Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

MAS

Modelling, Analysis and Simulation

Modelling, Analysis and Simulation

MAS

Airfoil optimization by using the Manifold Mapping method

M. van der Jagt

REPORT MAS-E0709 JUNE 2007

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3703

Airfoil optimization by using the Manifold Mapping method

ABSTRACT

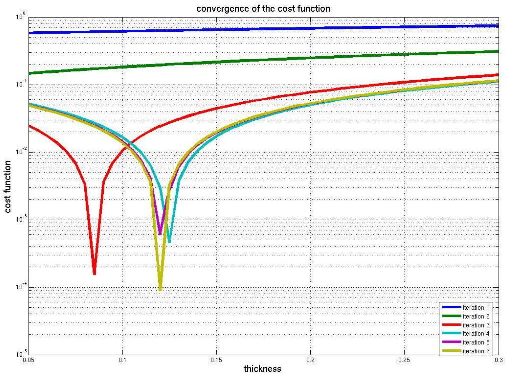
In this report it is investigated if the Manifold Mapping method can be used in airfoil optimization. Before the method can be implemented, a suitable airfoil parametrization must be chosen. Furthermore a coarse and fine model must be assigned. These models are the key to success for the Manifold Mapping method. If two models are chosen that are completely different from each other, the Manifold Mapping will not work properly. Furthermore, if two models are chosen that are very similar to each other, the benefit in reducing computational cost will be only marginal. In this report the Manifold Mapping method will be explained in detail and applied to airfoil design. The approach is validated by using a test case, which will also be explained in detail. Furthermore recommendations and extensions will be given in the last chapter.

2000 Mathematics Subject Classification: 49M99, 49Q99, 65K10, 76D55
Keywords and Phrases: Manifold Mapping; Defect correction; Airfoil optimization; Navier-Stokes; Potential equation; Finite Elements; Panel method; Response Surface Modelling

Airfoil optimization by using the Manifold Mapping method

M.van der Jagt

June 7, 2007



Abstract

In this report it is investigated if the Manifold Mapping method can be used in airfoil optimization. Before the method can be implemented, a suitable airfoil parametrization must be chosen. Furthermore a coarse and fine model must be assigned. These models are the key to success for the Manifold Mapping method. If two models are chosen that are completely different from each other, the Manifold Mapping will not work properly. Furthermore, if two models are chosen that are very similar to each other, the benefit in reducing computational cost will be only marginal. In this report the Manifold Mapping method will be explained in detail and applied to airfoil design. The approach is validated by using a test case, which will also be explained in detail. Furthermore recommendations and extensions will be given in the last chapter.

Keywords: Manifold Mapping, Defect correction, Airfoil optimization, Navier-Stokes, Potential equation, Finite Element, Panel method, Response Surface Modelling

Preface

This report describes the work done by Martin van der Jagt, as a trainee from TU Delft's Faculty of Aerospace Engineering, in CWI's group Scientific Computing and Control Theory, in the period January 15 - April 15, 2007.

In reading this report you might find one or more imperfections. If so, these will be compensated by one or more novelties for aerodynamic shape optimization.

Barry Koren.

Contents

Chapter 1

Introduction

The behavior of an airfoil is investigated by considering the airflow around the airfoil. In aerodynamics it is common to find the resultant aerodynamic force and to decompose it into the direction *perpendicular* to the free stream (lift) and into the direction *parallel* to the free stream velocity (drag).

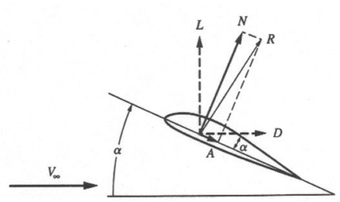


Figure 1.1: Lift (L) and Drag (D) definitions, picture taken from ref [2]

No matter how complex the shape is, there are always two sources contributing to the resulting aerodynamic force. They are the pressure and shear stress distribution along the airfoil. So, if the lift and drag forces have to be computed correctly, both physical mechanisms have to be included. The complete aerodynamic behaviour of the flow around the airfoil is described by the full Navier-Stokes equations. However these equations are quite complicated and difficult to solve. Furthermore accurate numerical simulations take a lot of time. Therefore airfoil optimization for the Navier-Stokes equations may take an extreme amount of work and computational cost. On the other hand simplifications on the Navier-Stokes equations can be introduced to obtain models which are more easy to solve; an example is the potential flow equation, where viscosity (and consequently shear stress) is neglected. A drawback is that an optimized airfoil obtained by such a simplified model will not be the optimized airfoil for the accurate model (the Navier-Stokes equations).

Now the Manifold Mapping comes into play. The simplified model is denoted as the coarse model and the accurate model is denoted as the fine model. The method works as follows: first an optimized airfoil for the coarse model is computed for certain specifications. A specification is for example a prescribed value for the lift coefficient, or a prescribed pressure distribution. Another possible specification is a prescribed maximum thickness or maximum camber. If this airfoil shape meets all the specifications it is linked to the fine model. Actually this mathematical linking (mapping) is the core of the Manifold Mapping method. How this mapping is constructed will be explained in detail in Chapter 4. Next this airfoil shape is evaluated in the fine model and it is checked if the design criteria are fullfilled. If not, the mapping is updated and the procedure is executed again.

In Chapter 2 first the building blocks of the Manifold Mapping algorithm are discussed. The topics that are described are the airfoil parametrization, the coarse model (potential equation and surface response model) and the fine model (incompressible Navier-Stokes equations). Then it is time to explain the Manifold Mapping method. This is done in a detailed mathematical description in Chapter 3. Finally when all the components of the Manifold Mapping are discussed, a program can be written in Comsol Multiphysics. The problem set up is described in Chapter 4. Results for a one dimensional problem are given in Chapter 5, whereas the two dimensional problem is considered in Chapter 6. The possible future applications of Manifold Mapping are discussed in Chapter 7. Finally conclusions are given in Chapter 8.

Chapter 2

Building Blocks of the Manifold Mapping

2.1 Airfoil Parametrization

In this section the airfoil parametrization will be discussed. The airfoil parametrization will determine the design parameters in the Manifold Mapping. As a first start a model suitable for thin airfoils is used. Based on the NACA profiles, the parameters that are chosen are the camber, the position where the camber occurs and the maximum thickness. To clarify these parameters, consider figure 2.1.

The most forward point of the airfoil is called the leading edge, whereas the most rearward point is called the trailing edge. The chord line is the straight line connecting the leading edge and the trailing edge. Next the mean camber line is an imaginary line which lies halfway between the upper surface and lower surface of the airfoil and intersects the chord line at the leading and trailing edges. It is also possible for the mean camber line to intersect the chord line between the leading and trailing edge. Measuring perpendicular to the chord line, the maximum distance between the mean

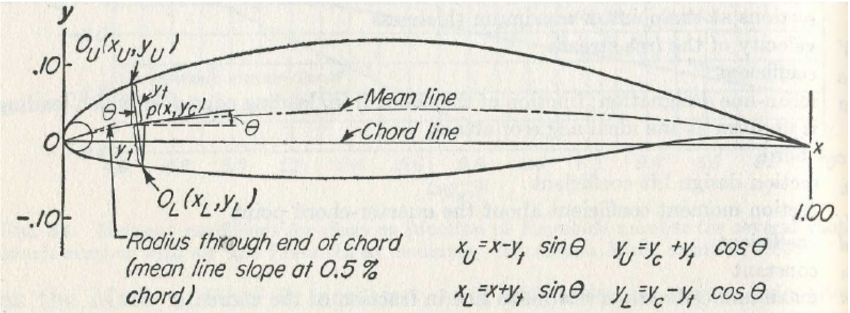

y
.10
$O_U(x_U, y_U)$
y_t
θ
$p(x, y_c)$
θ
Mean line
Chord line
O
y_t
x
$O_L(x_L, y_L)$
-.10
1.00
Radius through end of chord (mean line slope at 0.5 % chord)
$x_U = x - y_t \sin\theta$
$y_U = y_c + y_t \cos\theta$
$x_L = x + y_t \sin\theta$
$y_L = y_c - y_t \cos\theta$

Figure 2.1: Airfoil definitions, picture taken from Theory of wing sections, ref [1]

camber line and the chord line is called the camber of the airfoil. The position where the camber is defined is denoted as x_{cam}. Finally, measuring perpendicular to the chord line, the thickness is the distance between the upper and lower surfaces of the airfoil. For more information on aircraft nomenclature, see ref [2]. Now the contour of the airfoil can be built up as follows; in figure 2.1 it can be seen that for the upper surface the following equation holds:

$$y_u = y_c + y_t cos(\theta). \tag{2.1}$$

Basically it states that the y-coordinate of the upper surface (y_u) is the y-coordinate of the mean camber line (y_c) plus a contribution of the thickness ($y_t cos(\theta)$), where θ is the angle between the tangent of the mean camber line and the horizontal. Similarly for the lower surface it follows that:

$$y_l = y_c - y_t cos(\theta). \tag{2.2}$$

Here y_l is the y-coordinate of the lower surface. Assuming small values of θ it follows that:

$$y_{u/l} = y_c \pm y_t. \tag{2.3}$$

Based on ref [1] [1], the following equation for y_t can be obtained:

$$y_t = \frac{t}{0.20}(0.2982\sqrt{x} - 0.1271x - 0.3579x^2 + 0.2920x^3 - 0.1052x^4). \tag{2.4}$$

Here t is the maximum thickness of the airfoil and x is the running coordinate in x-direction as measured from the leading edge (where $x = 0$). If $x < x_{cam}$ the following equation is used for y_c:

$$y_c = \frac{camber}{x_{cam}^2}(2x_{cam}x - x^2). \tag{2.5}$$

And if $x > x_{cam}$ the following equation is used:

$$y_c = \frac{camber}{(1 - x_{cam})^2}(1 - 2x_{cam} + 2x_{cam}x - x^2). \tag{2.6}$$

By specifying (i) the camber, (ii) the x-position of the camber (x_{cam}) and (iii) the maximum thickness, a unique contour of the airfoil can now be found by using equations (2.3) to (2.6). Therefore these three parameters will be the design variables in the optimization process. Recall that equations (2.3) to (2.6) only apply to thin airfoils where θ is small. The assumption of a small θ is based on the airfoil data in ref [1]. With this assumption the parametrization becomes less complex. To obtain a wider range of airfoils, a different parametrization must be used, as for example in ref [6]. However, the current parametrization forms a good starting point.

[1] The formula on page 113 in ref [1] is valid for $0 \leq x \leq 1.0089$. A small modification is made here to make the equation valid for $0 \leq x \leq 1$

2.2 Fine and coarse model description

2.2.1 Fine model

Description

To describe the complete physical behaviour of a fluid flow three conservation laws can be used: the laws of conservation of mass, momentum and energy. From these conservation laws governing equations for a fluid flow can be derived, namely the Navier-Stokes equations. Although the Navier-Stokes equations are known for a long time already, no general analytical solution has been found yet.

The Navier-Stokes equations are coupled partial differential equations. By making the assumption of *incompressibility* the energy equation can be deleted. To compute the velocity and pressure field only the mass and momentum equation are needed. For an incompressible flow they are (see ref [13]):

Equation of mass

$$\nabla \cdot \, \mathbf{V} = 0. \tag{2.7}$$

Equation of momentum (Navier-Stokes)

$$\frac{D\mathbf{V}}{Dt} = \frac{-1}{\rho}\nabla p + \nu \nabla^2 \mathbf{V}. \tag{2.8}$$

First note that in the second equation body forces are neglected. Secondly equation (2.8) is a single *vector* equation. Finally D/Dt is not just the time derivative but the substantial derivative as defined as:

$$\frac{D..}{Dt} \equiv \frac{\partial ..}{\partial t} + (\mathbf{V} \cdot \nabla).. \tag{2.9}$$

Equations (2.7) and (2.8) will now serve as the fine model. To solve them numerically a finite element mesh is generated in Comsol Multiphysics. The main advantage of this program is that the equations can be implemented easily. Furthermore it is possible to specify geometries and form a (unstructured) grid in a short time. An example of such a grid generated by Comsol Multiphysics is given in figure 2.2.

Two different meshes will be used in the report. First a standard (coarse) mesh as generated by the default setting of Comsol Multiphysics is used. Secondly this mesh is refined to a large extent and this, although more costly, can approximate the equations more accurately.

Next the two dimensional incompressible laminar solver of Comsol Multiphysics is used to evaluate the Navier-Stokes equations.

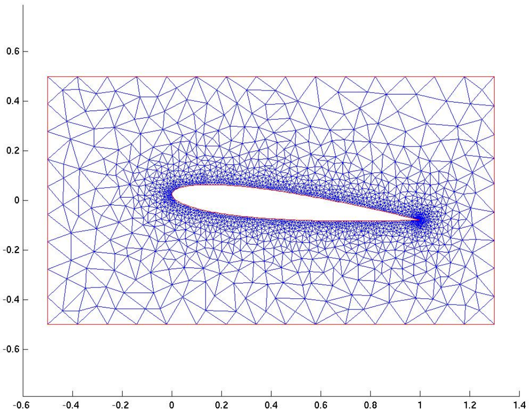


Figure 2.2: Unstructured mesh for an airfoil as generated by Femlab

Computation of the lift coefficient

Basically there are two ways of computing the lift coefficient. First it is possible to examine the pressure and shear stress distribution *on the airfoil* itself. Another way is to consider the integral form of the conservation of momentum for *the complete control volume*. After testing different (cambered) airfoils at zero angle of attack, it appeared that the differences between the two methods are negligible for the accuracy used in this report. The latter approach is used in this report, no difficulty with the integration method of Comsol Multiphysics is expected at higher angles of attack. The basic picture is shown in figure 2.3.

Due to the frictional force there is a defect in the velocity profile at the back of the contour. Furthermore due to the body there will be a different pressure distribution on the lower and upper surface of the control volume. These distributions can be related to the lift and drag as follows. According to ref [2] the integral form of the momentum equation can be written as:

$$\mathbf{R'} = -\oiint_{S} (\rho \mathbf{V} \cdot \mathbf{dS})\mathbf{V} - \iint_{abhi} p\mathbf{dS}. \tag{2.10}$$

The x-component of the resultant force ($\mathbf{R'}$) corresponds to the drag force whereas the y-component corresponds to the lift force. Leaving out all the zero terms, it can be derived that the equation for the lift will be:

$$L = -(\int_h^b [\rho uv + \rho v^2] dy + \int_a^b [p_{upper}] dx - \int_h^i [p_{bottom}] dx). \tag{2.11}$$

Here u is the horizontal component of the velocity and v is the vertical component

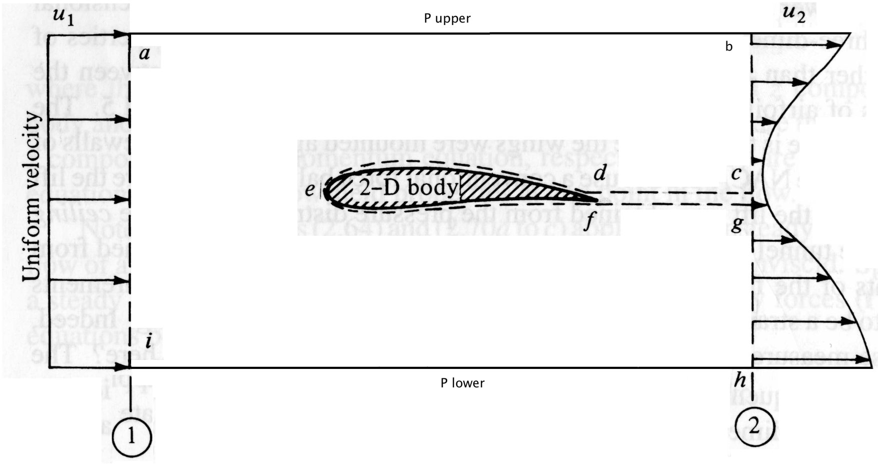


Figure 2.3: Control volume to determine the Lift

of the velocity. Furthermore the surface integral of the pressure reduces to a contour integral since a unit vector in the z-direction is taken. These integrals can be easily evaluated in Femlab. The lift coefficient follows then from [2]:

$$c_l = \frac{L}{\frac{1}{2}\rho_\infty V_\infty^2 S}. \tag{2.12}$$

2.2.2 The potential equation as the coarse model

Introduction

As a coarse model the potential equation is used. This potential equation is solved by means of the panel method. Since the theory behind the panel method is long and well investigated only a brief overview will be given in this section. The theory is by no means easy and for a better understanding it is advisable to read Chapter 3 and 4 of ref [2] as an introduction and Chapter 4 of ref [9] as a more detailed discussion.

Overview

If it is assumed that a flow is irrotational and incompressible, the following equation for the potential can be derived (see ref [2]):

$$\nabla^2\phi = 0. \tag{2.13}$$

The velocities follow from equations (2.14) and (2.15):

$$u = \frac{\partial \phi}{\partial x}, \tag{2.14}$$

[2] For the 2D situation it holds that S=c(1)

$$v = \frac{\partial \phi}{\partial y}. \tag{2.15}$$

These equations are now taken as the coarse model. Depending on the boundary conditions several elementary solutions can be obtained (ref [2]). The solutions that will be used in the panel method are the “uniform flow”, the “source and sink” and the “vortex”. The basic idea is to use combinations of these elementary solutions to simulate a body in a flow [3]. To construct an airfoil the idea is to subdivide the airfoil into straight segments, called source panels, as in figure 2.4. Each source panel has a specific strength which is constant *along* the panel itself. The effect of a source panel is that it induces a velocity at a point P (see figure 2.4). To find the strength of each panel, the point P is taken at the middle of a panel (called the “control point”) and the induced velocity due to all the panels is calculated at that point. In case of an incompressible steady wall, the normal component of the velocity must be zero.

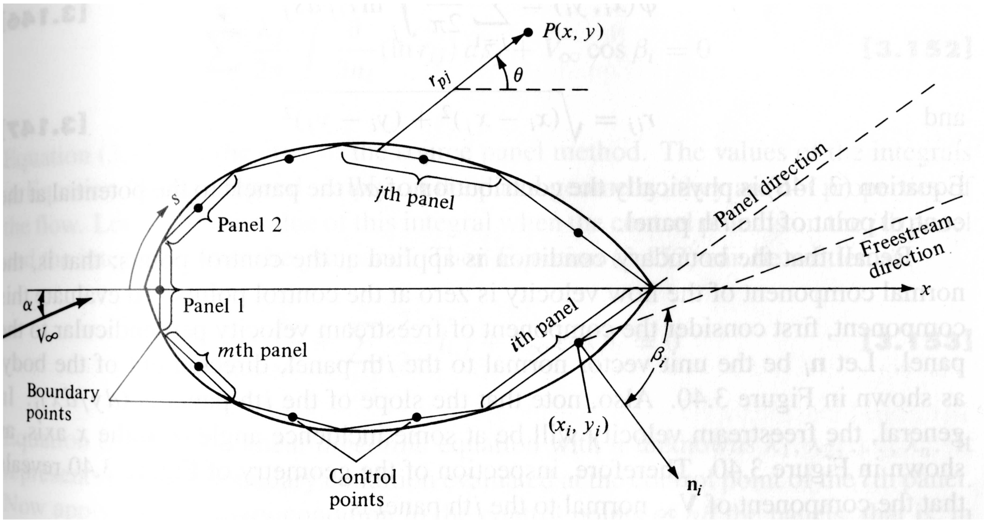


Figure 2.4: Source panel distribution over the surface of the body of arbitrary shape, ref [2]

Now taking the point P on *each* panel, a system of n equations can be formed (velocity induced at the control point of each panel) with n boundary conditions (at each control point the normal velocity should be zero). These equations can then be solved simultaneously to find the right strength of each source panel.

However, the problem is still that there is no lift. The reason behind this is that the Kutta condition is not satisfied (the airflow does not leave the airfoil smoothly at the trailing edge), see ref [2]. Since the physical mechanism (viscosity) is now absent, a different solution must be found. In the panel method used this is solved by imposing

[3] Or in aerodynamic terms:“to obtain a closed pattern of streamlines that display the body”

a constant vortex distribution on all the panels. This introduces an extra unknown (strength of the vortex, Γ_{vortex}); to close the system the Kutta condition is used as an extra boundary condition.

Computation of cl

For an incompressible flow the lift per unit span is given by the Kutta-Joukowski theorem:

$$L' = \rho_\infty V_\infty \Gamma. \tag{2.16}$$

Here Γ is the circulation. Each panel has a vortex imposed on it of strength Γ_{vortex}. Since the vortex strength is the same for each panel, the only thing to do to calculate the total circulation is counting the number of panels. Then it follows that:

$$\Gamma_{total} = n\Gamma_{vortex}. \tag{2.17}$$

Here n is the number of panels. The lift coefficient follows from the definition (equation (2.12)); for S=1:

$$c_l = \frac{2n\Gamma_{vortex}}{V_\infty}. \tag{2.18}$$

2.2.3 Response Surface Model as the coarse model

Response surface models are known for a long time already and a well-established theory exists [4, 12]. These models are built by interpolating a set of discrete sample points of the fine model. The two main aspects of response surface modelling are the choice of interpolating (basis) functions and the sample points, [5].

After choosing the sample points, the response surface model is built up by using so-called multiquadrics. First assume that only one response (say the lift coefficient) is present. Then the multiquadric $q_j(\mathbf{x})$, with $j = 1, \ldots, N$ (where N is the number of sampling points), centered on the point $\mathbf{x}_j \in X$ is defined as

$$q_j(\mathbf{x}) = \sqrt{\| \mathbf{x} - \mathbf{x}_j \|^2 + h}\,. \tag{2.19}$$

As will be explained in the following chapter, the vector $\mathbf{x}$ is the design vector containing parameters like the camber, thickness and x_{cam}. Furthermore $q_j(\mathbf{x})$ is the basis function; it gives a relation between a given point ($\mathbf{x}$) and the j^{th} sample point ($\mathbf{x}_j$). Here the parameter h (called the *shift-parameter*) allows the control of the curvature of $q_j(\mathbf{x})$. The formula for h is given by

$$h = \frac{\sum_{j=1}^{N} \|\mathbf{x}_j\|}{K\,N}. \tag{2.20}$$

Here K is an arbitrary constant. Based on ref [5] a value for K of 50 is used. The value of K is the constant determining the exact curvature. The relation between several points does not have to be linear, and it is this constant that takes care of the non-linearity. The larger the value of K, the more linear the basis function becomes. The coarse model $\mathbf{c}(\mathbf{x})$ can then be expressed as a combination of coefficients and basis functions:

$$\mathbf{c}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \, q_j(\mathbf{x}) \,. \tag{2.21}$$

For a given input $\mathbf{x}$ (the vector containing the maximum thickness, camber and x_{cam}) the coarse model $\mathbf{c}(\mathbf{x})$ returns a value for the lift and moment coefficients. The coefficients α_j are the solution of the following linear system of equations:

$$\mathbf{Q}\,\boldsymbol{\alpha} = \mathbf{F}. \tag{2.22}$$

Here $\mathbf{Q}$ is the matrix of interpolation functions evaluated in the sample points. The entries are given by $\mathbf{Q}_{ij} = q_j(\mathbf{x}_i)$. Furthermore $\boldsymbol{\alpha}$ is the vector of coefficients (i.e: $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_N)$) and $\mathbf{F}$ is the vector of the fine model evaluated in the sample points (i.e: $\mathbf{F} = (\mathbf{f}(\mathbf{x}_1), \mathbf{f}(\mathbf{x}_2), \ldots, \mathbf{f}(\mathbf{x}_N)))$. For an example see appendix A. The extension of this discussion to more responses is straightforward.

So for a given set of sampling points the linear system in equation (2.22) is solved to obtain the vector $\boldsymbol{\alpha}$. Then the coarse model is built up by using equation (2.21). With this generated coarse model the Manifold Mapping can be executed. The more sample points are taken the more accurate the model becomes. However a drawback is that the computational cost increase with increasing number of sample points. As will be mentioned in Chapter 6, every time a new point is calculated for the fine model in the Manifold Mapping this point will also be used as an extra sample point for the coarse model. In this way the coarse model will become more accurate with every iteration step of the Manifold Mapping algorithm.

2.2.4 Summary

In this section the fine and coarse models are described as well as the methods to approximate them. For the fine model the incompressible Navier-Stokes equations are chosen. These equations are approximated by a finite element method. Furthermore a coarse and a fine *mesh* will be used as generated by Comsol Multiphysics.

For the coarse model two options are possible. First the potential equation can be used. This equation is then approximated by the panel method implemented in Matlab. Secondly a response surface model can be taken. This model makes use of several sampling points, which are evaluated in the Navier-Stokes simulation of the fine model.

Chapter 3

Manifold Mapping

In aerodynamics the Navier-Stokes equations describe the complete physical behavior of an airfoil. However designing an airfoil by using the Navier-Stokes equations is time consuming and therefore simpler but less accurate models are used. The goal of Manifold Mapping is to accelerate the design process by using a simplified model (e.g. potential flow) and linking it to the accurate (Navier-Stokes) model. To understand this technique, first the nomenclature will be explained; this will be done in section 3.1. Next the mapping process and the algorithm are discussed in sections 3.2, 3.3 and 3.4. The optimization of an airfoil is taken as an example to clarify the mathematics.

3.1 Nomenclature

In the Manifold Mapping process there are two models, the fine and the coarse model. As has been mentioned, the fine model is accurate but expensive to compute whereas the coarse model is less accurate but easy to compute. In the airfoil optimization process, the full steady Navier-Stokes equations are considered as the fine model and the inviscid, irrotational potential equation is considered as the coarse model.

The design parameters are the parameters that are selected as the optimization parameters. For the airfoil they are the camber, x_{cam} and the maximum thickness as described in Chapter 2. In literature the design parameters are grouped in a vector, $\mathbf{x}$ for the fine model and $\mathbf{z}$ for the coarse model. In principle the design parameters could be different, but for the airfoil design they are the same for the coarse and fine model. However it is still useful to keep this notation since it gives a clear distinction between the coarse and fine model.

Next there are independent variables, i.e parameters that *are not* changed. For this case they are the angle of attack, the temperature and the density at infinity. The vector ξ is associated with these parameters.

Next responses are defined as the outcome of the model. The lift and moment coefficients are used as the responses. The response function is usually denoted as $\mathbf{R(x}, \xi)$ or $\mathbf{f(x)}$ for the fine model. For the coarse model the notation $\mathbf{R(z}, \xi)$ or $\mathbf{c(z)}$

is used. Again $\mathbf{x}$ and $\mathbf{z}$ are the design parameters. Once mentioned, the vector of independent variables may be dropped in the notation for convenience.

Specifications are constraints on the responses, an example would be a prescribed value for the lift coefficient. Other types of specifications would be a prescribed limit on the camber, thickness, etc ... All the specifications are grouped in the vector $\mathbf{y}$, also called the "design specifications vector". A design that does not meet the design specifications is simply called a non-feasible design. Now for both models the cost function is defined as:

$$F(\mathbf{x}) = \| \mathbf{f(x)} \text{ - } \mathbf{y} \|, \tag{3.1}$$

$$C(\mathbf{z}) = \| \mathbf{c(z)} \text{ - } \mathbf{y} \| \, . \tag{3.2}$$

Here $\| \dots \|$ is just the L_2 norm as defined as (see [10]):

$$\| \mathbf{a} \|_2 = [\sum_{k=1}^{N_p} |a_k|^2]^{\frac{1}{2}}, \tag{3.3}$$

where $\mathbf{a}$ is a vector of N_p elements. Actually the cost function is a measure to determine how close the responses are to the specifications. So it is a way to measure the error.

The ultimate goal of system design is to find an optimum for the design parameters of the fine model, $\mathbf{x}$. This means that the design parameters are chosen in such a way that they minimize the error. To put it a mathematical form:

$$\mathbf{x}^* = \underset{\mathbf{x} \in X}{argmin}[U(\mathbf{x})]. \tag{3.4}$$

Here $\mathbf{x}^*$ is the optimal design parameter vector and $U(\mathbf{x})$ is a measure for the error. The function *argmin* means that the argument ($\mathbf{x}$) should be taken such that it minimizes the function $U(\mathbf{x})$. So in airfoil design those values for the camber, x_{cam} and the maximum thickness are taken that result in an airfoil design with lift- and moment coefficients close to the lift- and moment coefficients specified by the designer. Note that this is not the only specification. If the thickness or any other design parameter falls outside the specified design range, the design is also not feasible.

The cost function is a way to express the error, so equation (3.4) could be written as:

$$\mathbf{x}^* = \underset{\mathbf{x} \in X}{argmin} \| \mathbf{f(x)} \text{ - } \mathbf{y} \|. \tag{3.5}$$

To express this more clearly, consider figure 3.1, which is valid for a one dimensional problem. On the x-axis the design parameter (for example the thickness) is shown and on the y-axis the cost function is shown. At a certain point, the cost function reaches a minimum and on that point the error is the smallest. The corresponding $\mathbf{x}^*$ is therefore the optimal value for the design parameter (thickness).

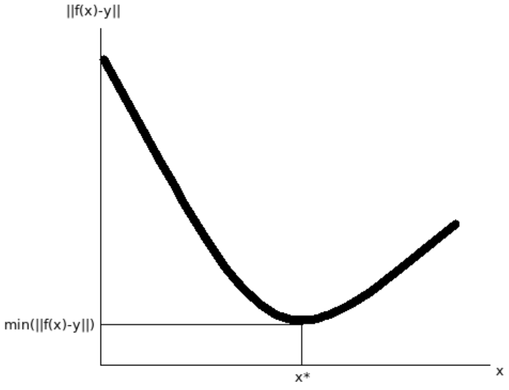


Figure 3.1: Schematic picture of the cost function

3.2 Manifold Mapping, the first steps

As explained in section 3.1 the optimum $\mathbf{x}^*$ must be found to get the best design. In this section it is explained how this is done.

The Manifold Mapping assumes a relation between the fine model parameters and the coarse model parameters. This function is expressed as:

$$\overline{\mathbf{p}}(\mathbf{x}) = \mathbf{z}. \tag{3.6}$$

In case of the airfoil optimization process, the design parameters are the same for both models (both models use the camber, x_{cam} and the maximum thickness to obtain the airfoil shape). Therefore it is logical to take $\overline{\mathbf{p}}(\dots)$ as the identity matrix. From equation (3.6) it then follows that:

$$\mathbf{x} = \mathbf{z}. \tag{3.7}$$

The first step is to find the optimal value for the coarse model, $\mathbf{x}_0$. This value can be found by solving the following equation:

$$\mathbf{x}_0 = \underset{\mathbf{x} \in X}{argmin} \| \mathbf{c}(\mathbf{x}) \text{ - } \mathbf{y} \|. \tag{3.8}$$

This will result in a vector of optimal design variables for the *coarse* model. In practice the evaluation is done by using the Fminsearch function, which is a robust solver. The main advantage is that in two or higher dimensions it always finds a minimum for the cost function.

The next step is to evaluate this coarse model optimum in the fine model. In other words $\mathbf{f}(\mathbf{x}_0)$ has to be computed. In theory it is possible that this optimum is satisfying the design criteria and therefore the following condition must be tested:

$$\| \mathbf{f}(\mathbf{x}_0) \text{ - } \mathbf{y} \| < \epsilon. \tag{3.9}$$

Here ϵ is just some tolerance specified by the designer. If this condition is satisfied, the optimum for the coarse model is the same as the optimum for the fine model and the optimization procedure is ended. However in almost all cases this condition is not satisfied. The next step will be to update the objective, **y**, as follows:

$$\mathbf{y}_1 = \mathbf{c}(\mathbf{x}_0) - (\mathbf{f}(\mathbf{x}_0) - \mathbf{y}). \tag{3.10}$$

The new optimum is then found as:

$$\mathbf{x}_1 = \underset{\mathbf{x}\in X}{argmin} \| \ \mathbf{c}(\mathbf{x}) \ \textbf{-} \ \mathbf{y}_1 \ \|. \tag{3.11}$$

Or, after substituting $\mathbf{y}_1$:

$$\mathbf{x}_1 = \underset{\mathbf{x}\in X}{argmin} \| \ \mathbf{c}(\mathbf{x}) \ \textbf{-} \ \mathbf{c}(\mathbf{x}_0) \ + \ (\mathbf{f}(\mathbf{x}_0)\textbf{-}\mathbf{y}) \ \|. \tag{3.12}$$

The step to obtain $\mathbf{y}_1$ is based on the defect correction process, as explained in ref [7]. To express this more clearly, consider the following non-linear operator equation:

$$Q\mathbf{x} = \mathbf{y}. \tag{3.13}$$

Then ref [7] gives the derivation which results into the following defect correction process:

$$\mathbf{x}_0 = \widetilde{G}_k\mathbf{y}, \tag{3.14}$$

$$\mathbf{x}_{k+1} = \widetilde{G}_{k+1}(\widetilde{Q}_k\mathbf{x}_k - Q\mathbf{x}_k + \mathbf{y}). \tag{3.15}$$

Here $\widetilde{Q}_k$ is the k^{th} approximation to Q. Furthermore $\widetilde{G}_k$ is the left inverse of $\widetilde{Q}_k$. For making the step to Manifold Mapping, note that

$$\widetilde{Q}_k\mathbf{x}_k = \mathbf{c}(\mathbf{x}_k). \tag{3.16}$$

This means that the k^{th} approximation is equal to the coarse model evaluated at the optimum $\mathbf{x}_k$. The second term $Q\mathbf{x}_k$ represents the non linear operator acting on the k^{th} approximation. The non linear approximation corresponds to the fine model:

$$Q\mathbf{x}_k = \mathbf{f}(\mathbf{x}_k). \tag{3.17}$$

Therefore the term in between brackets in equation (3.15) can be written as:

$$\widetilde{Q}_k\mathbf{x}_k - Q\mathbf{x}_k + \mathbf{y} = \mathbf{c}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_k) + \mathbf{y}. \tag{3.18}$$

This is exactly the newly updated **y** in equation (3.10). So the reason behind the updated $\mathbf{y}_1$ is that it is nothing more than a defect correction step.

Returning to equation (3.12), the new coarse model optimum can be found. Then the next step is to evaluate this optimum in the fine model, so computing $\mathbf{f}(\mathbf{x}_1)$. Again a convergence check is necessary, similar as before:

$$\| \ \mathbf{f}(\mathbf{x}_1) \ \textbf{-} \ \mathbf{y} \ \| < \epsilon. \tag{3.19}$$

The steps in the remaining part of the algorithm are similar to the steps described above. However an important difference is present. To understand the reason for this difference it is wise to consider first the aim of Manifold Mapping.

3.3 The aim of Manifold Mapping

In the previous section the following iteration process was used:

$$\mathbf{x}_0 = \underset{\mathbf{x}\in X}{argmin} \| \mathbf{c(x)} \text{ - } \mathbf{y} \|, \tag{3.20}$$

$$\mathbf{x}_1 = \underset{\mathbf{x}\in X}{argmin} \| \mathbf{c(x)} \text{ - } \mathbf{c(x_0)} + (\mathbf{f(x_0)}\text{-}\mathbf{y}) \|. \tag{3.21}$$

Now pretend that this defect correction process is continued until a final fixed point $\mathbf{x}_k$ is obtained. This $\mathbf{x}_k$ is also called the stationary solution $\overline{x}$ (the vector of fine design variables to which the iteration process is converged). Then in ref [7] the following lemma is given:

In case of convergence of (3.21), to a fixed point $\overline{x}$ it holds that:

$$\mathbf{f}(\overline{\mathbf{x}}) - \mathbf{y} \in \mathbf{c}(X)^{\perp}(\overline{\mathbf{x}}). \tag{3.22}$$

Here $\mathbf{c}$(X) is the collection (manifold) of all the possible responses of the coarse model. Now what does this formula mean? It basically states that the vector $\mathbf{f}(\overline{\mathbf{x}}) - \mathbf{y}$ lies perpendicular to the set of all possible responses, $\mathbf{c}$(X). For the airfoil, $\mathbf{c}$(X) is the collection of all lift- and moment coefficients of the coarse model. To clarify this an illustration in 2D is given in figure 3.2.

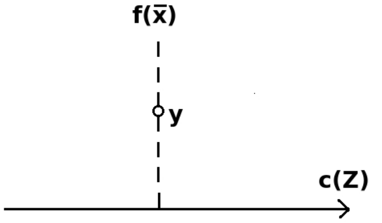


Figure 3.2: Illustration in 2D of the convergence lemma

Now $\mathbf{f}(\overline{\mathbf{x}})$ is just *a* solution of the manifold of the fine model responses, $\mathbf{f}(X)$. However this manifold doesn't have to be parallel to the manifold of the coarse model solutions. This is visualized in figure 3.3. Clearly the distance $\| \mathbf{f}(\overline{\mathbf{x}}) - \mathbf{y} \|$ is not the shortest distance from $\mathbf{y}$ to the manifold $\mathbf{f}$(X). The shortest distance is namely the one that makes an angle of 90 degrees with the manifold $\mathbf{f}$(X). Since the distance is the same as the cost function, it can be concluded that the cost function[1] is not minimized. Therefore, as is mentioned in ref [7], it follows that the fixed point does not coincide with the solution of the fine minimization problem. A physical consequence is that the designed airfoil does not behave as the optimal airfoil. For example its behaviour in lift and moment will not be as good as it can be.

[1] and so the error

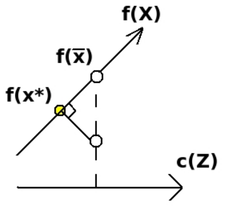


Figure 3.3: Location of the optimum fine model response

To make the fixed point the same as the fine model optimum, it is required that the manifolds of the coarse and fine model responses are parallel to each other. Furthermore it is desirable to have $\mathbf{f}(X)$ and $\mathbf{c}(X)$ as close to $\mathbf{y}$ as possible.

To improve the iteration procedure, the Manifold Mapping is used. Manifold Mapping makes sure that $\mathbf{f}(X)$ and $\mathbf{c}(X)$ are found *parallel in the neighbourhood* of $\mathbf{y}$. The way it operates is that it makes use of a mapping $\mathbf{S}$ such that:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{S}(\mathbf{c}(\mathbf{x})). \tag{3.23}$$

The establishment of this mapping is the important difference which was mentioned in section 3.2. Graphically the Manifold Mapping translates and rotates the manifold $\mathbf{c}(X)$ until it is parallel and close to $\mathbf{f}(X)$, see figure 3.4.

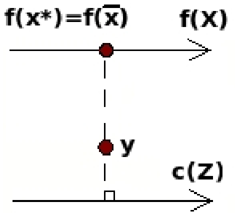


Figure 3.4: Parallel manifolds

3.4 The complete Manifold Mapping technique

3.4.1 Definition

In section 3.3 it was argued that the Manifold Mapping had to be used and the result was equation (3.23). Now the mapping itself is defined as (see ref [7]):

$$\mathbf{S}_k\mathbf{v} = \mathbf{f}(\mathbf{x}_k) + S_k(\mathbf{v} - \mathbf{c}(\mathbf{x}_k)). \tag{3.24}$$

Although this equation seems to come out of the blue, its meaning and construction will be explained in the coming sections. In this equation $\mathbf{v}$ is a vector on which the mapping is working. As will be shown in the following section, $\mathbf{v}$ corresponds to the coarse model $\mathbf{c(x)}$. The first part is the basic level to where the coarse model manifold has to be translated. This is nothing else than the fine model response $\mathbf{f(x}_k\mathbf{)}$. The matrix S_k takes care of the rotation.

Several important aspects are contained in this definition. The first thing to note is that $\mathbf{S}_k$ is the mapping itself whereas S_k is a matrix. Secondly it is interesting to note that after each iteration the mapping is updated.

Furthermore, according to ref [7], the following condition must be satisfied for the matrix S_k:

$$S_k[\mathbf{c(x}_l\mathbf{)}\text{ - }\mathbf{c(x}_k\mathbf{)}] = [\mathbf{f(x}_l\mathbf{)}\text{ - }\mathbf{f(x}_k\mathbf{)}]. \tag{3.25}$$

Here k is the number of the iteration and $l = k-1\ldots max(0, k-n)$. Note that l is in decreasing order.

In equation (3.25) the terms between brackets are matrices, where the columns are built up from $\mathbf{c(x}_l\mathbf{)}$ - $\mathbf{c(x}_k\mathbf{)}$ and $\mathbf{f(x}_l\mathbf{)}$ - $\mathbf{f(x}_k\mathbf{)}$. Therefore to obtain an equation for S_k, equation (3.25) has to be multiplied by the pseudo-inverse of $\mathbf{c(x}_l\mathbf{)}$ - $\mathbf{c(x}_k\mathbf{)}$.[2] Then equation (3.25) can be rewritten as:

$$S_k = [\mathbf{f(x}_l\mathbf{)}\text{ - }\mathbf{f(x}_k\mathbf{)}][\mathbf{c(x}_l\mathbf{)}\text{ - }\mathbf{c(x}_k\mathbf{)}]^\dagger. \tag{3.26}$$

Or more shortly:

$$S_k = \Delta F \Delta C^\dagger. \tag{3.27}$$

3.4.2 Implementation

With this mapping in mind it is now possible to find the new optimum as

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in X}{argmin} \parallel \mathbf{S}_k(\mathbf{c(x)}) - \mathbf{y} \parallel . \tag{3.28}$$

Or by substituting equation (3.24) it follows that:

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in X}{argmin} \parallel \mathbf{f(x}_k\mathbf{)} + S_k(\mathbf{c(x)} - \mathbf{c(x}_k\mathbf{)}) - \mathbf{y} \parallel . \tag{3.29}$$

Continuing by substituting equation (3.27) into (3.24) it follows that:

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in X}{argmin} \parallel \mathbf{f(x}_k\mathbf{)} + \Delta F \Delta C^\dagger(\mathbf{c(x)} - \mathbf{c(x}_k\mathbf{)} - \mathbf{y}) \parallel . \tag{3.30}$$

Or multiplying by the inverse of $\Delta F \Delta C^\dagger$ it follows that:

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}\in X}{argmin} \parallel \mathbf{c(x)} - \mathbf{c(x}_k\mathbf{)} + \Delta C \Delta F^\dagger(\mathbf{f(x}_k\mathbf{)} - \mathbf{y}) \parallel . \tag{3.31}$$

[2] The pseudo-inverse is nothing more than a definition to obtain the inverse of a rectangular matrix. It is expressed by the symbol †. For the definition see [10]

Now one last step can be taken if instead of $\Delta C \Delta F^{\dagger}$ the following expression is used:

$$\Delta C \Delta F^{\dagger} + I - U_c U_c^T. \tag{3.32}$$

The matrix U_c follows from the singular value decomposition:

$$\Delta C = U_c \Sigma_c V_c^T. \tag{3.33}$$

More information of the singular value decomposition can be found in ref [10]. Actually the replacement of $\Delta C \Delta F^{\dagger}$ by equation (3.32) is just a mathematical trick to stabilize the algorithm. As is explained in ref [7], the solution is not changed. Therefore the equation becomes:

$$\mathbf{x}_{k+1} = argmin \parallel \mathbf{c}(\mathbf{x}) - \mathbf{c}(\mathbf{x}_k) + (\Delta C \Delta F^{\dagger} + I - U_c U_c^T)(\mathbf{f}(\mathbf{x}_k) - \mathbf{y}) \parallel . \tag{3.34}$$

Finally this can be rewritten as:

$$\mathbf{x}_{k+1} = \underset{\mathbf{x} \in X}{argmin} \parallel \mathbf{c}(\mathbf{x}) - \mathbf{y}_k) \parallel, \tag{3.35}$$

where:

$$\mathbf{y}_k = \mathbf{c}(\mathbf{x}_k) - (\Delta C \Delta F^{\dagger} + I - U_c U_c^T)(\mathbf{f}(\mathbf{x}_k) - \mathbf{y}). \tag{3.36}$$

3.4.3 Construction of ΔC and ΔF

The matrices ΔC and ΔF can be computed as follows:

1. Compute

$$\Delta \mathbf{c}_i = \mathbf{c}(\mathbf{x}_{k-i}) - \mathbf{c}(\mathbf{x}_k), \tag{3.37}$$

$$\Delta \mathbf{f}_i = \mathbf{f}(\mathbf{x}_{k-i}) - \mathbf{f}(\mathbf{x}_k). \tag{3.38}$$

   Here k is just the iteration number and i runs from i=1... minimum of n and k. Here n is the number of design variables. See appendix B for an example.

2. Construct the matrices ΔC and ΔF by using $\Delta \mathbf{c}_i$ and $\Delta \mathbf{f}_i$ as columns. In appendix B this is shown for the first three iterations in case that n=2.

3.4.4 Convergence

Equations (3.35) and (3.36) are now the steps that are iterated until convergence is reached. In addition to the convergence check in equation (3.9) there are two more checks. All the convergence checks are given below:

$$\parallel \mathbf{f}(\mathbf{x}_k) \text{ - } \mathbf{y} \parallel < \epsilon, \tag{3.39}$$

$$|\mathbf{x}_k - \mathbf{x}_{k-1}| < \eta, \tag{3.40}$$

$$| \parallel \mathbf{f}(\mathbf{x}_k) \text{ - } \mathbf{y} \parallel - \parallel \mathbf{f}(\mathbf{x}_{k-1}) \text{ - } \mathbf{y} \parallel | < \theta. \tag{3.41}$$

As soon as only one of these three criteria is satisfied, the iteration is stopped. In the equations ϵ, η and θ are just some prescribed convergence criteria. The reason for the first check is obvious, here it is checked whether the optimum is satisfying the design criteria. Sometimes it is possible that a designer specifies a lift coefficient that cannot be reached. If this is the case, the first convergence criterion can never be reached. Therefore there are two more convergence criteria. The second check sees if the difference between the optima does not become too small (if it is too small the benefit of a more accurate solution does not account for the amount of work). Lastly it is checked if the difference in the cost functions does not become too small. All these statements are "if" statements.

3.5 Summary

The complete Manifold Mapping process can be summarized by equations (3.20), (3.21) and (3.34), repeated below:

$$\mathbf{x}_0 = \underset{\mathbf{x} \in X}{argmin} \| \ \mathbf{c(x)} \text{ - } \mathbf{y} \ \|, \tag{3.42}$$

$$\mathbf{x}_1 = \underset{\mathbf{x} \in X}{argmin} \| \ \mathbf{c(x)} \text{ - } \mathbf{c(x_0)}+\mathbf{I(f(x_0)}\text{-}\mathbf{y)} \ \|, \tag{3.43}$$

$$\mathbf{x}_{k+1} = \underset{\mathbf{x} \in X}{argmin} \ \| \ \mathbf{c(x)} - \mathbf{c(x}_k) + (\Delta C \Delta F^{\dagger} + I - U_c U_c^T)(\mathbf{f(x}_k) - \mathbf{y}) \ \| \ . \tag{3.44}$$

The third iteration is very similar to the second one, the difference is the matrix in front of $(\mathbf{f(x}_k) - \mathbf{y})$. So basically this is again a (modified) defect correction step. These steps are the core of the Manifold Mapping process and together with the convergence criteria, they are used to find the optimum design vector. A clear and brief overview of the implementation of the Manifold Mapping is given in ref [8] and a modified version for the airfoil optimization can be found in appendix C.

3.6 What is the difference between Space Mapping and Manifold Mapping?

As has been shown in this chapter, the Manifold Mapping uses the manifolds of both models to finally obtain the optimum design vector. The crucial difference with Space Mapping is that the Space Mapping seeks a relation with the design parameters of both models. In case of the airfoil optimization process the collections of all the lift and moment coefficients are used by the Manifold Mapping. If on the other hand the Space Mapping would be used, a mapping will be performed on the maximum thickness, camber and x_{cam}. Therefore Space Mapping sometimes is referred to as an input mapping and Manifold Mapping as an output mapping. Unfortunately the implementation of the Space Mapping is not considered in this report due to time constraints.

Chapter 4

Problem description

Two different set ups are used to validate and test the applicability of the Manifold Mapping. They are as follows:

Set up 1

1. Compute for a *given* airfoil shape the desired responses (Lift, quarter chord moment, etc) with the fine model.
2. Assign these calculated responses as specifications for the Manifold Mapping process and put the initial airfoil shape aside for a moment.
3. Find an airfoil shape by means of the Manifold Mapping process.
4. See if the Manifold Mapping airfoil and the initial airfoil are the same.

This first set up is a check to see if the Manifold Mapping works. If the test is positive, a more interesting approach can be taken:

Set up 2

1. Give certain specifications based on desired properties of the airfoil (a specified lift- and momentcoefficient).
2. Obtain the airfoil shape by means of Manifold Mapping.

To make the process as understandable as possible, first only one design variable is changed. By investigating the outcome (certain pictures for the cost function) important aspects can be visualized which are also present in higher dimensions. In one dimension both set ups as described above are investigated. The next step is to consider more dimensions. The starting point would be to vary two design parameters. Although the complete approach is the same, certain difficulties arise, which were not present in the one dimensional case. For higher dimensions the procedure is exactly the same as in case of two design parameters.

When possible, the Manifold Mapping is compared to other optimization techniques. By looking at the computational time a fair comparison can be made with other conventional methods. The computational cost is expressed in “equivalent fine model evaluations”. The word equivalent is used since the Manifold Mapping (unlike the other methods) makes use of the coarse model and this computational time should also be taken into account. For a coarse *mesh* used in the Navier-Stokes solver, the coarse *model* is approximately 50 times faster as the fine model. So by counting the number of coarse model evaluations the computational cost for the Manifold Mapping can be calculated as follows:

$$\sharp MM_{eq.fine.model.eval.} = \sharp fine.model.eval. + \frac{\sharp coarse.model.eval.}{50}. \tag{4.1}$$

For a fine mesh used in the Navier-Stokes solver, the coarse model is approximately 200 times faster. Therefore for a fine mesh, the computational cost are expressed as:

$$\sharp MM_{eq.fine.model.eval.} = \sharp fine.model.eval. + \frac{\sharp coarse.model.eval.}{200}. \tag{4.2}$$

Chapter 5

Single design variable Manifold Mapping

In this chapter the results of the Manifold Mapping are described in the case of changing one design variable. First the maximum thickness is varied, which showed a semi-linear behaviour in the responses. Secondly the camber is varied as the design parameter. Both cases show interesting properties which will be described in this chapter.

5.1 Single specification

5.1.1 Changing the maximum thickness, known solution

In this case only the lift coefficient (c_l) as a specification will be used. Following the same approach as in Chapter 4, the NACA0012 airfoil was used as a reference airfoil. It is a symmetric airfoil with a maximum thickness of 0.12c. When placing it at an angle of attack of 6 degrees and using a Reynolds number of 500, the value of the lift coefficient can be calculated with the Comsol Multiphysics Navier Stokes solver:

$$\text{fine model response} = [c_l] = [0.3115].$$

This value was obtained without refining the mesh. In accordance to Chapter 4, this value is now assigned as the specification. The goal of Manifold Mapping is thus to find the value for the maximum thickness of 0.12c back. It is interesting to note that there is one design variable and one specification. Since the design can be realized (that means the specification is in the range of fine model responses) the problem is not an optimization problem anymore. A true optimization problem would be to prescribe a lift coefficient that can never be reached. Then the algorithm has to find a design that comes as close to the prescribed lift coefficient as possible. In the approach taken in this paragraph basically a set of equations is solved. However even now the Manifold Mapping can be used to speed things up.

The Manifold Mapping process as described in Chapter 3 can almost be applied. The only resulting part is to set the tolerances. In table 5.1 the tolerances used are

type	$\parallel \mathbf{f}(\mathbf{x}_k)$ - $\mathbf{y} \parallel$	$\lvert \mathbf{x}_k - \mathbf{x}_{k-1} \rvert$	$F(\mathbf{x}_k) - F(\mathbf{x}_{k-1})$	max. it.
tolerance	10^{-3}	10^{-20}	10^{-20}	10

Table 5.1: Tolerances used in the Manifold Mapping code

iter	1	2	3	4	5	6
optimum	0.050195	0.050038	0.084753	0.124348	0.120851	0.120095
F(x)	$1.46 \cdot 10^{-1}$	$1.47 \cdot 10^{-1}$	$7.07 \cdot 10^{-2}$	$7.30 \cdot 10^{-3}$	$1.25 \cdot 10^{-3}$	$9.04 \cdot 10^{-4}$

Table 5.2: Results for the thickness and cost function for one specification after each iteration

displayed. Remember that the first criterion is needed to see if the design criterion is satisfied. The second one sees if the optima of subsequent iterations are not too close to each other. A similar tolerance holds for the cost function, which is displayed in the third column. Finally a tolerance is set on the maximum number of iterations.

Now everything is ready to execute the Manifold Mapping. The results of the Manifold Mapping can be found in table 5.2. From the table it can be seen that within six iterations the solution has converged. In the same table the values of the cost function after each iteration are displayed. The error is reduced within every iteration step. The algorithm was terminated because the cost function reached its tolerance (i.e. $F(\mathbf{x}_k) - F(\mathbf{x}_{k-1}) < 10^{-3}$).

It is interesting to note that the most important steps are taken in the third and fourth iteration. The reason behind this large step is that the gradient is now accurately determined. Since the responses are linear (see figure 5.2), two evaluations are necessary to approximate the gradient. This information is only available after the second iteration.

The behaviour of the cost function of the coarse model can be visualized. This is displayed in figure 5.1 where for each iteration the cost function is plotted against the thickness. Note that this plot is a semi-logarithmic plot. This figure is obtained by evaluating a lot of coarse model responses and calculating the corresponding cost function. The purpose of this figure is to provide an illustration of what is happening[1]. In this figure it can be seen that the optimum values of the first two iterations lie on the left y-axis. This is in agreement with the values in table 5.2. In the following iterations, the optimum is moved closer to the final value (t = 0.12c). The last step is mainly reducing the cost function in such a way that it reaches its tolerance.

Finally an interesting aspect can be found if attention is paid to the graphs of the fine and coarse model responses. In figure 5.2 the value of c_l is plotted against the thickness.

[1] As a side note it can be said that for the Space Mapping the pictures look very similar

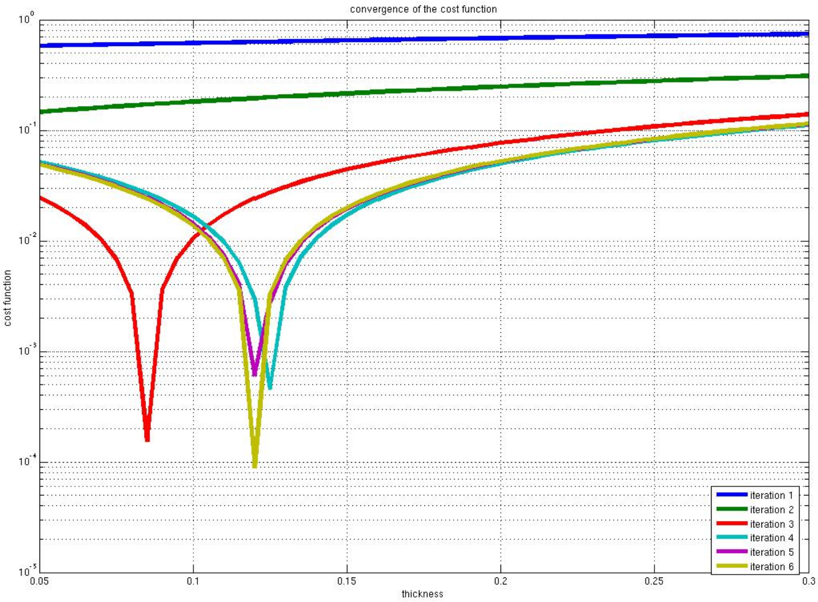


Figure 5.1: Visualized cost functions for each iteration

The behaviour of the two models is completely different (a larger thickness gives a lower value of c_l for the fine model but a higher value for the coarse model). Nevertheless the Manifold Mapping works properly since the manifold of the coarse model can be rotated and translated in such a way that the manifolds of the coarse and fine model are equal in the neigbourhood of the optimum (see Chapter 3). As will be shown in the following chapter, the Manifold Mapping algorithm no longer converges when two design variables are used.

5.1.2 Comparison with other methods

To investigate how well Manifold Mapping performs, it is useful to compare the computational costs and the optimum value found with other methods. As was explained in Chapter 4, the computational cost are expressed in terms of the number of fine model evaluations. To obtain the final converged value in the previous subsection, the Manifold Mapping executed 6 fine model evaluations and approximately 50 coarse model evaluations. The result was obtained with a coarse grid, so using formula (4.1), the equivalent number of fine model evaluations becomes 7. In table 5.3 the comparison is made with other conventional methods.

The values in this table are rounded off to four digits at most for clarity. As can be seen from the table, the Manifold Mapping is clearly faster than the first two

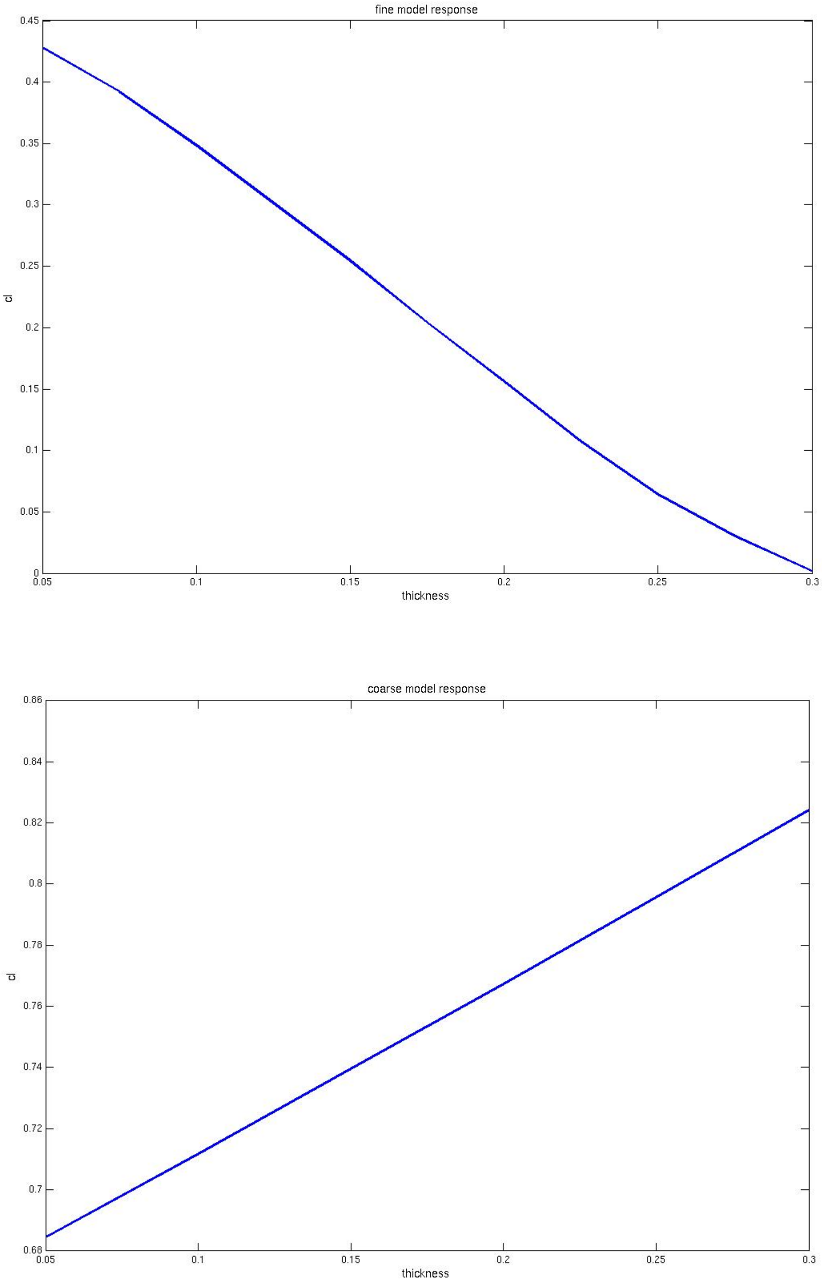


Figure 5.2: Behaviour of the fine and coarse model response

method	Fmincon	Fminsearch	Quasi-Newton	MM
optimum	0.1199	0.1209	0.1216	0.1201
c_l	0.3110	0.3103	0.3058	0.3104
cost function	$1.5 \cdot 10^{-3}$	$3.0 \cdot 10^{-3}$	$4.18 \cdot 10^{-3}$	$4.13 \cdot 10^{-4}$
Fine model eval.	10	18	4	7

Table 5.3: Comparison between various optimization techniques

conventional methods. However the Quasi-Newton method is even faster. The Quasi-Newton method makes use of Broyden's rank one update. Finally it can be said that this comparison is representative for all cases considered in one dimension. Unfortunately the Space Mapping is not considered here due to time constraints.

5.2 Two specifications

5.2.1 Changing the camber, known solution

The next step is to use two specifications and one design variable. The system now becomes overdetermined and this makes the problem truly an optimization problem. Now the NACA2412 airfoil is taken as a reference airfoil to include the camber. This airfoil has a camber of 0.02c. To avoid difficulties like flow separation[2] with a small change in camber (design variable), the angle of attack in this case was lowered to four degrees. The calculated responses of the fine model become for a Reynolds number of 500:

$$\text{fine model response} = [c_l\ c_m] = [0.2455185\ \text{-}0.0163686].$$

An interesting but important aspect becomes clear if a small deviation of this fine model response vector is taken as the specification vector. Taking the specification vector, y, as:

$$\text{y} = [0.2455\ \text{-}0.0164],$$

and using the same tolerances as in paragraph 5.1.1, the results in figures 5.3 and 5.4 are found.

Since only a small deviation of the fine model response vector was taken, it was expected that the solution would converge to a value of the camber close to 0.02c. However the first picture in figure 5.4 shows that iteration 4 overestimates the value. Next, iteration 5 makes a correction and obtains a much smaller cost function, which is desired. Unfortunately the following iteration shows a jump to a much larger value, quite close to the value obtained by the fourth iteration. Much worse is that besides the large deviation of the optimum, the cost function is *also* increased. After doing

[2] And thus a sudden jump in lift (fine model response)

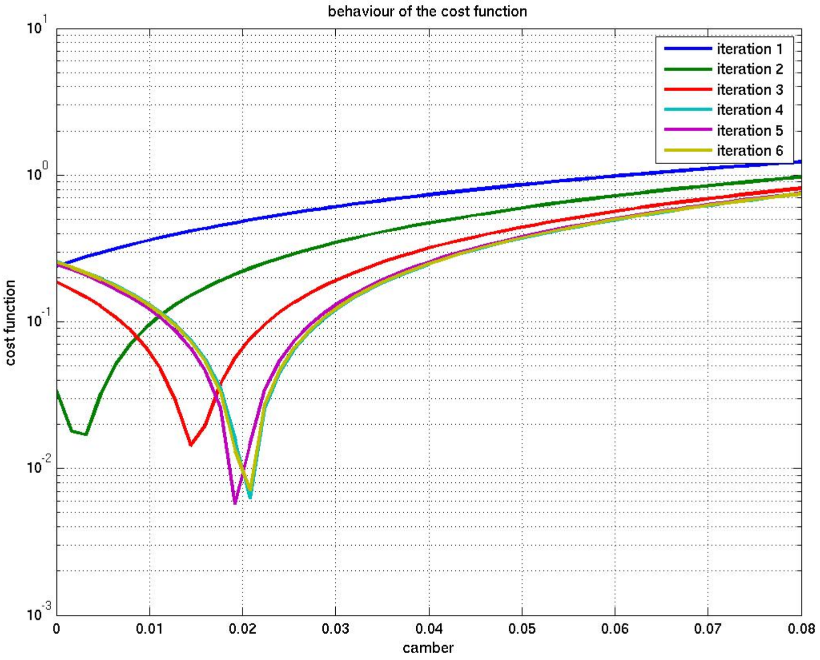


Figure 5.3: Behaviour of the cost function on a coarse mesh

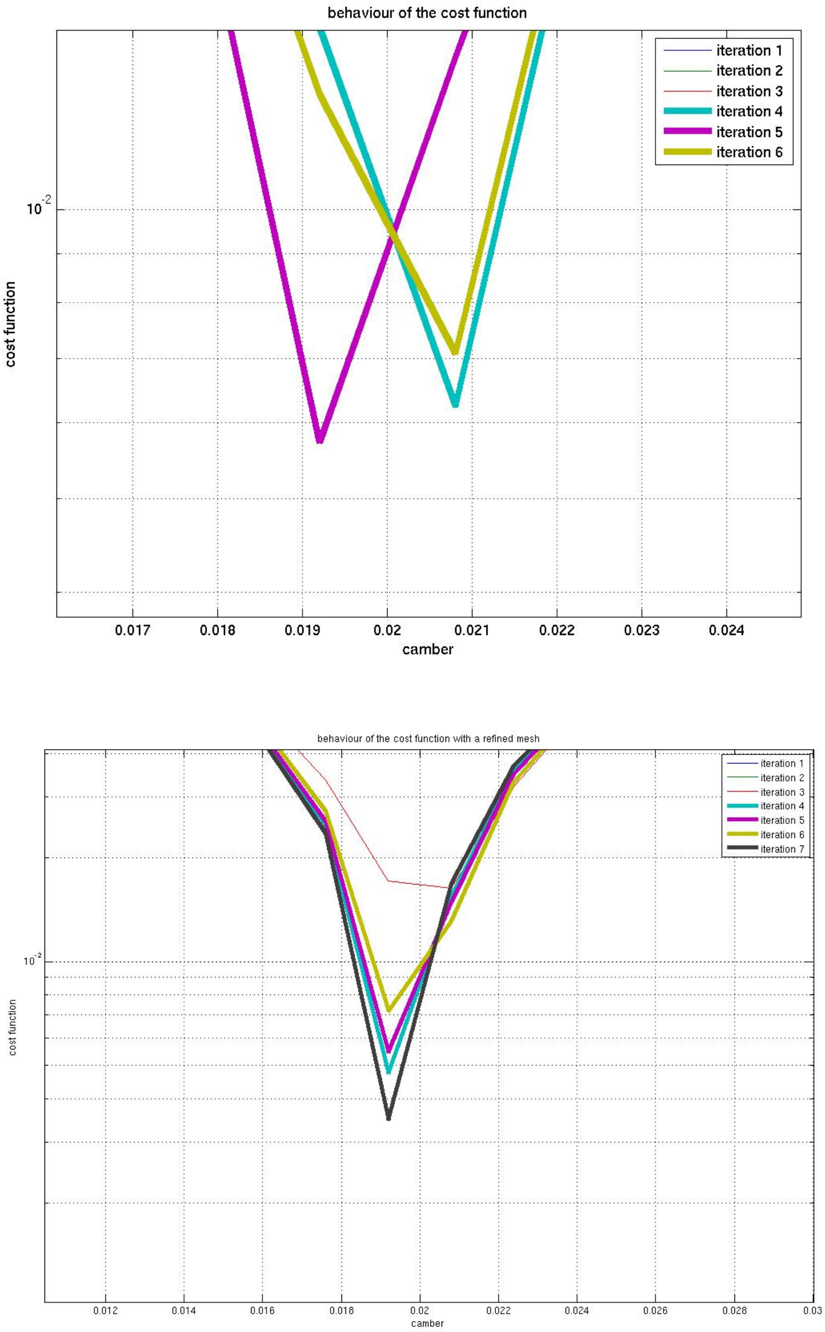


Figure 5.4: Detailed picture of the behaviour of the cost function on a coarse and fine mesh

type	$\parallel \mathbf{f}(\mathbf{x}_k)$ **-** $\mathbf{y} \parallel$	$\lvert \mathbf{x}_k - \mathbf{x}_{k-1} \rvert$	$F(\mathbf{x}_k) - F(\mathbf{x}_{k-1})$	max. it.
tolerance	10^{-3}	10^{-4}	10^{-4}	10

Table 5.4: Tolerances used in the optimization problem

iter	1	2	3	4	5
Camber	$9.54 \cdot 10^{-10}$	$6.47 \cdot 10^{-3}$	$4.39 \cdot 10^{-2}$	$4.24 \cdot 10^{-2}$	$4.15 \cdot 10^{-2}$
F(x)	$8.15 \cdot 10^{-2}$	$7.20 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$	$3.79 \cdot 10^{-2}$	$3.79 \cdot 10^{-2}$
response (c_l)	0.2198	0.2284	0.2785	0.2765	0.2755
response (c_m)	0.00451	-0.00192	-0.04151	-0.03983	-0.03894

Table 5.5: Results for the 1D optimization problem

simulations with more iterations, this oscillating behaviour did not disappear. Even the average cost function did not become smaller.

The solution to this problem is to use a refined mesh. Results obtained with the refined mesh are displayed in the second picture in figure 5.4. Here it can be seen that the solution does not jump back and forth anymore. It always stays close to the same value, near the optimum. There are variations in the magnitude of the cost function visible. However, in the last iteration the magnitude of the cost function is clearly the lowest.

The reason why a refined mesh works and a coarser mesh fails, is that there is a lot of numerical noise present in the coarse grid. This numerical noise is far less present in the case of a fine grid. This is clearly visible in figure 5.5.

So the important message is that a mesh refinement is necessary for obtaining good results.

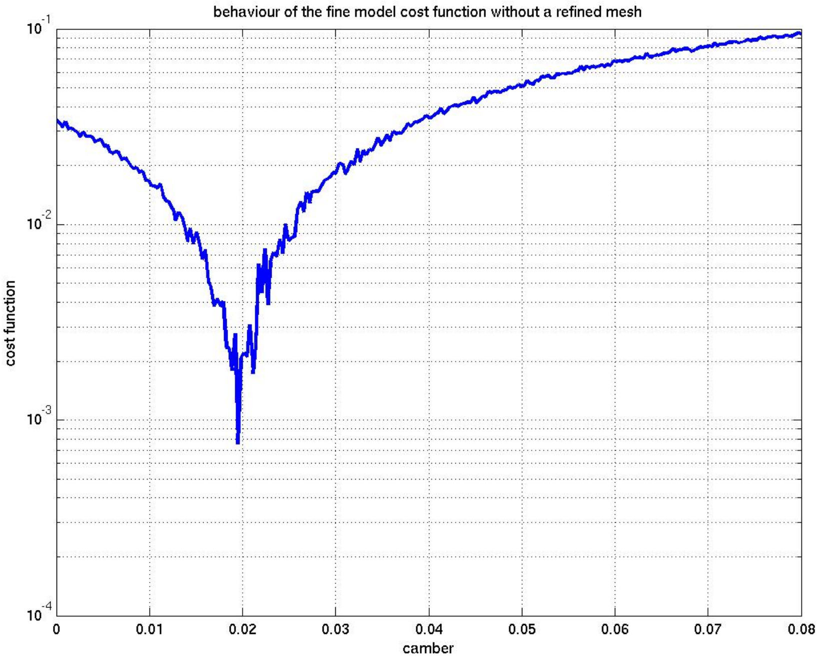
5.2.2 Changing the camber, unknown solution

Keeping in mind that a refined mesh is necessary to obtain good results, an optimization problem for which the solution is not known beforehand can be formulated. As (arbitrary) specifications, the following vector is chosen:

$$\text{y} = [c_l \; c_m] = [0.3000 \text{ -}0.0100].$$

Since it is not likely that the tolerance on the cost function will be met, the tolerances on the difference in the optimum values and responses on subsequent iterations are revisited. Instead of taking the tolerances in table 5.1, a more realistic approach would be to use the tolerances in table 5.4.

The results obtained after running the Manifold Mapping code can be found in table 5.5. The algorithm was terminated because the difference in cost functions met the tolerance requirement. The behaviour of the coarse model cost function can be found in figure 5.6.



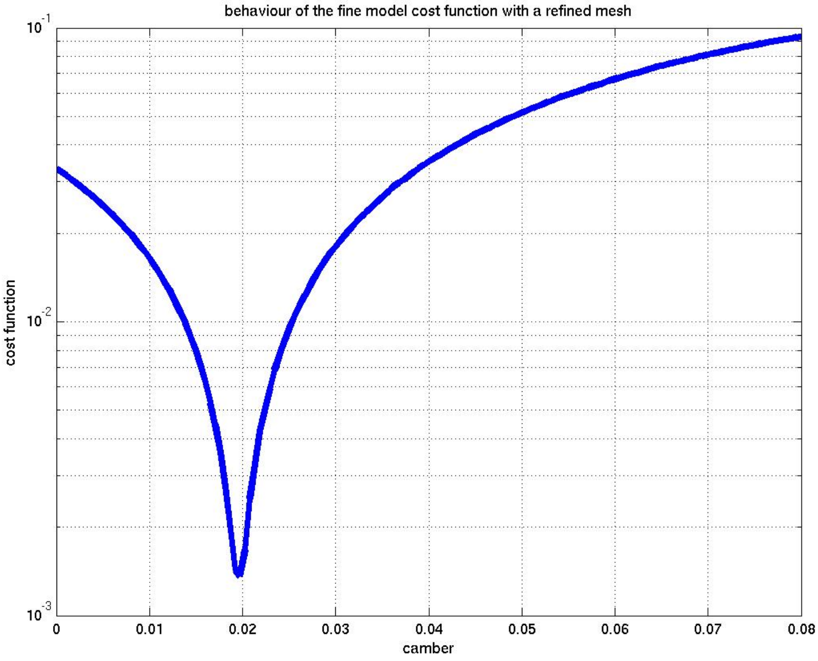


Figure 5.5: Behaviour of the fine model cost function without and with a refined mesh

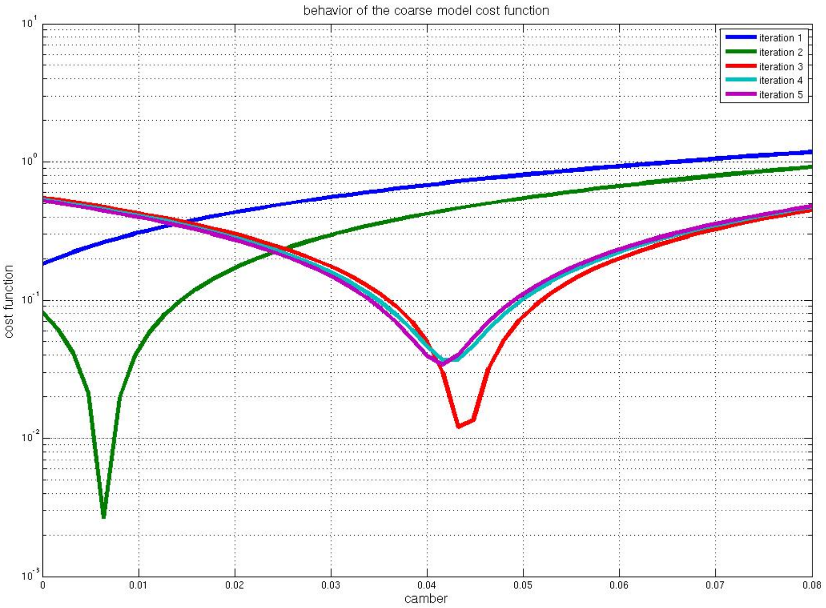


Figure 5.6: Behaviour of the coarse model cost function

As can be seen in the table and figure, the largest step is again taken after the second iteration.

Chapter 6

Manifold Mapping in more dimensions

In this chapter the Manifold Mapping is discussed in more dimensions. As will be shown, large troubles are present. To solve these problems only the first set up as described in chapter 4 is used.

6.1 Two dimensions, two specifications

After applying the Manifold Mapping technique successfully in one dimension, the next step is to vary two design variables. Based on the previous examples, the camber and thickness are chosen. Again the lift- and moment coefficients were taken as the specifications. Since *both* the design variables and the specifications differ an order of magnitude [1], the formula for the cost function is now taken as:

$$cost = \Sigma_{i=1}^{n} \frac{(f_i - y_i)^2}{y_i^2}. \tag{6.1}$$

Here n is the number of specifications. In this way any difficulties due to the difference in order of magnitude are removed. The reference airfoil is again taken as the NACA2412. Therefore the specifications become:

$$y = [c_l \; c_m] = [0.2455 - 0.0164]. \tag{6.2}$$

Thus the goal is now to find a thickness close to 0.12c and a camber close to 0.02c. However there appeared to be a problem when using the same Manifold Mapping code adjusted to two dimensions. The results for the design variables and cost function for the first five iterations are given in table 6.1.

Obviously the solution is not converged to the optimum. Even worse is that also the cost function is increasing, a behaviour that will not disappear if the number of

[1] $camber = O(10^{-2}), thickness = O(10^{-1})$ and $c_l = O(10^{-1}), c_m = O(10^{-2})$

iter	1	2	3	4	5
optimum (camber)	$4.47 \cdot 10^{-3}$	$7.29 \cdot 10^{-3}$	$5.51 \cdot 10^{-8}$	$9.80 \cdot 10^{-8}$	$1.06 \cdot 10^{-7}$
optimum (thickness)	$5.00 \cdot 10^{-2}$	$5.03 \cdot 10^{-2}$	$5.00 \cdot 10^{-2}$	$2.85 \cdot 10^{-1}$	$3.00 \cdot 10^{-1}$
Cost function	$3.17 \cdot 10^{-1}$	$1.66 \cdot 10^{-1}$	$7.76 \cdot 10^{-1}$	$5.12 \cdot 10^{-1}$	$5.15 \cdot 10^{-1}$

Table 6.1: Results of the Manifold Mapping in 2D

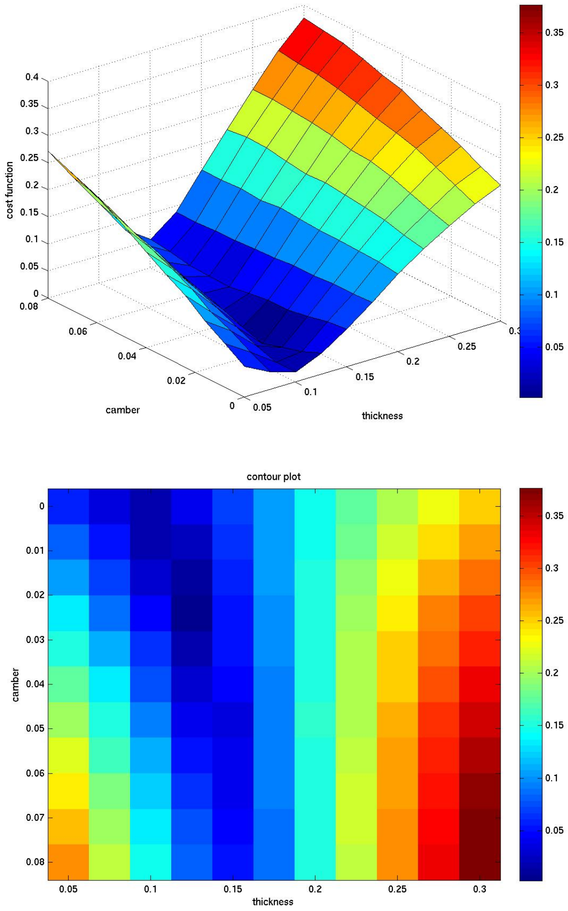


Figure 6.1: Behaviour of the fine model cost function in 2D

iterations is increased or the mesh refined. Another point to be mentioned is that the cost function is oscillating in the first three iterations. The cost function is expected to decrease after the second iteration.

Two underlying problems can be identified. First a region in the design space can be distinguished where the cost function becomes very small. This is illustrated in figure 6.1, where the behaviour of the cost function is displayed. These pictures are obtained by evaluating a lot of fine model responses and calculating the cost function for each point. The first picture in figure 6.1 clearly shows a trough. This is also expressed in the second picture. In the blue area the cost function only deviates a little bit and this poses a problem for the mapping. As will be explained in the following paragraph, the problem is that the pseudo inverse of ΔF becomes very large.

Secondly the behaviour of the responses of the coarse and fine models is quite different (due to the low Reynolds number). In appendix D this is illustrated with several pictures.

6.2 Applying a trust-region

6.2.1 Theory

A solution to the first underlying problem is to apply a trust region. Returning to table 6.1 it can be seen that between iterations 2 and 3, as well as between iterations 3 and 4, a large jump in the optimum is present. To locate the problem, equations (3.35) and (3.36) are repeated below:

$$\mathbf{x}_{k+1} = argmin \parallel \mathbf{c}(\mathbf{x}) - \mathbf{y}_k \parallel, \tag{6.3}$$

$$\mathbf{y}_k = \mathbf{c}(\mathbf{x}_k) - (\Delta C \Delta F^{\dagger} + I - U_c U_c^T)(\mathbf{f}(\mathbf{x}_k) - \mathbf{y}). \tag{6.4}$$

The value of the optimum of iteration k followed from the specification $\mathbf{y}_k$, which on its turn depends on the matrices ΔC and ΔF. The problem is that if the matrix ΔF becomes almost singular, its inverse becomes very large. This in turn forces a large value of $\mathbf{y}_k$ and a jump in the optimum. In the trough, this behaviour is happening. The remedy to this problem is to use a trust region. It operates as follows:

1. Check to see if the matrix ΔF becomes (almost) singular.

2. If so, adjust the matrix ΔF to make it less singular.

3. Obtain a new $\mathbf{y}_k$ and find the corresponding optimum.

4. Calculate the cost function.

5. If the cost function is lower, take this point as the new optimum in the Manifold Mapping. If it is not, reject the point and restart the cycle until the cost function becomes lower than the one obtained in the Manifold Mapping.

iter	1	2	3	4	5
Optimum (camber)	0.0562	0.0001	0.0001	0.0001	0.0001
Optimum (thickness)	0.0500	0.0500	0.0525	0.0500	0.0512
Cost function	14.8435	0.7676	0.7640	0.7569	0.7564

Table 6.2: Results of the Manifold Mapping in 2D with a trust region

Although the trust region approach is computationally more expensive, it eliminates the acceptance of a design vector of which the cost function is higher than its predecessor. A more detailed mathematical description is given in ref [3].

6.2.2 Results

The same Manifold Mapping set-up as in paragraph 6.1 is used to test the trust region approach. The results for 5 iterations can be found in table 6.2. It can be seen that the cost function decreases with each iteration step. This means that after each iteration the solution becomes closer to the true optimum[2]. But unfortunately, the true optimum will never be reached. Furthermore the difference in order of subsequent optima is in the order of 10^{-5}.

Since applying the trust region approach improves the solution only marginally, it is likely that the two models are too different from each other. This was already briefly mentioned in previous paragraphs and illustrated in appendix D. To test this hypothesis, a different coarse model is analyzed and tested. This will be the subject of the next section.

6.3 Response Surface Modelling

6.3.1 Results

Instead of the potential equation as the coarse model, the Response Surface Model as described in paragraph 2.2.3 will be used. To test the response surface modelling model, four points will be initially selected. After this test it will be investigated how many points are necessary to make the model work properly. Furthermore a brief survey will be made to investigate where in the design space they should be chosen. Since there are now two design variables, the design space is a plane which can be easily visualized. In total 4 cases will be considered:

1. four points on the corners of the design space.

2. two points on the diagonal of the design space.

3. two points on the right edge of the design space.

[2] ie: camber = 0.02c and thickness = 0.012c

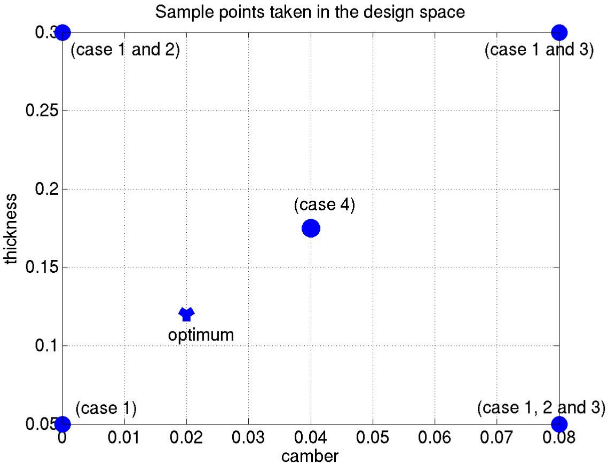


Figure 6.2: Response surface modelling sample points

iter	1	2	3	4	5
Optimum (camber)	0.0159	0.01956	0.0201	0.0199	0.0200
Optimum (thickness)	0.1077	0.1196	0.1198	0.1202	0.1201
Cost function	$1.778 \cdot 10^{-2}$	$4.841 \cdot 10^{-4}$	$1.131 \cdot 10^{-4}$	$4.130 \cdot 10^{-5}$	$1.952 \cdot 10^{-5}$

Table 6.3: Results for the Manifold Mapping in 2D with four sample points

4. one point in the center of the design space.

In figure 6.2 the points are visualized in the design space. The first case is considered to test the response surface model. Since the fine model evaluations can take a lot of time, a brief survey is done to see if the method can become even cheaper by evaluating less points. This is the goal of cases 2,3 and 4. The results for the first case can be found in table 6.3. In the first iterations the value is already in the neigbourhood of the optimum. This is a result of the much better correlation of the fine and coarse model. Furthermore it holds that every time an optimum is calculated and evaluated, it is included in the coarse model as an extra sample point. Going back to the table it can be seen that after every iteration the cost function keeps decreasing. After five iterations the algorithm is terminated since the accuracy of the fine model is reached.

After this successful test, case 2 is considered. The results are found in table 6.4. It can be concluded that with a coarser model (less points) the solution still converges to the optimum. However the convergence is much slower than the previous case. After performing a run with more iterations it appeared that to achieve a cost function of the order of 10^{-5}, nine iterations were necessary.

iter	1	2	3	4	5	6
Camber	0.0800	0.0049	0.0316	0.0250	0.0229	0.0193
Thickness	0.0666	0.0947	0.1291	0.1255	0.1232	0.1171
Cost function	$2.727{\cdot}10^{+1}$	$5.622{\cdot}10^{-1}$	$3.633{\cdot}10^{-1}$	$6.334{\cdot}10^{-2}$	$1.960{\cdot}10^{-2}$	$4.237{\cdot}10^{-4}$

Table 6.4: Results for the Manifold Mapping in 2D with two sample points placed on the diagonal

iter	1	2	3	4	5	6
Camber	0.0720	0.0720	0.0799	0.0330	0.0725	0.0538
Thickness	0.0228	0.0250	0.0234	0.0330	0.0725	0.1131
Cost function	$1.691 \cdot 10^{0}$	$1.504 \cdot 10^{0}$	$2.277 \cdot 10^{0}$	$3.447{\cdot}10^{-1}$	$1.240{\cdot}10^{+1}$	$5.5931{\cdot}10^{0}$

Table 6.5: Results for the Manifold Mapping in 2D with two sample points placed on the right side

Finally the last two cases are considered. The results can be found in tables 6.5 and 6.6. In both cases the model was inappropriate to converge to the optimum. However it must be said that these tests were performed without a trust function. With a trust function no points with a higher value of the cost function will be accepted. Therefore, when admitting enough iterations, the optimum value will eventually be found.

6.3.2 Conclusion on the Response Surface Model

In this section it was demonstrated that when using the Response Surface Model, the Manifold Mapping could be used for two design variables. The reason why the Response Surface Model works and a potential flow equation fails as a coarse model,

iter	1	2	3	4	5	6
Camber	0.0400	0.0401	0.0406	0.0430	0.0637	0.0770
Thickness	0.1750	0.1750	0.1750	0.1745	0.0715	0.1080
Cost function	$2.808{\cdot}10^{-1}$	$2.834{\cdot}10^{-1}$	$2.999{\cdot}10^{-1}$	$4.191{\cdot}10^{-1}$	$1.574{\cdot}10^{+1}$	$1.608 \cdot 10^{1}$

Table 6.6: Results for the Manifold Mapping in 2D with one sample point placed in the middle

Number of points	Placement in design space	Manifold Mapping
4	corners	succeed
2	end points of a diagonal	succeed
2	end points of a side	fails
1	middle	fails

Table 6.7: Different cases considered (without trust region)

is that there is a much better correlation with the fine model in case of the Response Surface Model. However, it appeared that a proper number and placing of the sample points is necessary for convergence of the Manifold Mapping algorithm. In table 6.7 the cases considered are summarized.

6.4 Conclusion

In this chapter it was demonstrated that without a reasonable correlation between the fine and coarse model, the Manifold Mapping will encounter problems. These problems could not be removed when applying a trust region or a mesh refinement. This problem did not occur in one dimension since in the neigbourhood of the solution the models could be made similar by rotating and translating the manifolds. In two and more dimensions this is not possible since *both* the manifolds of the lift and moment coefficients were completely different.

A solution to this problem was to use the Response Surface Modelling approach, which made use of so-called sampling points. Each iteration that followed produced a new optimum which was included in the sample points. In this way the coarse model became better with each iteration. Although this approach was successful, a proper placing of these sampling points is necessary in order to converge. If this is not possible, a trust region must be applied to ensure convergence.

Chapter 7

Recommendations and prospectus

7.1 Recommendations

The simulation of the fine model was done with a low Reynolds number of 500. This caused the huge difference between responses of the fine and coarse model. Ultimately, as was shown in Chapter 6, this leads to a failure of the Manifold Mapping in more than one dimension. For further investigations it is recommended to increase the Reynolds number up to at least 150.000. This is not only necessary to obtain a better correspondence but also to compare results with wind tunnel testing.

Secondly it is advised to use a structured grid instead of the unstructured grid shown in figure 2.2. A brief survey is done with rectangular cells, parallel to the airfoil surface. Figure 7.1 shows several levels of detail of the structured mesh. To obtain this structured mesh, the meshmap functionality of the package of Comsol Multiphysics was used. It appeared that the boundary layer was better captured and this will eventually lead to a better calculation of the coefficients. With a larger Reynolds number, the boundary layer becomes thinner and a capturing with a highly refined structured mesh is desirable.

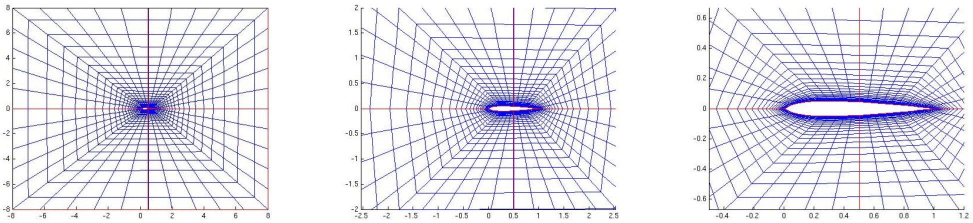


Figure 7.1: Several pictures of the structured mesh generated by Comsol Multiphysics

Next when going to more dimensions a different airfoil parametrization has to be chosen. For this purpose it is recommended to use the approach taken in ref [6]. In this paper the airfoil is parametrized by means of Chebysev polynomials. By adjusting the value of certain constants a large variety of airfoils can be simulated. In this way more cambered and thicker airfoils can be considered. Furthermore the number of degrees of freedom can be easily adjusted.

Finally to include calculations on drag coefficients a simple boundary layer calculation must be included in the panel method. To find a way to implement the boundary layer, it is advised to take a good look on the XFOIL code. Here a boundary layer computation is superposed on a panel method to include drag.

7.2 Prospectus

A natural way of continuing this research is to go to compressible flows. Today's aircraft have a cruising Mach number around 0.8. Increasing the Mach number is thus more close to reality. Therefore a way to optimize airfoils at high Mach numbers is certainly interesting from a business point of view.

A second prospectus might be to consider simple delta wings. Starting with a relatively easy triangular shape, it is possible to find a parametrization of several parameters. For example the sweep angle, the thickness distribution, etc..., can be taken. Next the shape can be optimized in an efficient way by using the Manifold Mapping.

Chapter 8

Conclusion

In this report it was demonstrated that the Manifold Mapping can be successfully applied to airfoil optimization. A careful selection of all the components was necessary to ensure convergence of the algorithm. The choice of the coarse model proved to be a key point. Selecting a coarse model which displays a large difference with the fine model leads to a failure of the Manifold Mapping in dimensions larger than one. This was demonstrated by using the potential equation as the coarse model and the Navier-Stokes equations with a low Reynolds number as the fine model. A solution to this problem was shown to be the Response Surface Modelling. However, when going to more dimensions this could be computationally more costly. Ultimately the cost may become unacceptable. It is likely that this problem will disappear when going to more (larger) practical Reynolds numbers.

Furthermore the topic of numerical accuracy was discussed. With a too coarse grid, the Navier-Stokes solver could only solve the equations up to a certain accuracy. This directly imposed a limit on the accuracy of the optimum found by the Manifold Mapping. To find a useful stopping criterion, knowledge about the accuracy of the solution is therefore necessary.

The trust region approach was also demonstrated. It was shown that with a trust region the cost function would always decrease after each iteration step. The indication that a trust region must be used is the alternating value for the cost function in the Manifold Mapping output obtained *without* a trust region.

When all these aspects were considered the Manifold Mapping proved to be a powerful technique for obtaining an optimum design. Based on the defect-correction approach, it updated the objective of the coarse model after each iteration. Then a new optimum was calculated and the procedure was executed all over again. For the problems discussed in this report, the solution was usually found after five or six iterations. This is much less than other conventional methods as used in Matlab.

The prospectus of the Manifold Mapping in aerospace engineering looks good. Starting from simple problems the complexity can be gradually increased. In this report an airfoil is investigated but for future applications more general and more complex structures can be examined.

Appendix A

Response Surface Model Example

As explained in paragraph 2.2.3, the coarse model (with as output the value of the lift coefficient) can be built up by:

$$\mathbf{c}(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \, q_j(\mathbf{x}). \tag{A.1}$$

Where the coefficients α_j are the solution of the following linear system of equations:

$$\mathbf{Q}\,\boldsymbol{\alpha} = \mathbf{F}. \tag{A.2}$$

Consider h as a given parameter that determines the shape of the basis function, $q_j(\mathbf{x})$. Furthermore:

$$q_j(\mathbf{x}) = \sqrt{\parallel \mathbf{x} - \mathbf{x}_j \parallel^2 + h}. \tag{A.3}$$

For this example, assume that three sample points are present. The matrix $\mathbf{Q}$ gives all the relations between all the sample points. Each row displays the relation of a taken point with respect to the other sample points. The first row corresponds to the first sample point. Its entries display the relation of this sample point to all the other sample points. They can be built up as follows:

$$Q_{11} = q_1(\mathbf{x}_1) = \sqrt{\parallel \mathbf{x}_1 - \mathbf{x}_1 \parallel^2 + h} = \sqrt{h}, \tag{A.4}$$

$$Q_{12} = q_2(\mathbf{x}_1) = \sqrt{\parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel^2 + h}, \tag{A.5}$$

$$Q_{13} = q_3(\mathbf{x}_1) = \sqrt{\parallel \mathbf{x}_1 - \mathbf{x}_3 \parallel^2 + h}. \tag{A.6}$$

The second row corresponds to the second sample point. Again the entries give the relation of this point to all the other points. In a similar way the entries can be built up as follows:

$$Q_{21} = q_1(\mathbf{x}_2) = \sqrt{\parallel \mathbf{x}_2 - \mathbf{x}_1 \parallel^2 + h}, \tag{A.7}$$

$$Q_{22} = q_2(\mathbf{x}_2) = \sqrt{\parallel \mathbf{x}_2 - \mathbf{x}_2 \parallel^2 + h} = \sqrt{h}, \tag{A.8}$$

$$Q_{23} = q_3(\mathbf{x}_2) = \sqrt{\parallel \mathbf{x}_2 - \mathbf{x}_3 \parallel^2 + h}. \tag{A.9}$$

In exactly the same way the equations of the entries of the third row can be found.

Next in equation A.2, **F** is the vector of the fine model responses (the lift coefficients as calculated by the Navier-Stokes equations). As a notation $\mathbf{f}(\mathbf{x}_1)$ is the lift coefficient for the first sample point, $\mathbf{f}(\mathbf{x}_2)$ is the lift coefficient for the second sample point and $\mathbf{f}(\mathbf{x}_3)$ is the lift coefficient for the third sample point. So the complete system of equations becomes:

$$\begin{bmatrix} \sqrt{h} & \sqrt{\parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel^2 + h} & \sqrt{\parallel \mathbf{x}_1 - \mathbf{x}_3 \parallel^2 + h} \\ \sqrt{\parallel \mathbf{x}_2 - \mathbf{x}_1 \parallel^2 + h} & \sqrt{h} & \sqrt{\parallel \mathbf{x}_2 - \mathbf{x}_3 \parallel^2 + h} \\ \sqrt{\parallel \mathbf{x}_3 - \mathbf{x}_1 \parallel^2 + h} & \sqrt{\parallel \mathbf{x}_3 - \mathbf{x}_2 \parallel^2 + h} & \sqrt{h} \end{bmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \end{pmatrix}.$$

This linear system of equations can now be solved to obtain the coefficients α_1, α_2 and α_3. Finally these coefficients can be used to construct the coarse model. For a given input $\mathbf{x}$, the lift coefficient for this example can then be calculated as follows:

$$c_l = \alpha_1 \sqrt{\parallel \mathbf{x} - \mathbf{x}_1 \parallel^2 + h} + \alpha_2 \sqrt{\parallel \mathbf{x} - \mathbf{x}_2 \parallel^2 + h} + \alpha_3 \sqrt{\parallel \mathbf{x} - \mathbf{x}_3 \parallel^2 + h}. \tag{A.10}$$

Appendix B

Construction of ΔC and ΔF

Repeating the formulae for ΔC and ΔF:

$$\Delta C = [c(\mathbf{x}_l) - c(\mathbf{x}_k)], \tag{B.1}$$

$$\Delta F = [f(\mathbf{x}_l) - f(\mathbf{x}_k)], \tag{B.2}$$

where $l = k - 1 \ldots max(0, k - n)$. Here k is the iteration number and n is the number of design variables. If as an example $n = 2$ is taken, the following process will result. For the first iteration it holds that $k = 1$ and $l = 0$.

$$k = 1,\ l = 0 : \quad \Delta C = [c(\mathbf{x}_0) - c(\mathbf{x}_1)], \tag{B.3}$$

For the second iteration it holds that $k = 2$. Using the expression for l, it follows that l runs in descending order from 1 to 0.

$$k = 2,\ l = 1, 0 : \quad \Delta C = [c(\mathbf{x}_1) - c(\mathbf{x}_2) \mid c(\mathbf{x}_0) - c(\mathbf{x}_2)], \tag{B.4}$$

For the third iteration it holds that $k = 3$. Using the expression for l, it follows that l runs in descending order from 2 to 1.

$$k = 3,\ l = 2, 1 : \quad \Delta C = [c(\mathbf{x}_2) - c(\mathbf{x}_3) \mid c(\mathbf{x}_1) - c(\mathbf{x}_3)]. \tag{B.5}$$

Here the sign | is used to visualize the different columns of the matrix. The matrix ΔF follows in exactly the same way by replacing C by F and c by f:

$$k = 1,\ l = 0 : \quad \Delta F = [f(\mathbf{x}_0) - f(\mathbf{x}_1)], \tag{B.6}$$

$$k = 2,\ l = 1, 0 : \quad \Delta F = [f(\mathbf{x}_1) - f(\mathbf{x}_2) \mid f(\mathbf{x}_0) - f(\mathbf{x}_2)], \tag{B.7}$$

$$k = 3,\ l = 2, 1 : \quad \Delta F = [f(\mathbf{x}_2) - f(\mathbf{x}_3) \mid f(\mathbf{x}_1) - f(\mathbf{x}_3)]. \tag{B.8}$$

In the first iteration only a limited amount of information about the history is present. Therefore the matrices ΔC and ΔF will contain only one column. Secondly it can be seen that the matrix builds up to a matrix consisting of two columns. The size of the matrix will not grow in every following iteration. This behavior will not change for more design variables. In general when there are n design variables there will be n columns. Finally it can be noted that the most recent data is used to construct the matrices. Therefore it looks as if the columns are moving to the right (look at the difference between the matrices of iteration step 2 and 3).

Appendix C

Overview of the Manifold Mapping implementation

1. Set $k = 0$ and compute
$$\mathbf{x}_0 = \underset{\mathbf{x} \in X}{argmin} \| \mathbf{c}(\mathbf{x}) \text{ - } \mathbf{y} \|. \tag{C.1}$$

2. Evaluate $\mathbf{f}(\mathbf{x}_k)$ and $\mathbf{c}(\mathbf{x}_k)$ and finish if appropriate stopping criteria are met. If $k > 0$, determine
$$\Delta \mathbf{c}_i = \mathbf{c}(\mathbf{x}_{k-i}) - \mathbf{c}(\mathbf{x}_k), \tag{C.2}$$
$$\Delta \mathbf{f}_i = \mathbf{f}(\mathbf{x}_{k-i}) - \mathbf{f}(\mathbf{x}_k). \tag{C.3}$$

3. Find ΔC and ΔF with $\Delta \mathbf{c}_i$ and $\Delta \mathbf{f}_i$ as columns.

4. Find the singular value decomposition of ΔC and the pseudo inverse of ΔF

5. Compute:
$$\mathbf{y}_k = \mathbf{c}(\mathbf{x}_k) - (\Delta C \Delta F^{\dagger} + I - U_c U_c^T)(\mathbf{f}(\mathbf{x}_k) - \mathbf{y}). \tag{C.4}$$

6. Compute
$$\mathbf{x}_{k+1} = \underset{\mathbf{x} \in X}{argmin} \ \| \mathbf{c}(\mathbf{x}) - \mathbf{y}_k \| \ . \tag{C.5}$$

7. Set $k = k$+1 and go to 2

Appendix D

Behaviour of the coarse and fine model responses in two dimensions

On the next two pages the fine and coarse model responses are given in several graphs. Note that in the coarse model the responses are linear whereas in the fine model the responses are non-linear.

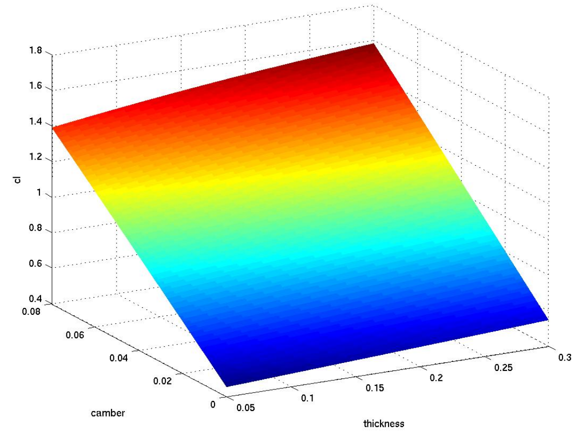


Figure D.1: Coarse model response (c_l)

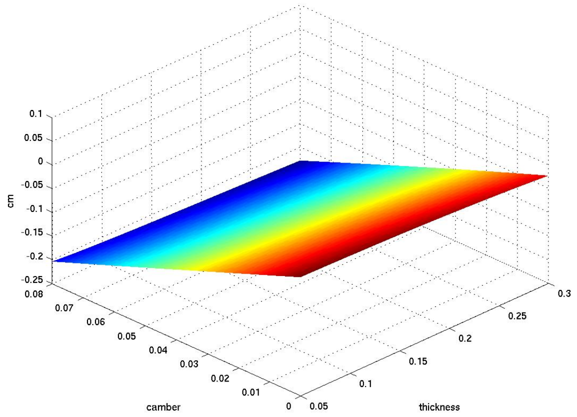


Figure D.2: Coarse model response (c_m)

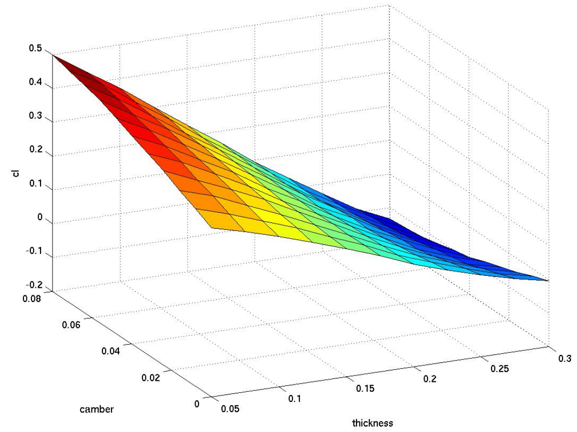


Figure D.3: Fine model response (c_l)

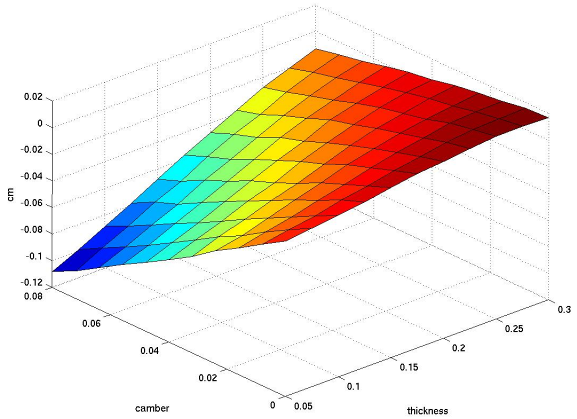


Figure D.4: Fine model response (c_m)

Appendix E

Navier-Stokes program in Comsol Multiphysics

The next pages give an example of the type of language used in Comsol Multiphysics. Note how easy a Navier-Stokes solver can be programmed and how easy the coefficients can be calculated.

```
function [vec] = fine(x)

% Extract values from input
c  = x(1);
cx = x(2);
t  = x(3);

% Geometry variables

alphadeg = 4; % degrees
alpha = -(alphadeg*pi)/180;

channel = rect2(-10,10,-10,10);

camber=c;
camber_x=cx;
dikte=t;

step=0.01;
x=[0:step:1];

for i=1:length(x);
[y(i), dydx(i)] = martincamber(x(i),1,camber,camber_x,dikte);
[z(i), dzdx(i)] = martincamber(x(i),0,camber,camber_x,dikte);
end

x1 = 0.25 - x;
x2 = 0.25 - x;
```

```
xnewrot1 = -(x1.*cos(alpha) + y.*sin(alpha));
xnewrot2 = -(x2.*cos(alpha) + z.*sin(alpha));

ynewrot1 = -x1.*sin(alpha)+y.*cos(alpha);
ynewrot2 = -x2.*sin(alpha)+z.*cos(alpha);

for j=1:length(x)-1
if j==1

else
xnew(j-1) = xnewrot2(length(x)+1-j);
ynew(j-1) = ynewrot2(length(x)+1-j);

end
end

assembly2=[xnewrot1,xnew;ynewrot1,ynew];

%..Make spline representation of the boundary..
%..The variable c holds the spline..
 c=geomspline(assembly2,'splinemethod','uniform' ,'closed','on');
 profile = solid2(c);

%..Define the 2D geometry..
fem.geom = channel-profile;

fem.mesh = meshinit(fem);
%fem.mesh =meshrefine(fem);

fem.sdim = {'x' 'y'};
fem.dim  = {'u' 'v' 'p'};
fem.shape = [2 2 1];


panels = 204;
% boundary groups: A: walls  B:inlet  C:outlet D: profile
fem.bnd.ind = {[2 3] [1] [4] [5:panels]};
fem.const.rho  = 1;
fem.const.eta  = 1/500;
fem.const.umax = 1;

fem.form = 'general';

fem.equ.ga = {{{'-p+2*eta*ux' 'eta*(uy+vx)'} ...
               {'eta*(uy+vx)' '-p+2*eta*vy'} ...
               {0 0}}};

fem.equ.f  = {{'rho*(u*ux+v*uy)' 'rho*(u*vx+v*vy)' 'ux+vy'}};
fem.bnd.r = {{0 'v' 0} {'u-1' 'v' 0} {0 0 'p'} {'u' 'v' 0}};

fem.bnd.g = {{ 0 0 0}};

% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Evaluate initial value using current solution
%init = asseminit(fem);

% Solve problem
if(1)
fem.sol=femnlin(fem, ...
               'solcomp',{'u','p','v'}, ...
               'outcomp',{'u','p','v'});

end

% Save current fem structure for restart purposes
fem0=fem;

clp=2*postint(fem,'p*ny','edim',1,'Dl',[5:panels]);


int = 0;
for i = 2:length(x)-1
int = int + (x(i)-x(i-1))*(cpdiff(i)+cpdiff(i-1))/2;
end

%voor cm

mint1 = 0;
mint2 = 0;
mint3 = 0;

for i = 2:length(x)-1

mint1 = mint1 + (x(i)-x(i-1))*(cpdiff(i)*x(i)+cpdiff(i-1)*x(i-1))/2;

end

for i=3:length(x)-1
   if i==2
    mint2 = mint2 + (x(i)-x(i-1))*(cpboven(i)*dydx(i)*y(i))/2;
    mint3 = mint3 + (x(i)-x(i-1))*(cponder(i)*dzdx(i)*z(i))/2;
    else
    mint2 = mint2 + (x(i)-x(i-1))*(cpboven(i)*dydx(i)*y(i)
            + cpboven(i-1)*dydx(i-1)*y(i-1))/2;

    mint3 = mint3 + (x(i)-x(i-1))*(cponder(i)*dzdx(i)*z(i)
            +cponder(i-1)*dzdx(i-1)*z(i-1))/2;

    end
end

for i=3:length(x)-1
cpboven(i-1)*dydx(i-1)*y(i-1);
end

mint=mint1+mint2-mint3;

cmkwartkoorde  =  mint + clp/4;

vec = [clp;cmkwartkoorde];
```

Bibliography

[1] I. H. Abbott, A. E. von Doenhoff, *Theory of Wing Sections*, Dover Publications (1995)

[2] John. D Anderson, *Fundamentals of Aerodynamics, third edition*, Mc Graw Hill (2001)

[3] M. H. Bakr, J. W. Bandler, K. Madsen, J. Søndergaard *Review of the Space Mapping Approach to Engineering Optimization and Modeling*, Optimization and Engineering, 1, p.241-276 (2000), Kluwer Academic Publishers

[4] G. Box, N. Draper *Empirical model-building and response surfaces*, Wiley, 1987 New York

[5] A. Canova, G. Grusso, M. Repetto *Magnetic design optimization and objective function approximation*, IEEE. Trans. on Magn., vol 39, No. 5, p.2154-2162, 2003

[6] G. Carpentieri, M. J. L. van Tooren, M Kelly, R Cooper *Airfoil Optimization using an Analytical Shape Parametrization*, CEIAT (2005)

[7] D.Echeverría, P.W. Hemker *Space Mapping and Defect Correction*, Computational Methods In Applied Mathematics, Vol. 5 (2005), No. 2, pp.107-136, Institute of Mathematics of the National Academy of Sciences of Belarus

[8] D.Echeverría, D.Lahaye, L.Encica, E.A.Lomonova, P.W.Hemker, A.J.A. Vandenput *Manifold-Mapping Optimization Applied to Linear Actuator Design*, IEEE transactions on magnetics, vol. 42, NO. 4, april 2006

[9] M. I. Gerritsma *Computational Fluid Dynamics, Incompressible flows*, Delft University Press, 4th December 2002

[10] G. H. Golub, C.F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, second edition, 1992, Baltimore Maryland

[11] D. Lahaye, A. Canova, G. Grusso, M. Repetto *Adaptive Manifold-Mapping Using Multiquadric Interpolation Applied to Linear Actuator Design*, COMPEL, Vol.26 2 pp. 230-240 (2007)

[12] A. Manella, M. Nervi, M.Repetto *Response surface method in magnetic optimization*, Int. J. Appl. Electromagn. Mater., vol.4, p.99-106, 1993

[13] Frank M. White *Viscous Fluid Flow, third edition* Mc Graw Hill (2006)