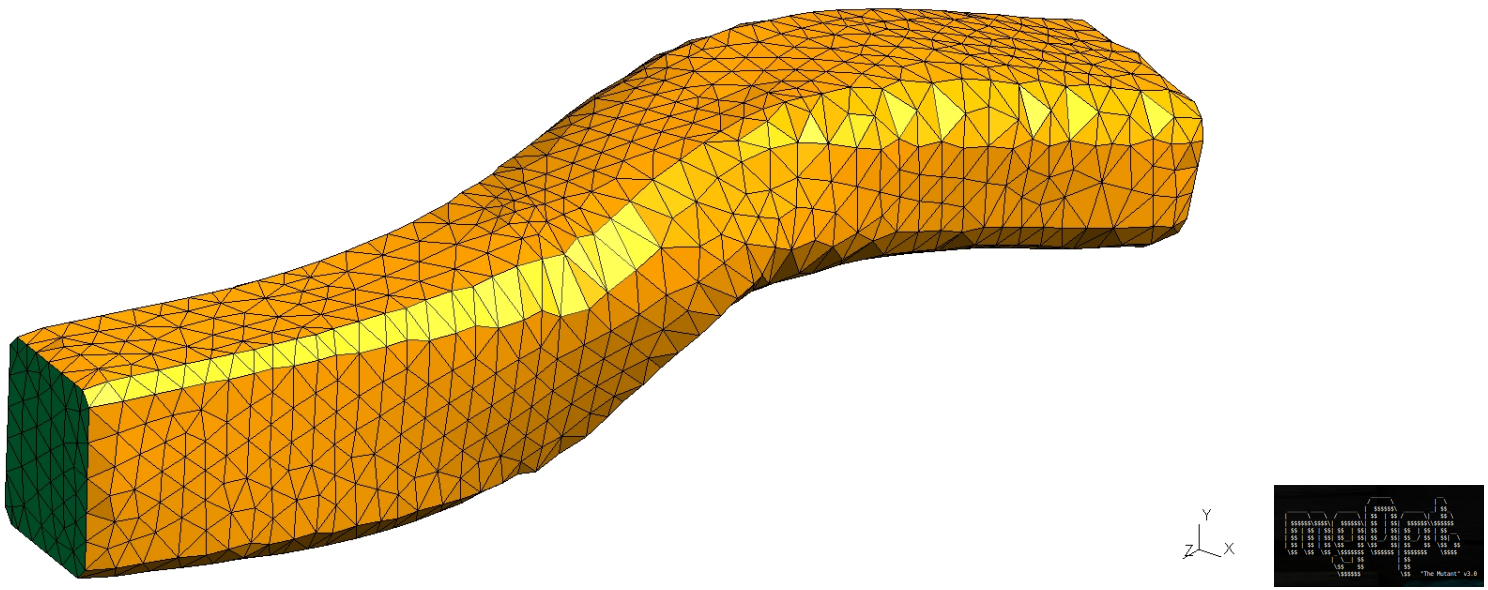


Figure: Optimisation of the entire S-Bend, Step 8

Entire S-Bend optimisation

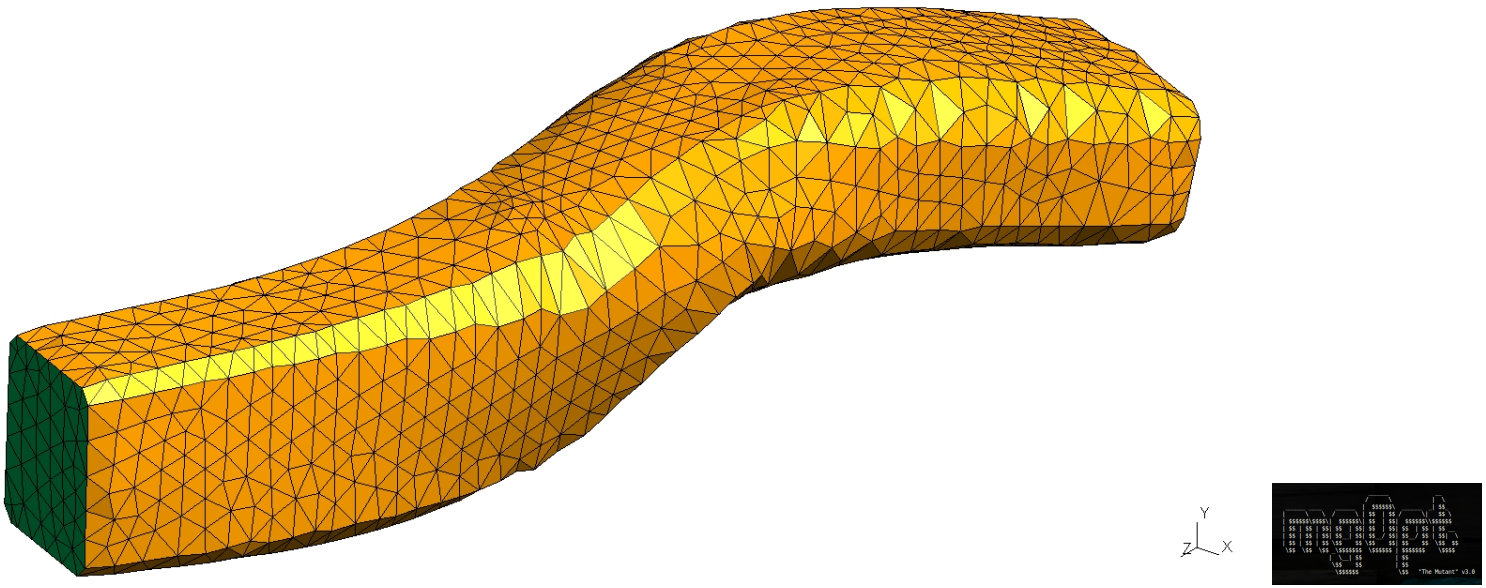


Figure: Optimisation of the entire S-Bend, Step 9

Entire S-Bend optimisation

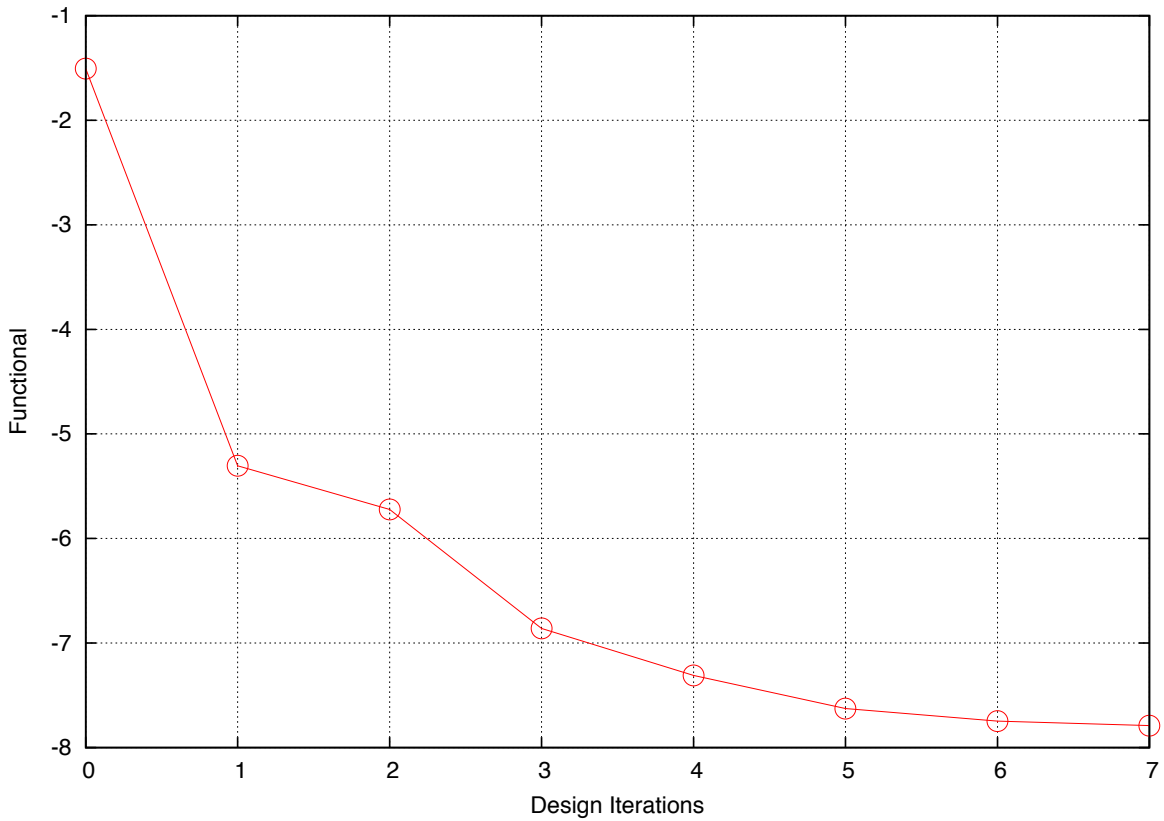


Figure: Functional convergence

Entire S-Bend optimisation

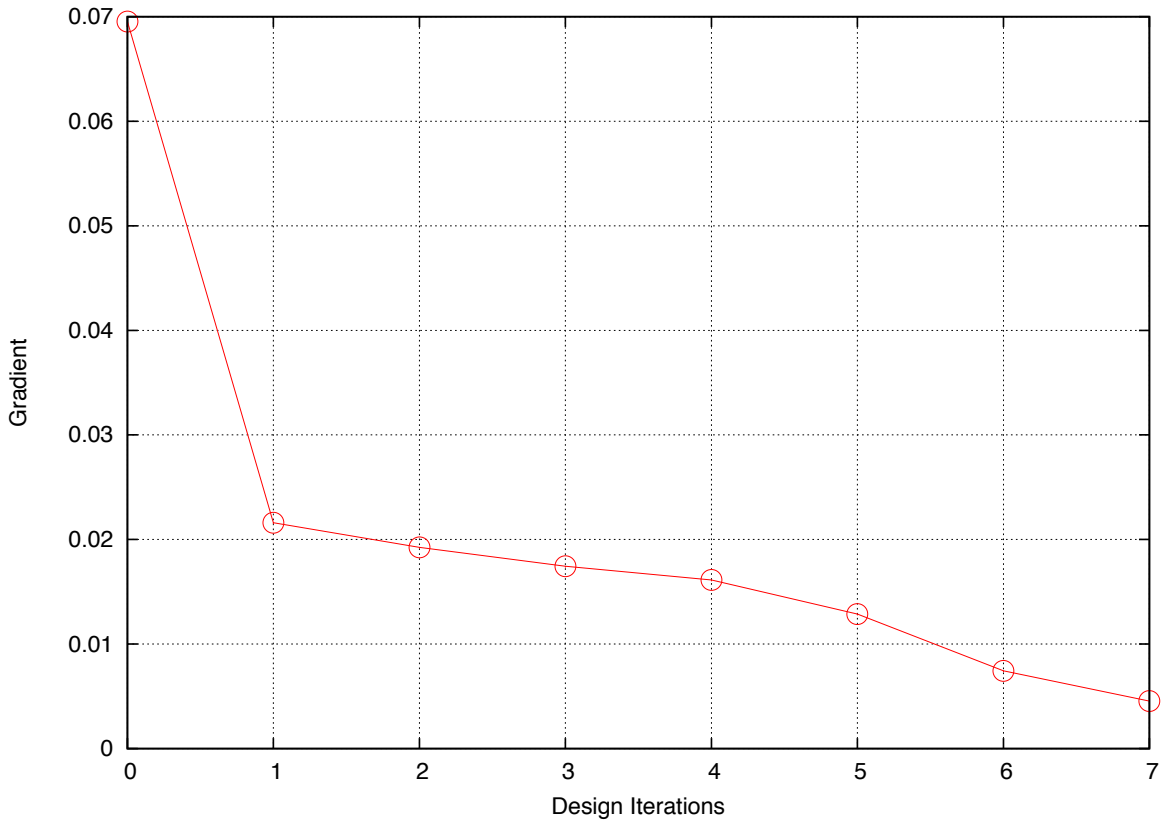


Figure: Gradient convergence

Adjoint Hessian via AD

Methods to derive 2^{nd} derivatives :

- Tangent over tangent (**ToT**). Cost $\sim N_{DV}^2$.
- Tangent over reverse (**ToR**). Cost $\sim N_{DV}$.
- Reverse over tangent (**RoT**). Cost $\sim N_{DV}$
- Reverse over reverse (**RoR**). Cost $\sim N_{DV}$ but high complexity.

[1] Martinelli, M. and Hascoët, L., *Tangent -on-Tangent vs. Tangent-on-Reverse for second differentiation of constrained functionals*

[2] L. Sherman et al, *First- and Second-Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods*

[3] M. Martinelli and R. Duvigneau, *On the use of second-order derivatives and metamodel-based Monte- Carlo for uncertainty estimation in aerodynamics*

Adjoint Hessian via AD

Methods to derive 2^{nd} derivatives :

- Tangent over tangent (**ToT**). Cost $\sim N_{DV}^2$.
- Tangent over reverse (**ToR**). Cost $\sim N_{DV}$.
- Reverse over tangent (**RoT**). Cost $\sim N_{DV}$
- Reverse over reverse (**RoR**). Cost $\sim N_{DV}$ but high complexity.

Preference[1][2][3] :

- ToT, for $N_{DV} \leq 40$
- ToR, for $N_{DV} > 40$

[1] Martinelli, M. and Hascoët, L., *Tangent -on-Tangent vs. Tangent-on-Reverse for second differentiation of constrained functionals*

[2] L. Sherman et al, *First- and Second-Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods*

[3] M. Martinelli and R. Duvigneau, *On the use of second-order derivatives and metamodel-based Monte- Carlo for uncertainty estimation in aerodynamics*

Tangent over tangent (ToT)

The second derivative via ToT :

$$\frac{dJ}{d\alpha_i} = \frac{\partial J}{\partial \alpha_i} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial \alpha_i} \Rightarrow \boxed{\frac{d^2 J}{d\alpha_i d\alpha_k} = D^2_{i,k} J + \frac{\partial J}{\partial U}\frac{d^2 U}{da_i da_k}}$$

where :

$$D^2_{i,k} F = \frac{\partial^2 F}{\alpha_i \alpha_k} + \frac{\partial^2 F}{\partial U \partial \alpha_i}\frac{dU}{d\alpha_k} + \frac{\partial^2 F}{\partial U \partial \alpha_k}\frac{dU}{d\alpha_i} + \frac{\partial}{\partial U}\left(\frac{\partial F}{\partial U}\frac{dU}{d\alpha_i}\right)\frac{dU}{d\alpha_k}$$

The term $d^2U/d\alpha_i d\alpha_2$ is given by :

$$\frac{\partial R}{\partial U}\frac{dU}{d\alpha} = -\frac{\partial R}{\partial \alpha_i} \Rightarrow \boxed{\frac{\partial R}{\partial U}\frac{d^2 U}{d\alpha_i d\alpha_k} = -D^2_{i,k} R}$$

which is in the same form as $\mathbf{A}u = f$.

Tangent over tangent (ToT)

Algorithm for ToT

1. For each $\alpha_i, i\epsilon[1,n]$, compute and store the solution of the equation : $\frac{\partial R}{\partial U}\frac{dU}{d\alpha_i} = -\frac{\partial R}{\partial \alpha_i}$
2. For each $\alpha_i, i \in [1,n]$:
   For each $\alpha_k, k \in [1,i]$:
   1. Compute the solution to the equation : $\frac{\partial R}{\partial U}\frac{d^2U}{d\alpha_i d\alpha_k} = -D^2_{i,k}R$
   2. Compute $\frac{d^2J}{d\alpha_i d\alpha_k} = D^2_{i,k} + \frac{\partial J}{\partial U}\frac{d^2U}{d\alpha_i d\alpha_k}$

Cost of ToT : $(N_{DV} + (N^2_{DV} + N_{DV})/2) \cdot N_{it}$

Alternative tangent-over-tangent

The equation used to compute $d^2U/d\alpha_i d\alpha_k$ can be written as :

$$\frac{\partial R}{\partial U}\frac{d^2U}{d\alpha_i d\alpha_k} = -D^2_{i,k}R \Rightarrow \frac{d^2U}{d\alpha_i d\alpha_k} = -\frac{\partial R}{\partial U}^{-1} D^2_{i,k}R$$

Substituting, the second derivative becomes :

$$\frac{d^2J}{d\alpha_i d\alpha_k} = D^2_{i,k} + \frac{\partial J}{\partial U}\frac{d^2U}{da_i da_k} \Rightarrow \boxed{\frac{d^2J}{d\alpha_i d\alpha_k} = D^2_{i,k}J - v^T D^2_{i,k}R}$$

where v the adjoint solution.

Tangent over reverse (ToR)

The second derivative via ToR :by :

$$\frac{dJ}{d\alpha_i} = \frac{\partial J}{\partial \alpha_i}^T + v^T f \Rightarrow \boxed{\frac{d^2 J}{d\alpha_i d\alpha_k} = \dot{\bar{J}}_\alpha - \dot{\bar{R}}_\alpha - \frac{\partial R}{\partial \alpha_i}^T \frac{d}{d\alpha_k}\left(\frac{dU}{d\alpha_i}^T\right)}$$

where the terms $\dot{\bar{J}}_\alpha$ and $\dot{\bar{R}}_\alpha$ are :

$$\dot{\bar{J}}_\alpha = \frac{\partial}{\partial \alpha_k}\left(\frac{\partial J}{\partial \alpha_i}^T\right) + \frac{\partial}{\partial U}\left(\frac{\partial J}{\partial \alpha_i}^T\right)\frac{dU}{d\alpha_k}$$

$$\dot{\bar{R}}_\alpha = \frac{\partial}{\partial \alpha_k}\left(\frac{\partial R}{\partial \alpha_i}^T v\right) + \frac{\partial}{\partial U}\left(\frac{\partial R}{\partial \alpha_i}^T v\right)\frac{dU}{d\alpha_k}$$

The term $d[(dU/d\alpha_i)^T]/d\alpha_k$ is given by :

$$\frac{\partial R}{\partial U}^T v = \frac{\partial J}{\partial U}^T \Rightarrow \boxed{\frac{\partial R}{\partial U}^T \frac{dv}{d\alpha_k} = \dot{\bar{J}}_U - \dot{\bar{R}}_U}$$

which is in the same form as $\mathbf{A}^T v = g$.

$$\dot{\bar{J}}_U = \frac{\partial}{\partial \alpha_k}\left(\frac{\partial J}{\partial U}^T\right) + \frac{\partial}{\partial U}\left(\frac{\partial J}{\partial U}^T\right)\frac{dU}{d\alpha_k}$$

$$\dot{\bar{R}}_U = \frac{\partial}{\partial \alpha_k}\left(\frac{\partial R}{\partial U}^T v\right) + \frac{\partial}{\partial U}\left(\frac{\partial R}{\partial U}^T v\right)\frac{dU}{d\alpha_k}$$

Algorithm for ToR

1. Compute the adjoint solution : $\mathbf{A}^T v = g$
2. For each $\alpha_i, i \in [1, n]$:
   1. Compute the tangent solution from : $\frac{\partial R}{\partial U}\frac{dU}{d\alpha} = -\frac{\partial R}{\partial \alpha_i}$
   2. Compute the terms $\dot{\dot{J}}_\alpha$, $\dot{\dot{R}}_\alpha$, $\dot{\dot{J}}_U$ and $\dot{\dot{R}}_U$.
   3. Solve $\frac{\partial R}{\partial U}^T \frac{dv}{d\alpha_k} = \dot{\dot{J}}_U - \dot{\dot{R}}_U$.
   4. Compute $\frac{d^2 J}{d\alpha_i d\alpha_k} = \dot{\dot{J}}_\alpha - \dot{\dot{R}}_\alpha - \frac{\partial R}{\partial \alpha_i}^T \frac{d}{d\alpha_k}\left(\frac{dU}{d\alpha_i}^T\right)$

Cost of ToR : $N_{it} + 2 \cdot N_{it} \cdot N_{DV}$

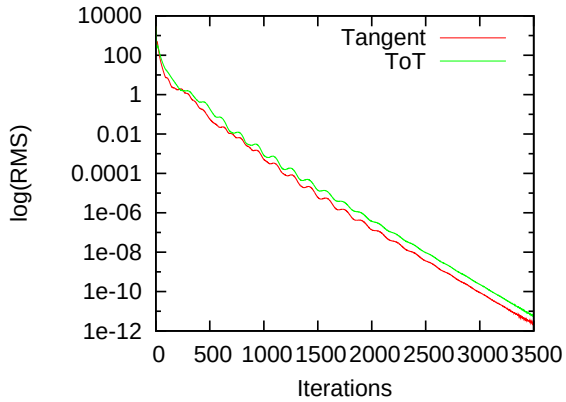
Sensitivity Validation

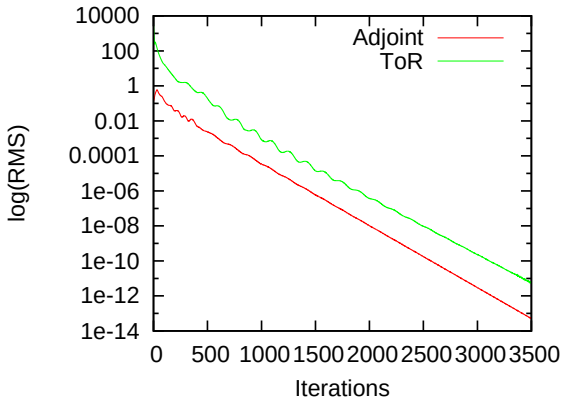
AUSM+ UP	Method	1^{st} Order Accuracy	2^{nd} Order Accuracy
First Derivative	FD	**0.453321**42571	**0.619166**853538
	Tangent	**0.453321292403**	**0.619166824022**
	Adjoint	**0.453321292403**	**0.619166824022**
Second Derivative	FD	**138.3578**951408	**142.2**067719136
	ToT	**138.3578906648**	**142.2184126185**
	ToR	**138.3578906648**	**142.2184126185**

Table: First and second order gradient validation.

Convergence acceleration

Use of the original multi grid

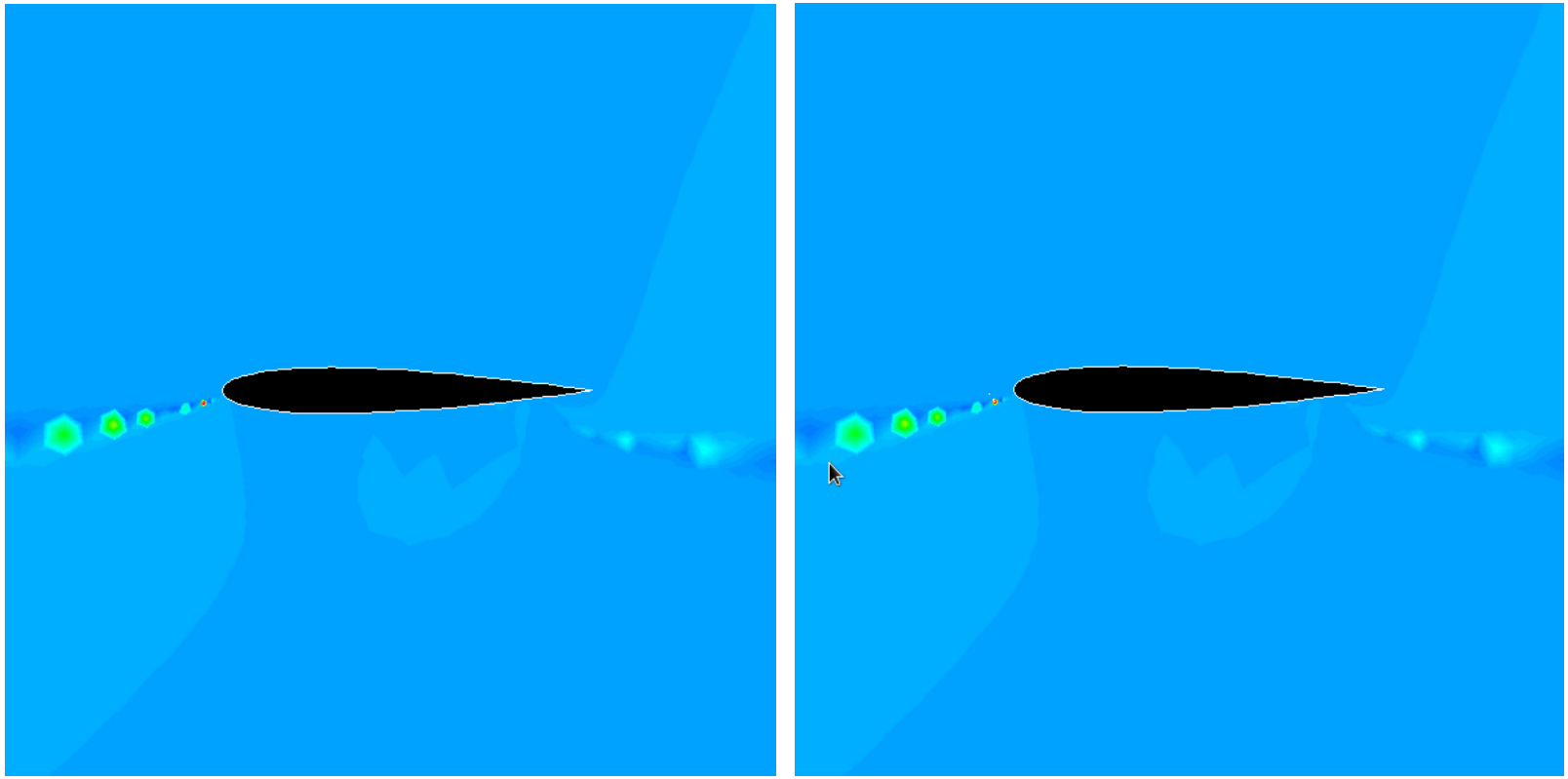




Use the altered time stepping

Methodology	**Runtime [sec]**
Brute force AD	2944.4360
Proposed methodology	2387.4854
Improvement	**19%**

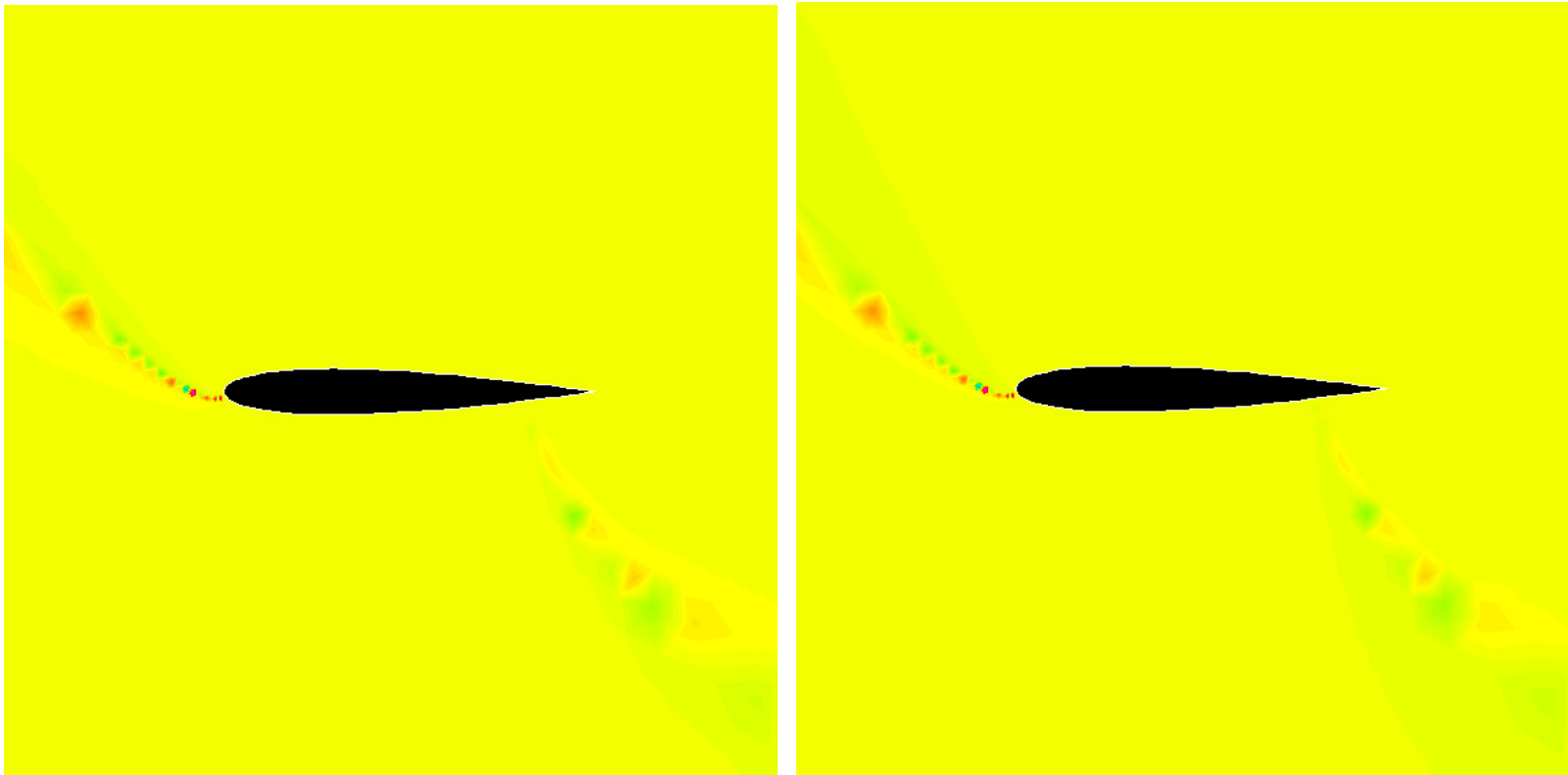
First order sensitivity fields



(c) FD, first derivative of y-velocity (d) Tangent, first derivative of y-velocity

Figure: First order sensitivity fields of y-velocity

Second order sensitivity fields



(a) FD, second derivative of y-velocity (b) ToT, second derivative of y-velocity

Figure: Second order sensitivity fields of y-velocity

Summary & Novelties

Novel developments in FlowHead:

- Completely automated sensitivity code generation [4]
- Highest level possible Fortran 90/95 sensitivity code via AD
- Single *Makefile* scripting (preprocessing, AD, compilation)
- Fast timestepping for Adjoint & Hessian[5]
- Hot-start enabled adjoint and Hessian via adjoint
- Use of non-differentiated multigrid for adjoint and Hessian
- CAD sensitivities in adjoint based optimisation [6]

---

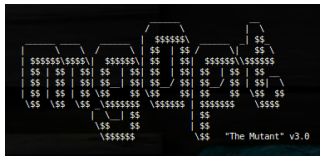
[4] Jones D., Christakopoulos F, Müller J.-D., *Preparation and assembly of discrete adjoint CFD codes*,Computers & Fluids, Vol. 46, Is. 1, 7/2011,P. 282-286

[5] Christakopoulos F. et al, *Pseudo-timestepping and validation for Automatic Differentiation derived CFD codes*, Computers & Fluids, Vol. 46, Is. 1, 7/2011,P. 174-179

[6] Yu G., Müller J.-D., Jones F., Christakopoulos F. *CAD-based shape optimisation using adjoint sensitivities*,Computers & Fluids, Vol. 46, Is. 1, 7/2011,P. 512-516

Thank you for your attention.

Queen Mary University of London



FLOWHEAD

Fluid Optimisation Workflows for Highly Effective Automotive Development Processes

European Commission, THEME SST.2007-RTD-1:
Competitive product development from February 2009 to January 2012
http://flowhead.sems.qmul.ac.uk/