RWTH AACHEN UNIVERSITY

Improvement of the aerodynamic shape optimization by adjoint methods in an MDO process

Verbesserung der aerodynamischen Formoptimierung durch adjungierte Methoden in einem MDO Prozess

Bachelorarbeit von **Thomas Camminady** aus Menden im Sauerland

in Computational Engineering Science, vorgelegt der Fakultät für Maschinenwesen der RWTH Aachen

Angefertig bei
Prof. Dr. Nicolas R. Gauger
LuFG Computational Mathematics, RWTH
in Zusammenarbeit mit der
Firma EADS Cassidian

Aachen, 2013

Acknowledgements

My sincere appreciation goes to Dr. Luca Nardin and Dr. Kurt Sermeus for their great support as my supervisors. Without their guidance, help and suggestions it would not have been possible to create this thesis. No less helpful has been Prof. Dr. Nicolas R. Gauger, who also made it possible for me to work in such an inspiring environment. Furthermore, I would like to thank everybody I have been working with at Cassidian, especially Dr. Herbert Rieger who allowed me to be a part of his team.

Besonders möchte ich mich an dieser Stelle auch für die Unterstützung meiner Familie bedanken, ohne deren Hilfe ich nicht bis hierhin gekommen wäre. Dies gilt vor allem für die Hilfe meiner Eltern.

Abstract

In this thesis, we are going to explain the process of aerodynamic and aero-structural optimization in aircraft design. We will explain the theoretical concept of the flow adjoint approach and the mesh adjoint approach for the evaluation of the gradients of the objective function with respect to the input parameters.

We will then implement this approach into an existing aerodynamic optimization process. Furthermore, the existing model will be extended towards an aero-structural model by introducing a CSM simulation additionally to the CFD calculation. The aerodynamic optimization and the structural optimization will be loosely coupled, which means that we take the aerodynamic optimized geometry as a starting point to perform the structural optimization afterwards.

The optimization process will be applied to a test case and the results will be analyzed.

This work has arisen from the AeroStruct program at EADS Cassidian. During my three month internship at EADS Cassidian additionally to the creation of this thesis, I have been introduced into the existing software environment that is related to AeroStruct.

All given wing geometries in this thesis are generic geometries, which are derived from standard aircrafts of the given type. Especially, the included geometries are not identical with the ones of the EADS Talarion or the Eurohawk.

Contents

1 Optimization in aerodynamics **1**

2 Computational fluid dynamics **3**

2.1 Euler equations . . . 3

2.2 Navier-Stokes equations . . . 4

3 Mathematical formulation of the optimization process **6**

3.1 Adjoint formulation of the optimization problem . . . 7

3.2 Extension towards the mesh adjoint . . . 8

3.3 The SQP algorithm . . . 10

4 Implementation and process chain **12**

4.1 Software and modelling . . . 12

4.1.1 CATIA v5 . . . 13

4.1.2 Mesher . . . 13

4.1.3 DLR TAU-code . . . 16

4.1.4 Weight estimation . . . 16

4.1.5 modeFrontier . . . 17

4.2 Process chain of the aerodynamic optimization without mesh adjoint . . . 17

4.2.1 Application-specific elements of the implementation . . . 21

4.2.1.1 Perturbing the geometry . . . 21

4.2.1.2 Calculation of the aerodynamic sensitivities . . . 22

4.3 Process chain of the aerodynamic optimization with mesh adjoint . . . 24

4.3.1 Application-specific elements of the implementation . . . 24

4.4 Process chain of the aeroelastic optimization . . . 24

5 Gradient comparison and numeric parameter influence **28**

5.1 Comparison of the gradients . . . 28

5.2 TAU parameter influence . . . 30

5.2.1 Number of inner iterations . . . 33

5.2.2 Number of preconditioning iterations 33

6 Test cases and results **36**

6.1 Wing planform optimization problem 36

6.2 Optimization without mesh adjoint 36

6.2.1 Results 37

6.3 Optimization with mesh adjoint 38

6.4 Aeroelastic optimization 41

7 Summary and outlook **46**

Bibliography **46**

List of Figures

1.1 Eurohawk at Manching 2

4.1 CAD model 15
4.2 Surface mesh 15
4.3 Boundary layer mesh 15
4.4 Simplex optimization with modeFrontier 18
4.5 Process chain for optimization without mesh adjoint 19
4.6 Extract of the process chain for optimization in modeFrontier 20
4.7 Process chain for optimization with mesh adjoint 25
4.8 Extract of the process chain for optimization in modeFrontier with mesh adjoint approach 26

5.1 Surface mesh of the wing configuration 29
5.2 Adjoint Solution, $Ma = 0.75$ and $\alpha = 2.1°$ 30
5.3 Accuracy of the gradients over accuracy of the flow adjoint solution, $Ma = 0.75$ and $\alpha = 2.1°$ 31
5.4 Gradients for kink point x-coordinate 31
5.5 Gradients for kink point y-coordinate 32
5.6 Quotient of gradients 32
5.7 Different values for number of inner iterations 34
5.8 Different values for number of preconditioning iterations 35

6.1 Converged geometry with pressure distribution 38
6.2 Pressure distribution for different iteration 39
6.3 Structural quantities over design iteration 40
6.4 Aerodynamic coefficients over design iteration 40
6.5 Aerodynamic coefficients over design iteration 40
6.6 Comparison between initial and optimized geometry 41
6.7 Structure of the geometry 43

6.9 von Mises stresses for the converged structure 43
6.8 Mesh for the CSM simulation 43
6.10 Mass over iterations 44
6.11 Stresses over iterations 44
6.12 Thickness over iterations 45

Chapter 1

Optimization in aerodynamics

The development process in the aerospace area represents a complex interaction of several scientific disciplines. Next to flight control, structures and loads, navigation and several other fields, aerodynamics is probably one of the most difficult. With the increasing computational power and the progress in modern applied mathematics, computational fluid dynamics (CFD) has become an important pillar of the aerodynamic design process next to wind tunnel tests and practical experience. In the past decades the importance of simulation has increased further, as it reduces the costs in the development process drastically. Numerical optimization uses the results obtained by simulations and suggests design improvements that are often difficult to deduce from practical experience and wind tunnel tests alone[15]. In general, one can formulate the optimization problem in aerodynamics such that a certain performance parameter like lift or range should be maximized, while satisfying constraints arising from structural requirements or design constraints in general. Shape optimization of an aircraft wing is only one example for the application of shape optimization in aircraft design.

Different designs can be optimal regarding different requirements. Therefore it is necessary to clarify the definition of an objective in advance. Additionally to that, it is also needed to correctly define the parameters that describes the wing and along with that, give meaningful boundaries for each parameter. One example for the parametrization of a wing is to define the shape by values for span length, chord length, profile and thickness. This is a simple and intuitive way to create generic shapes, but all of a certain similar type. Another possibility lies in the approximation of a shape by a sample of points which are defined by their deflection in x-,y- or z-direction, relative to a certain baseline. The overall shape is then interpolated on base of these points and smoothed afterwards. For more complex geometries, the second approach is the only suitable. This method allows to create nearly arbitrary complex shapes which are required in practical application. It is obvious that with higher geometrical resolution and therefore a larger number of parameters, the computational effort also rises. The increase in computational complexity is due to the fact that with a large number of parameters, finding an optimum becomes numerically more complex. Finite difference approaches for gradient computation have a complexity which is proportional to the number of design variables.

Figure 1.1: Eurohawk at Manching

This makes the finite difference approach computational expensive for optimization with a non-trivial number of design variables. The adjoint approach however, represents an effective method for optimization with a large number of design variables, as its complexity only depends on the number of objective function, which is for many engineering problems much smaller than the number of design variables.

An important cost function is the range of an aircraft in cruise mode. The cruise mode is thereby defined as the section of a flight, in which the aircraft has reached the desired altitude and travel velocity. For transport aircraft and unmanned aircraft vehicles (UAV), such as the Eurohawk in figure 1.1, the majority of the mission is in cruise flight.

Aircraft design is generally speaking a multidisciplinary optimization problem (MDO). The optimization goal from an aerodynamic point of view is for instance to minimize drag while providing a required lift. If we also consider structural aspects in our design process, we are speaking of a multidisciplinary optimization. The optimization goal from a structural point of view, is generally to minimize the structural weight, while keeping the given stresses under a certain level.

As we can see, the design process becomes a complex interaction of the different disciplines. In this thesis, the focus will lie on the aerodynamic and structural aspects of an optimization process.

Chapter 2

Computational fluid dynamics

To describe the motion of a fluid around a given body, it is necessary to have an accurate model that reproduces the physical behaviour of a fluid. The derivation of the model depends heavily on the required application of the model. Models that are used to describe global weather phenomena can neglect small scale properties that become essential when modelling vortices in a boundary layer. In general those models are systems of partial differential equations which have certain conservative properties. The two most common models are the Euler equations, which describe inviscid flow and represent the conservation of mass, momentum and energy and the Navier-Stokes equations. The Navier-Stokes equations are a set of nonlinear partial differential equations that describe a viscous and heat conducting fluid. As the Euler equations can be seen as a simplification of the Navier-Stokes equations, given certain assumptions on the underlying physics, also the computational costs decrease. Due to this fact, the Euler equations are often chosen, if the accuracy, which results from the application of the Navier-Stokes equations, is not necessary.

2.1 Euler equations

The Euler equations form a simplification of the Navier-Stokes equations under the assumption of zero viscosity and heat conduction. In conservative form, the set of equations can be written in the following form, derived in [12]

$$\frac{\partial \omega}{\partial t} + \frac{\partial \mathbf{f}_x}{\partial x} + \frac{\partial \mathbf{f}_y}{\partial y} + \frac{\partial \mathbf{f}_z}{\partial z} = \mathbf{0}, \tag{2.1}$$

with the vector of conserved quantities

$$\omega = \begin{pmatrix} \rho, & \rho u, & \rho v, & \rho w, & \rho E \end{pmatrix}^T \tag{2.2}$$

and the flux terms

$$\mathbf{f}_x = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ \rho uw \\ \rho uE + up \end{pmatrix} \qquad \mathbf{f}_y = \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ \rho vw \\ \rho vE + vp \end{pmatrix} \qquad \mathbf{f}_z = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ p + \rho w^2 \\ \rho wE + wp \end{pmatrix}$$

Here ρ is the density of the fluid, $\mathbf{u} = (u, v, w)^T$ the velocity vector in x-, y- and z-direction. E denotes the total energy and p the pressure. The first equation represents the conservation of mass, the second to fourth component is the conservation of momentum in x-, y- and z-direction and the last line conservation of energy.

The Euler equations are computational less expensive, but problems rise from the neglection of the viscosity. Close to the boundary, one can no longer ignore viscous effects. Furthermore, viscosity is the reason of dissipation of kinetic energy on small length scale in turbulence flow.

2.2 Navier-Stokes equations

Including also viscosity and heat conduction, one can extend the Euler equations towards the Navier-Stokes equations. In this case, we no longer assume that the stress tensor σ and the heat flux $\mathbf{q}$ equal zero. Writing the compressible Navier-Stokes equations again in conservative (divergence) form one obtains

$$\frac{\partial \omega}{\partial t} + \frac{\partial \mathbf{g}_x}{\partial x} + \frac{\partial \mathbf{g}_y}{\partial y} + \frac{\partial \mathbf{g}_z}{\partial z} = 0 \tag{2.3}$$

Where ρ, ω and $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)^T = (u, v, w)^T$ denotes the same variables as in the case of the Euler equations.

From [6] we know that the flux terms are

$$\mathbf{g}_x = \begin{pmatrix} \rho u \\ p + \rho u^2 + \sigma_{xx} \\ \rho uv + \sigma_{xy} \\ \rho uw + \sigma_{xz} \\ \rho uE + up + u\sigma_{xx} + v\sigma_{xy} + w\sigma_{xz} + q_x \end{pmatrix}$$

$$\mathbf{g}_y = \begin{pmatrix} \rho v \\ \rho uv + \sigma_{xy} \\ p + \rho v^2 + \sigma_{yy} \\ \rho vw + \sigma_{yz} \\ \rho vE + vp + u\sigma_{xy} + v\sigma_{yy} + w\sigma_{yz} + q_y \end{pmatrix}$$

$$\mathbf{g}_z = \begin{pmatrix} \rho w \\ \rho uw + \sigma_{xz} \\ \rho vw + +\sigma_{yz} \\ p + \rho w^2 + +\sigma_{zz} \\ \rho wE + wp + u\sigma_{xz} + v\sigma_{yz} + w\sigma_{zz} + q_y \end{pmatrix}$$

In addition to the Euler equations, we now furthermore have

- the heat flux $\mathbf{q} = (q_x, q_y, q_z)^T = -\lambda(T_x, T_y, T_z)$
- the stress tensor $\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}$ for a Newtonian fluid with $\sigma_{ij} = -\eta(\frac{\partial \mathbf{u}_i}{\partial j} + \frac{\partial \mathbf{u}_j}{\partial i}) + \delta_{ij}\frac{2}{3}\eta(\frac{\partial \mathbf{u}_i}{\partial i} + \frac{\partial \mathbf{u}_j}{\partial j})$
  for $i, j \in \{x, y, z\}$ and the Kronecker delta δ_{ij}.

Instead of the direct solution of the Navier-Stokes equation, one often uses the time-averaged Reynolds-averaged NavierStokes equations (RANS equation). This approach uses the Reynolds decomposition. The RANS equation reduce the computational requirements, compared to the direct numerical solution (DNS) with the Navier-Stokes equation.

Chapter 3

Mathematical formulation of the optimization process

Mathematical optimization is a process which finds its application in a variety of scientific fields. These range from financial science to medical engineering up to optimization in aircraft design. Independent of the concrete problem, we can write an optimization problem (OPT) via the following mathematical abstraction [5]:

$$(\text{OPT}) \begin{cases} & I(\omega, \alpha) \to \min, \\ \text{such that} & R(\omega, \alpha) = 0, \\ \text{and} & c(\omega, \alpha) \geq 0 \end{cases} \tag{3.1}$$

Where $I(\omega, \alpha)$ is the vector of objective functions, which should be minimized. This minimization takes place while fulfilling the state equation $R(\omega, \alpha) = 0$. Furthermore we need to take care of additional constraints $c(\omega, \alpha)$. It is α the vector of design variables and ω the vector of state variables, depending on the design variables. Note that this is a formulation for a minimization problem only. If we instead want to maximize a quantity, one can reformulate the given problem by multiplying the objective function with -1 and thus obtaining again a minimization problem. The same holds for the constraints. An example for a concrete formulation of the optimization problem above is given as:

$$\begin{cases} & C_{\text{drag}}(\omega, \alpha) \to \min, \\ \text{such that} & R(\omega, \alpha) = 0, \\ \text{and} & C_{\text{lift}}(\omega, \alpha) - C^0_{\text{lift}} \geq 0 \end{cases} \tag{3.2}$$

With $\omega = (\rho, u, v, w, p)^T$. $R(\omega, \alpha) = 0$ is supposed to describe the (discretized) system of Euler equations. This formulation denotes, that we want to reduce the drag coefficient, while keeping the initial lift coefficient. Furthermore we describe the flow by the Euler equations. The vector α is a parametrization of a wing geometry, e.g. span, thickness or sweep angle.

Several algorithms for solving the optimization problem have been developed in the past years. These algorithms can be divided into two classes. On the one hand we have the non gradient-based

algorithms and on the other the gradient-based algorithms. An algorithm is gradient-based, if it evaluates the derivative of the objective function with respect to the design variables during its process chain. The evaluation of these gradients results in high computational costs for complex application in CFD, if one follows a finite difference approach. As an alternative one can abandon the gradient calculation, as it is the case in evolutionary algorithms or the SIMPLEX algorithm, which results in a slower convergence behavior. This means that the amount of optimization iterations increases which also results in high computational costs. By following the approach of the adjoint formulation, one obtains the advantage of faster convergence of a gradient-based algorithm, while keeping the computational requirements per design iteration at a reasonable level. Due to this fact, adjoint algorithms are applied to several problems in the field of computational fluid dynamics with a large number of design variables, as further explained in [13].

3.1 Adjoint formulation of the optimization problem

In gradient-based algorithms, one has to evaluate the gradients of the objective function with respect to the design variables $dI_i/d\alpha_j,\ i = 1..N_o,\ j = 1..N_d$ during the optimization process, where N_o denotes the number of objective functions and N_d the number of design variables, respectively. In most practical application we have $N_d >> N_o$. As a first idea to evaluate these gradients, the finite difference approach is applied.

$$\frac{dI_i}{d\alpha_j} \approx \frac{I_i(\omega, \alpha + \Delta\alpha \cdot e_j) - I_i(\omega, \alpha)}{\Delta\alpha} \tag{3.3}$$

with e_j the j-th unit vector. This approach makes it necessary, to solve the state equations $N_d + 1$ times (or even $2N_d$ times when performing central differences). This is due to the fact, that $I_i(\alpha + \Delta\alpha \cdot e_j)$ can only be evaluated, iff we know ω for the perturbed design. As the vector ω is only obtained by solving the state equation $R(\omega) = 0$, this results in huge computational effort, as we need to solve the state equation for each perturbation. Furthermore the finite difference approach involves problems of numerical stability such as loss of significance due to round-off errors. Therefore, the choice of an optimal $\Delta\alpha$ is non trivial. For real world application, this approach is computational too expensive.

In the following, we now consider the adjoint approach. Therefore we introduce the Lagrange multiplier Λ and similar to [1] define

$$\mathcal{I}_i := I_i(\omega, \alpha) = I_i(\omega, \alpha) + \Lambda^T R(\omega, \alpha) \tag{3.4}$$

Using chain rule, we get

$$\frac{d\mathcal{I}_i}{d\alpha} = \Big[\frac{\partial \mathcal{I}_i}{\partial \omega}\frac{d\omega}{d\alpha} + \frac{\partial \mathcal{I}_i}{\partial \alpha}\Big] + \Lambda^T\Big[\frac{\partial R}{\partial \omega}\frac{d\omega}{d\alpha} + \frac{\partial R}{\partial \alpha}\Big] \tag{3.5}$$

$$= \Big[\frac{\partial \mathcal{I}_i}{\partial \omega} + \Lambda^T\frac{\partial R}{\partial \omega}\Big]\frac{d\omega}{d\alpha} + \Big[\frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T\frac{\partial R}{\partial \alpha}\Big] \tag{3.6}$$

As $d\omega/d\alpha$ is the computational expensive term, we choose Λ such that

$$\begin{aligned} &\frac{\partial \mathcal{I}_i}{\partial \omega} + \Lambda^T\frac{\partial R}{\partial \omega} = 0 \\ \Leftrightarrow &\Big[\frac{\partial R}{\partial \omega}\Big]^T\Lambda = -\Big[\frac{\partial \mathcal{I}_i}{\partial \omega}\Big]^T \end{aligned} \tag{3.7}$$

Equation (3.7) is called adjoint equation. Note, that in this case we only evaluate the partial derivatives and no total derivatives, which can be obtained via the differentiated version of the TAU code. With this solution Λ, we can evaluate the gradients as

$$\frac{d\mathcal{I}_i}{d\alpha} = \frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T\frac{\partial R}{\partial \alpha} \tag{3.8}$$

Instead of solving the state equation for each design variable, we now only need to solve one additional linear system for each objective function. As $N_d >> N_o$, this results in a huge reduction of computational effort, compared to the finite difference approach.

3.2 Extension towards the mesh adjoint

With the adjoint approach, we can eliminate the influence of the number of design variable on the number of flow solution. However, we can improve this method further, by making use of the adjoint approach for the perturbation of the mesh. In equation (3.8) we still have to evaluate the expression $\partial R/\partial\alpha$. This is usually done via

$$\frac{\partial R}{\partial \alpha} \approx \frac{R(\alpha + \Delta\alpha) - R(\alpha)}{\Delta\alpha} \tag{3.9}$$

To eliminate the cost of the finite difference approach in (3.9), we can reduce the computational cost even more which is described in [1] and is derived in the following.

Let $X = X(\alpha)$ be the vector of the coordinates of each node in the mesh. Obviously, we have $R(\alpha) = R(X(\alpha))$, as the residual is only an implicit function of the design variables, but an explicit function of the mesh. As the mesh is a function of the design variables, the above equation holds. Furthermore, one can choose the linear elastic equation as a state equation for X and α as explained in [2].

$$T(X, \alpha) = X_s - KX_v = 0, \tag{3.10}$$

where X_s describes the surface mesh, K the stiffness matrix and X_v the volume mesh, respectively. In equation (3.8) we now introduce the Lagrange multiplier Θ referring to the mesh state equation above.

$$\begin{aligned}\frac{d\mathcal{I}_i}{d\alpha} &= \frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha} \quad \text{as derived in equation (3.8)} \\ &= \frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha} + \Theta^T \frac{dT}{d\alpha} \quad \text{by the second Lagrange multiplier} \\ &= \frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha} + \Theta^T \Big[\frac{\partial T}{\partial \alpha} + \frac{\partial T}{\partial X}\frac{dX}{d\alpha}\Big]\end{aligned}$$

With $X = X(\alpha)$, it holds $dX/d\alpha = \partial X/\partial\alpha$ and by chain rule $\frac{\partial}{\partial\alpha} = \frac{\partial}{\partial X}\frac{\partial X}{\partial \alpha}$ it follows that:

$$\frac{d\mathcal{I}_i}{d\alpha} = \frac{\partial \mathcal{I}_i}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha} + \Theta^T \Big[\frac{\partial T}{\partial \alpha} + \frac{\partial T}{\partial X}\frac{\partial X}{\partial \alpha}\Big] \tag{3.11}$$

$$= \Big[\frac{\partial \mathcal{I}_i}{\partial X} + \Lambda^T \frac{\partial R}{\partial X}\Big]\frac{\partial X}{\partial \alpha} + \Theta^T \Big[\frac{\partial T}{\partial \alpha} + \frac{\partial T}{\partial X}\frac{\partial X}{\partial \alpha}\Big] \tag{3.12}$$

$$= \Big[\frac{\partial \mathcal{I}_i}{\partial X} + \Lambda^T \frac{\partial R}{\partial X} + \Theta^T \frac{\partial T}{\partial X}\Big]\frac{\partial X}{\partial \alpha} + \Theta^T \frac{\partial T}{\partial \alpha} \tag{3.13}$$

As before, we now choose the Lagrange multiplier such, that the computationally expensive term, i.e $\partial X/\partial\alpha$ drops out. Therefore, let Θ fulfill

$$\frac{\partial \mathcal{I}_i}{\partial X} + \Lambda^T \frac{\partial R}{\partial X} + \Theta^T \frac{\partial T}{\partial X} = 0 \tag{3.14}$$

Or written similar to the flow adjoint equation (3.8), the mesh adjoint equation is:

$$\Big[\frac{\partial T}{\partial X}\Big]^T \Theta = -\Big[\frac{\partial \mathcal{I}_i}{\partial X}\Big]^T - \Big[\frac{\partial R}{\partial X}\Big]^T \Lambda \tag{3.15}$$

It is obvious, that we first have to solve for the flow adjoint solution, before calculating the mesh adjoint solution. If we now choose Θ such, that (3.15) holds, from we get

$$\frac{d\mathcal{I}_i}{d\alpha} = \Theta^T \frac{\partial T}{\partial \alpha} \tag{3.16}$$

Together with (3.15) and the fact, that $\frac{\partial T}{\partial X} = -K$ and $\frac{\partial T}{\partial \alpha} = \frac{\partial X_s}{\partial \alpha}$, as only the surface mesh is partial dependent of α, but not the volume mesh, we finally obtain

$$\frac{d\mathcal{I}_i}{d\alpha} = \Theta^T \frac{\partial X_s}{\partial \alpha} \tag{3.17}$$

The advantage of equation (3.17), compared to (3.8) lies in the fact, that we do not need to evaluate $\partial R/\partial\alpha$. If we want to evaluate $\partial X_v/\partial\alpha$, we need to create a new volume mesh for each design variable, if we instead make use of the linear elastic equation we only have to evaluate $\partial X_s/\partial\alpha$. Together with the mesh adjoint approach, this reduces the computational effort, as the deformation of a surface mesh takes only a fraction of the time spend on the deformation of the volume mesh.

3.3 The SQP algorithm

Essential for an optimization process is, in addition to an effective evaluation of the gradients, the estimation of a new design in the next iteration. For a gradient-based optimization with constraints, SQP (Sequential Quadratic Programming) algorithm are often used. The software modeFrontier contains a SQP algorithm, called NLPQLP. NLPQLP is a modified SQP implementation and solves a variety of nonlinear optimization problems with constraints.

In its general form, a SQP algorithm is applied to a problem of form (3.1). At first we consider the case where we only deal with equality constraints.

$$(\text{OPT}) \begin{cases} & f(x) \to \min, \\ \text{such that} & h_i(x) = 0,\ i = 0..M \end{cases} \tag{3.18}$$

Therefore, we introduce the Lagrange function $\mathcal{I} := f(x) + \lambda^T h(x)$ with the Lagrange multiplier $\lambda^T = (\lambda_1, \lambda_2, \ldots, \lambda_M)$ and $h(x) = (h_1(x), h_2(x), \ldots, h_M(x))^T$, respectively. A necessary condition for the solution of equation (3.18) is therefore given by

$$\nabla \mathcal{I}(x, \lambda) = \begin{pmatrix} \nabla f(x) + \sum_i \lambda_i \nabla h_i(x) \\ h(x) \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.19}$$

Equation (3.19) is in its general formulation a nonlinear system [11]. The related Newton system for solving $\nabla \mathcal{I}(x, \lambda) = 0$ can be written as

$$H_{\mathcal{I}}(x^k, \lambda^k) \begin{pmatrix} \Delta x \\ \Delta \lambda \end{pmatrix} = -\nabla \mathcal{I}(x^k, \lambda^k) \text{ with } x^{k+1} = x^k + \Delta x,\ \lambda^{k+1} = \lambda^k + \Delta \lambda \tag{3.20}$$

Where $H_{\mathcal{I}}(x^k, \lambda^k)$ denotes the Hessian of $\mathcal{I}(x^k, \lambda^k)$. We can further manipulate the system and rewrite it as

$$\begin{pmatrix} \mathcal{I}_{xx}(x^k, \lambda^k) & h'(x^k)^T \\ h'(x^k) & 0 \end{pmatrix} = \begin{pmatrix} \Delta x \\ \lambda^{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) \\ h(x) \end{pmatrix} \tag{3.21}$$

Note that we do not explicitly evaluate $\mathcal{I}_{xx}$. This relates to a further increase of computational cost and is not possible in most real world application. Instead, one uses an approximation of the Hessian as described in [9] and [7]. If $\mathcal{I}_{xx}$ has full rank and the Hessian is symmetric positive definite, the solution of (3.21) is unique. Solving (3.21) for Δx is then equivalent to solving the quadratic subproblem

$$(\text{QP}) \begin{cases} & \text{minimize } r(\Delta x) = \frac{1}{2} \Delta x^T \mathcal{I}_{xx} \Delta x + \nabla f(x)^T \Delta x, \\ & \text{such that } h'(x) \Delta x + h(x) = 0 \end{cases} \tag{3.22}$$

The single steps of a SQP algorithm are described in an abstract way in Algorithm 1.

Algorithm 1 Solving 3.18

input : $x^0,\ \lambda^0,\ \epsilon$
output: Solution of 3.18

$x^i = x^0, \quad \lambda^i = \lambda^0$;
while $||\nabla \mathcal{I}|| > \epsilon$ *or* $\lambda_i < 0$ **do**
    Calculate $f, \nabla f,\ h,\ \nabla h, \nabla \mathcal{I}$;
    Solve the quadratic subproblem 3.22 for Δx;
    Solve the system 3.21 for λ;
    Update x^i and λ^i;
end
return x^i;

The quadratic subproblem in general can be solved by e.g. Conjugate Gradients, Augmented Lagrangian or with the Interior Point Method.

If we want to solve an optimization problem with inequality constraints, we can reformulate the quadratic subproblem as

$$\text{(QP)} \begin{cases} \text{minimize } r(\Delta x) = \frac{1}{2}\Delta x^T \mathcal{I}_{xx} \Delta x + \nabla f(x)^T \Delta x, \\ \text{such that } h'(x)^T \Delta x + h(x) = 0 \\ \text{and } g'(x)^T \Delta x + g(x) \leq 0 \end{cases} \tag{3.23}$$

This leads to a slightly different procedure, derived in [11].

Chapter 4

Implementation and process chain

For an optimization process, we need several tools to bring together the required components based on the theoretical analysis. Starting point for a CFD simulation is a CAD software, which generates a surface geometry description (e.g. NURBS surfaces) based on certain design parameters. Afterwards, the model is used to generate a computational mesh with a mesh generation tool as an input for further calculation. The CFD simulation itself is done by the use of a flow solver, which solves for the required flow variables. Afterwards the results need to be post processed, to extract the desired data. Following the decision to use a gradient based algorithm, the derivatives of the objective functions with respect to the design variables need to be calculated.

As this only describes the process chain for a single CFD simulation, we now need an optimization algorithm. With this optimization algorithm we can make use of the post processed data and the gradients to create an improved design, given certain optimization criteria. During each optimization iteration, we create a new design, based on the objective function and the corresponding gradients obtained from the previous designs. An exception is the first design, which has to be specified by the user. Afterwards the gradients are calculated and a new iteration starts. This process repeats until the design converges to an optimal solution, or the number of maximal design evaluation is reached.

We start with an approach, that does not use the mesh adjoint to obtain the mesh sensitivities. Afterwards, the existing model is extended by the use of the mesh adjoint approach. Finally we will move away from an aerodynamic optimization towards an aeroelastic optimization by introducing an internal structure into our design.

4.1 Software and modelling

Whereas the above description is a generic way to implement the optimization process, we are going to explain the software used in this optimization in the following sections, together with some modelling assumptions we made. Therefore, we will at first describe the different tools that have been used to perform the aforementioned processes.

4.1.1 CATIA v5

We use CATIA v5 in our optimization to generate the required CAD model. CATIA (Computer Aided Three-dimensional Interactive Application) is a commercial software available for Windows and the Unix/Linux envrionment [14]. Since its "Version 5" (v5), CATIA can define geometries as a non-static, parametrized model. Whereas a static CAD model has no variable input parameter, a parametrized model allows to reshape a given initial model by new input parameters obtained due to the optimization process. To simplify the IO-process, the parametrized model can generate a certain CAD model, if a specific design table is provided. This method also allows to easily generate perturbed models which are needed to calculate the gradient, e.g. directly by finite differences or through the mesh sensitivity in (discrete) adjoint methods.

An example of a design table is shown below.

```
LambdaPointY (mm)          2000.0
LambdaPointX (mm)           -80.0
KinkPointY (mm)            5000.0
KinkPointX (mm)             300.0
TipPointY (mm)             9000.0
TipPointX (mm)              700.0
ScalingLambdaProfile          0.5
ScalingKinkProfile            0.4
ScalingTipProfile             0.35
LambdaProfileTilt (deg)       0.0
KinkProfileTilt(deg)          0.0
TipProfileTilt(deg)           0.0
TipDeltaY (mm)              300.0
TipDeltaZ (mm)             -200.0
TipRotationAboutYsimilarAxis (deg)  0.0
TipRotationAboutZsimilarAxis (deg) 85.0
TipRotationAboutXsimilarAxis (deg)  0.0
```

4.1.2 Mesher

At EADS Cassidian, the mesh generation software is the in-house tool Mesher. Mesher is used to generate unstructured hybrids grids, given a CAD model, a logical model and an input script that controls the parameters required for the mesh specification. A logical model describes the hierarchy of the CAD groups (e.g. wing, engine, fuselage, farfield). The mesh generation is separated into different phases. The three most important ones are the surface mesh generation, followed by the layer generation and the volume mesh generation. During the surface mesh generation, only the surface of the CAD model, the farfield and, if available, the symmetry plane are meshed. Afterwards layers are created from the solid surfaces to accurately model the boundary layers. It is crucial to guarantee a sufficient resolution in this area. Finally the volume mesh in the computational domain is generated. For all the mentioned steps, a variety of parameters can be specified.

During the optimization process, it is necessary to evaluate slightly perturbed designs to calculate the finite differences of the flow residual with respect to the design variables. To evaluate the flow residual for a perturbed design variable, the mesh has to be deformed to fit the perturbed surface shape. Following the approach without the mesh adjoint, one needs the perturbed *volume* mesh. Using the mesh adjoint, it is instead sufficient to evaluate the perturbed *surface* mesh. As the mesh generation process is time consuming, techniques have been developed to enable a faster mesh perturbation. Instead of creating a complete new volume mesh for each perturbed design variable, one corrects the position of the existing nodes instead. This process can be compared to a network of nodes, connected by springs. By moving certain nodes to a new position, the remaining nodes rearrange according to the original position and the stiffness of the corresponding springs. In our example the stiffness is proportional to the inverse distance of two nodes. This method is called spring analogy and enables a faster mesh generation for perturbed meshes. As an example of the mesh generation process, we can see the surface mesh and the layers for to the DLR-F6.

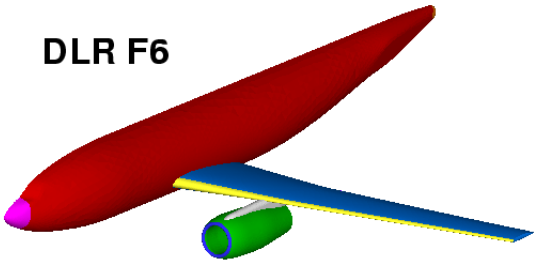


Figure 4.1: CAD model

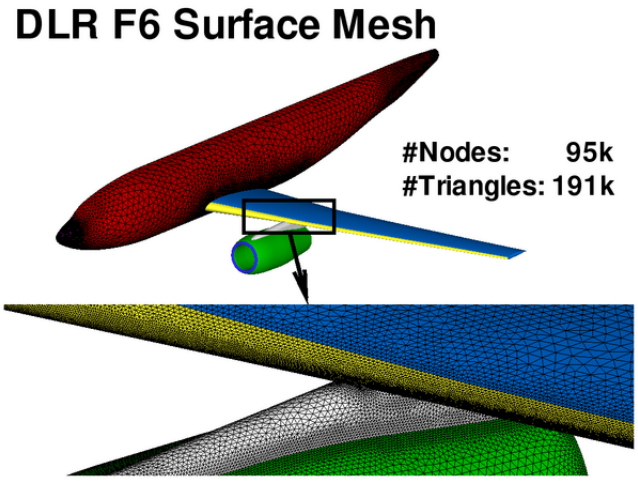


Figure 4.2: Surface mesh

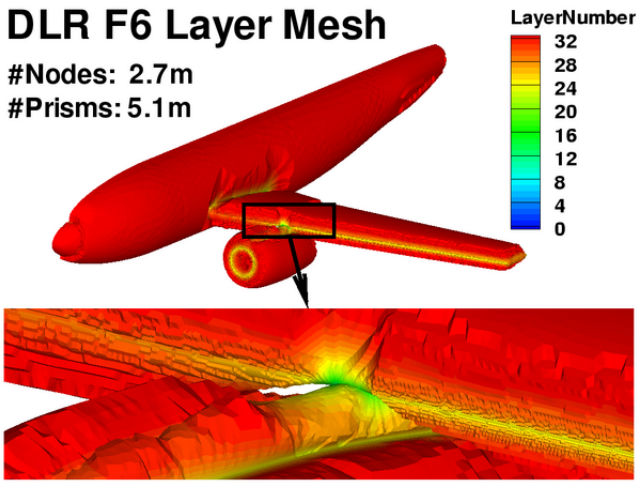


Figure 4.3: Boundary layer mesh

4.1.3 DLR TAU-code

To solve the governing flow equations for the different flow variables, a solver has to be applied to the given framework. In all following applications, the TAU-code from DLR (German Aerospace Center) is used to perform this task. The TAU-code allows to solve viscous and inviscid flows for hybrid unstructured grids and can be applied from low subsonic, up to hypersonic flow regimes. TAU contains modules for grid partitioning, pre processing and post processing, as well as for grid adaption and grid deformation[4]. For efficient computation, TAU is optimized for parallel computers by making use of the message passing interface (MPI). The internal solver "uses an edge-based dual-cell approach, i.e. a vertex-centred scheme, where inviscid terms are computed employing either a second-order central scheme or a variety of upwind schemes using linear reconstruction (of the left and right states) for second-order accuracy"[4]. Several modeling and numerical parameters can be defined via input scripts, to ensure a correct computational framework for the given flow phenomena; this includes the choice of different turbulence models, multigrid settings, parameters for flux calculation, solver type and further solver settings.

Especially important for the optimization process is the ability of TAU to solve the adjoint flow and the adjoint mesh equations. To make use of these features, the given input scripts for the flow equation only need to be extended by adding specific parameters for the calculation of the adjoint equation. For the computation of the (linear) adjoint equation, the possible solver types are PETSc and Facemat. As PETSc stores the complete Jacobian, it requires more memory. In the following test cases, we used Facemat for our computations, as it was not possible to allocate the required amount of memory, needed for the computations with PETSc. The reduced amount of memory, needed for Facemat goes hand in hand with a less precise solution, due to simplifications made in the computation of the Jacobians $\partial R/\partial \omega$, e.g. due to turbulence freezing.

4.1.4 Weight estimation

As we are performing an optimization with multidisciplinary aspects, but mainly focus on aerodynamics, wo do not perform a CSM simulation for the weight estimation. Therefore, it is necessary to approximate the weight of the wing geometry. The weight is needed to estimate the required lift, which has to be generated during cruise flight. In our model we also consider the stored fuel as a part of the weight. Due to this fact, the take off weight differs from the zero fuel weight. The required lift is therefore chosen to be the mean value of take off weight (TOW) and zero fuel weight (ZFW). As $L = C_L \cdot q \cdot S$ with q the dynamic pressure and S the wing area, we can calculate the target C_L, which corresponds to the estimated lift. To estimate the weight of our wing, we use the weight approximation presented in [8]. As we are dealing with a different type of wing, i.e. three sections instead of two, we use a modification of the approximation, presented in [10].

Let S denote the wing area, N_{ult} the ultimate load factor, b measures the span, $(t/c)_{avg}$ is the average airfoil thickness, λ the average taper ratio and Λ the sweep angle, respectively. Assuming the wing structure is sized by the wing bending stresses, we can then calculate the approximated weight W_{wing} as presented in [8]:

$$W_{wing} = 202 \cdot S + 8 \cdot 10^{-5} \cdot \frac{N_{ult} \cdot b^3 \sqrt{\mathrm{TOW} \cdot \mathrm{ZFW}}(1 + 2\lambda)}{(t/c)_{avg} \cos^2(\Lambda) S(1 + \lambda)} \tag{4.1}$$

Based on this estimation, we recalculate ZFW and get a new estimate for W_{wing} by applying the above formula again. This process repeats until the weight is converged.

4.1.5 modeFrontier

To control the process chain in our optimization process we use the software modeFrontier, a proprietary software developed by the company Esteco. ModeFrontier is a multidisciplinary and multiobjective software[3], which we use for an overriding structuring of the single optimization step. During each step, modeFrontier can execute internal operations, such as data copying or assigning variables, but it is also possible to call external applications via scripts (e.g. in python scripts or bash scripts) or directly for certain application. An important feature are the implemented optimization algorithms. These include non gradient based algorithms such as the Simplex algorithm or several genetic algorithms. But also gradient based algorithms are available, where the most common ones are the AfilterSQP algorithm and the NLPQLP algorithm, both implementations of a SQP algorithm. An example for a simple process structure, defined in the modeFrontier GUI, can be seen in the figure 4.4. Here we have the data path from left to right which controls the input variables, temporary variables and output variables and from bottom to top the logic path which controls the different steps of the optimization process. However, in this example we only execute a single shell script (denoted “ShScript”).

4.2 Process chain of the aerodynamic optimization without mesh adjoint

To describe the general process chain, we will explain the single steps, using the simplified structure presented in figure 4.5. The beginning of an optimization process is the initial design configuration, which has to be specified by the user and given as input. After the corresponding design table has been integrated into the modeFrontier environment, we generate a new CAD model by calling CATIA v5. This model is then used to create a corresponding CFD mesh by the use of Mesher.

Normally, the internal structure is a result of a computational structural mechanics (CSM) calculation which is based on a CSM mesh, but in this example, we use a simplified approach to estimate the weight by an analytical formula based on the one given in [8]. Together with the calculated surface area of our wing geometry, we estimate a target C_L which generates the required

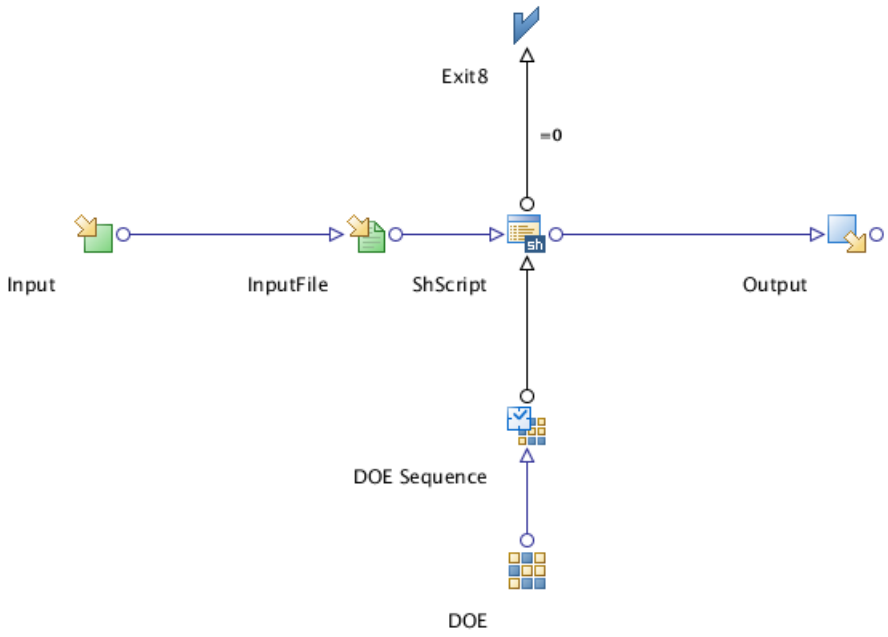


Figure 4.4: Simplex optimization with modeFrontier

lift. This target C_L is one of the input parameters for TAU. To reach a specified target C_L, TAU will adjust the angle of attack during its iteration to match the desired C_L. Having reached this point of the process chain, the following two processes can be executed in parallel. On the one side we have the calculation of the flow equation. After this calculation has finished, we will update the corresponding input files for TAU and start the computation of the flow adjoint for each objective function, also in parallel. On the other side, we will prepare the calculation of the mesh sensitivities. Therefore, we perturb the given design table for each variable and generate a new CAD model for each perturbation. From a given CAD model, we will then perturb the original mesh (note that we do not generate a complete new mesh). This also takes place in parallel. Afterwards we calculate the perturbed weights and surface areas for each design variable. When both parallel paths have finished, the computation of the gradients starts. These step combines the mesh sensitivities and the information obtained from the adjoint solution. Furthermore, we take into account changes in the angle of attack. The gradient is then handed to the optimization algorithm to generate a new design and start over. This cycle repeats, until a specified convergence in either the relevant gradients, or the change of the objective function is reached.

In figure 4.6 we can see the main process chain of the optimization process. Note that certain details for input and output have not been included to focus on the main parts.

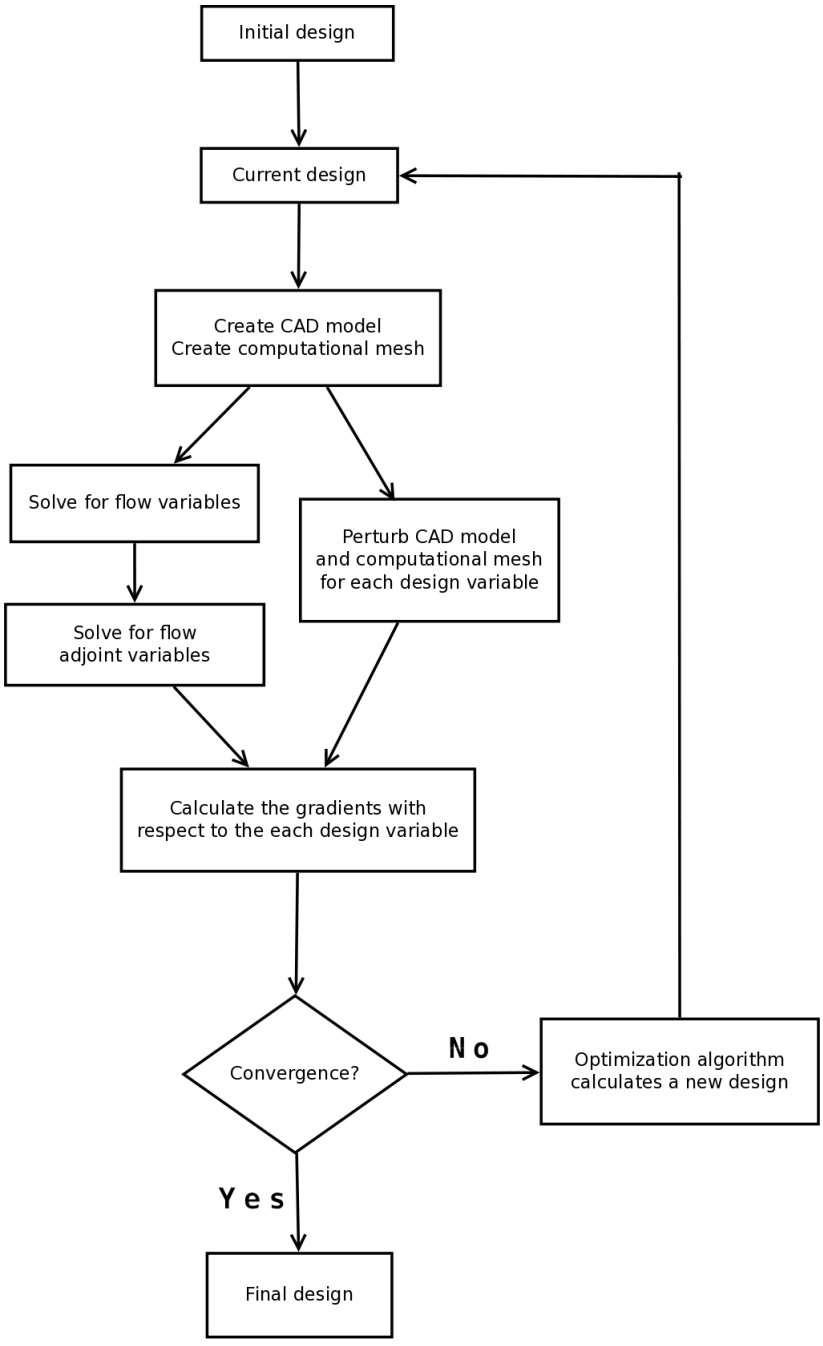


Figure 4.5: Process chain for optimization without mesh adjoint

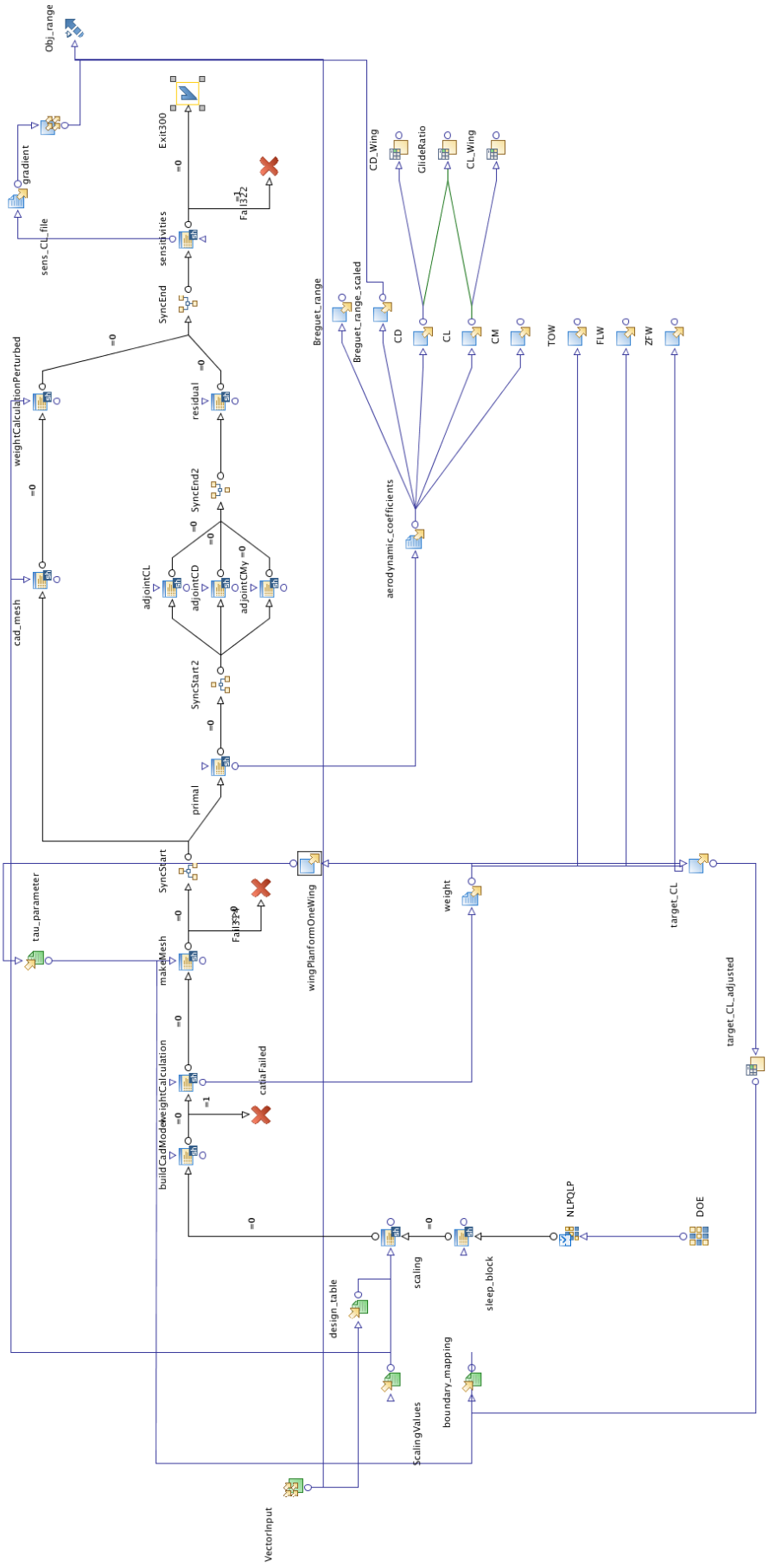


Figure 4.6: Extract of the process chain for optimization in modeFrontier

4.2.1 Application-specific elements of the implementation

As the implementation of certain procedures in the optimization is straightforward, we now want to focus on those steps, where potential error sources in the process chain have been observed, or modelling assumptions where made which have a large influence on the final output. To rate the quality of the converged design regarding physical correctness, we used similar existing geometries as references.

4.2.1.1 Perturbing the geometry

For the mesh sensitivities it is necessary to perturb the initial design table by a specific perturbation $\epsilon_i \, \forall i = 1, \ldots, N_d$, where N_d denotes the number of design parameters. This can produce a failure during two phases of the mesh perturbation process. The creation of a new CAD model can fail, which is due to the parameterization of the model. More common is a failure during the mesh generation process. The perturbed mesh could produce invalid shaped finite elements which can not be smoothed during the mesh generation process. If this happens, it becomes necessary to perturb the initial geometry by a different perturbation and repeat the mesh generation process. The process chain, that handles the mesh perturbation is described in Algorithm 2.

Algorithm 2 Mesh perturbation

input : Original design and perturbation for each parameter
output: Perturbed mesh for each parameter

ϵ_i = initial perturbation for parameter $i \quad \forall i = 1, \ldots, N_d$;
pm_i = Perturbed mesh for parameter i exists = false $\quad \forall i = 1, \ldots, N_d$;
for $i = 1..N_d$ **do in parallel**
  while pm_i =*false* **do**
    Create a new CAD model with CATIA v5;
    if *Creation of the new CAD model did not fail* **then**
      Call Mesher for mesh perturbation;
      if *Creation of the new mesh did not fail* **then**
        pm_i =true;
      end
    end
    Choose a different perturbation ϵ_i
  end
end

In practice, it is rarely the case, that the mesh generation process fails. If, however, it does happen that a failure during the mesh generation takes place, it usually requires only one or two new perturbation for ϵ to obtain a feasible perturbed mesh.

Note that, even though the complete process theoretically can take place in parallel, in our setup the creation of the CAD model takes place in serial, which is due to the limited amount of CATIA v5 licenses. As the amount of time that is spent on the CAD model creation is relatively small compared to the time spent for the mesh generation, we still speak of parallel process. In the

concrete implementation we will check for each parameter, if another process already uses CATIA v5. Given that case, the process is set to pause until CATIA v5 becomes unused again and will then be executed.

4.2.1.2 Calculation of the aerodynamic sensitivities

In chapter 3.1, we already derived the calculation of the gradients of the objective function with respect to the design variables. In addition to the theoretical background, we will now elaborate how this procedure is implemented in the process chain.

In the following, Br denotes the Breguet range, L the lift and D represents the drag. To avoid confusion, we will use p_i for the design variables and will use α for the angle of attack. One aspect, that we did not include in the derivation of the adjoint formulation is the constraint that in cruise flight the required lift equals the given weight. In each iteration, we expect a certain value for L which is based on the weight of the wing. We force TAU to reach this lift, by varying α to generate the corresponding C_L, i.e. α is not a free variable, but dependent on the design variables. Therefore we need to consider the influence of the gradient with respect to the the angle of attack. We introduce $\frac{d^{total}}{dp}$ to emphasize, that we include the rate of change of the angle of attack. From this analysis, it follows that we require the following gradients:

$$\frac{d^{total}L}{dp} = \underbrace{\frac{dL}{dp}}_{\text{TAU via adjoint}} + \underbrace{\frac{\partial L}{\partial \alpha}}_{\text{TAU via code derivatives}} \cdot \frac{d\alpha}{dp} \tag{4.2a}$$

$$\frac{d^{total}D}{dp} = \underbrace{\frac{dD}{dp}}_{\text{TAU via adjoint}} + \underbrace{\frac{\partial D}{\partial \alpha}}_{\text{TAU via code derivatives}} \cdot \frac{d\alpha}{dp} \tag{4.2b}$$

As we force TAU to reach a certain L we use the following approximation for the left hand side of equation (4.2a).

$$\frac{d^{total}L}{dp} \approx \frac{p_{\text{dyn}} \cdot A^{\epsilon} \cdot {C_L}^{\epsilon} - p_{\text{dyn}} \cdot A \cdot C_L}{\epsilon} \tag{4.3}$$

Here A^{ϵ} is obtained by calculating the area for the perturbed geometry and C_L^{ϵ} is the new target C_L required to generate a lift that matches the weight of the perturbed geometry. Furthermore, from the definitions $L = C_L \cdot q \cdot A$ and $D = C_D \cdot q \cdot A$ with q the dynamic pressure, we get

$$\frac{dL}{dp} = \frac{dC_L}{dp} \cdot q \cdot A + C_L \cdot p_{dyn} \cdot \frac{dA}{dp} \tag{4.4a}$$

$$\frac{dD}{dp} = \frac{dC_D}{dp} \cdot q \cdot A + C_D \cdot p_{dyn} \cdot \frac{dA}{dp} \tag{4.4b}$$

And

$$\frac{dA}{dp} = \frac{A^{\epsilon} - A}{\epsilon} \tag{4.5}$$

Where dC_L/dp and dC_D/dp are calculated by equation (3.8). The superscript ϵ is used for the perturbed area A and target C_L. We also make use of

$$\frac{\partial L}{\partial \alpha} = \frac{\partial C_L}{\partial \alpha} p_{dyn} A \tag{4.6}$$

$$\frac{\partial D}{\partial \alpha} = \frac{\partial C_D}{\partial \alpha} p_{dyn} A \tag{4.7}$$

where partial derivatives with respect to α are obtained from the adjoint solution using equation (3.8). If we use the approximation in (4.3) to rewrite equation (4.2a), we get

$$\frac{d\alpha}{dp} = \frac{1}{\frac{\partial L}{\partial \alpha}} \cdot \Big[\frac{d^{total}L}{dp} - \frac{dL}{dp}\Big] \tag{4.8}$$

which we can use to calculate $d^{total}D/dp$ via equation (4.2b) as

$$\frac{d^{total}D}{dp} = \frac{dD}{dp} + \frac{\partial D}{\partial \alpha} \cdot \frac{1}{\frac{\partial L}{\partial \alpha}} \cdot \Big[\frac{d^{total}L}{dp} - \frac{dL}{dp}\Big] \tag{4.9}$$

Note, that even though we can calculate $d^{total}L/dp$ without the use of an adjoint solution, however, we still need dL/dp to solve equation (4.2a) for $d\alpha/dp$ which gives $d^{total}D/dp$, together with the adjoint solution for dL/dp.

The Breguet range is defined as

$$Br = \frac{aM}{c_T} \frac{L}{D} \ln \frac{TOW}{ZFW} \tag{4.10}$$

With a the speed of sound, M the cruise Mach number and c_T the thrust specific fuel consumption. With L/D we denote the lift to drag ratio, TOW is the take off weight and ZFW the zero fuel weight, respectively. Note, that as we fixed TOW, ZFW is a function of p. As Br is the objective function for this simulation, we have to evaluate

$$\frac{d^{total}Br}{dp_i} \quad \forall i = 1, \ldots, N_d \tag{4.11}$$

Using standard calculus we get

$$\frac{d^{total}Br}{dp} = \frac{aM}{c_T} \Big[\Big(\frac{d^{total}L}{dp}\frac{1}{D} - \frac{L}{D}\frac{d^{total}D}{dp}\Big) \ln \frac{TOW}{ZFW} - \frac{L}{D}\frac{1}{ZFW}\frac{d^{total}ZFW}{dp}\Big] \tag{4.12}$$

Furthermore we again force $L = (TOW + ZFW)/2$, which we reformulate as $ZFW = 2L - TOW$ to eliminate ZFW and obtain

$$\begin{aligned}\frac{dBr}{dp} &= \frac{aM}{c_T} \Big[\Big(\frac{d^{total}L}{dp}\frac{1}{D} - \frac{L}{D}\frac{d^{total}D}{dp}\Big) \ln \frac{TOW}{2L - TOW} - \frac{L}{D}\frac{1}{2L - TOW}\frac{d^{total}(2L - TOW)}{dp}\Big] \\ &= \frac{aM}{c_T} \Big[\Big(\frac{d^{total}L}{dp}\frac{1}{D} - \frac{L}{D}\frac{d^{total}D}{dp}\Big) \ln \frac{TOW}{2L - TOW} - 2\frac{L}{D}\frac{1}{2L - TOW}\frac{d^{total}L}{dp}\Big]\end{aligned} \tag{4.13}$$

The gradient dBr/dp is then propagated towards the modeFrontier optimizer. Together with previous gradients and values for the objective function, a new design is calculated via the specified optimization algorithm, which is a SQP algorithm in our application.

4.3 Process chain of the aerodynamic optimization with mesh adjoint

If we introduce the mesh adjoint approach into our optimization, the process chain changes only slightly. In figure 4.7 we see a simplified structuring of the processes which are executed in comparison to the process chain with mesh perturbation in figure 4.5. Figure 4.8 shows the process chain in modeFrontier. In this case, we only perturb the surface mesh instead of the complete volume mesh. Additionally, we have to output the displacements in all three dimensions for each element. To calculate the mesh adjoint solution, we need a flow solution and the flow adjoint solution, together with the mesh stiffness matrix. The point of time at which we calculate the stiffness matrix is arbitrary, as long, as we obtain the matrix before the calculation of the mesh adjoint solution starts. In the current development version of TAU at Cassidian, we can only perform a serial calculation of the mesh adjoint solution. Therefore, we also need to preprocess the mesh a second time with the number of domains set to one. The computation of the mesh adjoint solution can start after the flow adjoint solutions is available. Similar to the flow adjoint solution, also the mesh adjoint solutions for each objective functions are computed in parallel.

4.3.1 Application-specific elements of the implementation

As we do only change the calculation of the gradients of the objective function, all other parts of the optimization process are identical to the flow adjoint approach with volume mesh deformation. The only difference is the fact, that it is now sufficient to perturb the surface mesh (instead of the complete volume mesh) and output the perturbation for each point that belongs to the surface mesh.

4.4 Process chain of the aeroelastic optimization

In our application, we will not perform a structural optimization during the process of an aerodynamic optimization. We will instead use the converged geometry obtained by the aerodynamic optimization as a starting point for the structural optimization, which is also considered a loosely coupled multidisciplinary optimization. This procedure is also called sizing. During the sizing, the goal is to minimize the weight while satisfying the structural constraints. These structural constraints arise from the loads extracted from the aerodynamic simulation. In the given implementation, the structural model is implemented in CATIA v5. The geometry is composed into about 60 elements. These elements include spars, ribs, flaps and panels for the surface of the wing. During the CSM simulation, we obtain the von Mises stresses for each single element. We will then compare these stresses, with the maximal value, depending on the given material and safety factors. During the next step, we will then increase the thickness of a given element if its stress is above the tolerated

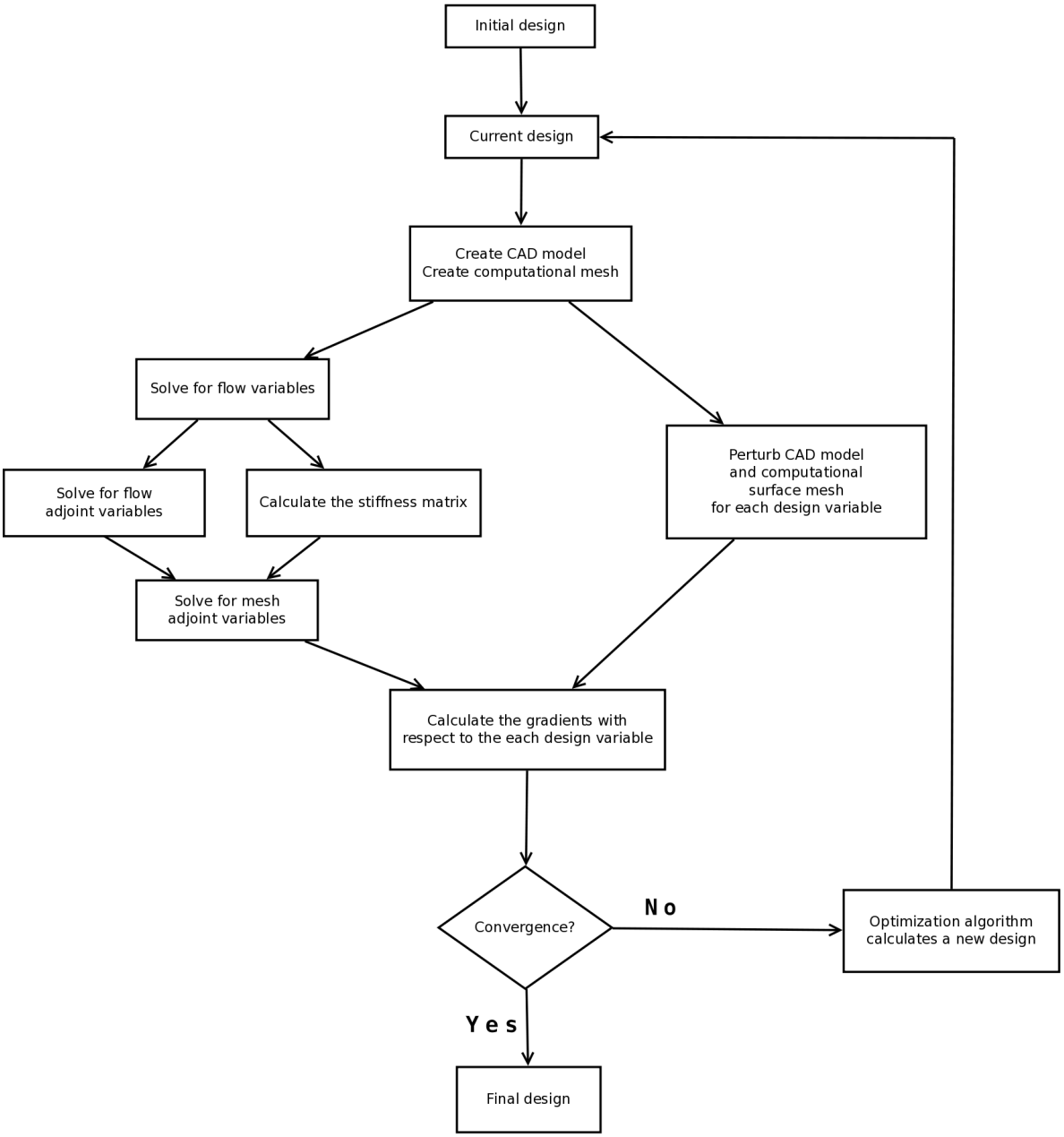
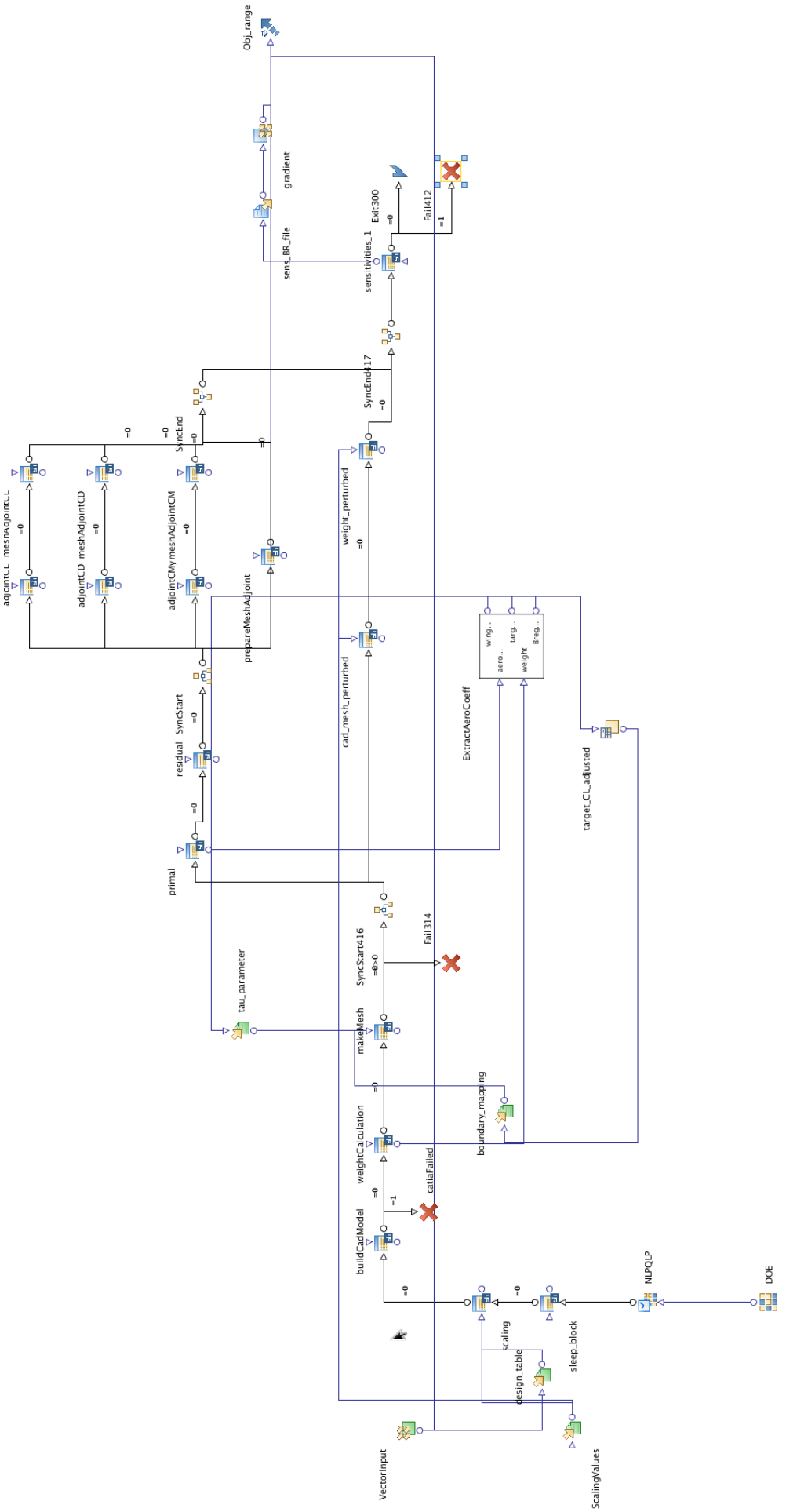


Figure 4.7: Process chain for optimization with mesh adjoint


Obj_range
gradient
sens_BR_file
sensitivities_1
Exit300
Fail412
SyncEnd417
SyncEnd
adjointCL_meshAdjointCL
adjointCD_meshAdjointCD
adjointCM_meshAdjointCM
prepareMeshAdjoint
weight_perturbed
cad_mesh_perturbed
SyncStart
residual
primal
ExtractAeroCoeff
target_CL_adjusted
tau_parameter
Fail314
SyncStart416
makeMesh
boundary_mapping
weightCalculation
catiaFailed
buildCadModel
NLPQLP
DOE
scaling
design_table
sleep_block
VectorInput
ScalingValues

maximum, or decrease the thickness to reduce the overall weight if the stress is below the tolerated maximum.

During the sizing, we will consider each element independent from its neighbours. Furthermore, we assume a linear dependency between the stresses and the thickness of each element. Therefore, we will not calculate any gradients in the process of sizing. We rather scale each element independent and rerun the CSM simulation with the new structure. We will repeat this process, until the stresses for each element are below the maximal value and the thickness is as small as possible. Following this approach, it is possible that for certain elements the thickness reduces arbitrarily, as the stress for the element is always below the critical value. This would indicate, that the spar or rib at that certain position would not be necessary and could be removed from the internal structure.

In our implementation we decided to keep the elements fixed, meaning that they do not get removed even though the stresses on a certain elements might be under the critical value for any thickness. Under this assumption we could either remove elements at the end of the sizing process if the given thickness is under a minimal thickness or we could force a minimal thickness value during the sizing. If we want to keep the structural layout and do not want to remove certain elements, it becomes necessary to force a minimal value for the thickness of each element, as CATIA does not allow an arbitrary small thickness.

Chapter 5

Gradient comparison and numeric parameter influence

Prior to the complete optimization, it is necessary to check each step of the process separately. Here we will focus on the output gradients and their dependency on a variety of numeric parameters and setups. Additionally, we will try to find an optimal set of parameters for the simulation with TAU to speed up the overall optimization.

5.1 Comparison of the gradients

For the gradient based optimization, it is necessary to have a sufficiently accurate evaluation of the gradients. As we use the gradient to obtain a direction in the design space for the next design evaluation it is not necessary to have fully converged gradients. This is due to the fact that one estimates a new direction in every single iteration, thus the direction is corrected in each step.

In figure 5.3 for the example of the viscous flow over a wing at $Ma = 0.75$ and $\alpha = 2.1°$,we can see the dependency of the accuracy of the gradients for different convergence levels of the flow adjoint solution. Hereby, we place the relative error of the gradient on the y-axis and the relative residual of the flow adjoint solution on the x-axis. The relative error of the gradient is defined as

$$\frac{||\nabla^{approx} f - \nabla f||_2}{||\nabla f||_2} \tag{5.1}$$

with ∇f the best approximation of the gradient. As we can see, the relative error of the gradient decreases as the relative error in the flow adjoint solution decreases.

In figure 5.2, we see a cutplane through the geometry with the adjoint variable v_z for C_L and C_D. Especially in 5.2 a) we can see the typical characteristic of the adjoint solution. This is the fact, that the contour of the adjoint variables behaves similar to the way a flow variable (e.g. density) would behave, if the geometry would be placed in the opposite direction for the given flow field.

To judge the quality of the gradients of the objective function with respect to the design variables, we can compare three different types of gradients. One obtained via finite differences, the second

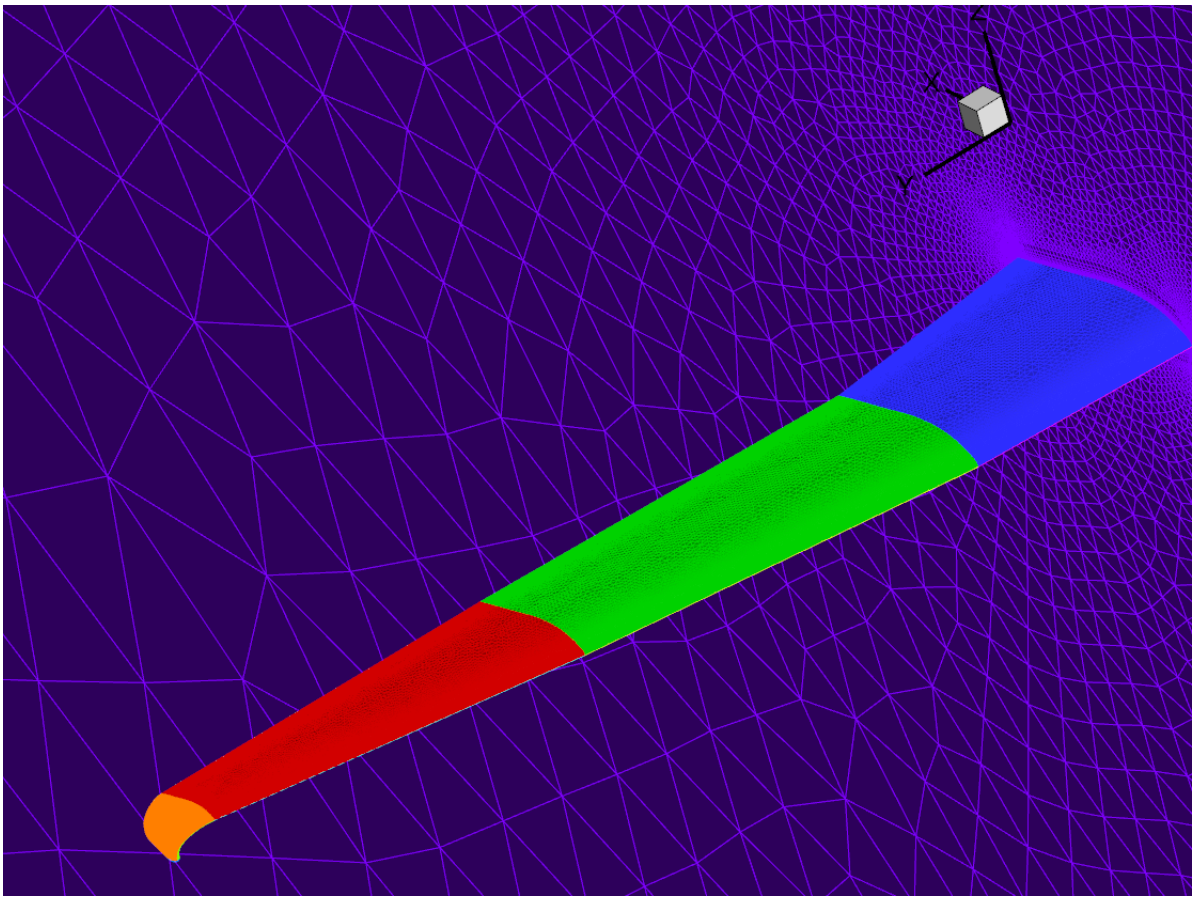
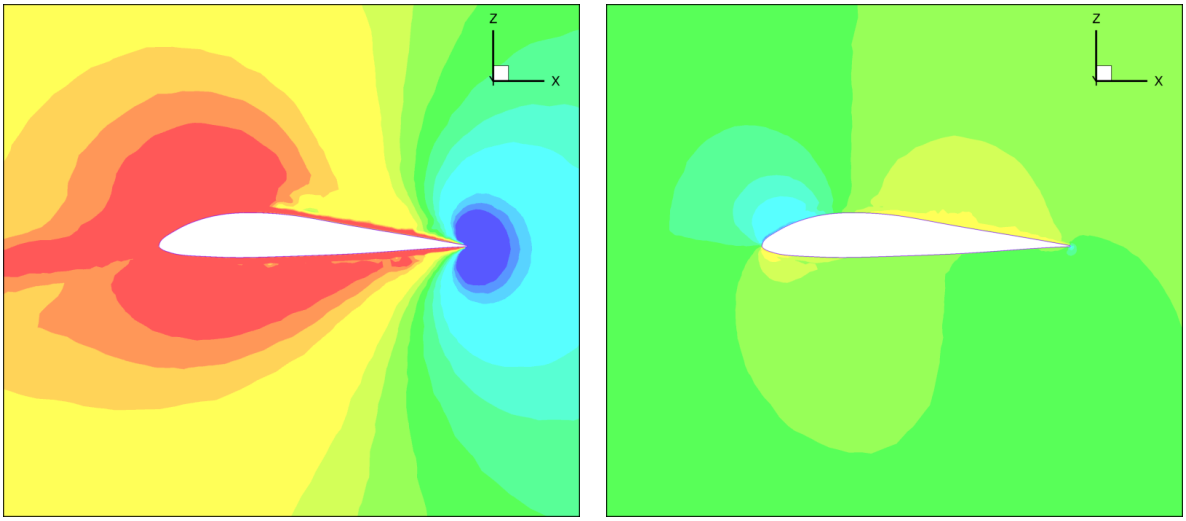


Figure 5.1: Surface mesh of the wing configuration

by making use of the flow adjoint and performing a volume mesh deformation and the third by the use of the mesh adjoint and a surface mesh deformation.

For this comparison we focus on two design variables, the x- and y-position of the kink point, to be seen in 5.1. In figure 5.4 and 5.5 we can see the gradients of C_L and C_D with respect to the x- and the y-position of the kink point. For the calculation we used two different meshes for the same geometry: a coarse computational mesh of 1.4 million points and a finer one with 9.7 million points. On the y-axis we have the gradients and on the x-axis different values for the perturbation magnitude ϵ (in mm) of the design parameters. From the figures 5.4 and 5.5 several information can be deduced. At first we can see, that for the flow adjoint approach and the finite difference approach, the gradients vary if we choose a too small perturbation, relative to the mesh size. Especially for the finite difference approach, this becomes a problem as we even observe a change in the sign of the gradients for too small perturbation in figure 5.5 b). The second information we can deduce from the data, is the fact that for larger perturbation, the gradients of the flow adjoint approach and the finite difference approach match. This can be seen for C_L and C_D for both parameters. If we now consider the gradients obtained via the mesh adjoint approach, we can deduce that the perturbation



a) Lift adjoint variable v_z on cutplane

b) Drag adjoint variable v_z on cutplane

Figure 5.2: Adjoint Solution, $Ma = 0.75$ and $\alpha = 2.1°$

has almost no influence on the quality of the gradients. However, for the two given parameters, we can see an offset between the mesh adjoint approach and the flow adjoint gradients.

In figure 5.6 we see the gradients obtained by the mesh adjoint approach, divide by the flow adjoint gradients. A value of one would suggest, that the gradients are equal. However, for the given setup we can observe that for certain input parameters the quotient is close to one, while there also exist parameters where the gradients of the mesh adjoint approach differ from the flow adjoint gradients.

Together with the information from 5.4 and 5.5, we can see that the finite difference approach matches the adjoint approach, at least for larger perturbation. Therefore, the gradients of the mesh adjoint approach are to be considered wrong for the parameters where the values in figure 5.6 differ from one. Thus we need to summarize, that the quality of the mesh adjoint gradients is not sufficient for all parameters. This difference likely stems from having used a different mesh stiffness matrix in the mesh adjoint solution as in the actual mesh deformation. This is due to the fact that the deformation used in Mesher is the spring analogy approach, whereas TAU uses the linear elasticity approach.

5.2 TAU parameter influence

For the computation of the flow solution, the default TAU parameter setup led to a satisfactory wall clock time. Therefore, we will focus on the parameters of the adjoint calculation. For the flow adjoint solution, the generalized minimal residual method (GMRes) is applied. GMRes is

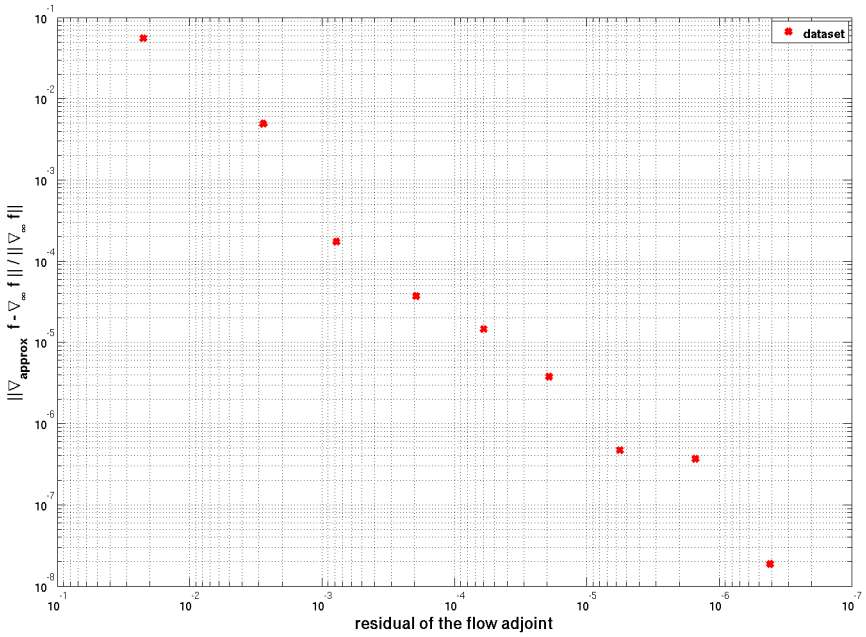
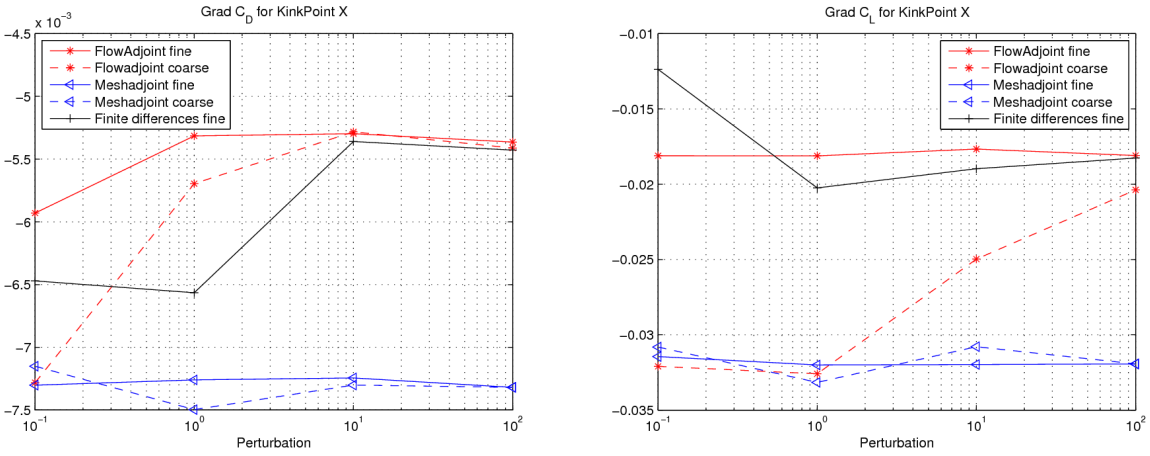


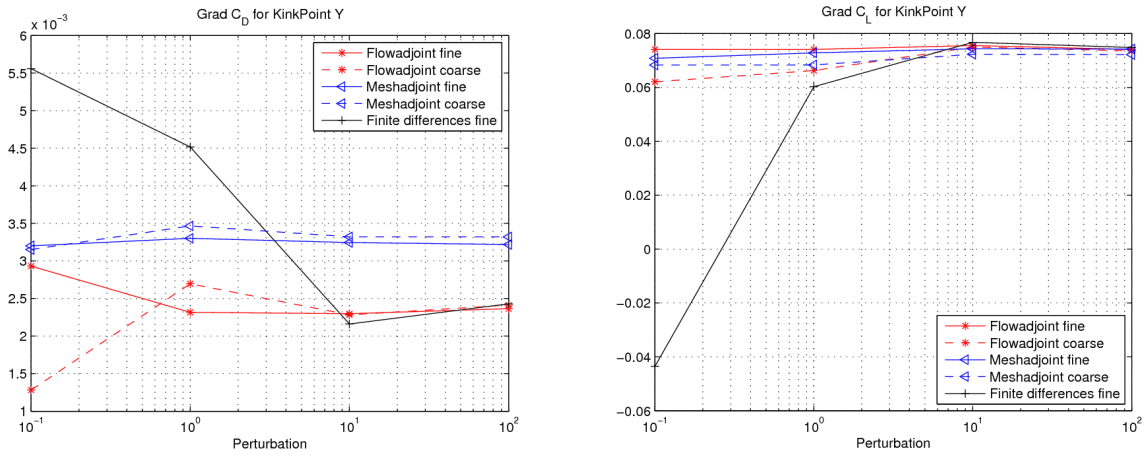
Figure 5.3: Accuracy of the gradients over accuracy of the flow adjoint solution, $Ma = 0.75$ and $\alpha = 2.1°$



a) Gradients for C_D

b) Gradients for C_L

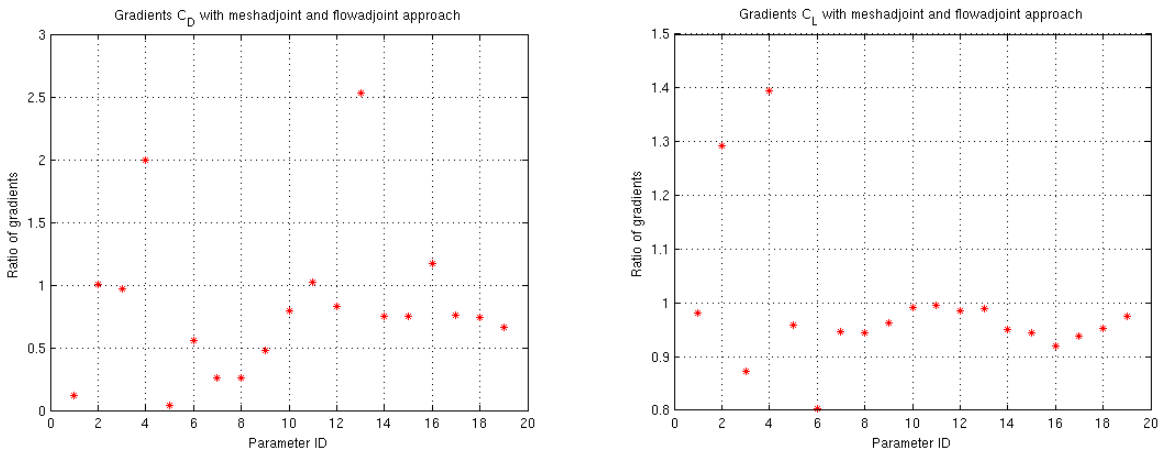
Figure 5.4: Gradients for kink point x-coordinate



a) Gradients for C_D

b) Gradients for C_L

Figure 5.5: Gradients for kink point y-coordinate



a) Gradients for C_D

b) Gradients for C_L

Figure 5.6: Quotient of gradients

an approximative iterative Krylov subspace solver for systems of linear equations, which are not symmetric. We can use either the Facemat framework or the PETSc framework. Due to the lower requirements in memory for the storage of the Jacobian, we decided to use Facemat. TAU allows the user to specify the number of inner iterations (the maximum Krylov subspace dimension) and the number of preconditioning iteration for the GMRes calculation.

5.2.1 Number of inner iterations

To find an optimal value for the number of inner iterations that should be performed, we will use the same number of preconditioning iterations for all experiments. We will also use the same flow solution for each adjoint calculation. The 5000 flow iterations correspond to a relative residual of 10^{-6} for the given test case. From figure 5.7 we can deduce, that increasing the number of inner iterations in the range from 10 to 80 reduces the wall clock time. A larger number of inner iterations corresponds to a larger memory requirement, which is no restrictive factor in our application. In accordance with the TAU user guide, we decide to take 80 as the value for inner iterations.

5.2.2 Number of preconditioning iterations

The number of preconditioning iterations corresponds almost proportional to the time spent on a single iteration. On the other side, a better preconditioning can increase convergence speed. In this experiment, we kept the number of inner iterations constant and used the same flow solution as in the previous case. We can see that an optimal value for the number of preconditioning iterations lies between 10 and 20. As the TAU user guide suggests a value between 1 and 10, we will chose 10 for the following computations. In addition to the number of preconditioning iterations, it is also possible to specify the level of incomplete lower-upper factorization corresponding to the preconditioning. In our case, choosing a value different from 0 (i.e. no fill-in) drastically increased the wall clock runtime wherefore, we kept the default value ILU-0.

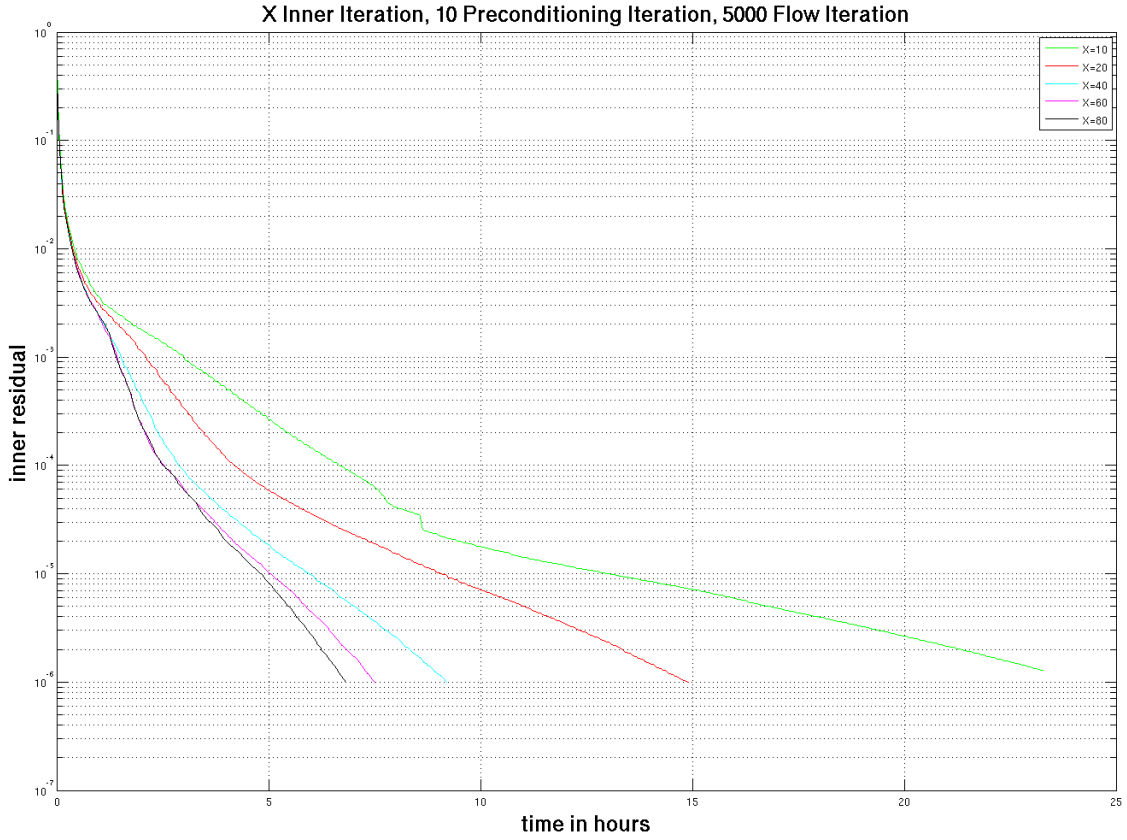


Figure 5.7: Different values for number of inner iterations

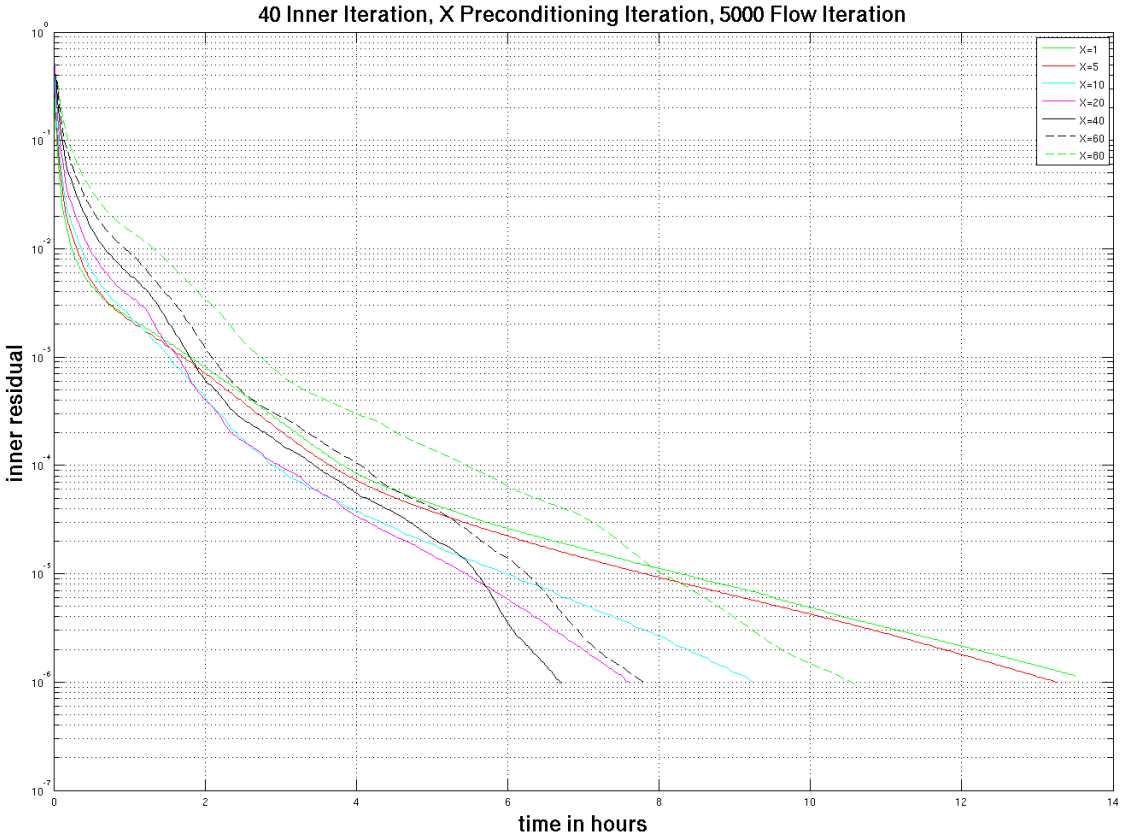


Figure 5.8: Different values for number of preconditioning iterations

Chapter 6

Test cases and results

6.1 Wing planform optimization problem

For this test case, the objective is to maximize range with a fixed take off weight, by optimizing the wing planform for low drag and low structural weight. We can see the initial geometry, together with the computational mesh in the figure 5.1. In this model, we can vary 19 parameters. Of these parameters, twelve describe the wing planform, while the remaining seven variables describe the tip geometry of the wing. The given mesh has a resolution of 1.4 million elements. The mach number is set to $M = 0.35$ as this is a realistic value for cruise flight of a UAV. The objective function is the Breguet range. Our optimization problem reads

$$\begin{cases} & \text{Breguet Range}(\omega, \alpha) \rightarrow \max, \\ \text{such that} & R(\omega, \alpha) = 0, \\ \text{and} & C_{\text{Lift}}(\omega, \alpha) - C_{\text{Lift}}^{\text{target}}(\alpha) = 0, \\ \text{where} & C_{\text{Lift}}^{\text{target}}(\alpha) \cdot p_{\text{dyn}} \cdot \text{Wing Area}(\alpha) = \text{Lift}(\alpha) = \text{Weight}(\alpha) \end{cases} \tag{6.1}$$

In this context, $R(\omega, \alpha) = 0$ denotes that the Navier-Stokes equation are satisfied. Note, that in this context $C_{\text{Lift}}(\omega, \alpha) - C_{\text{Lift}}^{\text{target}}(\alpha) = 0$ is not considered as a constraint for the optimization, but is handled within the TAU-code. The angle of attack will be adjusted automatically, to generate the required C_{Lift}. During this process, after 2000 flow iterations, every 50 iterations the angle of attack will be adjusted. This method allows to meet the required equality constraint by almost no additional computational effort. Choosing the Breguet range as our objective function has the advantage, that we combine lift, drag and wing weight within one quantity. It is therefore not necessary to add further inequality constraints which guarantee a certain minimal lift oder maximal drag.

6.2 Optimization without mesh adjoint

Based on our gradient accuracy experiments, discussed in Chapter 4, we limit the number of iteration for the flow equation to 6000, which corresponds to a residual of roughly $5 \cdot 10^{-6}$ to 10^{-5},

and a maximal residual of about 10^{-3} to $5 \cdot 10^{-3}$. The flow adjoint solution converges to a residual of 10^{-4}. A single optimization step takes about two hours (wall-clock time). All CFD simulation are executed on an Intel Xeon cluster, parallelized on 64 processors. The mesh perturbations are executed externally on a machine with a large memory, but on only a single core per design variable. This is due to the fact, that mesh generation is a memory consuming process, rather than a computationally expensive process.

As the optimization algorithm, we choose NLPQLP with the default setup provided by modeFrontier.

6.2.1 Results

The optimization performs 26 iterations until a converged local design optimum is reached. In this case, convergence refers to changes in the design space. The final design can be seen in figure 6.1. In figure 6.2 we can see the shape of the wing during subsequent iterations. The objective function can be seen in 6.4 b). The two quantities that influence the Breguet range are the weight in figure 6.3 a) and the glide ratio L/D in 6.4 a).

From the data in 6.3 a) we can deduce, that during the first three iterations, the increase in Breguet range is due to the decrease in weight. From iteration three up to iteration five the weight stays almost constant, while the glide ratio increases, which also increases the Breguet range. In iteration six, we can see that the Breguet range decreases for the first time. This is due to the fact that the design changes are too large, therefore in the next iteration the optimizer tries to reduce the step size for the performed changes in the design space. After about 13 iterations, the weight stays almost constant over rest of the optimization. Therefore, only changes in the glide ratio influence the objective function. During iteration 16 a peak in the Breguet range is reached. Afterwards the optimizer performs further iterations which have a lower value for the objective function. A possible explanation for this behaviour is the fact that the gradients do not vanish, as the gradients are not exact due to the perturbation that is chosen. Therefore, each new design moves away from the optimum. After no increase in the objective function can be observed, at iteration 22, the optimizer moves back to the design space close to design 16 and is converged three iterations later.

In figure 6.4 b) we can see an increase of the Breguet range of 10 % from 948km to 1043km. The average flight weight, which is the mean between take-off weight and zero fuel weight, decreased 1.6%, from 56528N to 55637N, while the glide ratio increase from 21.9 to 22.1 for 0.9%. A summary of all data can be seen in table 6.6. The planform area for a single wing decrease from $10.67m^2$ to $10.23m^2$ for 4.4%, which can be seen in 6.3 b). Even though the position of the Kink point, the Lambda point and the Tip point are independent in parameterization, it can be seen, that a nearly straight leading edge and a straight trailing edge are reached in the late iteration.

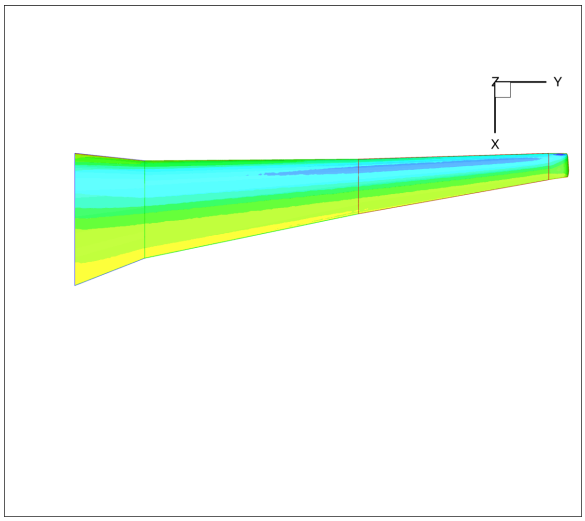
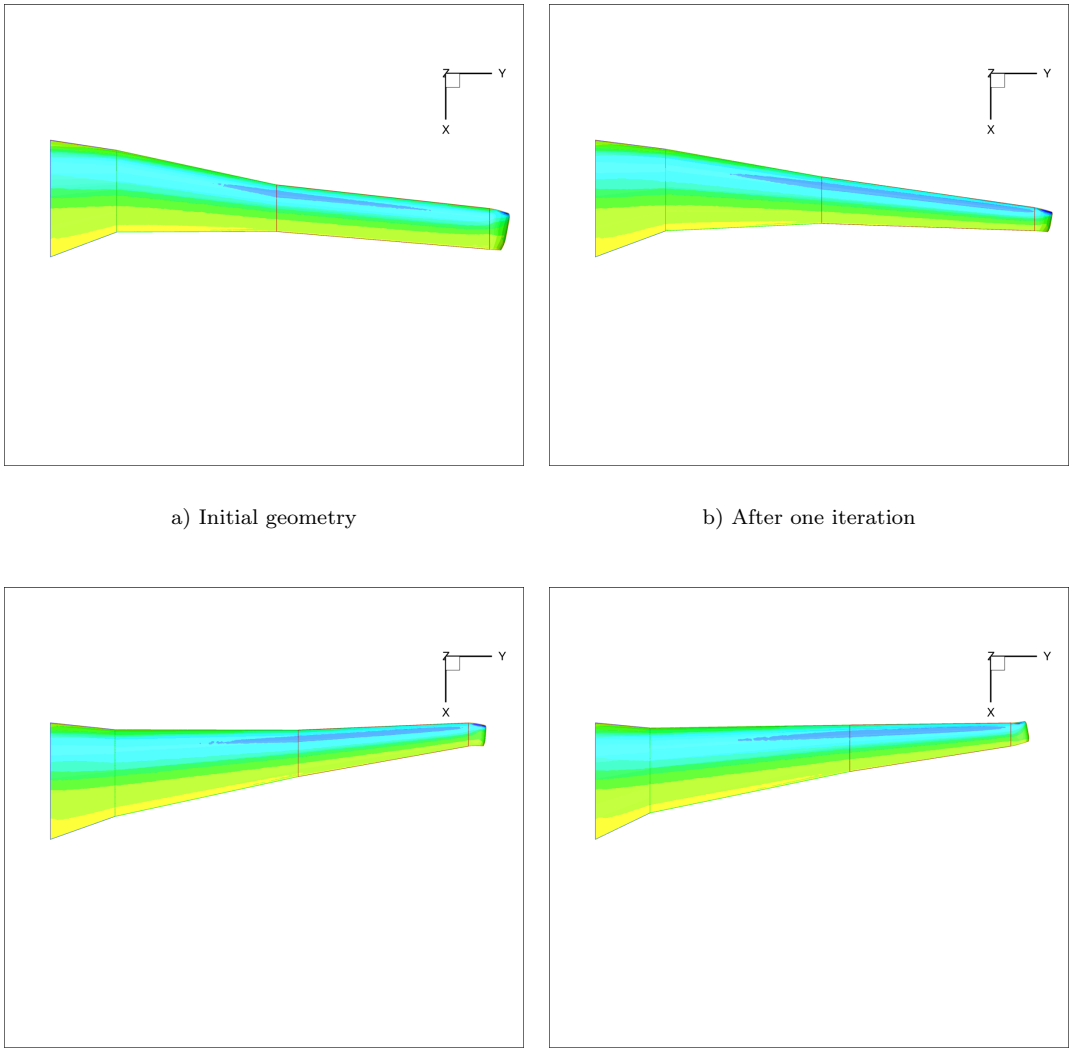


Figure 6.1: Converged geometry with pressure distribution

6.3 Optimization with mesh adjoint

As we already saw in section 5.1, the gradients obtained by the mesh adjoint approach do not match the ones obtained with the flow adjoint plus mesh perturbation. Therefore, the optimizer does not move towards the optimal design as it happens during the approach without the use of the mesh adjoint. From this it follows, that then number of overall optimization iterations does increase.

Comparing overall runtime it can be seen, that in our application the mesh adjoint approach also increases the runtime of a single iteration. This has several reasons. As we have a rather small number of design parameters, we can execute each volume mesh perturbation in parallel. For a large number of parameter, the number of mesh perturbation that can be executed in parallel would be limited by the available memory. This would not be possible, if the number of design parameters would exceed a certain value such that the required amount of memory force the machine to swap. This would drastically increase runtime. As this is not the case, the volume mesh perturbation can run parallel to the computation of the flow solution and the flow adjoint solution. Therefore, the fact that we only need a surface perturbation for the mesh adjoint approach does not bring an advantage for the given setup. Due to the fact, that the calculation of the mesh adjoint can only start after the flow adjoint solution is available, the time needed for the calculation of the mesh adjoint adds up to the runtime of the approach without the use of the mesh adjoint. In the available development version of TAU, the solution of the mesh adjoint equation is a serial computation. This is also problematic, as it requires a machine that has a large enough amount of RAM for a single processor. Another time consuming calculation is the evaluation of the stiffness matrix, but as this can be done in parallel to the computation of the flow solution it has no influence on the overall runtime.



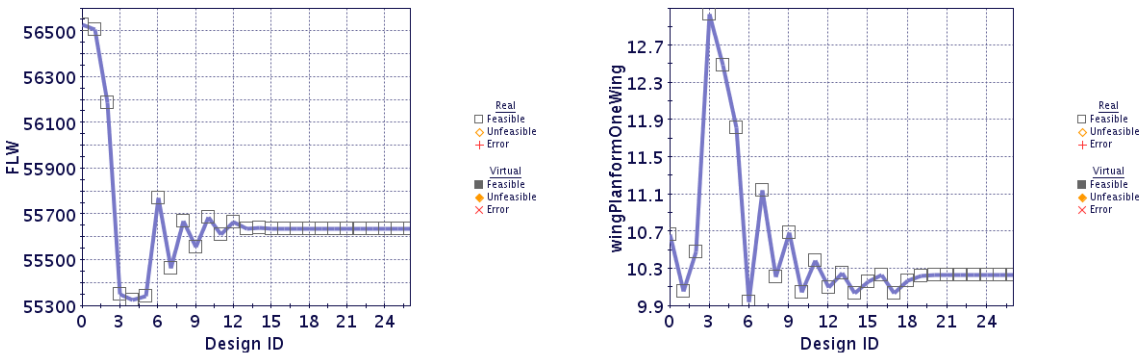
a) Initial geometry

b) After one iteration

c) After three iteration

d) After eleven iteration

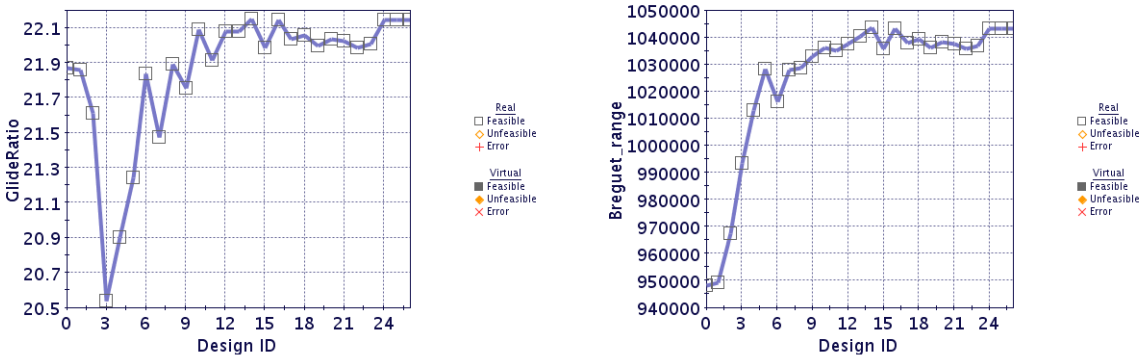
Figure 6.2: Pressure distribution for different iteration



a) Weight over design iteration

b) Planform area over design iteration

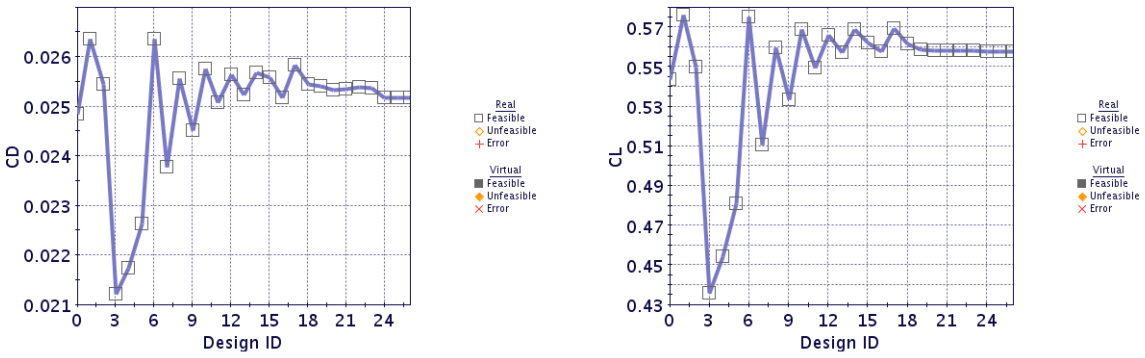
Figure 6.3: Structural quantities over design iteration



a) Glideratio over design iteration

b) Efficiency over design iteration

Figure 6.4: Aerodynamic coefficients over design iteration



a) CD over design iteration

b) CL over design iteration

Figure 6.5: Aerodynamic coefficients over design iteration

	Initial geometry	Optimized Geometry
Area	10.670	10.229
TOW	70000	70000
ZFW	43056	41275
log(TOW/ZFW)	0.4859	0.5282
α	2.9291	3.1608
C_L	0.543527	0.557437
C_D	0.024852	0.025172
Lift	56521.2	55571.9
Drag	2584.3	2509.4
Lift/Drag	21.87	22.15
Range	948048.9	1043401.4

Figure 6.6: Comparison between initial and optimized geometry

In summary, it can be stated that for the given setup and the relatively few number of design variables, the mesh adjoint does increase the overall runtime. This is mostly due to the fact, that the mesh adjoint approach gradients do not match the gradients obtained by the standard approach. Furthermore, the mesh adjoint approach would only give an advantage for a computation in which the number of design variables is that large, that a parallel computation of the mesh perturbation is no longer possible and therefore, the time required for the mesh perturbation dominates the overall computation.

However, the implemented process flow for modeFrontier gives the user an alternative to the original process flow in the case, that the number of design variables becomes larger, if the development version of the mesh adjoint approach comes to a point where the gradients match the original approach.

6.4 Aeroelastic optimization

After the aerodynamic optimization is converged, we use the structural model for the final geometry. In figure 6.7 we see the partitioning of the wing. Each surface element has a thickness value which can be modified individually. Each edge is either a spar if it runs in horizontal direction, or a rib if it runs in vertical direction. For the modelling of flaps, the trailing edge is separated into larger segments than the leading edge. In this model, we have a total of 60 elements for which we can vary the thickness. Similar to the aerodynamic optimization, we use a design table to provide CATIA with the required thicknesses. The material we used for this case is aluminium 2024-T3. For the CSM simulation, we again need a computational mesh, but in this case only for the surface elements including spars and ribs, but no volume elements as it was necessary in the CFD calculation. The mesh for the CSM simulation is presented in figure 6.8. Note, that in comparison with the CFD

simulation, for the structural calculation it is sufficient to work with a relatively coarse mesh. Only the mesh for the tip and at the boundaries of the flaps are refined. The presented model contains about 12000 nodes and 4600 elements. In a next step, we had to export the forces from the TAU simulation, to apply them for the CSM simulation in CATIA. Therefore, we extract a force vector for each point on the CFD surface mesh. As the CFD surface mesh and the CSM surface mesh are not topologically identical, CATIA needs to interpolate the forces to fit the CSM mesh. With the given CSM mesh and the forces we can perform a CSM simulation to obtain the von Mises stresses and the displacements for each point. We then export the stresses and update the design table for the next iteration. For our setup, we chose 80 MPa as the maximal value for the von Mises stress. We choose the initial structural design table to be the minimal value of 0.001mm for each element. This corresponds to a minimal weight, but as the stresses are not bounded by 80 MPa, this is not a valid design.

In figure 6.12 we can see the thickness for all elements over the number of iterations and in figure 6.11 the corresponding stresses. After 30 iterations, the thicknesses of the elements is converged and the corresponding von Mises stresses are all bounded by the 80 MPa. In figure 6.12 we notice that for some parameters we reach the minimal thickness. The stresses for those stay below the maximal value for any feasible thickness. Therefore, the von Mises stresses do not converge towards 80MPa. For all other parameters the von Mises stresses converges to 80MPa. In figure 6.10 we can see that we start with the lowest possible weight. In the next iteration we observe a large overshoot of the estimated mass, but already after about 10 iterations the mass no longer varies and the converged wing has a mass of 295kg. Figure 6.9 shows the von Mises stresses for the converged structure. We can clearly see the underlying structure, as the stresses change rather rapidly at the boundaries between two elements. Note that, even though we perform roughly 60 CSM simulation, compared to the performed CFD simulation, the CSM simulation has a significant smaller wall-clock runtime in the order of 30-60 seconds. Using this process in an optimization, the procedure can be accelerated by taking the last structural design table as an initial guess for the CSM simulation in the next step.

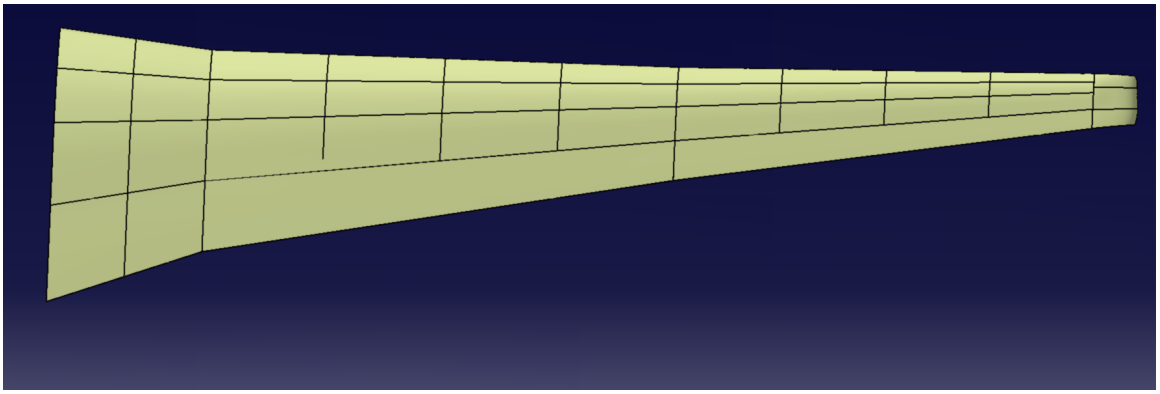


Figure 6.7: Structure of the geometry

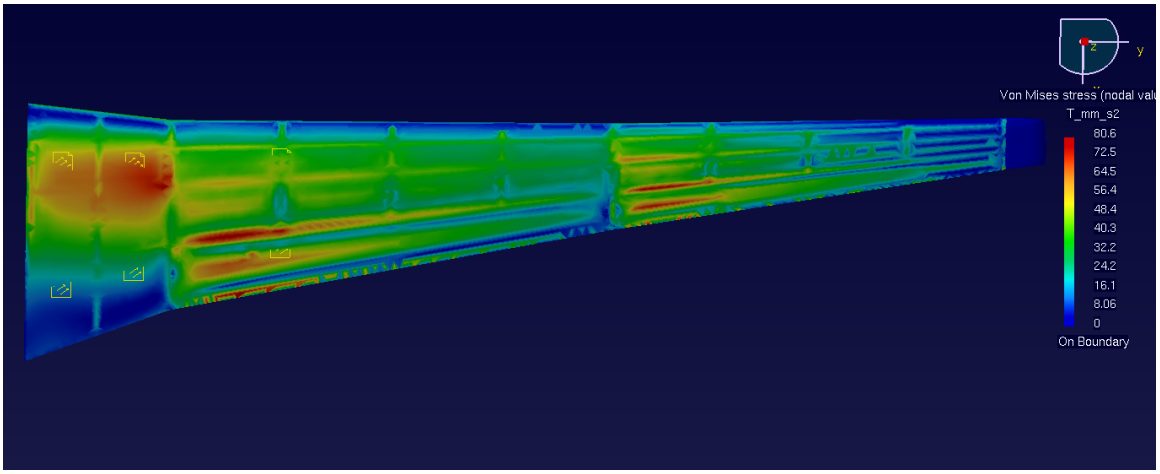


Figure 6.9: von Mises stresses for the converged structure

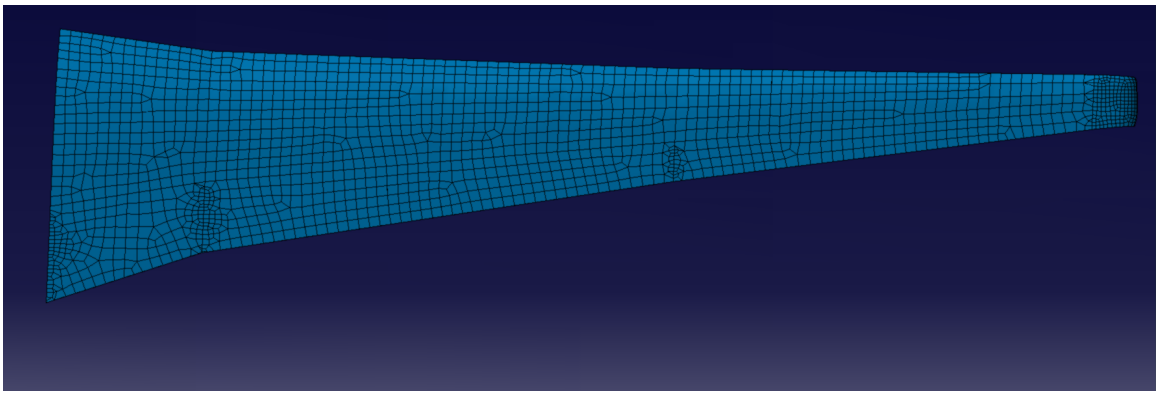
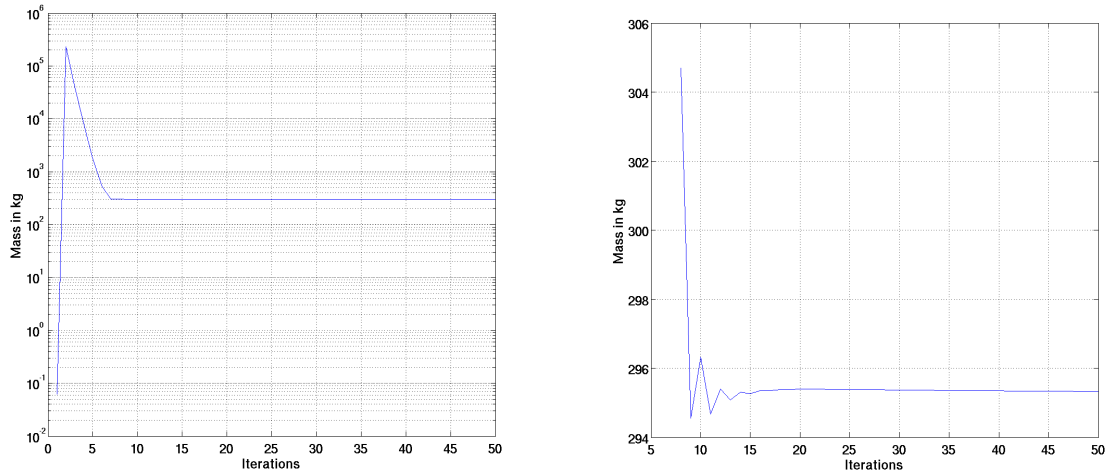


Figure 6.8: Mesh for the CSM simulation



a) All iterations

b) After initial over- and undershoot

Figure 6.10: Mass over iterations

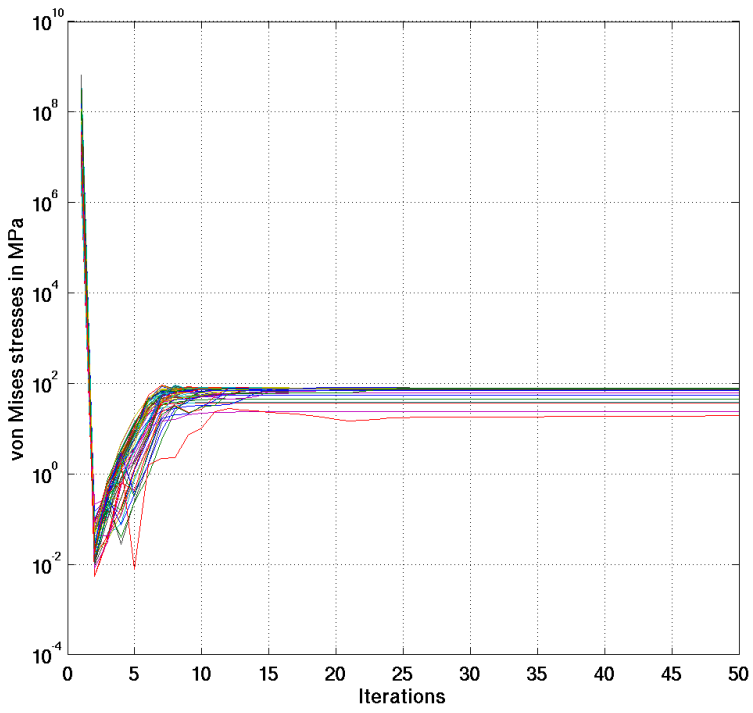


Figure 6.11: Stresses over iterations

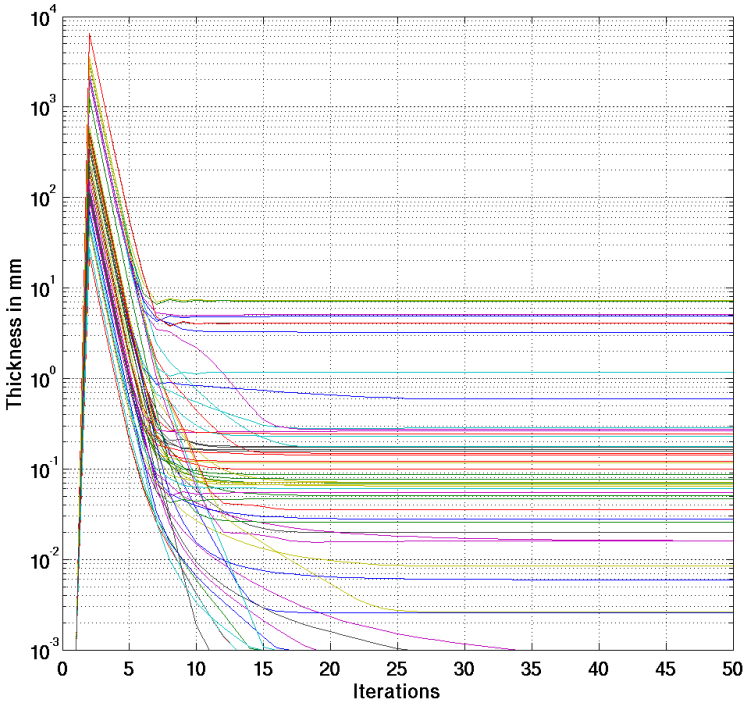


Figure 6.12: Thickness over iterations

Chapter 7

Summary and outlook

The work that is presented in this thesis shows the theoretical and practical steps for a multidisciplinary design optimization in the field of aerodynamics in an industrial environment. The adjoint approach has been derived in theory and applied in the optimization chain. A comparison with the finite difference approach ensured the quality of the gradients under certain assumptions. The results of the test case show, that the range of a complex wing geometry can be optimized. Furthermore, the existing optimization chain can be easily adapted to work on different geometries.

Furthermore, the mesh adjoint approach has been derived, implemented and tested against the existing process. Here we observed significant differenes in the obtained gradients. This leads to the conclusion, that the mesh adjoint approach is not applicable in its current development version. However, the optimization chain is fully funcional and can be tested again, with improved versions of the mesh adjoint implementation. Therefore, future versions of the mesh adjoint approach should be compared to the flow adjoint approach, to see whether the quality of the gradients has increased and makes the mesh adjoint approach applicable.

Additionally to the aerodynamic optimization, a structural optimization, called sizing, has been implemented and applied to the test case. The results show, that the weight can be successfully minimized, while the structural constraints are satisfied.

In the current implementation, the sizing is applied to the converged geometry of the aerodynamic optimization. In future versions, the structural optimization can replace the analytical formula which is currently used to estimate the wing weight. To accelerate the convergence behaviour, it is then possible to use the converged structure obtained by the structural optimization in the last iteration as an initial design for the next structural optimization.

In future applications, the optimization chain can be applied to more complex geometries with a higher number of design parameter. Given a sufficient quality of the mesh adjoint gradients, the use of the mesh adjoint should accelerate the overall runtime, as the costs of the volume mesh perturbation can become a bottleneck for a larger number of design variables and for meshes with a higher resolution.

Bibliography

[1] Caslav Ilic, Markus Widhalm, Joel Brezillon. Efficient polar optimization of transport aircraft in transonic rans flow using adjoint gradient based approach. *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 3–5, 2012.

[2] R. Dwight. Robust mesh deformation using the lienar elasticity equations. *EUROGEN*, 2005.

[3] Esteco. On modefrontier, `http://www.modefrontier.com/homeMF.html`, 2013.

[4] Deutsches Zentrum für Luft-& Raumfahrt. Description of the DLR TAU code. 2010.

[5] Nicolas Gauger. Topology optimization. *Lecture Script RWTH Aachen University*, 2012.

[6] Hänel. Computational fluid dynamics 1&2. *Lecture Script RWTH Aachen University*, 2010.

[7] Florian Jarre. On an approximation of the Hessian of the Lagrangian, `http://www.opt.uni-duesseldorf.de/~jarre/papers/lag_fin.pdf`.

[8] Ilan Kroo. Aircraft design: Synthesis and analysis.

[9] Ilan Kroo and Joaquim Martins. Optimization and MDO: Taxonomy and examples. *Research Consortium for Multidisciplinary System Design Workshop*, 2006.

[10] Luca Nardin. Multiobjective optimisation of aircraft systems, 2009-2010.

[11] H.J. Oberle. Optimierung. *Lecture Script University of Hamburg*, 2011.

[12] W. Schröder. *Fluidmechanik*. Aachener Beiträge zur Strömungsmechanik. Mainz, 2005.

[13] Caslav Ilic, Stephan Schmidt, Nicolas Gauger, Volker Schulz. Detailed aerodynamic shape optimization based on adjoint method with shape derivatives. *Fifth SIAM Workshop on Combinatorial Scientific Computing*, 2011.

[14] Dassault Systmes. About catia, `www.3ds.com`, 2013.

[15] Howard P. Buckley, Beckett Y. Zhou, David W. Zingg. Aifoil optimization using practical aerodynamic design requirements. *JOURNAL OF AIRCRAFT Vol. 47, No. 5, September*, 2010.

Erklärung

Hiermit erkläre ich, Thomas Camminady, an Eides statt, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Zitate und Quellen sind als solche kenntlich gemacht.

Thomas Camminady	Datum, Ort