European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

ADJOINT APPROACHES IN AERODYNAMIC SHAPE OPTIMIZATION AND MDO CONTEXT

Nicolas R. Gauger[*†]

[*]German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany
e-mail: Nicolas.Gauger@dlr.de
web page: http://www.dlr.de/as/
[†]Humboldt University Berlin, Department of Mathematics,
Unter den Linden 6, 10099 Berlin, Germany
e-mail: gauger@mathematik.hu-berlin.de

Key words: Aerodynamic Shape Optimization, Adjoint Approaches, MDO

Abstract. *Methods for aerodynamic shape optimization based on the calculation of the derivatives of the cost function with respect to the design variables suffer from the high computational costs if many design variables are used. However, these gradients can be efficiently obtained by solution of the adjoint flow equations, even for multidisciplinary optimizations.*

This paper has been also published in the Lecture Series 2006-03 (J. Periaux and H. Deconinck, eds.), VKI LS 2006-03 (ISBN 2-930389-65-6) and has been invited by the organizers J. Periaux and H. Deconinck of the STS 5: VKI course summary report on introductory optimization and multidisciplinary design with applications to aeronauts and turbomachinery.

1 INTRODUCTION

Because detailed aerodynamic shape optimizations still suffer from high computational costs, efficient optimization strategies are required. Regarding the deterministic optimization methods, the adjoint approach is seen as a promising alternative to the classical finite difference approach [15, 16]. With the adjoint approach the sensitivities needed for the aerodynamic shape optimization can be efficiently obtained by solution of the adjoint flow equations. Here, one is independent of the number of design variables with respect to the numerical costs for determining the sensitivities.

One distinguishes between continuous and discrete adjoint approaches. In the continuous case one formulates the optimality condition first, then derives the adjoint problem and finally does the discretization of the so obtained adjoint flow equations. On the other hand, in the discrete case one takes the discretized flow equations for the derivation of the

discrete adjoint problem. This can be automated by so called algorithmic differentiation (AD) tools.

The different adjoint approaches will be explained for single disciplinary aerodynamic shape optimization first and then their extension to MDO problems will be discussed for aero/structure cases. Finally, we will discuss the so-called one-shot methods. Here one breakes open the simulation loop for optimization.

All activities and results presented in the following sections are linked to the project MEGADESIGN [18] within the framework of the German aerospace research program. The main goal within this project is the development of efficient numerical methods for shape design and optimization.

2 NOMENCLATURE

$(x, y) \in \mathbb{R}^2$	cartesian coordinates	M_∞	Mach number
$(\xi, \eta) \in [0, 1]^2$	body fitted coordinates	$)_\infty$	... at free stream
$D \subset \mathbb{R}^2$	flow field domain	γ	ratio of specific heats
$\partial D = B \cup C$	flow field boundary	C_{ref}	cord length
$B = \{(\xi, 1)\}$	farfield	C_p	pressure coefficient
$C = \{(\xi, 0)\}$	solid wall	C_D	drag coefficient
$\vec{n} = \begin{pmatrix} n_x \\ n_y \end{pmatrix} \perp D$	outward pointing normal unit vector	C_L	lift coefficient
α	angle of attack	C_m	pitching moment coefficient
ρ	density	(x_m, y_m)	pitching moment's reference point
$\vec{v} = \begin{pmatrix} u \\ v \end{pmatrix}$	velocity	I	cost function
p	pressure	$-d(I)$	adjoint boundary condition's RHS on C
E	specific total energy	$X \in \mathbb{R}^n$	vector of design variables
H	total enthalpy	Z	displacement field

3 OPTIMIZATION CHAIN

In aerodynamic shape optimization a geometry is either given by a parameterization or can be changed by parameterized deformation. This means in detail that based on these parameters a shape can be built up or deformed by a design vector. Furthermore the obtained shape has some aerodynamic properties like the drag coefficient or pressure distribution. Therefore the task of the aerodynamic shape optimization is to optimize this design vector and its dependent shape for some aerodynamic cost function.

When optimizing, there must be some chain to calculate the cost function value at a given parameterization. This can be done by deforming a static initial shape or surface

mesh and its dependent computational grid based on the parameterization and afterwards evaluating the cost function. This procedure is visualized in Figure 1.

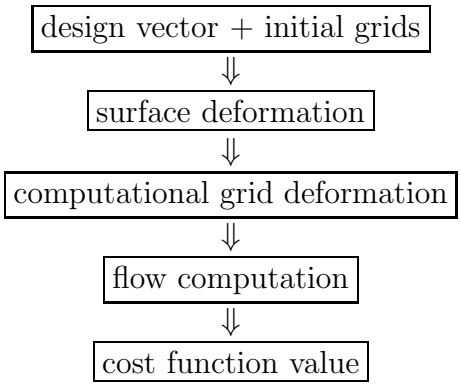

Figure 1: Cost function computation

3.1 Surface deformation

The basic idea for deforming the surface of an airfoil is to compute functions and adding their values to the upper and lower side of the surface. Therefore every design parameter is used to scale a specific function which is afterwards added to the shape. The result is a surface deformation which maintains the airfoil thickness. This maintaining of the airfoil thickness is needed to prevent the optimization to converge to a flat plate.

Several kinds of functions are considered for the deformation. The first are the Hicks-Henne functions which are defined as

$$h_{a,b} : [0,1] \to [0,1] : h_{a,b}(x) = (\sin(\pi x^{\frac{\log 0,5}{\log a}}))^b.$$

These functions have the positive property that they are defined in the interval $[0,1]$ and map to the interval $[0,1]$ where their peak is at position a. Furthermore they are analytically smooth at zero and one.

The used parameterization operates with Hicks-Henne functions with a fixed b of 3.0 and a varies from $\frac{3}{n+5}$ to $\frac{n+3}{n+5}$ where n is the number of design parameters.

The second considered kind of functions are transferred cosine functions. These cosine functions are defined for $q \in [0,1]$ as $c_q(x) : [0,1] \to [0,1]$ where

$$c_q(x) = \left\{ \begin{matrix} \frac{1}{2}(1-\cos(\frac{x}{q}\pi)) & \text{for} & x \leq 2q \\ 0 & \text{for} & x > 2q \end{matrix} \right\} \quad \text{for} \quad q \leq \frac{1}{2}$$

and

$$c_q(x) = \left\{ \begin{matrix} 0 & \text{for} & x < 2q-1 \\ \frac{1}{2}(1-\cos(\frac{x-2q+1}{1-q}\pi)) & \text{for} & x \geq 2q-1 \end{matrix} \right\} \quad \text{for} \quad q > \frac{1}{2}.$$

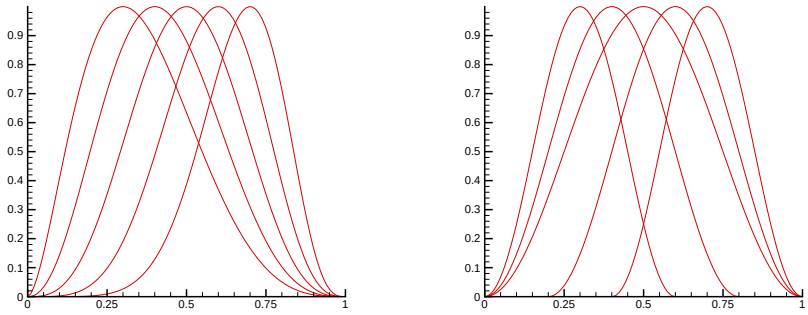

Figure 2: Hicks-Henne functions for $b = 3$ and $a = 0.3 - 0.7$ (left) and transformed cosine functions for $q = 0.3 - 0.7$ (right).

These functions have the positive property that their impact on the surface deformation is local because their support is $2 * \min(q, 1 - q)$. The used parameterization operates with these functions where q varies from $\frac{3}{n+5}$ to $\frac{n+3}{n+5}$.

3.2 Grid deformation

After having deformed the surface there is a need to deform the computational grid as well. This deformation should be related to the changes of the surface. Within the following work this is done via the volume spline method by Hounjet et al. (see [14]). This method is a general interpolation approach for n interpolation points (x_i, y_i, z_i) and their values $f_i (1 \leq i \leq n)$ which is given by

$$f(x, y, z) = \alpha_1 + \alpha_2 * x + \alpha_3 * y + \alpha_4 * z + \sum_{i=1}^{n} \beta_i * \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}. \quad (1)$$

The coefficients α_i and β_i can be determined by the condition that the interpolation f should be exact at its n interpolation points

$$f(x_i, y_i, z_i) = f_i \quad (1 \leq i \leq n)$$

and the four additional conditions

$$\begin{aligned}
\sum_{i=1}^{n} \beta_i &= 0, \\
\sum_{i=1}^{n} \beta_i * x_i &= 0, \\
\sum_{i=1}^{n} \beta_i * y_i &= 0, \\
\sum_{i=1}^{n} \beta_i * z_i &= 0,
\end{aligned}$$

which can be physically interpreted as equilibrium equations.

This results in solving the following linear system of equations

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 0 & x_1 & x_2 & \dots & x_n \\ 0 & 0 & 0 & 0 & y_1 & y_2 & \dots & y_n \\ 0 & 0 & 0 & 0 & z_1 & z_2 & \dots & z_n \\ 1 & x_1 & y_1 & z_1 & 0 & \epsilon_{12} & \dots & \epsilon_{1n} \\ 1 & x_2 & y_2 & z_2 & \epsilon_{21} & 0 & \dots & \epsilon_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_n & y_n & z_n & \epsilon_{n2} & \epsilon_{n2} & \dots & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix}$$

where $\epsilon_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ is the Euklidian distance between the interpolation points (x_i, y_i, z_i) and (x_j, y_j, z_j).

After solving this system of equations the interpolation is ready to be used with the given formula (1) for arbitrary points (x, y, z).

This general interpolation method is now applied to the differences of the original and deformed surface dx, dy, dz. These functions are each interpolated with the differences of the surfaces as interpolation points. Afterwards dx, dy, dz are applied to the computational grid and therefore yield a grid deformation.

Therefore, let $(x_{old,i}, y_{old,i}, z_{old,i})$ be the old and $(x_{new,i}, y_{new,i}, z_{new,i})$ be the new surface points $(1 \leq i \leq n)$. Then the functions dx, dy, dz can be interpolated with the interpolation point values

$$\begin{aligned}
dx_i &= x_{new,i} - x_{old,i}, \\
dy_i &= y_{new,i} - y_{old,i}, \\
dz_i &= z_{new,i} - z_{old,i}
\end{aligned}$$

at $(x_{old,i}, y_{old,i}, z_{old,i})$. These obtained functions dx, dy, dz can then be computed at arbitrary points. Now let $(a_{old,j}, b_{old,j}, c_{old,j})$ be the old computational grid points and $(a_{new,j}, b_{new,j}, c_{new,j})$ their corresponding new points $(1 \leq j \leq m)$. Finally, the grid deformation is given by

$$\begin{aligned} a_{new,j} &= a_{old,j} + dx(a_{old,j}, b_{old,j}, c_{old,j}), \\ b_{new,j} &= b_{old,j} + dy(a_{old,j}, b_{old,j}, c_{old,j}), \\ c_{new,j} &= c_{old,j} + dz(a_{old,j}, b_{old,j}, c_{old,j}). \end{aligned}$$

Instead of deforming the computational grid there is also the possibility to generate a new grid at each optimization step. But this results in a computation overhead because grid generation is expensive. Therefore the present work uses the above explained volume spline interpolation method to deform the grid and save computation time.

4 GRADIENT-BASED AERODYNAMIC SHAPE OPTIMIZATION

For convenience reasons, the following analysis is restricted to the 2D Euler equations. Let $X \in \mathbb{R}^n$ denote the vector of design variables. Then X determines the airfoil $C(X)$ and its physics $w(X)$, where $w = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}$ is the vector of the conserved variables. w is assumed to be the solution of the quasi-unsteady Euler equations

$$\frac{\partial w}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \qquad \text{in } D, \tag{2}$$

where $\vec{n}^\top \vec{v} = 0$ on $C = C(X)$, with $f = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}$ and $g = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{pmatrix}$. On the farfield free stream conditions are assumed. For a perfect gas

$$p = (\gamma - 1)\rho(E - \frac{1}{2}(u^2 + v^2)) \tag{3}$$

holds for the pressure, and finally C_p, C_D, C_L and C_m are defined as

$$C_p := \frac{2(p - p_\infty)}{\gamma M_\infty^2 p_\infty} \ , \tag{4}$$

$$C_D := \frac{1}{C_{\text{ref}}} \int_C C_p(n_x \cos\alpha + n_y \sin\alpha) dl \ , \tag{5}$$

$$C_L := \frac{1}{C_{\text{ref}}} \int_C C_p(n_y \cos\alpha - n_x \sin\alpha) dl \ , \tag{6}$$

$$C_m := \frac{1}{C_{\text{ref}}^2} \int_C C_p(n_y(x-x_m) - n_x(y-y_m))\ dl\ . \tag{7}$$

If the geometry is now perturbed from $C(X)$ to $C(X+\delta X)$, then via the solution of

$$\begin{aligned} & \frac{\partial(w+\delta w)}{\partial t} + \frac{\partial(f+\delta f)}{\partial x} + \frac{\partial(g+\delta g)}{\partial y} = 0 \\ \Leftrightarrow \quad & \frac{\partial(\delta w)}{\partial t} + \frac{\partial(\delta f)}{\partial x} + \frac{\partial(\delta g)}{\partial y} = 0 \qquad \text{in } D, \end{aligned} \tag{8}$$

where

$$\vec{n}^\top \vec{v} = 0 \ \text{ on } \ C = C(X+\delta X)\ , \tag{9}$$

the associated variation of pressure is as follows

$$\delta C_p = \frac{2\delta p}{\gamma M_\infty^2 p_\infty} \approx \frac{2(p(X+\delta X) - p(X))}{\gamma M_\infty^2 p_\infty}\ . \tag{10}$$

Finally via

$$\delta n_x \approx n_x(X+\delta X) - n_x(X) \tag{11}$$

and

$$\delta n_y \approx n_y(X+\delta X) - n_y(X)\ , \tag{12}$$

the variations of C_D, C_L and C_m are obtained as

$$\begin{aligned} \delta C_D \ &= \ \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}} \int_C \delta p(n_x \cos\alpha + n_y \sin\alpha) dl \\ & \quad + \frac{1}{C_{\text{ref}}} \int_C C_p(\delta n_x \cos\alpha + \delta n_y \sin\alpha) dl\ , \end{aligned} \tag{13}$$

$$\begin{aligned} \delta C_L \ &= \ \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}} \int_C \delta p(n_y \cos\alpha - n_x \sin\alpha) dl \\ & \quad + \frac{1}{C_{\text{ref}}} \int_C C_p(\delta n_y \cos\alpha - \delta n_x \sin\alpha) dl\ , \end{aligned} \tag{14}$$

$$\begin{aligned} \delta C_m \ &= \ \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}^2} \int_C \delta p(n_y(x-x_m) - n_x(y-y_m))\ dl \\ & \quad + \frac{1}{C_{\text{ref}}^2} \int_C C_p \delta(n_y(x-x_m) - n_x(y-y_m))\ dl\ . \end{aligned} \tag{15}$$

Proceeding as described above for the n perturbations $\delta_i X$ in each of the n components of the design vector X, the gradient of the cost function I (e.g. drag, lift or pitching moment coefficients) is obtained as $\nabla_X I = (\delta_i I)_{i=1,\ldots,n}$ after $n+1$ flow calculations. The easiest

gradient-based optimization strategy is the steepest descent method. There a recursive line search in the direction $-\nabla_{X^{(k)}} I$, starting from the point $X^{(k)}$, leads to an optimal geometry

$$X^{(k+1)} = X^{(k)} - \varepsilon^{(k)} \nabla_{X^{(k)}} I \tag{16}$$

with respect to the cost function I in that direction. This is repeated until the norm of the gradient of the cost function becomes zero.

But one can see that the numerical costs for the determination of the gradient of the cost function, are directly proportional to the number of design variables. This finite differences or brute force approach becomes more and more inefficient as the number of design variables increases.

5 SENSITIVITY COMPUTATIONS

In aerodynamic shape optimization, the task of computing sensitivities is very important in order to have the possibility to use gradient based optimization strategies. Gradient computations for a given cost function $I(X)$ for a design vector X out of a defined design space can generally be done with several methods.

5.1 Finite difference method

The first is the finite difference method (FD) which approximates the gradient as follows

$$\frac{\partial I}{\partial x_i}(X) \approx \frac{I(X + h * e_i) - I(X)}{h} \quad (1 \leq i \leq n) \tag{17}$$

where n is the number of design parameters, e_i the ith unit vector, and h is the scalar step size. Problems with this method occur if the computation of the cost function is extremely expensive. As can be seen in the approximation (17), this cost function has to be calculated once at point X and further n times at $(X + h * e_i)$ for $1 \leq i \leq n$. This results in $(n + 1)$ cost function evaluations which can take a long time. Another problem may occur if the step size h is not accurately chosen. This is based on the fact that

$$I(X + h * e_i) = I(X) + \frac{\partial I}{\partial x_i}(X) * h + \frac{\partial^2 I}{\partial^2 x_i}(X) * h^2 + O(h^3)$$

which can be transformed into

$$\frac{\partial I}{\partial x_i}(X) = \frac{I(X + h * e_i) - I(X)}{h} - \frac{\partial^2 I}{\partial^2 x_i}(X) * h + O(h^2). \tag{18}$$

If h is chosen too large, the first order term on the right side of equation (18) would have a large impact on the quality of the approximation in (17). This means on the one hand, that h has to be chosen relatively small. On the other hand, h can not be chosen arbitrarily small, because of numerical stability. This is based on the division in the approximation term (17) which will be error intensive if h is too small and therefore result in numerical noise. Therefore the step length h has to be manually tuned with respect to the cost function, parameterization and used geometry.

5.2 Continuous adjoint formulation

The second method to compute sensitivities is the continuous adjoint approach. Just for convenience reasons, the following analysis is restricted to the 2D Euler equations. In order to determine the gradient of the cost function independently of the design variables with respect to the numerical costs one can use the following continuous adjoint formulation. Let $\psi = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix}$ denote the vector of the adjoint variables. Instead of solving $n+1$ times the quasi-unsteady Euler equations to get the gradient, the Euler equations are solved just once in order to get the transposed Jacobians $\left(\frac{\partial f}{\partial w}\right)^\top$, $\left(\frac{\partial g}{\partial w}\right)^\top$ and then the quasi-unsteady continuous adjoint Euler equations

$$-\frac{\partial \psi}{\partial t} - \left(\frac{\partial f}{\partial w}\right)^\top \frac{\partial \psi}{\partial x} - \left(\frac{\partial g}{\partial w}\right)^\top \frac{\partial \psi}{\partial y} = 0 \qquad \text{in } D, \tag{19}$$

where

$$n_x\psi_2 + n_y\psi_3 = -d(I) \ \text{ on } \ C = C(X) \ , \tag{20}$$

and

$$\delta x_\xi, \ldots, \delta y_\eta = 0, \ \delta w = 0 \ \text{ on } \ B = B(X) \ , \tag{21}$$

are also solved just once.

The right hand side $-d(I)$ of the wall boundary condition of the quasi-unsteady adjoint Euler equations is dependent on the cost function I. The adjoint farfield boundary condition describes just that the geometrical position of the farfield is fixed and free stream conditions apply there.

Finally the components of the gradient $\nabla_X I = (\delta_i I)_{i=1,\ldots,n}$ can now be determined via an integration just over the adjoint solution and the metric sensitivities $\delta x_\xi, \ldots, \delta y_\eta$ and

$$\begin{aligned} \delta I \ &= \ -\int_C p(-\psi_2\delta y_\xi + \psi_3\delta x_\xi)\ dl + K(I) \\ &\quad -\int_D \psi_\xi^\top(\delta y_\eta f - \delta x_\eta g) + \psi_\eta^\top(-\delta y_\xi f + \delta x_\xi g)\ dA \end{aligned} \tag{22}$$

is obtained, where $K(I)$ is again a term dependent on the cost function I.

For the gradient of the drag, the following right hand side adjoint boundary on C is used

$$d(C_D) = \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}}(n_x\cos\alpha + n_y\sin\alpha) \tag{23}$$

and to get the corresponding gradient, $K(I)$ is

$$K(C_D) = \frac{1}{C_{\text{ref}}}\int_C C_p(\delta n_x\cos\alpha + \delta n_y\sin\alpha)dl \ , \tag{24}$$

for the gradient of the lift

$$d(C_L) = \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}}(n_y \cos\alpha - n_x \sin\alpha) \tag{25}$$

and

$$K(C_L) = \frac{1}{C_{\text{ref}}} \int_C C_p(\delta n_y \cos\alpha - \delta n_x \sin\alpha) dl \tag{26}$$

are used, and for the gradient of the pitching moment

$$d(C_m) = \frac{2}{\gamma M_\infty^2 p_\infty C_{\text{ref}}^2}(n_y(x - x_m) - n_x(y - y_m)) \tag{27}$$

and

$$K(C_m) = \frac{1}{C_{\text{ref}}^2} \int_C C_p \delta(n_y(x - x_m) - n_x(y - y_m))\ dl \tag{28}$$

are used. For more details see [6] or [5].

5.3 Algorithmic differentiation (AD)

A third method is a discrete approach which is usually called algorithmic or automatic differentiation (AD) and depends on a specific implementation of the cost function. This implementation consists of various elemental operations (like $+, -, *$) which build up the cost function as their concatenation. Therefore, applying the chain rule to this concatenation results in a differentiation of the cost function (after having dealt with possible inconsistencies and other problems which are beyond the scope of this lecture, see [9] for detailed information).

To give the reader a feeling of the principal ideas of AD we will introduce the two basic concepts with the help of a simple example. Let f be a cost function which depends on two input parameters x_1 and x_2 which is given by

$$f(x_1, x_2) = \sin(x_1/x_2) + x_1/x_2.$$

We now wish to compute the value of $y = f(1.5, 0.5)$ and its derivative by AD. Then a possible evaluation trace is given in Table 1.

The first possibility to apply the chain rule is to differentiate every single operation in the order of the evaluation trace. Let us suppose we want to differentiate the output variable y with respect to x_1. Then we associate with every variable v_i of the evaluation trace another variable $\dot{v}_i = \partial v_i/\partial x_1$. Applying the chain rule to each line in the evaluation trace, in order, leads to a numeric value of $\dot{y}$ which is the wanted sensitivity of y with respect to x_1. Clearly, $\dot{v}_{-1} = \partial v_{-1}/\partial x_1 = 1.0$ and $\dot{v}_0 = \partial v_0/\partial x_1 = 0.0$. Augmenting the evaluation trace of Table 1 gives the derived trace in Table 2. The total floating point operation count of the added lines to evaluate $\partial y/\partial x_1$ is a small multiple of that for the underlying code to evaluate y.

$v_{-1} =$	x_1	$= 1.5$	
$v_0 =$	x_2	$= 0.5$	
$v_1 =$	v_{-1}/v_0	$= 1.5/0.5$	$= 3.0000$
$v_2 =$	$\sin(v_1)$	$= \sin(3.0)$	$= 0.1411$
$v_3 =$	$v_1 + v_2$	$= 3.0 + 0.1411$	$= 3.1411$
$y =$	v_3	$= 3.1411$	

Table 1: An evaluation trace of the simple example.

$v_{-1} =$	x_1	$= 1.5$	
$\dot{v}_{-1} =$	$\dot{x}_1$	$= 1.0$	
$v_0 =$	x_2	$= 0.5$	
$\dot{v}_0 =$	$\dot{x}_2$	$= 0.0$	
$v_1 =$	v_{-1}/v_0	$= 1.5/0.5$	$= 3.0000$
$\dot{v}_1 =$	$\dot{v}_{-1}/v_0 - v_{-1} * \dot{v}_0/v_0/v_0$		
$=$	$(\dot{v}_{-1} - v_1 * \dot{v}_0)/v_0$	$= (1.0 - 3.0 * 0.0)/0.5$	$= 2.0000$
$v_2 =$	$\sin(v_1)$	$= \sin(3.0)$	$= 0.1411$
$\dot{v}_2 =$	$\cos(v_1) * \dot{v}_1$	$= (-0.99) * 2.0$	$= -1.9800$
$v_3 =$	$v_1 + v_2$	$= 3.0 + 0.1411$	$= 3.1411$
$\dot{v}_3 =$	$\dot{v}_1 + \dot{v}_2$	$= 2.0 - 1.98$	$= 0.0200$
$y =$	v_3	$= 3.1411$	
$\dot{y} =$	$\dot{v}_3$	$= 0.0200$	

Table 2: Forward differentiated evaluation trace.

Exactly the same code can be used to evaluate $\partial y/\partial x_2$ as well; the only change is to set $\dot{x}_1 = 0.0$ and $\dot{x}_2 = 1.0$ at the beginning.

The second possibility is to apply the chain rule in reverse order and is called the reverse mode. This concept can be seen as a discrete adjoint approach. Therefore we associate for every v_i another variable $\overline{v}_i = \partial y/\partial v_i$ called the adjoint variable. By definition $\overline{y} = 1.0$ and since the only ways in which v_1 can affect y are via the definitions $v_2 = \sin(v_1)$ and $v_3 = v_1 + v_2$ it is

$$\begin{aligned} \overline{v}_1 = \frac{\partial y}{\partial v_1} &= \frac{\partial y}{\partial v_3} * \frac{\partial v_3}{\partial v_1} && (29) \\ &= \overline{v}_3 * \frac{\partial (v_1 + v_2)}{\partial v_1} \\ &= \overline{v}_3 * \frac{\partial v_1}{\partial v_1} + \overline{v}_3 * \frac{\partial v_2}{\partial v_1} && (30) \\ &= \overline{v}_3 * (1 + \frac{\partial (\sin v_1)}{\partial v_1}) \\ &= \overline{v}_3 + \overline{v}_3 * \cos(v_1). && (31) \end{aligned}$$

This can also be evaluated by the iterative equations

$$\begin{aligned} \overline{v}_1 &= \overline{v}_3 \\ \overline{v}_2 &= \overline{v}_3 \\ \overline{v}_1 &= \overline{v}_1 + \overline{v}_2 * \cos(v_1). \end{aligned}$$

Thus applying the chain rule to every line in the evaluation trace of Table 1 we obtain the reverse differentiated code in Table 3. Note that the adjoint statements are lined up vertically underneath the original statements that spawned them.

1	$v_{-1} = x_1 = 1.5$
2	$v_0 = x_2 = 0.5$
3	$v_1 = v_{-1}/v_0 = 1.5/0.5 = 3.0$
4	$v_2 = \sin(v_1) = \sin(3.0) = 0.1411$
5	$v_3 = v_1 + v_2 = 3.0 + 0.1411 = 3.1411$
6	$y = v_3 = 3.1411$
7	$\overline{v}_3 = \overline{y} = 1.0$
8	$\overline{v}_1 = \overline{v}_3 = 1.0$
9	$\overline{v}_2 = \overline{v}_3 = 1.0$
10	$\overline{v}_1 = \overline{v}_1 + \overline{v}_2 * \cos(v_1) = 1.0 + 1.0 * \cos(3.0) = 0.01$
11	$\overline{v}_0 = -\overline{v}_1 * v_1/v_0 = -0.01 * 3.0/0.5 = -0.06$
12	$\overline{v}_{-1} = \overline{v}_1/v_0 = 0.01/0.5 = 0.02$
13	$\overline{x}_2 = \overline{v}_0 = -0.06$
14	$\overline{x}_1 = \overline{v}_{-1} = 0.02$

Table 3: Reverse differentiated evaluation trace.

Note also that line 7 of Table 3 belongs to the expansion of Equation (29), lines 8 and 9 to the expansion of Equation (30) and line 10 to the expansion of Equation (31).

As with the forward propagation method, the floating point operation count of the added lines is a small multiple of that for the underlying code to evaluate y. But this time the complete gradient has been computed.

The main advantage of the automatic differentiation method over the above two mentioned methods is that gradients can be computed with best possible accuracy. In forward mode the gradient computation speed is dependent on the number of design variables which can be expensive if the evaluation trace for the cost function is long. In reverse mode the gradient computation is independent on any input and therefore very efficient if many design variables are needed.

There are mainly two possible implementations of AD methods. The first is the so called source to source which means that the primal evaluation trace is transferred into a

differentiated evaluation trace. The second is based on operator overloading which only yields an executable.

However, the problem in reverse mode for both implementation strategies is that primal computation informations have to be recomputed and/or stored to compute backwards again. This results in dependence on the chosen approach in a need for much memory.

For more information, including vector valued functions, for multiple output and multiple input variables, please refer to [9].

6 ADJOINT FLOW SOLVERS

6.1 Continuous adjoint flow solvers

Within the MEGAFLOW project [17] an adjoint solver following the continuous adjoint formulation has been developed and widely validated for the block-structured flow solver FLOWer [6, 5]. The adjoint solver, which was implemented by hand, can deal with the boundary conditions for drag, lift and pitching-moment sensitivities. The adjoint option of the FLOWer code is validated for several 2D as well as 3D optimization problems [7, 2] controlled by the (adjoint) Euler equations. Within MEGADESIGN the robustness and efficiency of the adjoint solver will be improved, especially for the Navier-Stokes equations. In case of Navier-Stokes applications, currently the turbulence model is frozen in the adjoint mode. It is planned to make use of Algorithmic Differentiation (AD) in order to create adjoint turbulence models, which will then be linked to the hand coded adjoint solver.

Furthermore, it is planned to transfer the adjoint solver implemented in FLOWer, to the unstructured Navier-Stokes solver TAU. Here, the implementation work is already completed and validated for the inviscid adjoint solver [21] (see also Figure 3).

6.2 Discrete adjoint flow solvers

In addition to the continuous one, a discrete adjoint flow solver has been developed by hand within the unstructured Navier-Stokes solver TAU [1]. The implementation consists of the explicit construction of the exact Jacobian of the spatial discretization with respect to the unknown variables allowing the adjoint equations to be formulated and solved. Different spatial discretizations available in TAU have been differentiated, including the Spalart-Almaras-Edwards one-equation, and the Wilcox k-omega two-equation turbulence models.

For both solvers, FLOWer as well as TAU, first activities are launched for the automated generation of discrete adjoint solvers by the use of AD tools. For the FLOWer code the AD tool TAF [8] is used and ADOL-C [10] for the TAU code.

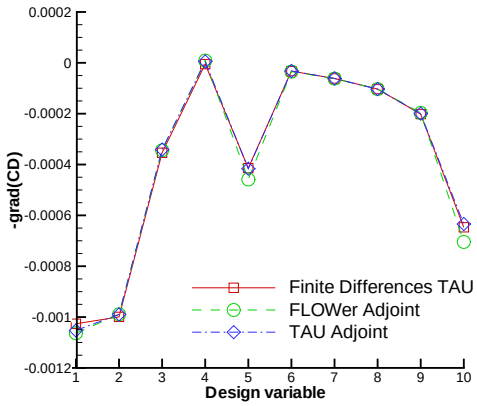

Figure 3: Gradient of the drag computed for 20 B-spline variables by finite differences with TAU and by the continuous adjoint approach with FLOWer and TAU. (RAE 2822, $M_\infty = 0.73$ and $\alpha = 2.0°$)

7 AUTOMATIC DIFFERENTIATION APPLIED TO AN ENTIRE OPTIMIZATION CHAIN

For the surface deformation the tool defgeo has been used. This tool has been implemented in order to compute deformations based on Hicks-Henne as well as cosine functions.

The grid deformation within our optimization chain is done by a tool named meshdefo. This tool uses a public domain linear equations solver to compute the above mentioned coefficients of the interpolation.

For the optimizations presented in this section, DLR's flow solver TAUij is used. TAUij solves the 2D Euler equations on structured grids with a standard central Jameson-Schmidt-Turkel scheme. Moreover TAUij provides the dimensionless lift and drag coefficients.

To compute the difference vectors of the original to the transformed shape geometry, another program named difgeo had to be implemented.

The chain to compute the cost function value is visualized in Figure 4. This entire optimization chain has been differentiated by the Dresden University of Technology with the use of ADOL-C (see [10]) which operates in reverse mode based on operator overloading. This differentiated chain can be written as

$$\frac{\partial \mathrm{drag}}{\partial \mathrm{p}} = \frac{\partial \mathrm{drag}}{\partial \mathrm{m}} \cdot \frac{\partial \mathrm{m}}{\partial \mathrm{dx}} \cdot \frac{\partial \mathrm{dx}}{\partial \mathrm{x}} \cdot \frac{\partial \mathrm{x}}{\partial \mathrm{p}}.$$

Note that the first term on the right side corresponds to the differentiation of TAUij, the second term to the differentiation of meshdefo, the third term to the differentiation of difgeo and the last term to the differentiation of defgeo. Since difgeo computes only the

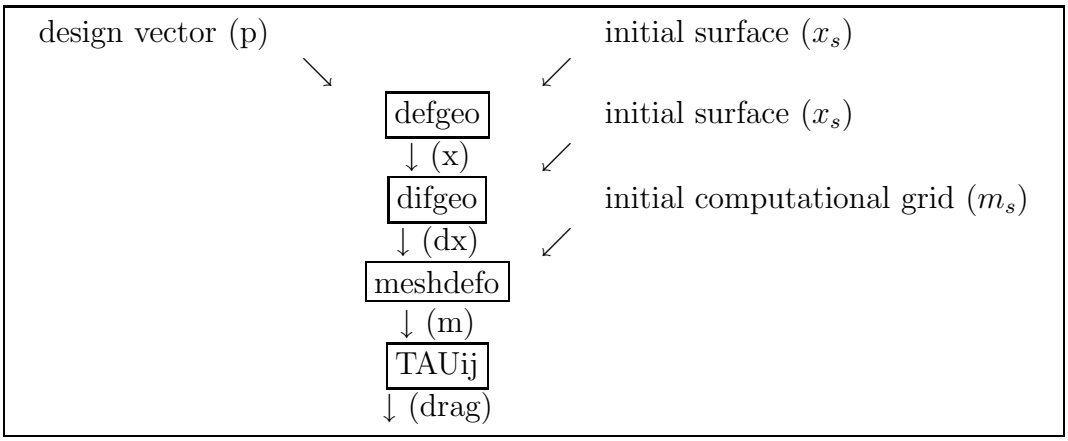

Figure 4: Chain to compute the cost function value.

differences $dx = x - x_s$ and x_s is a static initial surface, its corresponding factor becomes the unit matrix and therefore

$$\frac{\partial \text{drag}}{\partial \text{p}} = \frac{\partial \text{drag}}{\partial \text{m}} \cdot \frac{\partial \text{m}}{\partial \text{dx}} \cdot \frac{\partial \text{x}}{\partial \text{p}}. \tag{32}$$

The optimization strategy in the following computations is a steepest descent method which was implemented as an optimizer into the optimization framework Synaps Pointer Pro. This framework has the possibility to read in user defined gradients. Therefore, the gradients are calculated by separate routines and are then submitted to the optimizer.

7.1 Test case definition

As test case for the validation and application of AD generated adjoint sensitivity calculations an RAE2822 airfoil is chosen with a Mach number of 0.73 and an angle of attack of 2°. The drag coefficient for this test case has been optimized with both parameterizations, Hicks-Henne and cosine function parameterizations (see subsection 3.1). In both optimizations 20 design parameters have been used. The computational grid has 161x33 grid points.

7.2 Finite differences

To compute the finite differences in order to have a validation framework for the AD sensitivities, the first task was to tune the stepsize h for the approximation

$$\frac{\partial I}{\partial x_i}(X) \approx \frac{I(X + h * e_i) - I(X)}{h} \quad (1 \leq i \leq n) \tag{33}$$

as mentioned in section 5. Therefore the quotients of both parameterizations have been calculated for varying stepsizes with respect to all $n = 20$ parameters.

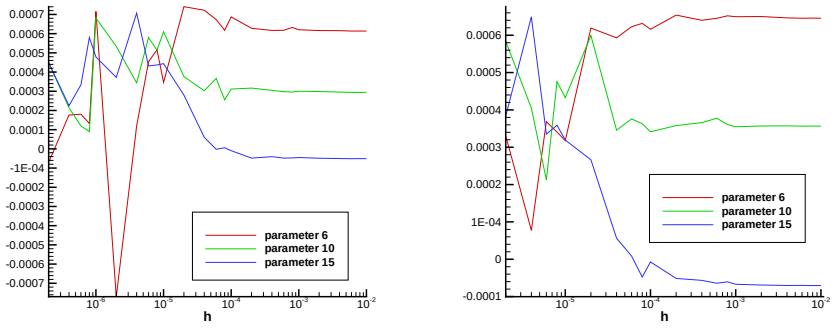

Figure 5: Quotients of Hicks-Henne (left) and cosine (right) functions parameterization for parameters 6, 10 and 15 and varying stepsizes.

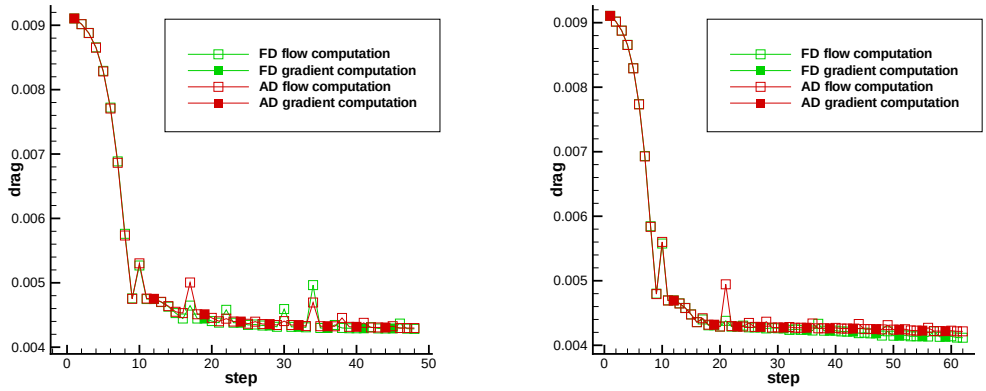

Figure 6: Optimization history of FD and AD for Hicks-Henne (left) and cosine (right) functions parameterization.

As can be seen in Figure 5 a good choice for the stepsize h is $10^{(-3)}$ for both parameterizations. Another possibility which has not been used is to tune the stepsize for each parameter seperately, which means selecting n stepsizes h_i for every approximation in (33). With this possibility a more accurate result might be achieved for the original airfoil, but based on the fact that this tuning cannot be done for every optimization step due to the high computational effort, it might cause worse optimization results at the end. Therefore a stepsize of $10^{(-3)}$ has been used for all gradient computations within the optimizations for both parameterizations.

In Figure 6 the optimization history for both parameterizations can be seen. In case of the Hicks-Henne functions parameterization the optimization converges after nine gradient computations which are marked by a filled out green square. The optimization with cosine functions converges after 13 gradient computations. The pressure distribution for both

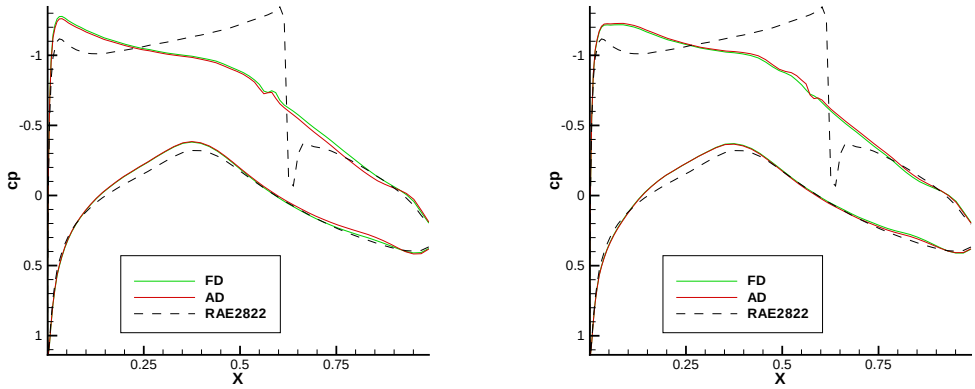

Figure 7: Pressure distribution of the original test case and the optimum of FD and AD for Hicks-Henne (left) and cosine (right) functions parameterization.

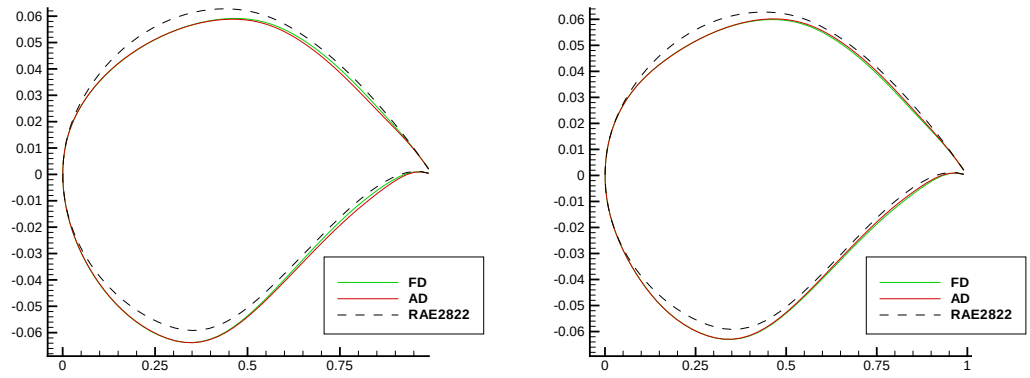

Figure 8: Surface geometry of the original test case and the optimum of FD and AD for Hicks-Henne (left) and cosine (right) functions parameterization.

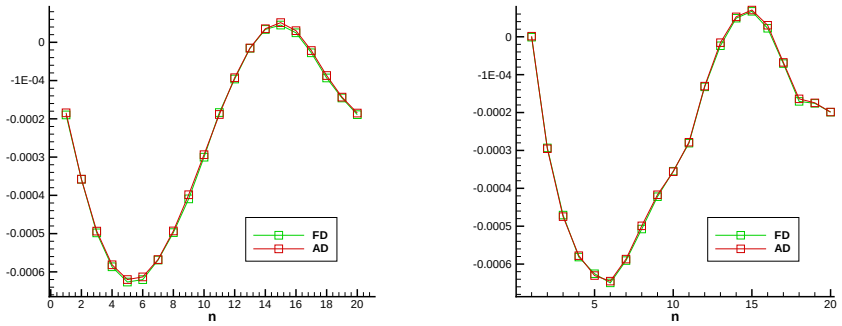

Figure 9: Comparison between FD and AD gradients on RAE2822 with Hicks-Henne (left) and cosine (right) functions parameterization.

optimizations is drawn in Figure 7 and the optimal geometries can be found in Figure 8.

As one can see, the strong shock of the original baseline geometry nearly vanished in both cases and one ends up with more than 50 % decrease in drag. In contrast to the optimum with the help of Hicks-Henne functions, the optimum with cosine functions parameterization shows slight oscillations in the pressure distribution, which is due to the fact that cosine functions only have local impacts on the surface deformation whereas Hicks-Henne functions have global impacts.

7.3 Automatic differentiation

In Figure 9 one can see the comparisons between the FD and AD gradients for the original RAE2822 airfoil with Hicks-Henne and cosine functions parameterization. This validates the AD approach and shows also that the chosen stepsize for the FD gradient is accurate enough.

As with the finite differences method the optimizations have also been done with the automatic differentiation approach. The optimization histories, the pressure distributions and the surface geometries of the optimum with AD are also visualized in Figures 6, 7 and 8. This clearly validates the automatic differentiation approach.

8 ADJOINT APPROACH FOR AERO/STRUCTURE COUPLING

The use of successively performed single disciplinary optimizations in case of a multi-disciplinary optimization problem is not only inefficient but in some cases has been shown to lead to wrong, non-optimal designs [19]. Although multidisciplinary optimization is possible with classical approaches for sensitivity evaluation by means of finite differences, this method is extremely expensive in terms of calculation time, requiring the reiterated solution of the coupled problem for every design variable.

A new approach that allows the evaluation of the gradient with low computational cost takes advantage of the adjoint formulation of the multidisciplinary optimization problem

[19, 20]. Therefore, the FLOWer adjoint option has been coupled with the structure solver MSC Nastran for an efficient coupled aero-structure adjoint solver. This approach, its implementation and validation is described in detail in [3, 4].

8.1 Adjoint formulation for aero/structure coupling

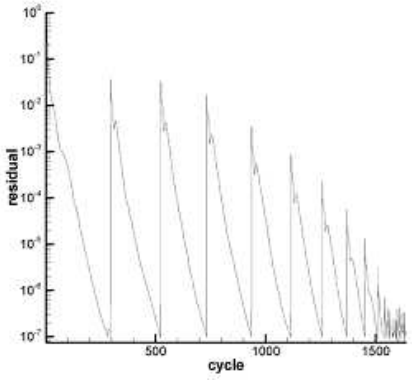

Figure 10: Plot of residual (log scale) of flow equation during coupled computation (multigrid is used): AMP wing, $M_\infty = 0.78$, $\alpha = 2.83°$, 2-block structured grid of about 140.000 nodes each.

The derivation of the adjoint equations in case of a multidisciplinary problem is similar to what has been carried out for the pure aerodynamic case, with the difference that we will end up with a dual adjoint variable for each set of state variables of the problem. An adjoint formulation is possible for any problem involving the calculation of the gradient of a function of one or more sets of variables obeying one or more constraint equations. We will restrict ourselves to the case of two sets: one represents the flow variables, the other the structure nodal displacement. As already seen $I(X, w, Z)$ denotes the cost function of the optimization problem, dependent now also on the displacement field Z, the solution of the structural problem. Then, the gradient takes the form

$$\nabla I = \frac{dI}{dX} = \frac{\partial I}{\partial X} + \frac{\partial I}{\partial w}\frac{\partial w}{\partial X} + \frac{\partial I}{\partial Z}\frac{\partial Z}{\partial X}, \tag{34}$$

or , in terms of differentials

$$dI = \frac{\partial I}{\partial X}\delta X + \frac{\partial I}{\partial w}\delta w + \frac{\partial I}{\partial Z}\delta Z. \tag{35}$$

The fields (w, Z) are the solution of the system of partial differential equations

$$R(X, w, Z) = 0, \tag{36}$$

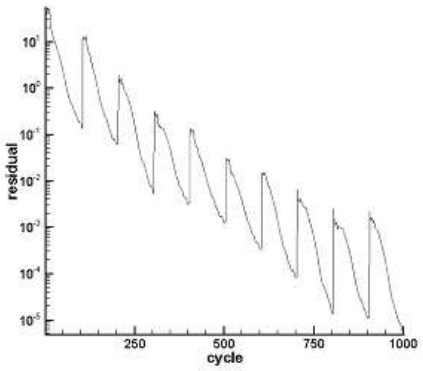

Figure 11: Plot of residual (log scale) of adjoint flow equation during coupled computation: AMP wing, $M_\infty = 0.78$, $\alpha = 2.83°$, 2-block structured grid of about 140.000 nodes each. Each 100 iterations, the boundary conditions of the adjoint flow solver are updated.

$$S(X, w, Z) = 0, \tag{37}$$

being (36) the flow and (37) the structural equations. We take the first variation of the PDEs. This yields

$$\delta R = \frac{\partial R}{\partial X}\delta X + \frac{\partial R}{\partial w}\delta w + \frac{\partial R}{\partial Z}\delta Z = 0, \tag{38}$$

$$\delta S = \frac{\partial S}{\partial X}\delta X + \frac{\partial S}{\partial w}\delta w + \frac{\partial S}{\partial Z}\delta Z = 0. \tag{39}$$

We multiply Equations (38) and (39) with the Lagrange multipliers ψ and ϕ respectively and add the result to the expression for the differential increment of I in terms of the differentials of the independent set(X, w, Z), obtaining

$$\begin{aligned} dI &= \left(\frac{\partial I}{\partial X} + \psi^T\frac{\partial R}{\partial X} + \phi^T\frac{\partial S}{\partial X}\right)\delta X \\ &+ \left(\frac{\partial I}{\partial w} + \psi^T\frac{\partial R}{\partial w} + \phi^T\frac{\partial S}{\partial w}\right)\delta w + \left(\frac{\partial I}{\partial Z} + \psi^T\frac{\partial R}{\partial Z} + \phi^T\frac{\partial S}{\partial Z}\right)\delta Z. \end{aligned} \tag{40}$$

Since we want to avoid recalculation of the (w, Z) fields, we cancel the terms in δw and δZ from dI by imposing the fields ϕ and ψ, to be the solution of the equations

$$\left(\frac{\partial I}{\partial w} + \psi^T\frac{\partial R}{\partial w} + \phi^T\frac{\partial S}{\partial w}\right) = 0, \tag{41}$$

$$\left(\frac{\partial I}{\partial Z} + \psi^T\frac{\partial R}{\partial Z} + \phi^T\frac{\partial S}{\partial Z}\right) = 0. \tag{42}$$

Figure 12: Wing structure model

These are the adjoint equations for the problem of coupled aeroelasticity. After their solution, the gradient can be recovered from the expression

$$dI = \left(\frac{\partial I}{\partial X} + \psi^T \frac{\partial R}{\partial X} + \phi^T \frac{\partial S}{\partial X} \right) \delta X. \tag{43}$$

We can assume the cost function to be a functional in the form

$$I(X, w, Z) = \int_V i(X, w, Z) dV, \tag{44}$$

with

$$i(X, w, Z) = \frac{C_p}{C_{\text{ref}}} (n_x \cos\alpha + n_y \sin\alpha) \delta(\eta), \tag{45}$$

where $\delta(\eta)$ being the Dirac delta function and $\eta = 0$ the equation defining the airfoil shape in the body fitted coordinates (ξ, η). For the Dirac delta function under integration the following equation holds

$$\int \delta(\eta) f(\eta) d\eta = f(0). \tag{46}$$

In the context of Equation (44), it reduces the volume integral to a surface integral. We suppose that the fluid obeys the Euler equations, which in body fitted coordinates take the form

$$\frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0, \tag{47}$$

where the transformed F, G are appropriate combinations of f and g

$$F = J \frac{\partial \xi}{\partial x} f + J \frac{\partial \xi}{\partial y} g = \begin{bmatrix} \rho U \\ \rho u U + \frac{\partial \xi}{\partial x} p \\ \rho v U + \frac{\partial \xi}{\partial y} p \\ \rho H U \end{bmatrix}. \tag{48}$$

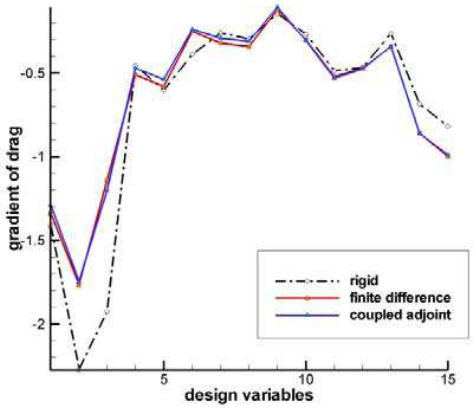

Figure 13: Validation of the aero-structural coupled adjoint with finite differences (AMP wing, $M_\infty = 0.78$ and $\alpha = 2.83°$).

Since our cost function I is of the form shown in Equation (44), as first step we have to formulate Equations (41) and (42) in an appropriate way, using the following property

$$\begin{aligned} \delta I(X,w,Z) &= \int_V \delta i(X,w,Z) dV \\ &= \int_V \left(\frac{\partial i(X,w,Z)}{\partial X}\delta X + \frac{\partial i(X,w,Z)}{\partial w}\delta w + \frac{\partial i(X,w,Z)}{\partial Z}\delta Z \right) dV. \end{aligned} \tag{49}$$

The derivation is identical to what has already been seen, and gives the adjoint equations

$$\int_V \left(\frac{\partial i}{\partial w} + \psi^T \frac{\partial R}{\partial w} + \phi^T \frac{\partial S}{\partial w} \right) dV = 0, \tag{50}$$

$$\int_V \left(\frac{\partial i}{\partial Z} + \psi^T \frac{\partial R}{\partial Z} + \phi^T \frac{\partial S}{\partial Z} \right) dV = 0. \tag{51}$$

And for the gradient we get

$$\begin{aligned} &\delta I(X,w,Z) = \\ &\int_V \left(\frac{\partial i(X,w,Z)}{\partial X}\delta X + \psi^T \frac{\partial R(X,w,Z)}{\partial X}\delta X + \phi^T \frac{\partial S(X,w,Z)}{\partial X}\delta X \right) dV. \end{aligned} \tag{52}$$

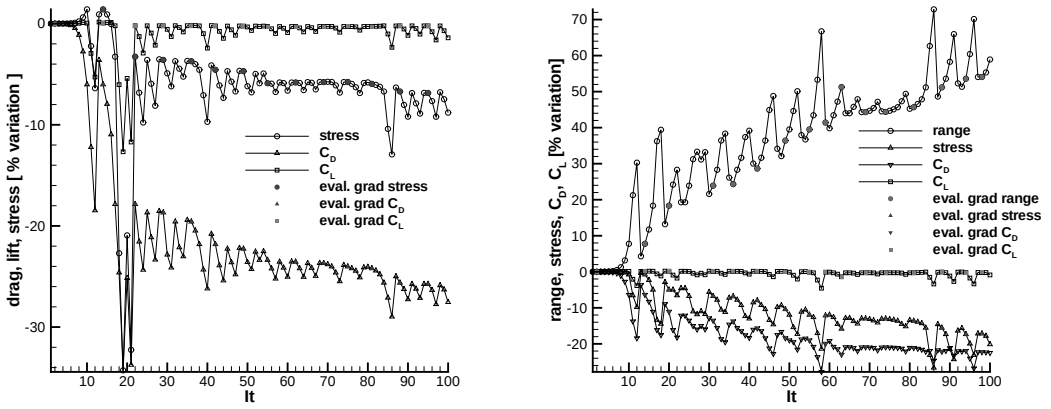

Figure 14: Optimization history for the drag reduction by constant lift while taking into account the static deformation (left picture) and for the range maximization (right picture) of the AMP wing ($Mach = 0.78$, $\alpha = 2.83°$). For both optimizations free-form deformation with 240 design variables was used for parameterization and feasible directions was used as optimization strategy.

It can be shown that Equation (50) is equivalent to the equation

$$\int_V \left(\left(\frac{\partial \psi}{\partial \xi} \right)^T \frac{\partial F}{\partial w} + \left(\frac{\partial \psi}{\partial \eta} \right)^T \frac{\partial G}{\partial w} \right) dV = 0 \tag{53}$$

and the boundary condition (in the case of the drag)

$$\psi_2 n_x + \psi_3 n_y + n_x \cos(\alpha) + n_y \sin(\alpha) - n^T \phi = 0. \tag{54}$$

Note that the structural adjoint variables appear only in the boundary condition (54), while the adjoint flow equation (53) is unchanged. This implies that in order to implement the coupling, only the boundary condition treatment in the FLOWer code has to be modified. Equation (42) represents the structural adjoint equation and its boundary conditions. The structural equation reads in the case of linear elasticity

$$S(X, w, Z) = K \cdot Z - a = 0, \tag{55}$$

where K is the symmetric stiffness matrix and a is the aerodynamic force. The derivative $\frac{\partial S}{\partial Z}$ in (42) can thus be replaced by K and the product $\phi^T K$ by $K\phi$. In this way, the same solver can be used for the structural direct and adjoint equation, with different boundary conditions, given by the first and second term in Equation (42). The first term is reduced to a surface integral by the presence of the Dirac delta function, giving a vector defined by

$$V_i = \frac{\partial \int_S I(X, w, Z) dS}{\partial Z_i}, \tag{56}$$

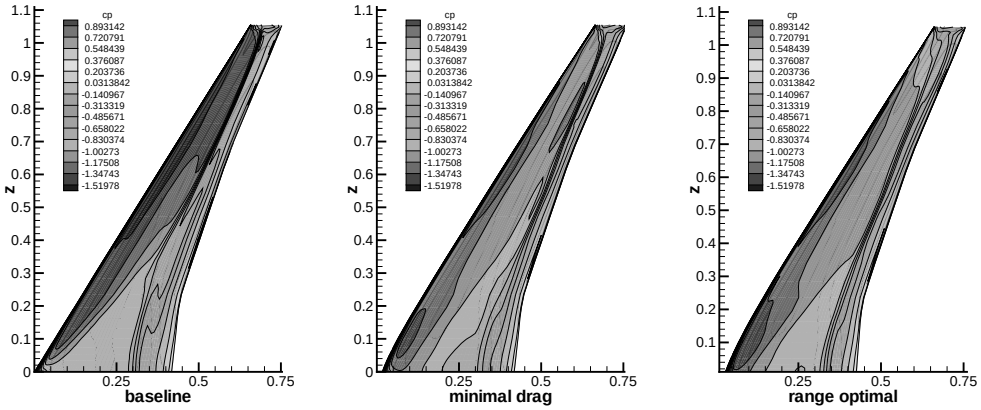

Figure 15: Pressure distribution for the baseline AMP wing shape and for the optimal wing shapes for drag minimization and range maximization ($Mach = 0.78$, $\alpha = 2.83°$).

that is the derivative of the cost function with respect to a structural degree of freedom. The second term, namely

$$\int_V \left(\psi^T \frac{\partial R}{\partial Z} \right) dV, \tag{57}$$

represents the integral of the scalar product of the adjoint field ψ and the partial derivative of the flow operator $R(X, w, Z)$ with respect to a structural degree of freedom, thus keeping the flow field and the design variables constant. It is evaluated by making use of the finite volume formulation implemented in FLOWer. A similar term appears in the expression for gradient (52), which explicitly becomes

$$\frac{dI}{dX} = \frac{\partial I}{\partial X} + \int_V \left(\psi^T \frac{\partial R}{\partial X} \right) dV + \int_V \left(\phi^T \frac{\partial S}{\partial X} \right) dV. \tag{58}$$

We already know how to evaluate the first two terms. The third term reduces to the surface integral of the adjoint field ϕ multiplied by the term

$$\frac{\partial S}{\partial X} = \frac{\partial K}{\partial X} Z - \frac{\partial a}{\partial X}. \tag{59}$$

Of the two terms on the right hand side, the first has been neglected, which is equivalent to assuming that shape deformations do not act on the structural mesh and thus on the stiffness matrix.

8.2 Implementation

In order to solve the coupled equations of the aero-structural system, a sequential staggered method has been implemented, where forces are transferred from the flow mesh to

the structure mesh and give the nodal loads, and deflections are transferred back from the structure mesh to the flow mesh which is consequently deformed. The flow around the body described by the Euler equation is solved by the DLR solver FLOWer, while the structural problem is solved by MSC Nastran. The transfer of information between the two meshes is managed by a module developed in-house based on B-spline volume interpolation [6, 5]. Typically, 20 exchanges of information between the two codes are more than enough to reach a converged aeroelastic solution, as shown in Figure 10.

The same staggered scheme has been used to solve the systems of the coupled adjoint equations, with the difference that now only adjoint deflections are interpolated from the structural mesh to the flow mesh, in order to evaluate the boundary condition (54) for the new adjoint flow computation. Each 100 steps of the adjoint flow solver, boundary conditions coming from the coupling are exchanged and updated, as shown in Figure 11.

8.3 Validation and Application

The validation of both the theory and the implementation of the adjoint formulation for the aeroelastic system has been achieved by comparison with the finite difference method.

As test case for the validation the AMP wing has been chosen. The structure has been modelled with a simplified model of 126 nodes, all lying on the wing surface, connected by 422 tria/quad shell and 198 beam elements. Such a model, unlike its fluid counterpart, is not state of the art, but is sufficient to demonstrate the features of the method. In order to underline the effect of aeroelasticity, the thickness of the beam elements of the wing has been tuned to reach a deflection of about 10% of the wing span at the wing tip.

Making use of the finite difference method, the gradient of the drag with respect to the shape parameters has been calculated, this time including the effect of aeroelastic interaction. This means that after a deformation of the jig shape (undeflected shape), an aeroelastic coupling was called and a stationary state was reached, as shown in Figure 10. This operation was repeated for every design parameter.
On the other hand, after the solution of the coupled adjoint equations, both the flow and structural adjoint fields have been used to reconstruct the gradient according to Equation (58). The comparison of both methods is shown in Figure 13, together with the gradient obtained when neglecting the aeroelastic coupling (rigid).

Finally, Figures 14 and 15 illustrate the application of the coupled aero-structural adjoint approach to the drag reduction of the AMP wing by constant lift while taking into account the static deformation of this wing caused by the aerodynamic forces as well as for the Breguet formula of aircraft range, where in addition to the lift to drag ratio the weight of the wing is taken into account.

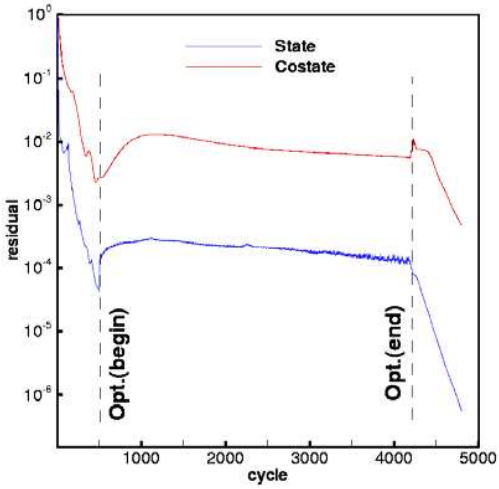

Figure 16: Convergence history of state and costate (RAE 2822, $M_\infty = 0.73$ and $\alpha = 2.0°$).

9 ONE-SHOT METHODS

The algorithmic approach of the so-called one-shot methods is based on an embedding of optimization strategies within the iterations of the respective flow solver. A continuous reduced SQP method is developed to solve the optimization problem in one joint pseudo-timestepping iteration for all variables (flow state, adjoint and wing design variables) [11, 12]. In this way we look for the steady states of the pseudo-time embedded non-stationary system of state, costate (or adjoint state) and design equations. The preconditioner used corresponds to Karush-Kuhn-Tucker matrices, which are used in an approximate reduced SQP method.

A first demonstration of the capability of the one-shot method is given for the drag reduction of the RAE 2822 airfoil in inviscid flow with $M_\infty = 0.73$ and $\alpha = 2.0°$. Figure 16 presents the convergence history of the optimization iterations. The optimization is started with the initial solution of the state and costate equations obtained after 500 steps with Runge-Kutta time integration. The convergence of the optimization is achieved after 3,700 optimization iterations. After convergence is achieved for optimization, we perform another 600 time iterations for state and costate solvers to reduce the residual of these two variables further to get more accurate values of surface pressure and force coefficients. Figure 17 shows the inexact and exact drag reduction during the optimization iterations. Inexact here means, that the drag is evaluated for the less converged state and costate variables used in the design loop. Afterwards, on the trace of modified shapes generated during the one-shot approach, the drag was recomputed up to an accuracy of 7 digits and compared with the inexact one. The final drag reduction after the optimization is about 68% and the shock completely vanished (Figure 18) as expected for inviscid cases. Figure

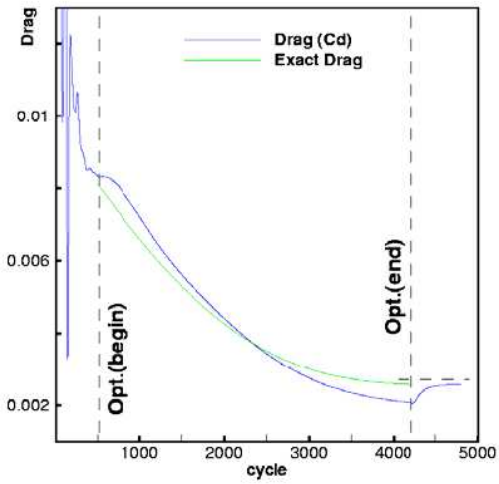

Figure 17: Convergence history of design (inexact/exact drag), RAE 2822, $M_\infty = 0.73$ and $\alpha = 2.0°$.

18 presents the comparison of the initial and final surface pressure distributions achieved with the one-shot approach (present) and with the conventional gradient based adjoint approach (steepest descent).

Altogether, the numerical cost of the one-shot optimization is of the magnitude of just 4 flow simulations, which is a dramatic reduction in computation time compared to the conventional approach.

10 ACKNOWLEDGEMENTS

The author thanks his colleagues at DLR, in particular A. Fazzolari and J. Brezillon, as well as the MEGADESIGN partners V. Schulz and S. Hazra from University of Trier for their contributions to this lecture note. Furthermore, the author thanks A. Walther and C. Moldenhauer from TU Dresden for their support and contributions w.r.t. algorithmic differentiation.

REFERENCES

[1] Brezillon, J., Dwight, R., "Discrete adjoint of the Navier-Stokes equations for aerodynamic shape optimization", Proceedings of EUROGEN05, Munich, 2005.

[2] Brezillon, J., Gauger, N.R., "2D and 3D aerodynamic shape optimization using the adjoint approach", Aerospace Science and Technology, 8, 8, pp. 715-727, 2004.

[3] Fazzolari, A., Gauger, N.R., Brezillon, J., "Sensitivity evaluation for efficient aerodynamic shape optimization with aeroelastic constraints", Proceedings of ECCOMAS 2004, Jyväskylä, Finland, 2004.

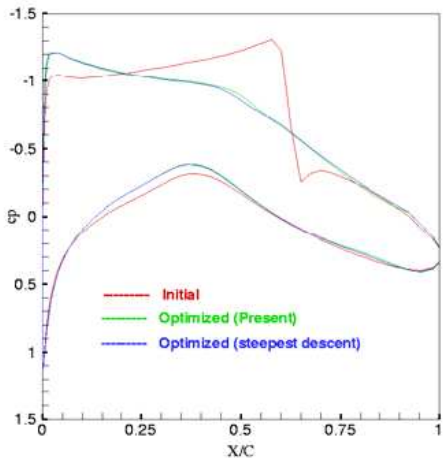

Figure 18: Initial and optimized pressure distribution (RAE 2822, $M_\infty = 0.73$ and $\alpha = 2.0°$).

[4] Fazzolari, A., Gauger, N.R., Brezillon, J., "An aero-structure adjoint formulation for efficient multidisciplinary wing optimization", Proceedings of EUROGEN05, Munich, 2005.

[5] Gauger, N.R., "Aerodynamic shape optimization using the adjoint Euler equations", Proceedings of the GAMM Workshop on Discrete Modelling and Discrete Algorithms in Continuum Mechanics, pp. 87-96, Logos Verlag Berlin, 2001.

[6] Gauger, N.R., "Das Adjungiertenverfahren in der aerodynamischen Formoptimierung", DLR-Report No. DLR-FB–2003-05 (ISSN 1434-8454), 2003.

[7] Gauger, N.R., Brezillon, J., "Aerodynamic shape optimization using adjoint method", Journal of the Aeronautical Society of India, 54, 3, pp. 247-254, 2002.

[8] Giering, R., Kaminski, T., Slawig, T., "Applying TAF to a Navier-Stokes solver that simulates an Euler flow around an airfoil", Future Generation Computer Systems 21(8),Elsevier 2005.

[9] Griewank, A., "Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation", Philadelphia, Society for Industrial and Applied Mathematics, 2000.

[10] Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O., Walther, A, "ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++", Technical University of Dresden, Institute of Scientific Computing and Institute of Geometry, 1999.

[11] Hazra, S. B., Schulz, V, "Simultaneous pseudo-timestepping for PDE-model based optimization problems", Bit Numerical Mathematics, Vol. 44, No. 3, pp. 457-472, 2004.

[12] Hazra, S.B., Schulz, V., Brezillon, J., Gauger, N.R., "Aerodynamic shape optimization using simultaneous pseudo-timestepping", Journal of Computational Physics, 204, 1, pp. 46-64, 2005.

[13] Heinrich, R., Ahrem, R., Günther, G., Krüger, W., Neumann, J., "Aeroelastische Simulation unter Verwendung des AMANDA Simulationssystems", VDI-Berichte Nr. 1682 (ISBN 3-18-091682-6), 2002.

[14] Hounjet, M.H.L., Prananta, B.B., Zwaan, R., "A Thin Layer Navier Stokes Solver and its Application for Aeroelastic Analysis of an Airfoil in Transonic Flow", Netherlands, DLR-Publication, 1995.

[15] Jameson, A., "Aerodynamic design via control theory", Journal of Scientific Computing, Vol. 3, pp. 233-260,1988.

[16] Jameson, A., Martinelli, L., Pierce, N.A., "Optimum Aerodynamic Design Using the Navier-Stokes Equations", Theoretical and Computational Fluid Dynamics, vol 10, pp. 213-237, 1998.

[17] Kroll, N., Rossow, C.C., Schwamborn, D., Becker, K., Heller, G., "MEGAFLOW - A Numerical Flow Simulation Tool for Transport Aircraft Design", ICAS-2002-1105.20, 2002.

[18] Kroll, N., Gauger, N.R., "Ongoing activities in shape optimization within the German Project MEGADESIGN", Proceedings of ECCOMAS 2004, Jyväskylä, Finland, 2004.

[19] Martins, J.R., Alonso, J.J., Reuther, J.J., "High-Fidelity aero-structural design optimization of a supersonic business jet", AIAA 2002-1483, 2002.

[20] Martins, J.R., Alonso, J.J., Reuther, J.J., "Complete configuration aero-structural optimization using a coupled sensitivity analysis method", AIAA 2002-5402, 2002.

[21] Widhalm, M., Gauger, N.R., Brezillon, J., "Implementation of an continuous adjoint solver in TAU", DLR-Report, to appear.