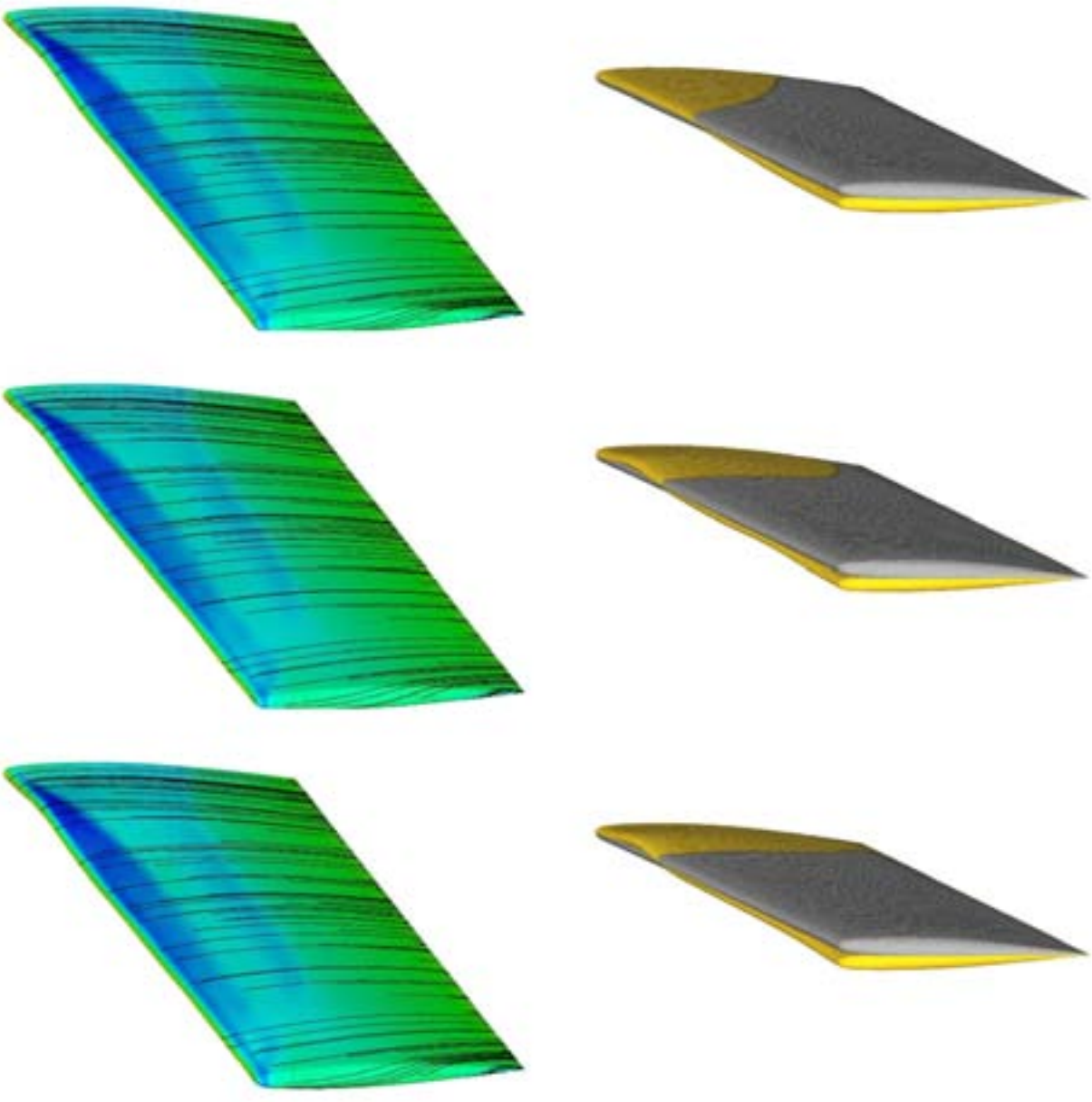
Mesh Deformation using Radial Basis Functions for Gradient-based Aerodynamic Shape Optimization

S. JAKOBSSON AND O. AMOIGNON



FOI-R--1784--SE Scientific report **Systems Technology**
ISSN-1650-1942 December 2005

S. Jakobsson and O. Amoignon

Mesh Deformation using Radial Basis Functions for Gradient-based Aerodynamic Shape Optimization

FOI-R--1784--SE
ISSN-1650-1942

Systems Technology
Scientific report

December 2005

Issuing organisation	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Systems Technology SE-164 90 STOCKHOLM	FOI-R--1784--SE	Scientific report
	Month year December 2005	Project number A64011
	Research area code Mobility and Space Technology, incl Materials	
	Sub area code Air Vehicles	
	Sub area code 2 Air Vehicles Technologies	
Author(s) S. Jakobsson and O. Amoignon	Project manager Martin Berggren	
	Approved by Monica Dahlen Head, Systems Technology	
	Scientifically and technically responsible Martin Berggren	
	Sponsoring agency FMV	

Report title

Mesh Deformation using Radial Basis Functions for Gradient-based Aerodynamic Shape Optimization

Abstract

Gradient-based aerodynamic shape optimization using Computational Fluid Dynamics (CFD), and time dependent problems in aeroelasticity, that is, coupled calculations between Computational Structural Mechanics (CSM) and CFD, require repeated deformations of the CFD mesh.

An interpolation scheme, based on Radial Basis Functions (RBF), is devised in order to propagate the deformations from the boundaries to the interior of the CFD mesh. This method can lower the computational costs due to the deformation of the mesh, in comparison with the usual Laplace smoothing. Moreover, the algorithm is independent of the mesh connectivities. Therefore, structured and unstructured meshes are equally treated as well as hybrid meshes.

The application of this interpolation scheme in problems of aerodynamic shape optimization is also carefully investigated. When the optimization is executed by a gradient-based algorithm the cost function is differentiated with respect to the design parameters in order to obtain the gradient. The gradient is most efficiently and accurately calculated by solving a certain adjoint equation derived from the discretized flow equations. The calculation of the gradient, which is detailed in this presentation, involves the Jacobian matrix of the mesh deformation.

Finally, we present the results of an optimization of the ONERA M6 wing at transonic speed using the interpolation algorithm. The results are used for comparison with another technique of mesh deformation. The quality of the mesh obtained by the new algorithm, and the interpolation error, are analyzed with respect to the parameters of the interpolation scheme: the type of RBF, the RBFs shape parameter, and the sets of control points.

Keywords

interpolation, radial basis functions, mesh deformation, gradient optimization, inviscid compressible flow , adjoint equations

Further bibliographic information

ISSN	Pages	Language
ISSN-1650-1942	36	English
Distribution By sendlist	Price According to price list	
	Security classification Unclassified	

Utgivare	Rapportnummer, ISRN	Klassificering
FOI – Totalförsvarets forskningsinstitut Systemteknik 164 90 STOCKHOLM	FOI-R--1784--SE	Vetenskaplig rapport
	Månad år December 2005	Projektnummer A64011
	Forskningsområde Farkost- och rymdteknik, inkl material	
	Delområde Luftfarkoster	
	Delområde 2 Flygfarkostteknik	

Författare	Projektledare
S. Jakobsson och O. Amoignon	Martin Berggren
	Godkänd av Monica Dahlen Chef, Systemteknik
	Tekniskt och/eller vetenskapligt ansvarig Martin Berggren
	Uppdragsgivare/kundbeteckning FMV

Rapporttitel

Nätdeformering med radiella basfunktioner för gradient baserad aerodynamisk formoptimering

Sammanfattning

För både gradientbaserad aerodynamisk optimering och aeroelasticitet, kopplade struktur och strömningsberäkningar, behövs snabba metoder för nätdeformering. I den här rapporten presenteras en interpolationsmetod baserad på radiella basfunktioner (RBF) för att propagera förkjutningarna av noderna på randen av beräkningsnätet till det inre. Metoden kan minska beräkningskostnaden för nätdeformationen jämfört med deformations metoder baserade på diskreta Laplace ekvationern (*Laplace Smoothing*). Algoritmen använder ej konnektiviten hos nätet. Därför kan strukturerade, ostrukturerade samt hybridnät behandlas likvärdigt. Tillämpningar på aerodynamisk formoptimering är också undersökta. För gradientbaserade optimeringsalgoritmer skall kostfunktionen deriveras med avseende på designparametrarna. För aerodynamisk formoptimering beräknas gradienten effektivast och noggrannast med den adjungerade ekvationen till de diskretiserade strömningsekvationerna. Beräkningen av gradienten involverar Jacobian matrisen för nätdeformeringen vilket beskrivs i detalj i rapporten.
Slutligen presenteras resultatet av optimering av ONERA M6 vingen i transonisk hastighet med interpolationsalgoritmen och jämförs med Laplace tekniken för nätdeformation. Kvaliten på beräkningsnätet och interpolationsfelet jämförs för olika parametrar till interpolationsschemat: typen av basfunktion, skalparametern till basfunktionen samt olika mängder av kontrollpunkter.

Nyckelord

interpolation, radiella basfunktioner, nätdeformering, gradient optimering, inviskös strömning, adjunkta ekvationer

Övriga bibliografiska uppgifter

ISSN	Antal sidor	Språk
ISSN-1650-1942	36	Engelska
Distribution Enligt missiv	Pris Enligt prislista	
	Sekretess Öppen	

Contents

1 Introduction

In aeronautic design it is common to use Computational Fluid Dynamics (CFD) to simulate air flowing around wings, airplanes, or through the inlet and outlet of the propulsion units. CFD simulations have a lower cost than traditional experiments, which enables to carry out studies requiring many simulations such as parameter optimization. In aerodynamic shape optimization, the goal is to improve an indicator of the aerodynamic performance of the design given by a set of parameters. The improvement is measured by the reduction of a cost function, which, typically, is an expression based on the drag, the lift, and the moment coefficients [8, 10, 16, 18, 17, 2].

The minimization of the cost function by a gradient-based algorithm, which is an efficient approach, still requires many function and gradient evaluations [19]. In aerodynamic applications, each function evaluation requires a CFD solution on a different design, which involves different CFD meshes. Deforming the mesh is an efficient alternative to re-meshing and it enables to build a smooth mapping from the design parameters to the cost function. In aerodynamic shape optimization, we identify two main approaches of the mesh deformation. The first is to use an interpolation algorithm, which may easily be devised in the case of structured meshes, for example by displacing the nodes along grid lines [8, 18, 17]. The second approach, suitable for unstructured or hybrid meshes, is to use a mesh smoother, also called Laplace smoothing [10, 16, 2]. This last technique involves solving large systems of equations. This report proposes an alternative technique of mesh deformation that can be used on arbitrary meshes and that reduces the computational cost.

The new algorithm is an interpolation method. It is derived from a problem of interpolation presented in Section 2.1. By requiring that the interpolating function is translation and rotation invariant, and linear in the displacements, a representation formula is derived (Theorem 2.1.2). Combined with radial basis functions, in Section 2.2, this yields the required mesh deformation algorithm. It turns out that this is the same expansion as used by Beckert and Wendland [5] for aeroelastic applications. They required that all first order polynomials should to be interpolated exactly which leads to translation and rotation invariance. The method is tested in 2D and 3D in Section 2.3.

Our purpose is to apply this method in aerodynamic shape optimization and, in coming studies, for aeroelastic calculations. Section 3 details an example of optimization of the ONERA M6 wing with the purpose to minimize a cost function that depends on the CFD solution and on the design of the wing. The particularity of this application, in contrast to aeroelasticity, is that the mapping defined by the mesh deformation algorithm is involved at two stages in the calculations. In the *forward mode*, when the cost function is evaluated for a new design, the mapping deforms the mesh for a given deformation of the wing, as explained in Section 3.1. In the *backward mode*, the Jacobian of the mapping is used to calculate the gradient of the cost function with respect to the parameters of design. Section 3.4 details the calculation of gradients of a cost function.

The performances of the mesh mapping, with respect to the computational cost, is investigated in Section 4.1. The results of optimization of the ONERA M6 wing are given in Section 4.2. In Section 4.3 we analyze the performances of the mesh mapping, with respect to mesh qualities and to the interpolation error.

2 Interpolation scheme for mesh deformation

In this paper we take an abstract approach to define the mesh deformation. The idea is to construct an interpolant of the displacements of some boundary nodes, in a CFD mesh, in such a way that it extrapolates those displacements into the entire domain of the CFD mesh, with some constraints. The presentation of the abstract problem stresses the connection to aeroelastic interpolation since the two problems set common requirements.

2.1 Geometric interpolation problem

The following geometric interpolation/extrapolation problem is of relevance for both the aeroelastic interpolation problem and mesh deformations.

Problem 2.1.1. Assume that we have a deformable structure Ω with rest state/position Ω^0. There are a number of control points $\{P_k\}_{k\in\mathcal{V}_P}$ with coordinates $\{\boldsymbol{x}_k\}_{k\in\mathcal{V}_P}$ in Ω, where $\mathcal{V}_P$ is the set of control nodes indexes. In the rest position, the point P_k has the coordinates $\boldsymbol{x}_k^0$. Given that the points P_k have coordinates $\boldsymbol{x}_k$, $k \in \mathcal{V}_P$, compute an approximation of the coordinates $\boldsymbol{x}$ to a point P which in the rest state has the coordinates $\boldsymbol{x}^0$.

In the sections relative to mesh deformation $\mathcal{V}_P$ is a subset of the mesh nodes, therefore, it is convenient to use the following notations:

$$N = |\mathcal{V}_P| \text{ , and , } \mathcal{V}_P = \{k_i\}_{1\leq i\leq N} .$$

The approximation of the coordinates of P depends on $\boldsymbol{x}^0$, $\{\boldsymbol{x}_{k_i}\}_{i=1}^N$ and $\{\boldsymbol{x}_{k_i}^0\}_{i=1}^N$

$$\boldsymbol{x} = G\left(\boldsymbol{x}^0, \{\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}_{k_i}^0\}_{i=1}^N\right) .$$

We will call the function G the *geometric interpolation function.*

For applications in aeroelasticity, Ω might be a wing of an aircraft and the control points are nodes of the grid used for the Computational Structural Mechanics (CSM) analysis. In this context, the geometric interpolation function is used to calculate the displacements of the points on the wing from the displacements given by the CSM calculations at the control points. For mesh deformation, the control points would be nodes, of the CFD grid, that are on the wing. In this other context, the geometric interpolation function is used to calculate the displacements of all the nodes in the CFD mesh. In all applications of this method, if $\boldsymbol{x}^0$ is within the convex hull of the control points then the coordinate vector of P is interpolated, otherwise it is extrapolated. As the mesh deformation is used here for aerodynamic shape optimization, and because we use a gradient-based algorithm, we impose that the geometric interpolation function is differentiable with respect to the displacements of the control points.

We require that the geometric interpolation function satisfies the following four conditions.

1. If P is one of the control points, then the interpolation condition should be satisfied

$$\boldsymbol{x}_{k_i} = G\left(\boldsymbol{x}^0_{k_i}, \{\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right), \qquad \forall k \in \mathcal{V}_P. \tag{2.1}$$

2. For all translation operators T_α, $T_\alpha \boldsymbol{x} = \boldsymbol{x} + \alpha$, and $\boldsymbol{x}^0 \in \mathbb{R}^3$, we have

$$T_\alpha G\left(\boldsymbol{x}^0, \{\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right) = G\left(\boldsymbol{x}^0, \{T_\alpha \boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right). \tag{2.2}$$

3. For all rotation operators R and $\boldsymbol{x}^0 \in \mathbb{R}^3$, we have

$$RG\left(\boldsymbol{x}^0, \{\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right) = G\left(\boldsymbol{x}^0, \{R\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right). \tag{2.3}$$

4. The displacement of any point should depend linearly on the displacement of the control points.

Condition 1 states that G is an interpolant. Conditions 2 and 3 impose that the interpolation method is invariant with respect to rigid motions. In aeroelastic applications, the three first conditions guarantee the conservation of virtual work, total load and total momentum, when the displacements are transferred from the CSM grid to the CFD grid. Condition 4 limits in a natural way the space of possible interpolation functions. Most existing aeroelastic interpolation methods satisfy condition 4. One exception is the CVT method (constant volume tetrahedra) of L. Goura [12] which satisfies the conditions 1 to 3 but is non-linear in the displacements. For an overview of existing methods we refer to [14]. The program *SPIVOL*, developed by EADS-CASA, uses *volume splines* for interpolation and satisfies these conditions [21]. We argue that for mesh deformation it is also natural to require invariance under rigid motions since these mappings preserves the qualitative properties of the mesh. For convenience, we introduce the displacement vectors for the coordinates

$$\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{x}^0, \tag{2.4}$$

$$\boldsymbol{v}_{k_i} = \boldsymbol{x}_{k_i} - \boldsymbol{x}^0_{k_i}, \qquad 1 \le i \le N. \tag{2.5}$$

The linearity condition can be written

$$\boldsymbol{v} = G\left(\boldsymbol{x}^0, \{\boldsymbol{x}_{k_i}\}_{i=1}^N, \{\boldsymbol{x}^0_{k_i}\}_{i=1}^N\}\right) - \boldsymbol{x}^0 = \sum_{i=1}^N \boldsymbol{A}_i(\boldsymbol{x}^0)\left(\boldsymbol{x}_{k_i} - \boldsymbol{x}^0_{k_i}\right) = \sum_{i=1}^N \boldsymbol{A}_i(\boldsymbol{x}^0)\,\boldsymbol{v}_{k_i}. \tag{2.6}$$

The following theorem gives a representation formula for all the geometric interpolation function which satisfy conditions 1 to 4.

THEOREM 2.1.2. *The geometric interpolation function satisfies the conditions 1 to 4 if and only if it has the following representation in three dimensions*

$$\boldsymbol{x} = \boldsymbol{x}^0 + \sum_{i=1}^N a_i(\boldsymbol{x}^0)\left(\boldsymbol{x}_{k_i} - \boldsymbol{x}^0_{k_i}\right) = \boldsymbol{x}^0 + \sum_{i=1}^N a_i(\boldsymbol{x}^0)\boldsymbol{v}_{k_i}, \tag{2.7}$$

where $a_i : \Omega^0 \to \mathbb{R}$ are scalar valued functions defined on Ω^0 that satisfy

$$\sum_{i=1}^N a_i(\boldsymbol{x}^0) = 1 \tag{2.8}$$

and

$$\boldsymbol{x}^0 = \sum_{i=1}^N a_i(\boldsymbol{x}^0)\boldsymbol{x}^0_{k_i}. \tag{2.9}$$

Radial basis function	$\phi(r)$
Spline type (R_n)	$\lvert r\rvert^n$, n odd
Thin Plate Spline (TPS_n)	$\lvert r\rvert^n \log \lvert r\rvert$, n even
Multiquadric (MQ)	$\sqrt{1+r^2}$
Inverse multiquadric (IMQ)	$\frac{1}{\sqrt{1+r^2}}$
Inverse quadratic (IQ)	$\frac{1}{1+r^2}$
Gaussian (GS)	e^{-r^2}

Table 2.1: Common radial basis functions.

In particular, the coordinate vectors $\{\boldsymbol{x}^0_{k_i}\}_{i=1}^N$ *cannot be contained in any plane. Moreover, the equations 2.8) and (2.9) together are equivalent to that*

$$q(\boldsymbol{x}^0) = \sum_{i=1}^{N} a_i(\boldsymbol{x}^0) q(\boldsymbol{x}^0_{k_i})$$

holds for all first degree polynomials q*; that is, all first degree polynomials are interpolated exactly.*

In two dimensions, the scalar valued functions a_i *should be replaced by scalar valued functions times a rotation matrix. Otherwise the theorem holds if the equations are modified appropriately, that is,* 1 *in (2.8) should be replaced by the identity matrix.*

A proof is given in Appendix A. Due to (2.9), the representation (2.7) can also be written

$$\boldsymbol{x} = \sum_{i=1}^{N} a_i(\boldsymbol{x}^0) \boldsymbol{x}_{k_i}.$$

2.2 Realization based on Radial Basis Functions

To obtain a geometric interpolation function that fulfills the conditions 1 to 4 above, we will apply radial basis functions interpolation for the displacements. We begin with a review of some results on radial basis functions.

Radial basis functions provide a very general and flexible way of interpolation in multi-dimensional spaces, even for unstructured data where it is often impossible to apply polynomial or spline interpolation. Because of its good approximation properties and ease of implementation, the method is a popular choice in many different areas, ranging from statistics to the approximation of partial differential equations, see for example [6] and the books [7] and [25].

According to Theorem 2.1.2, all first degree polynomials must be exactly interpolated in order to satisfy conditions 1 to 4. Therefore, we need to consider RBF expansions that satisfy this condition.

Suppose that g is the function to be interpolated at the set $\mathbf{X}_P = \{\boldsymbol{x}_{k_i}\}_{i=1}^N$ of data points, all in $\mathbb{R}^d$. Let ϕ be a function defined on the positive real axis. The *interpolation space* then consists of all functions of the form

$$s(\mathbf{x}) = \sum_{l=1}^{N} \gamma_l \phi(\|\mathbf{x} - \mathbf{x}_l\|) + h(\mathbf{x}), \tag{2.10}$$

for coefficients $\gamma_l \in \mathbb{R}$ and first degree polynomials h. Here $\|\cdot\|$ denotes the standard Euclidean norm. In many cases it is appropriate to scale the

basis function with a so-called *shape parameter* ε. The basis function is then replaced by $\phi_\varepsilon(r) = \phi(\varepsilon r)$. For mesh deformation we need three interpolating functions, one for each coordinate direction. The data points are sometimes called centers, and the basis functions are radial around these centers. The coefficients $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_N)^T$ and the polynomial are then chosen so that s interpolates g exactly at the data points

$$s(\boldsymbol{x}_{k_i}) = g(\boldsymbol{x}_{k_i}), \qquad 1 \leq i \leq N,$$

and $\boldsymbol{\gamma}$ must also be so that

$$0 = \sum_{i=1}^{N} \gamma_i q(\boldsymbol{x}_{k_i})$$

for all first degree polynomials q. This leads to a linear equation system for the coefficients $\boldsymbol{\gamma}$ and the coefficients for the polynomial. If ϕ is one of the basis function given in Table 2.1, then this system is non-singular, which implies that a unique solution exists.

This linear equation system for the coefficient vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ takes the form

$$\begin{pmatrix} \mathbf{M} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ 0 \end{pmatrix}, \tag{2.11}$$

where $\mathbf{M}$ is the *interpolation matrix*

$$M_{ij} = \phi(\|\boldsymbol{x}_{k_i} - \boldsymbol{x}_{k_j}\|), \qquad 1 \leq i, j \leq N.$$

and $\mathbf{P}$ is the matrix defined by constraint to interpolate all first degree polynomials exactly

$$\mathbf{P} = \begin{pmatrix} 1 & x_{k_1}^0 & y_{k_1}^0 & z_{k_1}^0 \\ 1 & x_{k_2}^0 & y_{k_2}^0 & z_{k_2}^0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{k_N}^0 & y_{k_N}^0 & z_{k_N}^0 \end{pmatrix}. \tag{2.12}$$

Here we have introduced the vectors $\mathbf{g} = \{g(\boldsymbol{x}_{k_i})\}_{i=1}^N$, and $\boldsymbol{\beta}$ is the vector of coefficients for the polynomial h. Given that the interpolation matrix $\mathbf{M}$ is invertible (as it is for many RBFs), we can solve for $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ to obtain

$$\boldsymbol{\gamma} = \mathbf{M}^{-1}\mathbf{g} - \mathbf{M}^{-1}\mathbf{P}\mathbf{M}_\mathbf{P}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{g} \tag{2.13}$$

and

$$\boldsymbol{\beta} = \mathbf{M}_\mathbf{P}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{g}, \tag{2.14}$$

where

$$\mathbf{M}_\mathbf{P} = \left(\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\right)^{-1}.$$

Finally, the polynomial h in (2.10) becomes

$$h(\boldsymbol{x}) = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 z,$$

where $\boldsymbol{x} = (x, y, z)^T$.

Now we apply this type of approximation to the displacements v_x, v_y, and v_z in each coordinate direction

$$v_x = s_x(\boldsymbol{x}) = \sum_{i=1}^{N} \gamma_i^x \phi(\|\boldsymbol{x} - \boldsymbol{x}_{k_i}\|) + \beta_1^x + \beta_2^x x + \beta_3^x y + \beta_4^x z,$$

$$v_y = s_y(\boldsymbol{x}) = \sum_{i=1}^{N} \gamma_i^y \phi(\|\boldsymbol{x} - \boldsymbol{x}_{k_i}\|) + \beta_1^y + \beta_2^y x + \beta_3^y y + \beta_4^y z,$$

$$v_x = s_z(\boldsymbol{x}) = \sum_{i=1}^{N} \gamma_i^z \phi(\|\boldsymbol{x} - \boldsymbol{x}_{k_i}\|) + \beta_1^z + \beta_2^z x + \beta_3^z y + \beta_4^z z,$$

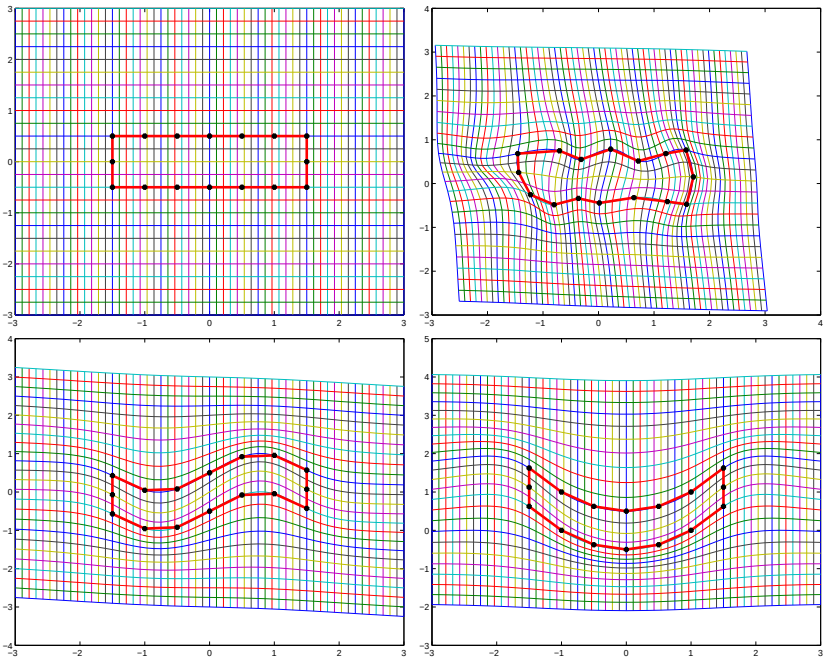


Figure 2.1: The undeformed domain together with its control points are shown to the upper left. The upper right figure shows the deformation due to a random deflection of the control points. The lower left and lower right figure show the deformation resulting from sinusoidal and quadratic deflections, respectively.

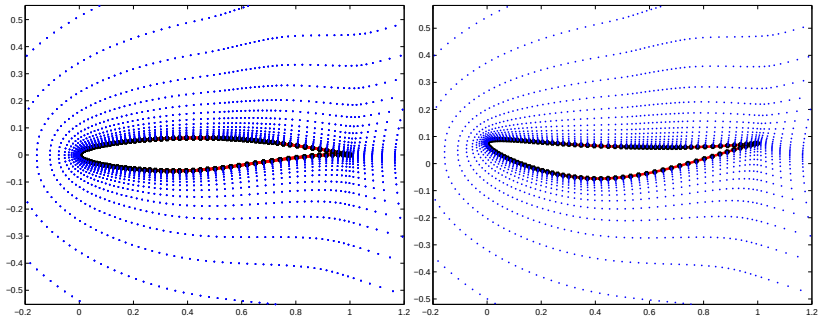


Figure 2.2: Deformation around a two dimensional airfoil. The right figure shows the mesh after quadratic deflection of the control points on the surface

where the set of coefficients satisfy (2.11).

Clearly, the coefficients depend linearly on the displacements. Thus, according to Theorem 2.1.2, the conditions 1 to 4 are satisfied. However, here the linear dependence between the displacements of the control points and an arbitrary point is implicit through the coefficients $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. If needed, it is not hard to derive explicit formulas for the coefficient vector $\mathbf{a}(\boldsymbol{x}_0)$.

2.3 Simple test cases

To illustrate the method, two simple two-dimensional examples of mesh deformation have been simulated in MATLAB$^{\text{TM}}$.

The result of the first test is shown in Figure 2.1. Here the control points

lie on the outer boundary of a 3-unit-long and 1-unit-wide rectangle. The test is performed with the inverse multiquadric basis function with the shape parameter equal to 1. Although the deformation for all points outside the structure is extrapolated rather then interpolated, the deformation is, in all cases, very regular throughout the external mesh. A possible interpretation is that the constraint to interpolate all first degree polynomials exactly has a stabilizing effect. This is promising for applications to mesh deformation where it is important to preserve the properties of the original mesh in order to obtain good computational results. A slightly more complicated case is shown in Figure 2.2.

3 Optimization of a wing

In this section we describe how the the mesh deformation algorithm defined earlier is used in the optimization of the ONERA M6 wing, described in the AGARD report [22], at Mach number M = 0.8395 and angle of attack $\alpha = 3.06°$. The numerical results are given in Section 4. The flow is modelled by the Euler equations of gas dynamics.

The cost function J is designed in order to reduce the pressure drag, and in addition penalize changes in the coefficients of lift and pitching moment:

$$J = \lambda_D C_D + \frac{1}{2}\lambda_L \left(C_L - C_L^0\right)^2 + \frac{1}{2}\lambda_M \left(C_M - C_M^0\right)^2 , \tag{3.1}$$

where superscript ^{0} denotes values at initial design. The coefficients of drag (C_D), lift (C_L), and pitching moment (C_M), are calculated from the pressure at nodes i, denoted p_i, on the wing ($i \in \mathcal{V}(\partial\Omega_w)$):

$$\begin{aligned}
C_D &= \sum_{i \in \mathcal{V}(\partial\Omega_w)} \frac{p_i \boldsymbol{n}_i \cdot \boldsymbol{d}_D}{\frac{1}{2}\rho_\infty \boldsymbol{u}_\infty^2 S_{\text{ref}}} , \\
C_L &= \sum_{i \in \mathcal{V}(\partial\Omega_w)} \frac{p_i \boldsymbol{n}_i \cdot \boldsymbol{d}_L}{\frac{1}{2}\rho_\infty \boldsymbol{u}_\infty^2 S_{\text{ref}}} , \\
C_M &= \sum_{i \in \mathcal{V}(\partial\Omega_w)} \frac{p_i \boldsymbol{d}_M \cdot \left(\boldsymbol{x}_i - \boldsymbol{O}_{\text{ref.}}\right) \times \boldsymbol{n}_i}{\frac{1}{2}\rho_\infty \boldsymbol{u}_\infty^2 S_{\text{ref}} L_{\text{ref}}} .
\end{aligned} \tag{3.2}$$

Here, $\boldsymbol{d}_D$ is a unit vector in the direction of the farfield velocity $\boldsymbol{u}_\infty$, $\boldsymbol{d}_L$ is a unit vector orthogonal to $\boldsymbol{d}_D$ such that C_L is positive for lifting pressure forces, and $\boldsymbol{d}_M$ is a unit vector orthogonal to $\boldsymbol{d}_D$ and $\boldsymbol{d}_L$. Other notations are ρ_∞, the density in the farfield, S_{ref} a reference area of the wing, L_{ref} a reference length of the wing, $\boldsymbol{O}_{\text{ref}}$ the reference center of rotation for the calculation of the moments, $\boldsymbol{x}_i$ the vector of coordinates of node i, and $\boldsymbol{n}_i$ the normal surface vector at node i.

The parameterization, given appendix B, is also used in [2]. It parameterizes the wing in terms of twist, camber and thicknesses distributions. The mesh is deformed by the RBF algorithm presented in Section 2.2 with additional treatments at the symmetry and far-field boundaries, explained in Section 3.1. The discretized Euler equations are presented in Section 3.2. The *adjoint method* is used to compute the gradient of the cost function with respect to the design parameters. In contrast to perturbation methods such as finite differences, the cost of the adjoint method is independent from the number of design variables. For a presentation of this approach we refer to [2, 20, 11, 13]. The adjoint equations are presented in discretized form in Section 3.3, and Section 3.4 details the calculation of the gradient with respect to the design parameters.

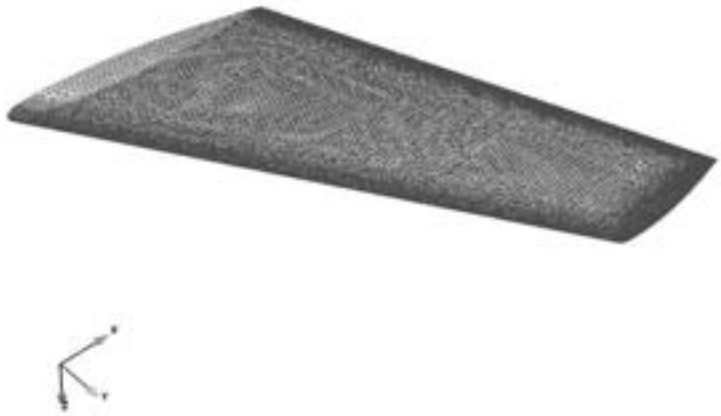


Figure 3.1: Euler mesh on the ONERA M6 wing.

3.1 Constrained mesh deformation

The symmetry boundary is at the root of the wing $y = 0$, see Figure 3.1. It is required that all points in the symmetry boundary ($y = 0$) must stay in this plane for all allowed deformations. Moreover, we shall impose the constraint that the points on the far-field boundaries are fixed. The latter constraint is easily implemented using a suitable cut-off function. This implies that the translation and rotation invariance are lost in the farfield.

We assume that the deformation of the control points is such that

$$\frac{\Delta y_k}{y_k^0}$$

is bounded. This means that the deformation in the y-coordinate of the control points is controlled by its distance to the plane $y = 0$. To describe the interpolation of the deformation in the y direction we define the function

$$\kappa(y, \alpha) = 1 - \exp(-\alpha y). \tag{3.3}$$

Clearly, $\kappa(y, \alpha) \approx y$ for small y and $\kappa(y, \alpha) \approx 1$ for large values (everything relative α which defines the scale of the problem).

According to our assumptions, the quantities

$$\eta_k = \frac{\Delta y_k}{\kappa(y_k^0, \alpha)}$$

are bounded. Let s_η be a RBF approximation of this function as described in Section 2.2. Then

$$s_{\Delta y}(\boldsymbol{x}) = \kappa(y, \alpha) s_\eta(\boldsymbol{x})$$

gives a deformation in the y-coordinate which leaves all points in the plane $y = 0$ fixed and interpolates the deformation at all control points. To complete the deformation, we choose a smooth cut-off function ω which is identically

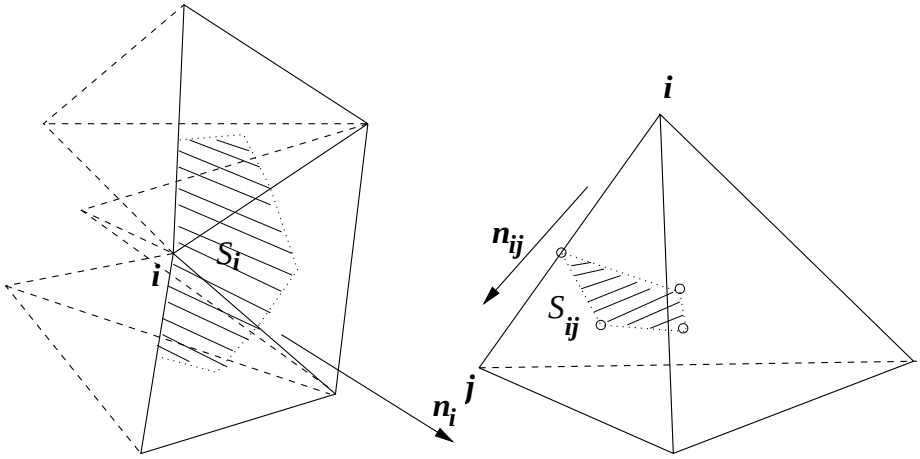


Figure 3.2: The dual grid control surfaces and associated surface normals, on the boundary (left) and in a volume element (right). $\boldsymbol{n}_{ij}$ is the surface normal of $S_{ij} = V_i \bigcap V_j$ (partially represented), and $\boldsymbol{n}_i$ is the surface normal of S_i (partially represented), the intersection of V_i with the boundary $\partial\Omega$.

1 in a neighborhood of the wing and decays smoothly to zero at the far field boundaries. If $s_{\Delta x}$ and $s_{\Delta z}$ are the deformations in the x and z coordinate, respectively, and $\mathbf{s} = (s_{\Delta x}, s_{\Delta y}, s_{\Delta z})^T$ then the function

$$\boldsymbol{x} = \boldsymbol{x}^0 + \omega(\boldsymbol{x}^0)\mathbf{s}(\boldsymbol{x}^0)$$

fulfills all imposed conditions for the mesh deformation.

3.2 Flow equations discretized in Edge

The program Edge [9, 23] solves a node-centered and edge-based finite-volume approximation of the system of compressible Euler or Reynolds Averaged Navier Stokes equations. The use of edge-based data is particularly efficient when using unstructured meshes. The discretization is based on the median dual grid as depicted in Figure 3.2. For an introduction to this type of discretization, we refer to Barth [4].

The Euler equations express the conservation of the mass, momentum and total energy densities. The vector field of the conserved flow variables is denoted $\mathbf{w} = [\rho, \mathbf{m}, E]^T$. In the finite volume method, a bounded region $\overline{\Omega}$ of the flow domain is discretized by control volumes V_i (see Figure 3.2), $i \in \mathcal{V}(\overline{\Omega})$, with $\mathcal{V}(\overline{\Omega})$ the set of the nodes indexes in $\overline{\Omega}$. The finite-volume approximation used here reduces the conservation equations, applied to each control volume V_i, to a system of ordinary differential equations (ODEs) involving only the values of the conserved flow variables at the mesh nodes. The system of ODEs reads:

$$\mathrm{V}_i \frac{\mathrm{d}\mathbf{w}_i}{\mathrm{d}t} + \mathbf{R}_i = 0, \qquad \forall i \in \mathcal{V}(\overline{\Omega})\,, \tag{3.4}$$

where $\mathbf{w}_i$ approximates the exact conservative variables at node i, that is $\mathbf{w}(\boldsymbol{x}_i)$

$$\mathbf{w}_i = [\rho_i, \boldsymbol{m}_i, E_i]^T\,, \quad \text{where} \quad \boldsymbol{m}_i = \rho_i \boldsymbol{u}_i\,. \tag{3.5}$$

For a given node i in the domain $\overline{\Omega}$, the vector $\mathbf{R}_i$, in (3.4), sums up the

numerical fluxes through the boundaries of V_i:

$$\begin{aligned}\mathbf{R}_i &= \sum_{j\in\mathcal{N}_i} (\boldsymbol{n}_{ij}\cdot\mathbf{f}_{ij}+\mathbf{d}_{ij}) \qquad \forall i\in\mathcal{V}(\Omega)\,,\\ \mathbf{R}_i &= \sum_{j\in\mathcal{N}_i} (\boldsymbol{n}_{ij}\cdot\mathbf{f}_{ij}+\mathbf{d}_{ij}) + \boldsymbol{n}_i\cdot\mathbf{f}_i^{\mathrm{bc}} \qquad \forall i\in\mathcal{V}(\partial\Omega)\,,\end{aligned} \tag{3.6}$$

where $\mathcal{N}_i$ is the set of nodes connected to i with an edge $\vec{ij}$. The numerical flux through the control surface S_{ij} of volume V_i (Figure 3.2) is the sum of an advection flux $(\boldsymbol{n}_{ij}\cdot\mathbf{f}_{ij})$, fluxes implementing the boundary conditions $(\boldsymbol{n}_i\cdot\mathbf{f}_i^{\mathrm{bc}})$, and artificial dissipation $(\mathbf{d}_{ij})$. In a central scheme approximation, the convection flux density of the conserved variables through the control surface V_{ij} is supposed constant and replaced by

$$\mathbf{f}_{ij} = (\mathbf{f}_i+\mathbf{f}_j)/2 \tag{3.7}$$

where

$$\mathbf{f}_i = \mathbf{f}(\mathbf{w}_i)\,, \quad \text{and} \quad \mathbf{f} = [\boldsymbol{m}\,,\,(\boldsymbol{m}\otimes\boldsymbol{m})/\rho+\boldsymbol{I}p\,,\,\boldsymbol{m}(E+p)/\rho]^T\,. \tag{3.8}$$

In expression (3.8), the pressure p is calculated assuming an ideal fluid and the law of perfect gas:

$$p = (\gamma-1)\left(E-\frac{1}{2}\frac{\boldsymbol{m}^2}{\rho}\right). \tag{3.9}$$

Integration of the flux density $\mathbf{f}_{ij}$ through the surface V_{ij} is then obtained by the dot products $\boldsymbol{n}_{ij}\cdot\mathbf{f}_{ij}$ where $\boldsymbol{n}_{ij}$ is the surface normal associated with V_{ij}. The artificial dissipation flux $\mathbf{d}_{ij}$ is a blend of second- and fourth-order differences [15].

An impermeability boundary condition $(\boldsymbol{u}_i\cdot\boldsymbol{n}_i=0)$ is applied on wall and symmetry boundaries yielding the boundary fluxes

$$\mathbf{f}_i^{\mathrm{bc}} = [\mathbf{0}\,,\,\boldsymbol{I}p_i\,,\,\mathbf{0}]^T\,. \tag{3.10}$$

On a farfield boundary the fluxes (3.8) are computed using the characteristic primitive variables $(\mathbf{v}^c)$ based on either the farfield data $(\mathbf{v}_\infty)$, for incoming characteristics, or, the flow data at the previous time step $(\mathbf{v}_i)$, for outgoing characteristics. It is expressed in [9] as:

$$\begin{aligned}\mathbf{v}_i^c(\hat{\boldsymbol{n}}_i) = {}& \mathbf{L}(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty)\,\mathbf{H}(\lambda_i)\,\mathbf{L}^{-1}(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty)\,\mathbf{v}_i\\ &+\mathbf{L}(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty)\,(\mathbf{I}-\mathbf{H}(\lambda_i))\,\mathbf{L}^{-1}(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty)\,\mathbf{v}_\infty\,,\end{aligned} \tag{3.11}$$

where $\mathbf{L}(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty)$ is a matrix of right eigenvectors that diagonalizes the Jacobian matrix of the flux in primitive variables along the outward-directed unit normal $\hat{\boldsymbol{n}}_i$, $\mathbf{H}(\lambda_i)$ is a diagonal matrix whose diagonal is 0 for negative eigenvalues and 1 for positive ones, and $\mathbf{I}$ is the identity matrix. The boundary flux takes the form

$$\mathbf{f}_i^{\mathrm{bc}} = \mathbf{f}(\mathbf{v}_i^c(\hat{\boldsymbol{n}}_i))\,. \tag{3.12}$$

Convergence of (3.4) to steady state is accelerated by local time stepping and agglomeration multigrid. We will use the following notations for the set of all the surface normals associated with the control volumes of the dual mesh:

$$\mathbf{n}_h = \left[\{\boldsymbol{n}_i\}_{i\in\mathcal{V}(\overline{\Omega})}\,,\,\{\boldsymbol{n}_{ij}\}_{\vec{ij}\in\mathcal{E}(\overline{\Omega})}\right], \quad \text{and} \quad \mathbf{n}_h\equiv\mathbf{n}_h(\mathbf{X}_h)\,. \tag{3.13}$$

The definition of the surface normals, $\boldsymbol{n}_i$ and $\boldsymbol{n}_{ij}$, is based on the coordinates of the centroids of the elements and of the element surfaces, and on the coordinates of the mid-points of the edges ij. Recalling that $\mathbf{X}_h$ is the vector of the nodal coordinates, we denote by $\mathbf{n}_h(\mathbf{X}_h)$, in (3.13), that the set $\mathbf{n}_h$ depends on the coordinate of the nodes. For the definitions of $\boldsymbol{n}_i$ and $\boldsymbol{n}_{ij}$ we refer to [4, 1, 2].

3.3 Adjoint of the flow equations in Edge

The adjoint of the discretized Euler equations, (3.4)–(3.12) in steady state and including boundary conditions, are also solved by time marching of a system of ODEs:

$$V_i \frac{\mathrm{d}\mathbf{w}_i^*}{\mathrm{d}t} + \mathbf{R}_i^* = \mathbf{0} \qquad \forall i \in \mathcal{V}(\overline{\Omega})\,, \tag{3.14}$$

until steady state. The complete derivation based on discrete sensitivities can be found in [1, 2]. The following gives expression for the numerical fluxes in the adjoint equation (3.14):

$$\begin{aligned}
\mathbf{R}_i^* &= \sum_{j\in\mathcal{N}_i} \left[\frac{\partial(\mathbf{f}_i\cdot\boldsymbol{n}_{ij})}{\partial\mathbf{w}_i}\right]^T \frac{\left(\mathbf{w}_i^*-\mathbf{w}_j^*\right)}{2} + \sum_{j\in\mathcal{N}_i}\mathbf{d}_{ij}^* \qquad \forall i\in\mathcal{V}(\overline{\Omega})\,,\\
\mathbf{R}_i^* &= \sum_{j\in\mathcal{N}_i} \left[\frac{\partial(\mathbf{f}_i\cdot\boldsymbol{n}_{ij})}{\partial\mathbf{w}_i}\right]^T \frac{\left(\mathbf{w}_i^*-\mathbf{w}_j^*\right)}{2} + \sum_{j\in\mathcal{N}_i}\mathbf{d}_{ij}^*\\
&\quad + \left[\frac{\partial(\mathbf{f}_i^{\mathrm{bc}}\cdot\boldsymbol{n}_i)}{\partial\mathbf{w}_i}\right]^T \mathbf{w}_i^* \qquad \forall i\in\mathcal{V}(\partial\Omega)\,, i\notin\mathcal{V}(\partial\Omega_w)\,,\\
\mathbf{R}_i^* &= \sum_{j\in\mathcal{N}_i} \left[\frac{\partial(\mathbf{f}_i\cdot\boldsymbol{n}_{ij})}{\partial\mathbf{w}_i}\right]^T \frac{\left(\mathbf{w}_i^*-\mathbf{w}_j^*\right)}{2} + \sum_{j\in\mathcal{N}_i}\mathbf{d}_{ij}^*\\
&\quad + \left[\frac{\partial(\mathbf{f}_i^{\mathrm{bc}}\cdot\boldsymbol{n}_i)}{\partial\mathbf{w}_i}\right]^T \mathbf{w}_i^* - \frac{\partial J}{\partial\mathbf{w}_i} \qquad \forall i\in\mathcal{V}(\partial\Omega_w)\,,
\end{aligned} \tag{3.15}$$

where $\mathcal{V}(\partial\Omega_w)$ is the set of nodes at which the cost function J is evaluated given by (3.1), and expressions of the Jacobian matrices are detailed in [1, 2]. The partial derivatives of the cost function at each node in $\mathcal{V}(\partial\Omega_w)$, with respect to the flow solution, are directly obtained from the expressions (3.1)-(3.2). The artificial dissipation flux $\mathbf{d}_{ij}^*$ uses the stencil of $\mathbf{d}_{ij}$, in the flow equations, but applied to the adjoint flow $\mathbf{w}^*$ and with artificial viscosities computed for the flow solution $\mathbf{w}$. The Jacobian of the farfield boundary fluxes is given by:

$$\frac{\partial(\mathbf{f}_i^{\mathrm{bc}}\cdot\boldsymbol{n}_i)}{\partial\mathbf{w}_i} = \frac{\partial\left(\mathbf{f}_i\cdot\boldsymbol{n}_i\right)}{\partial\mathbf{v}_i}\,\mathbf{L}\left(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty\right)\mathbf{H}\left(\lambda_{\mathbf{i}}\right)\mathbf{L}^{-1}\left(\hat{\boldsymbol{n}}_i,\mathbf{v}_\infty\right)\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}\mathbf{w}_i}\,. \tag{3.16}$$

The Jacobian of the Euler wall flux function is

$$\frac{\partial\left(\mathbf{f}_i^{\mathrm{bc}}\cdot\boldsymbol{n}_i\right)}{\partial\mathbf{w}_i} = (\gamma-1)\left[\frac{1}{2}|\boldsymbol{u}_i|^2\,,\,-\boldsymbol{u}_i\,,\,1\right]^T\,. \tag{3.17}$$

3.4 Computation of gradients

Gradient with respect to the dual and primal grid data

Given an arbitrary variation of the dual mesh data $\mathbf{n}_h$ (3.13), and neglecting the effect of the artificial viscosity, the first variation of the cost function (3.1) can be expressed in terms of the solution to the adjoint equations (3.14)-(3.17), and of the numerical fluxes of the Euler equations (3.7)-(3.12):

$$\delta J = -\sum_{\vec{ij}\in\mathcal{E}(\overline{\Omega})}\left(\mathbf{w}_i^*-\mathbf{w}_j^*\right)^T\mathbf{f}_{ij}\cdot\delta\boldsymbol{n}_{ij} - \sum_{i\in\mathcal{V}(\partial\Omega_h)}\mathbf{w}_i^{*T}\mathbf{f}_i^{\mathrm{bc}}\cdot\delta\boldsymbol{n}_i + \frac{\partial J}{\partial\mathbf{n}_h}\delta\mathbf{n}_h\,. \tag{3.18}$$

The proof can be found in [1, 2]. The notations $\delta\boldsymbol{n}_{ij}$ and $\delta\boldsymbol{n}_i$, in (3.18) are the variations of the surface normals, and $\delta\mathbf{n}_h$ denotes the vector of all those variations.

Observing that the objective function (3.1)-(3.2) depends explicitly on the normal vectors $\boldsymbol{n}_i$, where i is the index of a node on the boundary, expression (3.18) gives the gradient of J with respect to the dual data, denoted ∇J_n:

- for all edges $\vec{ij}$
$$(\nabla J_n)_{ij}^T = -\left(\mathbf{w}_i^* - \mathbf{w}_j^*\right)^T \mathbf{f}_{ij}\,, \tag{3.19}$$
- for all node i on a boundary, not on the wing:
$$(\nabla J_n)_i^T = -\mathbf{w}_i^{*T}\mathbf{f}_i^{\mathrm{bc}}\,, \tag{3.20}$$
- for all node i on the wing (where J is evaluated):
$$(\nabla J_n)_i^T = -\mathbf{w}_i^{*T}\mathbf{f}_i^{\mathrm{bc}} + \frac{\partial J}{\partial \boldsymbol{n}_i}\,, \tag{3.21}$$

The partial derivatives in (3.21) are directly obtained from the expression of J and of the aerodynamic coefficients (3.2). Details can be found in [2].

Denoting by N_h the number of mesh nodes, that is $|\mathcal{V}(\overline{\Omega})|$, the gradient ∇J_X is the N_h-by-d vector whose entries are the partial derivatives of J with respect to the nodal coordinates

$$\begin{aligned}(\nabla J_X)_{i,1} &= \frac{\partial J}{\partial x_i}\,,\\ (\nabla J_X)_{i,2} &= \frac{\partial J}{\partial y_i}\,,\\ (\nabla J_X)_{i,3} &= \frac{\partial J}{\partial z_i}\,.\end{aligned}$$

Recalling that the data $\mathbf{n}_h$ depend on $\mathbf{X}_h$, the set of coordinates of all the nodes in the CFD mesh, the gradient of J with respect to the mesh coordinates, denoted ∇J_X, is calculated from (3.19)-(3.21) using the chain rule:

$$\nabla J_X = \left(\frac{\mathrm{d}\mathbf{n}_h}{\mathrm{d}\mathbf{X}_h}\right)^T \nabla J_n + \left(\frac{\partial J}{\partial \mathbf{X}_h}\right)^T\,. \tag{3.22}$$

For a detailed algorithm to calculate ∇J_X we refer to [1, 2].

Gradient with respect to the control points

Given ∇J_X, we calculate the gradient ∇J_P of the cost function J with respect to the control points P_k introduced in Section 2.1.

First, the derivatives with respect to the coefficients $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$ in the RBF-expansion are computed using the chain rule. Since the expansions in the x, y and z directions are independent, the computation of the derivatives in the different directions are independent as well.

The following expansions are valid for the x, y and z coordinate, respectively:

$$\begin{aligned}x &= x^0 + \sum_{i=1}^{N+4} \lambda_i^x \varphi_i\left(\boldsymbol{x}^0\right),\\ y &= y^0 + \kappa(y^0,\alpha)\sum_{i=1}^{N+4} \lambda_i^y \varphi_i\left(\boldsymbol{x}^0\right),\\ z &= z^0 + \sum_{i=1}^{N+4} \lambda_i^z \varphi_i\left(\boldsymbol{x}^0\right),\end{aligned}$$

where $\kappa(y^0,\alpha)$ is given by (3.3) and we have introduced the notation

$$\varphi_i(\boldsymbol{x}^0) = \begin{cases} \omega(\boldsymbol{x}^0)\phi\left(\|\boldsymbol{x}^0 - \boldsymbol{x}^0_{k_i}\|\right), & i = 1,\dots,N \\ \omega(\boldsymbol{x}^0), & i = N+1, \\ x^0\omega(\boldsymbol{x}^0), & i = N+2, \\ y^0\omega(\boldsymbol{x}^0), & i = N+3, \\ z^0\omega(\boldsymbol{x}^0), & i = N+4, \end{cases}$$

and $\lambda^{x,y,z}_{N+i} = \beta^{x,y,z}_i$ for $i = 1,2,3,4$, so all coefficients, in each direction, are gathered in one long vector, e.g. $\boldsymbol{\lambda}^x, \boldsymbol{\lambda}^y, \boldsymbol{\lambda}^z$. We can now differentiate the cost function with respect to $\boldsymbol{\lambda}$

$$\begin{aligned} \frac{\partial J}{\partial \lambda^x_i} &= \sum_{k=0}^{N_h} \frac{\partial J}{\partial x_k}\frac{\partial x_k}{\partial \lambda^x_i} = \sum_{k=0}^{N_h} \frac{\partial J}{\partial x_k}\varphi_i\left(\boldsymbol{x}^0_k\right), \\ \frac{\partial J}{\partial \lambda^y_i} &= \sum_{k=0}^{N_h} \frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial \lambda^y_i} = \sum_{k=0}^{N_h} \frac{\partial J}{\partial y_k}\kappa(y^0_k,\alpha)\varphi_i\left(\boldsymbol{x}^0_k\right), \\ \frac{\partial J}{\partial \lambda^z_i} &= \sum_{k=0}^{N_h} \frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial \lambda^z_i} = \sum_{k=0}^{N_h} \frac{\partial J}{\partial z_k}\varphi_i\left(\boldsymbol{x}^0_k\right). \end{aligned}$$

Second, we compute, also by the chain rule, the derivatives of J with respect to the deflections of the control points $\boldsymbol{v}_k$, defined in §2.1. Using the usual notations, $\mathbf{v}^x, \mathbf{v}^y, \mathbf{v}^z$ denote the vectors of all control points displacements in each of the directions x, y, z.

Denoting by $\widetilde{\mathbf{M}}$ the 2-by-2 block matrix (2.11), we have the relation

$$\widetilde{\mathbf{M}}\begin{pmatrix} \boldsymbol{\lambda}^x & \boldsymbol{\lambda}^y & \boldsymbol{\lambda}^z \end{pmatrix} = \begin{pmatrix} \mathbf{v}^x & \widetilde{\mathbf{v}}^y & \mathbf{v}^z \\ 0 & 0 & 0 \end{pmatrix}$$

so

$$\begin{pmatrix} \boldsymbol{\lambda}^x & \boldsymbol{\lambda}^y & \boldsymbol{\lambda}^z \end{pmatrix} = \widetilde{\mathbf{M}}^{-1}\begin{pmatrix} \mathbf{v}^x & \widetilde{\mathbf{v}}^y & \mathbf{v}^z \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$$

which yields

$$\frac{\partial \lambda^x_j}{\partial v^x_{k_i}} = \frac{\partial \lambda^y_j}{\partial \widetilde{v}^y_{k_i}} = \frac{\partial \lambda^z_j}{\partial v^z_{k_i}} = \widetilde{\mathbf{M}}^{-1}_{ji} \quad \text{for } 1 \le i \le N \text{ and } 1 \le j \le N+4.$$

Here $\widetilde{v}^y_{k_i}$ is defined by

$$\widetilde{v}^y_{k_i} = \frac{v^y_{k_i}}{\kappa(y^0_{k_i},\alpha)}.$$

Therefore

$$\frac{\partial \widetilde{v}^y_{k_i}}{\partial v^y_{k_i}} = \frac{1}{\kappa(y^0_{k_i},\alpha)}.$$

The derivatives with respect to the displacements of the control points $\boldsymbol{v}_k$, defined by (2.5), are the derivatives with respect to the control points coordinates $\boldsymbol{x}_k$, that is, for $1 \le i \le N$:

$$(\nabla J_P)^x_{k_i} = \frac{\partial J}{\partial x_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^x_j}\frac{\partial \lambda^x_j}{\partial x_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^x_j}\widetilde{\mathbf{M}}^{-1}_{ji}, \tag{3.23}$$

$$(\nabla J_P)^y_{k_i} = \frac{\partial J}{\partial y_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^y_j}\frac{\partial \lambda^y_j}{\partial \widetilde{v}^y_{k_i}}\frac{\partial \widetilde{v}^y_{k_i}}{\partial y_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^x_j}\frac{\widetilde{\mathbf{M}}^{-1}_{ji}}{\kappa(y^0_{k_i},\alpha)}, \tag{3.24}$$

$$(\nabla J_P)^z_{k_i} = \frac{\partial J}{\partial z_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^z_j}\frac{\partial \lambda^z_j}{\partial z_{k_i}} = \sum_{j=0}^{N+4} \frac{\partial J}{\partial \lambda^z_j}\widetilde{\mathbf{M}}^{-1}_{ji}. \tag{3.25}$$

Gradient with respect to the design parameters

Given the gradient with respect to the control points (3.25), we calculate here the gradient ∇J with respect to the variables of optimization in (B.1), the parameters $\left[c_{n,i}, \theta_{n,i}, d^+_{n,i}, d^-_{n,i}\right]_{1\leq i\leq n+1}$. It is obtained by the chain rule:

$$
\begin{aligned}
(\nabla J)_{c_{n,i}} &= \sum_{k\in\mathcal{V}_P} \left(\frac{\partial \boldsymbol{x}_k^{\text{new}}}{\partial c_{n,i}}\right) \cdot (\nabla J_P)_k \ , \\
(\nabla J)_{\theta_{n,i}} &= \sum_{k\in\mathcal{V}_P} \left(\frac{\partial \boldsymbol{x}_k^{\text{new}}}{\partial \theta_{n,i}}\right) \cdot (\nabla J_P)_k \ , \\
(\nabla J)_{d^+_{n,i}} &= \sum_{k\in\mathcal{V}_P} \left(\frac{\partial \boldsymbol{x}_k^{\text{new}}}{\partial d^+_{n,i}}\right) \cdot (\nabla J_P)_k \ , \\
(\nabla J)_{d^-_{n,i}} &= \sum_{k\in\mathcal{V}_P} \left(\frac{\partial \boldsymbol{x}_k^{\text{new}}}{\partial d^-_{n,i}}\right) \cdot (\nabla J_P)_k \ ,
\end{aligned}
\tag{3.26}
$$

where the sum is over all the control points ($\mathcal{V}_P$) and the coordinates $\boldsymbol{x}_k^{\text{new}}$ of those points are calculated by (B.1)-(B.5). For this particular parameterization, the partial derivatives of the coordinates vectors $\boldsymbol{x}_k^{\text{new}}$ with respect to the design parameters are obtained at a cost that is negligible in regard to the mesh deformation, using a perturbation technique. The present technique involves complex arithmetic [24], which avoids the round-off errors of divided differences methods.

4 Numerical applications

An important feature of the method presented here is that it allows to lower the CPU time per deformed mesh, which is confirmed by the tests carried out in 3D in Section 4.1. In Section 4.2 we analyse the optimized designs obtained by shape optimization. The differences with the results obtained in [2] come from the different mesh deformation algorithms. Moreover, the method used here involves an approximation of the geometry of the parameterized design. These differences are analysed, numerically, in Section 4.3.

4.1 Efficiency

We compare the RBF method for deforming meshes with a Laplace smoother (DL) as in [2].

The tests are carried out on two unstructured meshes around the M6-wing. The 'Euler mesh' is suitable for inviscid flows calculations and contains about 136000 nodes, in 736000 tetrahedral elements. The wing itself is represented in this mesh by approximately 16000 points. The 'Navier-Stokes mesh' is generated for viscous calculations, and it has about 1 million nodes in 3 millions tetrahedral and prismatic elements. About 19000 points describe the wing in this last mesh. The results of the tests are presented in two tables. Table 4.1 summarizes the calculations on the Euler mesh, and Table 4.2 shows the results on the Navier-Stokes mesh.

The computational times (CPU) are measured in seconds. The gradient calculation involving the RBF method has been detailed in Section 3.4. In the case of the Laplace smoothing the gradient computation is detailed in [2]. In contrast with the RBF method, the DL method requires to solve a large sparse linear system of equations in each of the directions x , y , z. These equations are solved here by a diagonal preconditioned conjugate gradient (PCG) [3]. The CPU times for solving the smoothing equations vary greatly with the tolerance which decides on the convergence of the conjugate gradient methods. This is the reason why we indicated, for all results the time for two tolerances (see the tables). The choice of the tolerances influences the error on the displacement

'Euler mesh'

CPU time (s)	DL (PCG)	RBF (2166)	RBF (1378)	RBF (730)
Inverting matrix	0	34	9	3
Deformation (x,y,z)	66 (36)	17	9	4
Gradient	63 (33)	44	14	4

Table 4.1: CPU time in seconds for the calculation of deformed Euler meshes and gradients, for the Laplace smoother (DL), and the RBF method with the inverse quadratic (IQ) basis function. The number of control points in the RBF method are in parentheses. The numbers in parentheses are for TOL=10^{-7}, otherwise TOL=10^{-14}.

'Navier-Stokes mesh'

CPU time (s)	DL (PCG)	RBF (3602)	RBF (1379)	RBF (653)
Inverting matrix	0	173	9	1
Deformation (x,y,z)	1275 (515)	123	35	17

Table 4.2: CPU time in seconds for the calculation of deformed Navier-Stokes meshes, for the Laplace smoother (DL), and the RBF method with the inverse quadratic (IQ) basis function. The number of control points in the RBF method are in parentheses. The numbers in parentheses are for TOL=10^{-7}, otherwise TOL=10^{-14}.

of the nodes in the interior of the domain. The maximum distance between the deformed meshes indicates this error:

$$||\mathbf{X}_h(TOL = 10^{-14}) - \mathbf{X}_h(TOL = 10^{-7})||_\infty = 8 \times 10^{-9}\,,$$

$$||\mathbf{X}_h(TOL = 10^{-14}) - \mathbf{X}_h(TOL = 10^{-3})||_\infty = 2 \times 10^{-4}\,.$$

A tolerance of 10^{-3} would not be sufficient for deforming meshes for viscous calculations and may produce an additional error when calculating gradients in shape optimization.

Finally, for the deformation of the Euler mesh (Table 4.1), the factor between the CPU times varies from 2 to 15, in favor of the RBF method, depending on the number of control points (RBF method) and the tolerance (DL method). This factor is between 4 and 75, when deforming the Navier-Stokes mesh (Table 4.2), in favor of the RBF method. The differences in CPU times between the tests on the Euler mesh and the test on the Navier-Stokes mesh is likely to be observed for other meshes of the same types independently from the geometries. Indeed, the numerical solution of elliptic equations, like the ones solved for the DL method, is known to be penalized on highly stretched meshes like the meshes for viscous calculations, whereas this does not affect the RBF method.

4.2 Optimization of the ONERA M6 wing

The minimization of the cost function (3.1) aims to produce a design with a lower pressure drag while keeping the lift and pitching moment coefficients constant, at the prescribed Mach number and angle of attack, see Section 3. The values obtained for these functionals before and after optimization are given in Table 4.3 for various parameterizations (varying n in (B.1)). The optimizations carried out here are denoted 'RBF' and the results obtained in reference [2], where a Laplace smoothing is used to deform the mesh, are denoted 'DL'. The control points are chosen according to the rules 1 to 4 in Section 4.3 with $d = 0.47$. The basis function in the calculation was the inverse quadratic (IQ) basis function with relative shape parameter $\varepsilon_{\text{rel}} = 0.3$, and the parameter α defined in (3.3) was set to 1.0. In this case this parameter had no effect since there were no deformation in the y direction.

Figures 4.1 and 4.2 represent the pressure coefficients (C_p) calculated on the designs obtained here by optimization ('RBF') for $n = 1$ and $n = 6$, and on the initial design of the M6 wing. These are interpolated data on cutting planes which location is indicated in % of wingspan, where 0 is at the root and 100 is at the tip.

The reduction of the drag is achieved in all cases by a weakening of the shocks and of the lift-induced drag. In Figure 4.1 we observe that the shock waves are smoothened at all positions on the wing for both parameterizations

	$C_D^0 = 144$ $10^4.\Delta C_D$	$C_L^0 = 0.331$ $\Delta C_L/C_L^0$	$C_M^0 = 0.113$ $\Delta C_M/C_M^0$	$J^0 = 0.0144$ $\Delta J/J^0$
RBF-1s	-30	$-1.382E-02$	$3.133E-03$	$-1.934E-01$
RBF-3s	-33	$-9.597E-03$	$6.429E-04$	$-2.240E-01$
RBF-6s	-33	$-9.466E-03$	$5.196E-04$	$-2.256E-01$
DL-1s	-30	$-1.633E-02$	$1.376E-03$	$-1.927E-01$
DL-3s	-31	$-8.430E-03$	$8.615E-05$	$-2.075E-01$
DL-6s	-31	$-1.162E-02$	$-1.803E-04$	$-2.091E-01$

Table 4.3: Variations of the aerodynamic coefficients and cost function, in comparison to the values at initial design. RBF indicates the present calculations (using RBF-based mesh deformation) and DL indicates similar optimization results from [2]. The notations -1s, -3s, -6s, indicate the number of splines for the parameterization, that is $n = 1$, 3, or 6 in (B.1).

RBF-1s and RBF-6s. The C_p obtained for RBF-3s are not shown because they are very close to the C_p for RBF-6s. The smoothing of the shocks is more important for RBF-6s (and RBF-3s) than for RBF-1s. In Figure 4.3, we can observe a downstream shift of the wingtip vortex, and it appears to be more pronounced as the parameterization is finer (larger n in (B.1)). The plots of the C_p for the results 'DL' from [2] are similar to the C_p curves obtained by the method 'RBF'. This also appears in the summary of the aerodynamic coefficients in Table 4.3.

The differences between the 'RBF' and the 'DL' optimization results are only due to the different mesh deformation algorithms:

- The interpolation method maps the displacements of the control points to the displacements of all the other nodes in the mesh. The control points are chosen here as a subset of the nodes that lay on the wing. That is, for a node on the wing which is not a control point, the displacement is not imposed by the parameterization, but it is mapped by interpolation from the control points. We can also say that the interpolation method, used to deform the mesh, makes a second parameterization of the wing on top of the parameterization of the twist, camber and thicknesses used here. Alternatively, we can consider this as an approximation of the parameterization.

- Suppose that for a given parameterization, the interpolation would be exact at the nodes on the wing that are not control points. Then, the difference between the meshes would be only due to the differences in the mesh qualities, possibly leading to different results to the discretized flow equations, and different aerodynamic coefficients.

These differences are analyzed in the next section.

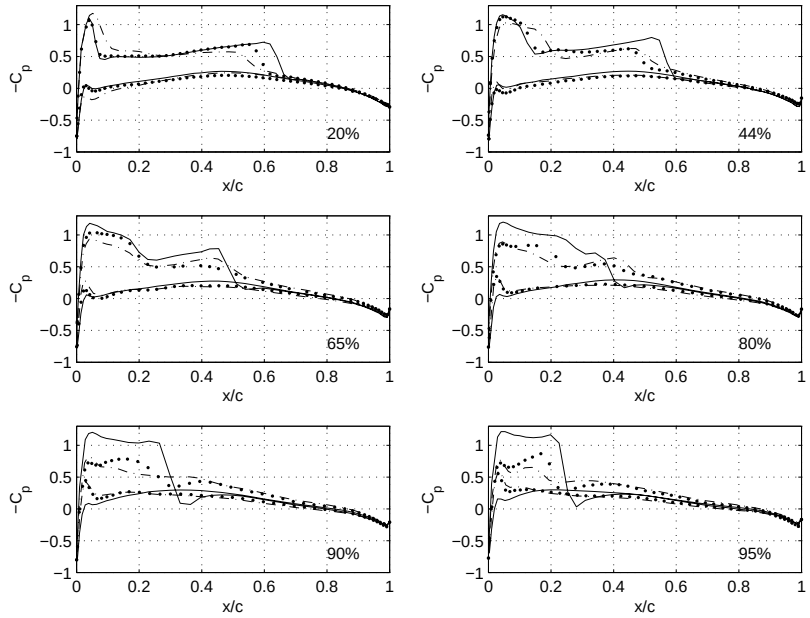


Figure 4.1: Pressure coefficients at original design (line), optimized design RBF-1s (dots) and RBF-6s (dashed line).

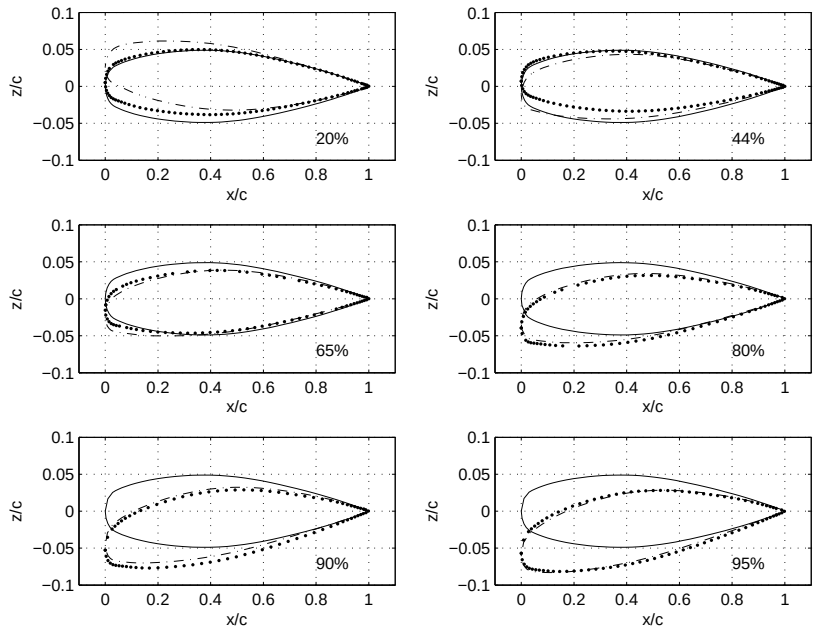


Figure 4.2: Profiles at original design (line), optimized design RBF-1s (dots) and RBF-6s (dashed line).

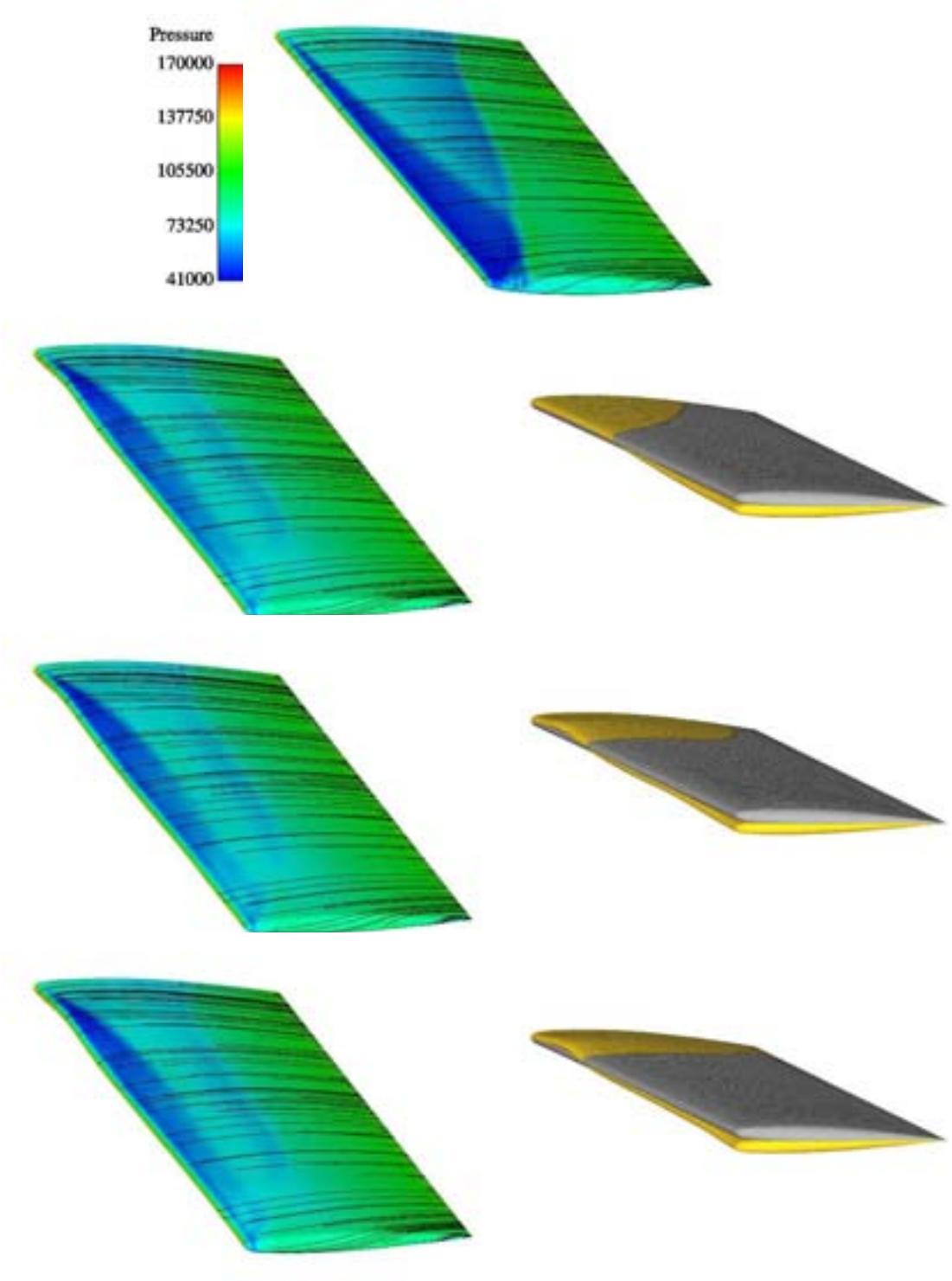


Figure 4.3: Pressure iso-surfaces and particle tracing around the original design (top) and, from the second row to the bottom, left column, the optimized designs RBF-1s, RBF-3s, and RBF-6s. In the right column in grey (dark grey in BW) the shape at initial design and in yellow (light grey in BW) the optimized shape.

d_{rel}	d (m)	Number of points
1.0	0.236093	2166
1.3	0.306965	1763
1.5	0.354205	1378
1.7	0.401341	920
2.0	0.472333	730
2.5	0.590232	505
3.0	0.708748	322
3.5	0.82673	219

Table 4.4: The number of control points for different values of the distance d.

4.3 Mesh qualities and interpolation accuracy

In this section, we test the mesh deformation method with respect to interpolation accuracy and mesh quality for different RBFs, sets of control points and shape parameters. The input to the mesh deformation program is the boundary nodes for the optimized wing RBF-3s given i Table 4.3. The tested basis functions are: inverse quadratic (IQ), multiquadric (MQ), inverse multiquadric (IMQ), Gaussian (GS) and the compactly supported Wendland function W_{33}.

The interpolation properties for RBFs are best on even distribution of control points. In our application, the control points is a subset of the mesh points on the wing which, for usual meshes, have inhomogeneous distribution. An algorithm is devised here in order to extract control points from the set of mesh nodes with an approximately even distribution:

1. Choose a minimal distance d between control points. We relate the value of d to the mesh density on the wing

$$d = d_{\text{rel}} h_{\mathcal{V}(\partial\Omega_w)}.$$

   where the mesh density is measured by

$$h_{\mathcal{V}(\partial\Omega_w)} = \max_{i\in\mathcal{V}(\partial\Omega_w)} \min_{j\in\mathcal{V}(\partial\Omega_w), i\neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2.$$

2. Assume that all points on the wing are given in a long array. Pick the first point in the array to be the first control point.

3. The next control point is chosen as the point in the array whose distance to all previously chosen control points is at least d.

4. Continue until reaching the end of the array.

Table 4.4 shows the relation between d_{rel}, d, and the number of control points. The shape parameter ε for the RBF, defined in Section 2.2, should be related to the density of the control points. Too small values of ε give ill conditioned matrices whereas too large values give poor interpolation properties. Therefore we relate the shape parameter to the approximate distance d between control points

$$\varepsilon = \varepsilon_{\text{rel}}/d.$$

where ε_{rel} is a dimensionless quantity.

The quality of the mesh deformation is measured with respect to two different objectives. First, since not all points on the wing are used there will be a discrepancy between the points on the wing in the deformed mesh and

the deformed points on the wing given as in data. If all points on the wing were used as control points then this discrepancy would of course be zero. The discrepancy will be measured in both l^2 and l^∞ norm. Second, the quality of the mesh is essential for accurate calculations. Mesh quality can be measured in a number of different ways. Here we will use three different measures:

1. Let r be the radius of the largest inscribed sphere in a tetrahedra and let R the radius of the circumscribed sphere and let $\rho = 3r/R$. For a equilateral tetrahedra this ratio is 1.

2. The maximal dihedral angle χ of an element. For an ideal equilateral tetrahedra this is 72°.

3. The aspect ratio η is defined as the quotient between lengths of the longest and the shortest edge.

In Table 4.5 to 4.14 we give the results for five different basis functions and two different values of d_{rel} , 1.3 and 2.0, and with the relative shape parameter varying from 0.1 to 1.0 in step of 0.1 except for 0.7 and 0.9 which are excluded. If we first look at the approximation properties, we see that the l^2 norm of the discrepancies are in general very small, ranging from one to a couple of millimeters. The l^∞ norms indicates that there are some point whose discrepancy are about two centimeters. However, if we compare this with the length of the wing, which is $15m$, the relative error is of the order 10^{-3}, which should be sufficient for most engineering applications. It seems as small values of the shape parameter give better interpolation results.

For the mesh quality, all results are compared relative the undeformed mesh. For the radii-ratio, dihedral and aspect ratio measures, we define for each element $i = 1, \ldots, N_{el}$, the quantities

$$\begin{aligned} \text{rad}_i &= \frac{\rho_i}{\rho_i^0}, \\ \text{aps}_i &= \frac{\eta_i}{\eta_i^0}, \\ \text{dihed}_i &= \chi_i - \chi_i^0. \end{aligned}$$

The average and maximum of these quantities are given in all tables with subscripts av and max, respectively. Except for a few cases, these measures indicate that the qualities of the deformed meshes are similar to the qualities of the initial undeformed mesh. For some basis functions problems occur for small shape parameters. For example, for IMQ with $d_{\text{rel}} = 2.0$ and $\varepsilon_{\text{rel}} = 0.2$, the aspect ratio quotient is $\text{aps}_{\max} = 13.8$ which implies that the quality of some cells is very bad. As can be seen from the tables, there are few results for relative shape parameters less than 0.3 since the deformation has resulted in inverted cells in these cases.

ε_{rel}	rad_{av}	$\text{rad}_{\max}$	dihed_{av}	$\text{dihed}_{\max}$	asp_{av}	$\text{asp}_{\max}$	l^2	l^∞
0.3	0.99	1.40	0.9°	23.4°	1.02	1.74	0.0008	0.0139
0.4	0.99	1.29	0.8°	14.1°	1.02	1.36	0.0009	0.0172
0.5	0.99	1.29	0.8°	13.0°	1.02	1.36	0.0011	0.0186
0.6	0.99	1.32	0.8°	12.3°	1.02	1.35	0.0012	0.0191
0.8	0.99	1.36	0.8°	13.9°	1.02	1.38	0.0014	0.0216
1.0	0.99	1.39	0.8°	15.1°	1.02	1.46	0.0015	0.0242

Table 4.5: IQ $d_{\text{rel}} = 1.3$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.4	0.98	1.47	1.0°	26.2°	1.02	1.95	0.0008	0.0141
0.5	0.99	1.28	0.9°	19.0°	1.02	1.46	0.0009	0.0161
0.6	0.99	1.28	0.9°	15.0°	1.02	1.36	0.0009	0.0173
0.8	0.99	1.27	0.8°	12.2°	1.02	1.33	0.0010	0.0185
1.0	0.99	1.26	0.8°	11.8°	1.02	1.31	0.0011	0.0191

Table 4.6: MQ $d_{\text{rel}} = 1.3$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.3	0.98	1.52	1.0°	27.6°	1.02	2.15	0.0007	0.0129
0.4	0.99	1.29	0.8°	16.7°	1.02	1.40	0.0009	0.0164
0.5	0.99	1.27	0.8°	13.2°	1.02	1.35	0.0010	0.0180
0.6	0.99	1.27	0.8°	12.2°	1.02	1.34	0.0011	0.0189
0.8	0.99	1.29	0.7°	11.5°	1.02	1.31	0.0012	0.0193
1.0	0.99	1.30	0.7°	11.7°	1.02	1.30	0.0013	0.0208

Table 4.7: IMQ $d_{\text{rel}} = 1.3$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.5	0.98	1.65	1.2°	41.7°	1.03	3.53	0.0006	0.0100
0.6	0.99	1.48	0.8°	21.7°	1.02	1.66	0.0008	0.0131
0.8	0.98	1.53	0.9°	22.2°	1.02	1.78	0.0012	0.0189
1.0	0.98	1.58	1.1°	27.8°	1.03	2.11	0.0017	0.0260

Table 4.8: GS $d_{\text{rel}} = 1.3$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.1	0.98	1.51	1.0°	29.7°	1.03	2.26	0.0007	0.0135
0.2	0.99	1.49	0.8°	18.9°	1.02	1.60	0.0010	0.0177
0.3	0.98	1.58	1.1°	27.8°	1.03	2.12	0.0017	0.0253
0.4	0.97	1.65	1.3°	37.9°	1.04	2.86	0.0028	0.0385
0.5	0.97	1.65	1.6°	47.2°	1.04	4.21	0.0048	0.0597

Table 4.9: Wend33 $d_{\text{rel}} = 1.3$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.2	0.98	1.58	1.1°	38.4°	1.03	2.81	0.0012	0.0145
0.3	0.99	1.25	0.8°	11.5°	1.02	1.28	0.0016	0.0213
0.4	0.99	1.26	0.8°	10.4°	1.02	1.25	0.0019	0.0230
0.5	0.99	1.27	0.8°	10.4°	1.02	1.26	0.0020	0.0248
0.6	0.99	1.28	0.7°	10.9°	1.02	1.28	0.0022	0.0267
0.8	0.99	1.32	0.7°	11.8°	1.02	1.33	0.0025	0.0310
1.0	0.99	1.34	0.7°	12.9°	1.02	1.35	0.0028	0.0355

Table 4.10: IQ $d_{\text{rel}} = 2.0$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.3	0.99	1.39	1.0°	21.1°	1.02	1.49	0.0014	0.0175
0.4	0.99	1.24	0.9°	12.2°	1.02	1.29	0.0016	0.0197
0.5	0.99	1.24	0.8°	10.5°	1.02	1.25	0.0017	0.0203
0.6	0.99	1.23	0.8°	9.9°	1.02	1.24	0.0017	0.0206
0.8	0.99	1.22	0.8°	9.6°	1.02	1.23	0.0018	0.0211
1.0	0.99	1.22	0.8°	9.4°	1.02	1.22	0.0019	0.0215

Table 4.11: MQ $d_{\text{rel}} = 2.0$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.2	0.97	1.64	1.6°	56.1°	1.04	13.79	0.0011	0.0121
0.3	0.99	1.25	0.9°	12.5°	1.02	1.30	0.0016	0.0204
0.4	0.99	1.23	0.8°	10.4°	1.02	1.25	0.0018	0.0219
0.5	0.99	1.25	0.8°	9.9°	1.02	1.24	0.0019	0.0230
0.6	0.99	1.25	0.8°	9.9°	1.02	1.24	0.0020	0.0242
0.8	0.99	1.26	0.7°	10.3°	1.02	1.25	0.0022	0.0267
1.0	0.99	1.28	0.7°	10.7°	1.02	1.27	0.0024	0.0292

Table 4.12: IMQ $d_{\text{rel}} = 2.0$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.5	0.99	1.47	0.8°	19.7°	1.02	1.50	0.0015	0.0199
0.6	0.99	1.41	0.8°	15.2°	1.02	1.44	0.0019	0.0251
0.8	0.99	1.45	0.8°	17.4°	1.02	1.54	0.0026	0.0325
1.0	0.99	1.51	0.9°	20.2°	1.02	1.64	0.0032	0.0417

Table 4.13: GS $d_{\text{rel}} = 2.0$

ε_{rel}	rad_{av}	rad_{max}	dihed_{av}	$\text{dihed}_{\text{max}}$	asp_{av}	asp_{max}	l^2	l^∞
0.1	0.99	1.32	0.8°	13.5°	1.02	1.34	0.0015	0.0184
0.2	0.99	1.39	0.8°	14.8°	1.02	1.48	0.0022	0.0260
0.3	0.98	1.51	0.9°	20.2°	1.02	1.64	0.0032	0.0412
0.4	0.98	1.59	1.0°	27.2°	1.03	1.93	0.0046	0.0583
0.5	0.98	1.62	1.2°	33.5°	1.03	2.48	0.0077	0.0790
0.6	0.98	1.65	1.3°	38.5°	1.03	3.20	0.0133	0.1238

Table 4.14: Wend33 $d_{\text{rel}} = 2.0$

5 Summary and perspectives

5.1 Summary

The method presented here performs deformation of meshes, structured or unstructured, in 2D or 3D, based on an interpolation method. It was be verified on numerical examples that the qualities of the deformed meshes are comparable to the qualities of the initial mesh. Another advantage of the method, when repeated deformations of the flow domain are necessary, is the low computational cost, without increasing the memory requirements.

The method of adjoint, in gradient-based aerodynamic shape optimization, is gaining in popularity because it allows an efficient calculation of the gradients of the objective function. In order to fully take advantage of the method, it requires to formulate the *backward* transformation of the mesh deformation. The backward algorithm, for the mesh deformation, calculates the gradient with respect to the control points when the gradient with respect to the nodal coordinates is provided by the adjoint approach. Therefore a particular effort was made here to present in details all the steps of the mesh deformation mapping and how it is involved in the gradient calculation, in order to allow efficient and straight-forward implementations.

5.2 Perspectives

The application of the same algorithm to the coupling of the CFD and CSM calculations, in aeroelasticity, is the object of current investigations [23]. There, the purpose of the interpolating function is twofold: to transfer displacements of the structural grid, into the CFD grid and to transfer loads from the CFD into the structural counter part in a physically reasonable way.

For the deformation of CFD meshes, as tested here, special constraints are imposed on the interpolation in order to preserve the far-field and symmetry boundaries of the mesh. The derivation of suitable boundary conditions for more complicated geometries than the ones investigated here could be investigated.

Finally, the method could be modified in order to construct an algorithm for the parameterization of geometrical shapes. Algorithms of parameterization play an important role in aerodynamic shape optimization as they strongly influence the outcome of the optimization. Parameterization is not a trivial issue because it determines the regularity of the shape as well as it should enable to formulate industrial-like constraints involving the angle of wedges, the volume, or the thickness at some locations.

Acknowledgement

The authors thank Jonathan Smith for the introduction to the aeroelastic interpolation problem and fruitful discussions during the preparation of this work, as well as Adam Jirasek and Martin Berggren for their comments and suggestions during the preparation of this report.

Bibliography

[1] O. Amoignon, *Adjoint-based aerodynamic shape optimization*, Tech. Report IT Licentiate theses 2003-012, Department of Information Technology, Division of Scientific Computing, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden, October 2003.

[2] O. Amoignon and M. Berggren, *Adjoint of the median dual finite volume method applied to 2D and 3D transonic aerodynamic shape optimization*, (2005), In review.

[3] O. Axelsson, *Iterative Solutions Methods*, second ed., Cambridge University Press, 1996.

[4] T.J. Barth, *Aspects of unstructured grids and finite-volume solvers for the Euler and Navier–Stokes equations*, Special Course on Unstructured Methods for Advection Dominated Flows, AGARD Report 787, May 1991, pp. 6–1–6–61.

[5] A. Beckert and H. Wendland, *Multivariate interpolation for fluid - structure-interaction problems using radial basis functions*, Aerosp. Sci. Technol. **1** (2001), no. 11, 1–11.

[6] M. D. Buhmann, *Radial basis functions*, Acta numerica, 2000, Acta Numer., vol. 9, Cambridge Univ. Press, Cambridge, 2000, pp. 1–38.

[7] ———, *Radial basis functions*, Cambridge Monographs on Applied and Computational Mathematics, vol. 12, Cambridge University Press, Cambridge, 2003.

[8] G.W. Burgreen, O. Baysal, and M.E. Eleshaky, *Improving the efficiency of aerodynamic shape optimization*, AIAA Journal **32** (1994), no. 1, 69–76.

[9] P. Eliasson, *Edge, a Navier–Stokes solver, for unstructured grids*, Tech. Report FOI-R–0298–SE, Swedish Defence Research Agency, Stockholm, November 2001.

[10] J. Elliot and J. Peraire, *Aerodynamic design using unstructured meshes*, AIAA Paper (1996), no. 96-1941.

[11] M.B. Giles and N.A. Pierce, *An introduction to the adjoint approach to design*, Flow, Turbulence and Combustion **65** (2000), 393–415.

[12] G. S. L. Goura, *Time marching analysis of flutter using computational fluid dynamics*, Ph.D. thesis, University of Glasgow, 2001.

[13] Max D. Gunzburger, *Perspectives in flow control and optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[14] M. H. L. Hounjet and J. J. Meijer, *Evaluation of elastomechanical and aerodynamic data transfer methods for non-planar configuration in computational aeroelastic analysis*, Tech. report, National Aerospace Laboratory NLR, 1995.

[15] A. Jameson, W. Schmidt, and E. Turkel, *Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes*, AIAA Paper (1981), no. 81-1259.

[16] Hyoung-Jin Kim, Daisuke Sasaki, Shigeru Obayashi, and Kazuhiro Nakahashi, *Aerodynamic optimization of supersonic transport wing using unstructured adjoint method*, AIAA Journal **39** (2001), no. 6, 0001–1452.

[17] A. Le Moigne and N. Qin, *Variable-fidelity aerodynamic optimization for turbulent flows using a discrete adjoint formulation*, AIAA Journal **42** (2004), no. 7.

[18] M. Nemec and D.W. Zingg, *Towards efficient aerodynamic shape optimization based on the Navier–Stokes equations*, AIAA Paper (2001), no. 2001-2532.

[19] J. Nocedal and S. Wright, *Numerical optimization*, Springer Series in Operations Research, 1999.

[20] O. Pironneau, *Optimal shape design for elliptic systems*, Springer Verlag, 1984.

[21] R. Ripollés, M. Cordero, M. Hermanns, and Valero E., *Spivol: A volume spline interpolation tool for elastomechanical and aerodynamic data transfer problems*, EADS-CASA, 2003.

[22] V. Schmitt and F. Charpin, *Pressure distributions on the ONERA-M6-WING at transonic mach numbers*, Experimental Data Base for Computer Program Assessment, AGARD-AR-138, May 1979, pp. B1–1–B1–44.

[23] J. Smith, *Aeroelastic Functionality in Edge, Initial Implementation and Validation*, Tech. Report FOI-R–1485–SE, Swedish Defence Research Agency, Stockholm, December 2005.

[24] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM Review **40** (1998), no. 1, 110–112.

[25] H. Wendland, *Scattered data approximation*, Cambridge Monographs on Applied and Computational Mathematics, vol. 17, Cambridge University Press, Cambridge, 2005.

A Proof of Theorem 2.1.2

To complete the paper here is a proof of Theorem 2.1.2.

*P*roof. Condition 4, the displacement for a point P depends linearly on the displacements for the control points, can be expressed as

$$\boldsymbol{v} = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\boldsymbol{v}_k,$$

where $\mathbf{A}_k(\boldsymbol{x}^0)$, $k = 1, \ldots, N$, are matrix-valued functions. Translated to the coordinates this is

$$\boldsymbol{x} = \boldsymbol{x}^0 + \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\left(\boldsymbol{x}_k - \boldsymbol{x}_k^0\right). \tag{A.1}$$

The translation invariance condition implies that

$$\boldsymbol{x} + \alpha = \boldsymbol{x}^0 + \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\left(\boldsymbol{x}_k + \alpha - \boldsymbol{x}_k^0\right) \tag{A.2}$$

for all $\alpha \in \mathbb{R}^3$. Combined with (A.1), it follows that

$$\alpha = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\alpha, \qquad \forall \alpha \in \mathbb{R}^3$$

Since $\alpha \in \mathbb{R}^3$ is arbitrary, the matrix valued functions must sum up to the identity matrix everywhere

$$\mathrm{I} = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0), \qquad \boldsymbol{x}^0 \in \Omega^0. \tag{A.3}$$

For each rotation operator $\mathbf{R}$ we should have, according to condition (4),

$$\mathbf{R}\boldsymbol{x} = \boldsymbol{x}^0 + \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\left(\mathbf{R}\boldsymbol{x}_k - \boldsymbol{x}_k^0\right).$$

By subtracting equation (A.1) from this relation we obtain

$$(\mathbf{R} - \mathrm{I})\,\boldsymbol{x} = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\,(\mathbf{R} - \mathrm{I})\,\boldsymbol{x}_k. \tag{A.4}$$

Another application of (A.1) yields

$$(\mathbf{R} - \mathrm{I})\,\boldsymbol{x}^0 + (\mathbf{R} - \mathrm{I}) \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\left(\boldsymbol{x}_k - \boldsymbol{x}_k^0\right) = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0)\,(\mathbf{R} - \mathrm{I})\,\boldsymbol{x}_k. \tag{A.5}$$

By putting $\boldsymbol{x}_k = \boldsymbol{x}_k^0$ in (A.4) we can replace the first term in (A.5), which yields

$$(\mathbf{R} - \mathrm{I}) \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0) \left(\boldsymbol{x}_k - \boldsymbol{x}_k^0\right) = \sum_{k=1}^{N} \mathbf{A}_k(\boldsymbol{x}^0) (\mathbf{R} - \mathrm{I}) \left(\boldsymbol{x}_k - \boldsymbol{x}_k^0\right).$$

In terms of the displacements of the control points $\boldsymbol{v}_k$, we have after simplification

$$\sum_{k=1}^{N} \left\{\mathbf{R}\mathbf{A}_k(\boldsymbol{x}^0) - \mathbf{A}_k(\boldsymbol{x}^0)\mathbf{R}\right\} \boldsymbol{v}_k = 0.$$

Since the displacements can be arbitrary, we conclude that

$$\mathbf{R}\mathbf{A}_k(\boldsymbol{x}^0) - \mathbf{A}_k(\boldsymbol{x}^0)\mathbf{R} = 0, \qquad \forall k = 1, \ldots, N.$$

A matrix which commutes with all rotations must be a constant times the identity matrix (the constant may depend on $\boldsymbol{x}^0$). If we denote this constant by $a_k(\boldsymbol{x}^0)$ the formula (2.7) follows. The relation (2.8) is then a consequence of (A.3). Now we look at relation (A.4) with this new information and $\boldsymbol{x}_k = \boldsymbol{x}_k^0$

$$(\mathbf{R} - \mathrm{I}) \left(\boldsymbol{x}^0 - \sum_{k=1}^{N} a_k(\boldsymbol{x}^0) \boldsymbol{x}_k^0 \right) = 0. \tag{A.6}$$

This relation should be true for all rotation operators, which implies that (2.9) follows. The last statement in the proposition is a consequence of (2.8) and (2.9). The proof is complete. □

B Parameterization of the ONERA M6 wing

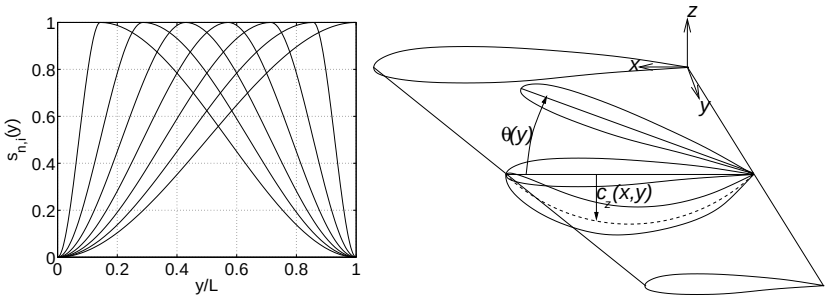


Figure B.1: Left: Basis functions ($n = 6$) for the parameterization of the distributions of twist, camber and thickness, in the spanwise direction (fixed geometry at the root $y = 0$). Right: a sketch representing effects of the twist ($\theta(y)$) about the trailing edge and of a camber deformation on the parameterized shape.

For the sake of the presentation, we derived a parameterization of simple modifications of the geometry. Functions controlling the camber ($c(y)$), the twist about the trailing edge ($\theta(y)$), the relative distance ($d^+(y)$) to the $z = 0$ plane of the nodes in the $z > 0$ half domain, and the relative distance ($d^-(y)$) to the $z = 0$ plane of the nodes in the $z < 0$ half domain, are parameterized in the spanwise direction (y), by $n + 1$ splines (see figure B.1):

$$\begin{aligned} c(y) &= \sum_{i=1}^{n+1} c_{n,i} s_{n,i}(y) \quad \text{(camber)}, \\ \theta(y) &= \sum_{i=1}^{n+1} \theta_{n,i} s_{n,i}(y) \quad \text{(twist)}, \\ d^+(y) &= \sum_{i=1}^{n+1} d^+_{n,i} s_{n,i}(y) \quad \text{(thickness}, z > 0), \\ d^-(y) &= \sum_{i=1}^{n+1} d^-_{n,i} s_{n,i}(y) \quad \text{(thickness}, z < 0), \end{aligned} \tag{B.1}$$

The parameters of the optimization are $c_{n,i}$, $\theta_{n,i}$, $d^+_{n,i}$ and $d^-_{n,i}$, which are real

numbers. The functions $s_{n,i}(y)$ are $\mathcal{C}^1$ piecewise cubic polynomials defined as:

$$\begin{cases} s_{n,i}(y) = 3\left(\dfrac{y}{h^L_{n,i}}\right)^2 - 2\left(\dfrac{y}{h^L_{n,i}}\right)^3 , i \leq n\,,\ y \leq h^L_{n,i}\,, \\ s_{n,i}(y) = 1 - 3\left(\dfrac{y-h^L_{n,i}}{h^R_{n,i}}\right)^2 + 2\left(\dfrac{y-h^L_{n,i}}{h^R_{n,i}}\right)^3 , i \leq n\,,\ h^L_{n,i} \leq y \leq L\,, \\ s_{n,n+1}(y) = 3\left(\dfrac{y}{L}\right)^2 - 2\left(\dfrac{y}{L}\right)^3 ,\ y \leq L\,, \end{cases} \tag{B.2}$$

where L is the span width and:

$$\begin{cases} h^L_{n,i} + h^R_{n,i} = L\,, \\ h^L_{n,i} = \dfrac{iL}{n+1}\,. \end{cases} \tag{B.3}$$

The leading and trailing edges are lines in the plane $z = 0$. The streamwise coordinate is x, and we denote by $x_T(y)$ and $x_L(y)$ the x-coordinate of the point on the trailing edge and on the leading edge, respectively, at coordinate y in the spanwise direction. Thus, the x- and z- coordinates of any point on the parameterized geometry, denoted with upperscript $^{\text{new}}$, if it is in the section y of the wing, is mapped from the reference geometry, denoted (x, z), and the parameters of the camber and twist (B.1) as:

$$\begin{aligned} y^{\text{new}} &= y\,, \\ \tilde{c}(y) &= c(y)\left(\frac{x_4 - x_2}{x_T(y) - x_L(y)}\right)^2 , \text{ (scaling)} \\ c_z(x,y) &= \tilde{c}(y)\,(x - x_T(y))\,(x_L(y) - x)\ , \\ n_z(x,y) &= \frac{1}{\sqrt{1 + \tilde{c}(y)^2\,(x_T(y) + x_L(y) - 2x)^2}}\,, \\ n_x(x,y) &= \tilde{c}(y)\,n_z(x,y)\,(2x - x_T(y) - x_L(y))\ , \end{aligned} \tag{B.4}$$

In (B.4), $c_z(x,y)$ describes a camber line at constant y, and $(n_x(x,y), n_z(x,y))$ are the coordinates of the unit normal vector to this line. We denote by x_4 and x_2 the streamwise coordinate of the leading edge and the trailing edge, respectively, at the tip of the wing. Adding the thicknesses, to the camber and the twist parameterizations yields:

$$\begin{aligned} \tilde{z}(x,y) &= \left(1 - \omega \sin\left(d^+(y)\right)\right) z\,, \text{ if } z \geq 0 \\ \tilde{z}(x,y) &= \left(1 - \omega \sin\left(d^-(y)\right)\right) z\,, \text{ if } z \leq 0 \\ x^{\text{new}}(x,y) &= x_T(y) + \cos(\theta(y))\,(x + \tilde{z}(x,y)\,n_x(x,y) - x_T(y)) + \\ &\qquad \sin(\theta(y))\,(c_z(x,y) + \tilde{z}(x,y)\,n_z(x,y))\ , \\ z^{\text{new}}(x,y) &= -\sin(\theta(y))\,(x + \tilde{z}(x,y)\,n_x(x,y) - x_T(y)) + \\ &\qquad \cos(\theta(y))\,(c_z(x,y) + \tilde{z}(x,y)\,n_z(x,y))\ , \end{aligned} \tag{B.5}$$

where c_z, n_x, n_z are calculated as in (B.4), and $\omega = 0.3$ is used to bound the changes of thickness.

FOI is an assignment-based authority under the Ministry of Defence. The core activities are research, method and technology development, as well as studies for the use of defence and security. The organization employs around 1350 people of whom around 950 are researchers. This makes FOI the largest research institute in Sweden. FOI provides its customers with leading expertise in a large number of fields such as security-policy studies and analyses in defence and security, assessment of different types of threats, systems for control and management of crises, protection against and management of hazardous substances, IT-security an the potential of new sensors.

FOI
Swedish Defence Research Agency
SE-164 90 STOCKHOLM

Tel: +46 8 5550 3000
Fax: +46 8 5550 3100

www.foi.se