Aerodynamic Propeller Model
for Load Analysis

M A R I O  H E E N E

Master's Thesis in Scientific Computing (30 ECTS credits)
Master Programme in Scientific Computing 120 credits
Royal Institute of Technology year 2012
Supervisor at KTH was David Eller
Examiner was Michael Hanke

TRITA-MAT-E 2012:02
ISRN-KTH/MAT/E--12/02--SE

Royal Institute of Technology
School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci
Abstract

An aerodynamic propeller model, which can contribute to the prediction of structural loads experienced by aircraft in different flight maneuvers is presented.

The model is based on Blade Element Momentum theory and is able to predict the unsymmetrical and frequency-dependent forces and moments induced by the propeller on the airplane structure at steady and unsteady inflow-conditions.

In order to validate the model, a comparison with experimental results was performed and it can be seen that the model is in agreement with the experimental data providing that the aerodynamic data used for the calculations has good accuracy.
Referat

Aerodynamisk propellermodell för simuleringsbaserade lastberäkningar

En modell har utvecklats för att beräkna aerodynamiska krafter som orsakas av propellern vid manöverflygning. Modellen använder sig av klassiska bladelementteorin för predikering av osymmetriska stationära krafter som uppstår vid snedanblåsning av propellerskivan. Modellen kommer att användas inom ett forskningsprojekt om effektiv beräkning av aerodynamiska laster vid flygmanövrar och i vindbyar. En vidareutveckling av den klassiska metoden används för att ta fram instationära kraftbidrag i frekvensplanet i en form som är lämpligt för aeroelastiska stabilitetsanalys och beräkning av vindbylasterna.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, David Eller, from the Division of Flight Dynamics at KTH Stockholm, for his valuable advice throughout all stages of my thesis.

I also thank Michael Hanke, from KTH Stockholm, and Ulrich Rüde and Harald Köstler, from the University of Erlangen-Nuremberg, for their assistance concerning the double-degree Master’s program.

I wish to thank Jesper Oppelstrup, from KTH Stockholm, for his hospitality and all the interesting discussions at our lunch meetings.

Furthermore, I am very grateful to my parents for their support throughout all steps of my education.

Special thank goes to the Erich-Becker-Stiftung, whose financial support for my thesis has been a great help for me.
# Contents

1 Introduction

2 Theory of propeller modelling
   2.1 Blade Element Theory ............................................. 3
   2.2 Momentum Theory ................................................... 6
   2.3 Combined Blade Element Momentum Theory ...................... 9
   2.4 Non-dimensional formulation .................................... 11
   2.5 Unsteady propeller model ........................................ 12
   2.6 Steady propeller model with oscillatory forces ............. 15

3 Implementation
   3.1 Solution of the governing BEM equations and evaluation of section
       force and moment coefficients .................................... 17
   3.2 Adaptive integration of force and moment coefficients ........ 19
   3.3 Representation of geometry and aerodynamic data ............. 21

4 Validation
   4.1 Parallel inflow-conditions ...................................... 25
   4.2 Non-parallel inflow-conditions .................................. 30

5 Conclusion

Bibliography

A Jacobian matrix for the Newton solver

B Linearization of the BEM equations with respect to $\alpha$

C Linearization of the BEM equations with respect to $\beta$

D Code documentation
Chapter 1

Introduction

Prediction of flight loads plays an important role in the design process of aircraft. This requires models to describe the structural loads due to aerodynamic forces and aeroelastic effects with sufficient accuracy, but low enough computational effort in order to be feasible for practical application.

This work describes the theory, implementation and validation of an aerodynamic propeller model based on the Blade Element Momentum Theory. Applying the model in flight simulation, the forces and moments induced by the propeller on the airplane structure in different flight maneuvers can be analyzed and serve as basis for structural load analysis. Furthermore, the model can be applied to predict the frequency-dependent forces for unsteady inflow-conditions, as they occur due to structural vibrations of the wing on which the propeller is mounted. This is of interest for gust load analysis and fatigue failure predictions.

The Blade Element Momentum theory combines Blade Element Theory with Momentum Theory. Blade Element Theory dates back to the work of Drzewiecki [6] and is based on the idea of predicting the performance of a given propeller geometry by considering the aerodynamic forces produced by the two-dimensional blade sections. Momentum Theory was initially developed in the 19th century by Rankine [14] and Froude [7] for marine propellers. They described the important relation between thrust and acceleration of the airflow by the propeller. The Wright brothers were the first to combine Blade Element Theory and Momentum Theory in order to design airplane propellers [17]. The corrections to the theory presented by Goldstein [9] and Prandtl [13] were an important contribution in order to improve the accuracy of the theory.

Aerodynamic forces

This section will provide a short introduction to the basic principles and definitions in aerodynamics. It is based on the textbook by Anderson [1].

The geometry of every body being exposed to a fluid flow causes a certain pressure distribution along its surface. The shape of a wing, usually, causes a lower
pressure along its upper surface than on its lower surface, resulting in a force that pushes the wing upwards.

\[ \frac{L}{R} \]
\[ \alpha \]
\[ c \]

Figure 1.1. Aerodynamic forces acting on an airfoil.

The forces acting on an airfoil are sketched in Figure 1.1. \( V_\infty \) denotes the velocity and direction of the freestream, e.g. the flow far away from the body. The force \( R \) acting on the airfoil is composed by the lift force \( L \), acting perpendicular to \( V_\infty \), and the drag force \( D \), acting parallel to \( V_\infty \). The chord line \( c \) is defined as the line connecting the leading edge and the trailing edge of the airfoil. The normal force \( N \), perpendicular to \( c \), and the axial force \( A \), parallel to \( c \), can be expressed by the following geometrical relation:

\[
N = L \cos \alpha + D \sin \alpha \\
A = -L \sin \alpha + D \cos \alpha
\]

(1.1)

(1.2)

It is common to express lift and drag by the non-dimensional coefficients

\[
c_L = \frac{L}{\frac{1}{2} \rho_\infty V_\infty^2 c} \\
c_D = \frac{D}{\frac{1}{2} \rho_\infty V_\infty^2 c}
\]

(1.3)

(1.4)

where \( \rho_\infty \) denotes the density of the freestream. The variation of \( c_L \) and \( c_D \) with respect to \( \alpha \) is a very important characteristic of every airfoil.
Chapter 2

Theory of propeller modelling

2.1 Blade Element Theory

The Blade Element Theory (BET) is a relatively simple model to predict the performance of a certain propeller geometry. It was initially presented by W. Froude [7], in 1878, and later refined by Drzewiecki [6]. In BET the propeller blade is subdivided in small elements and the two-dimensional flow around each element is analyzed individually. The theory is based on the assumption that no interference between adjacent blade elements occurs, which is, however, without physical justification [17]. The representation of Blade Element Theory used in this work was adapted from the textbooks of Glauert [8] and Weick [18].

As shown in Figure 2.1, the distance from the propeller axis to the centerline of each element is denoted by \( r \) and the width of the element by \( dr \). \( R \) is the distance from the propeller axis to the tip of the blade, also called tip radius. Every blade section is further specified by its airfoil geometry, the chord length \( c \), and the twist angle \( \tau \) (see also Section 3.3).

The forces and velocities acting at the blade element are visualized in Figure 2.2. The angle \( \theta \) between chord line and propeller plane is composed of the twist angle of the blade section and the pitch angle of the propeller blade. Here, the pitch angle of the blade is defined as the angle between the chord line of the tip section and
the propeller plane. The twist angle of the blade section is defined as the angle between the chord line of the blade section and the propeller plane, when the blade pitch angle is zero. The flow to which the airfoil is exposed is denoted by $V_p$ and is composed of the axial velocity $V_x$ and the tangential velocity $V_t$. The velocity components will be examined in more detail later. The angle between $V_p$ and the propeller plane is denoted by

$$\varphi = \tan^{-1} \frac{V_x}{V_t}$$

(2.1)

and thus the local angle of attack of the airfoil is denoted by

$$\alpha_b = \theta - \varphi.$$  

(2.2)

According to the definitions in Section 1, the elemental lift force $dL$ and drag force $dD$ acting on the blade element can be formulated as

$$dL = \frac{1}{2} \rho V_p^2 c_L(\alpha_b)c N_b dr$$

(2.3)

$$dD = \frac{1}{2} \rho V_p^2 c_D(\alpha_b)c N_b dr,$$

(2.4)

where $N_b$ is the number of propeller blades.

The elemental thrust $dT$ and torque $dQ$ can be derived by the following geometrical transformations:

$$dT = dL \cos \varphi - dD \sin \varphi$$

(2.5)

$$\frac{dQ}{r} = dL \sin \varphi + dD \cos \varphi.$$  

(2.6)

As it will be discussed in more detail in Section 2.2, the propeller induces velocity in axial direction, which increases the flow speed at the propeller plane by $v_a$. The
2.1. BLADE ELEMENT THEORY

propeller also causes a rotation in the flow field, in the same direction as the propeller rotates, and thus the tangential velocity seen by the blade element is reduced by $v_r$.

The components of $V_p$ can then be expressed as

$$ V_x = V_x' + v_a \quad (2.7) $$
$$ V_t = V_t' - v_r. \quad (2.8) $$

$V_x'$ and $V_t'$ depend on the angle of attack $\alpha$ and the sideslip angle $\beta$ between the propeller axis and the freestream velocity $V_\infty$. Additionally, $V_t'$ depends on the angular position of the blade, the local radius $r$ of the blade element, and the rotation speed of the propeller $\omega$.

From Figure 2.3 the following relations for the velocity components of $V_\infty$ can be derived:

$$ v_x = V_\infty \cos \beta \cos \alpha \quad (2.9) $$
$$ v_y = V_\infty \sin \beta \quad (2.10) $$
$$ v_z = V_\infty \cos \beta \sin \alpha. \quad (2.11) $$

The propeller axis is defined in the opposite direction than $v_x$.

The axial velocity $V_x'$ is independent of the radius and the angular position of the blade and thus for all blade elements holds

$$ V_x' = v_x = V_\infty \cos \beta \cos \alpha. \quad (2.12) $$

The tangential velocity $V_t'$ depends on the angular position $\gamma$ of the blade, as visualized in Figure 2.4:

$$ V_t' = \omega r + v_y \sin \gamma - v_z \cos \gamma $$
$$ = \omega r + V_\infty \sin \beta \sin \gamma - V_\infty \cos \beta \sin \alpha \cos \gamma. \quad (2.13) $$

\begin{figure}
\centering
\includegraphics[width=0.8\textwidth]{figure2.3.png}
\caption{Components of the freestream velocity $V_\infty$.}
\end{figure}
Therefore, it is not only necessary to consider different radii along the blade, but also to perform the analysis at a number of different angular positions. Incorporating the dependency on $\gamma$ in the above equations for thrust and torque

$$dT = dT(r, \gamma) = \frac{1}{2} \rho V_p^2 \left[ c_L(\alpha_b) \cos \varphi - c_D(\alpha_b) \sin \varphi \right] c_N b r \frac{d\gamma}{2\pi}$$ (2.14)

$$dQ = dQ(r, \gamma) = \frac{1}{2} \rho V_p^2 \left[ c_L(\alpha_b) \sin \varphi + c_D(\alpha_b) \cos \varphi \right] c_N r dr \frac{d\gamma}{2\pi}.$$ (2.15)

Since the induced velocities, $v_a$ and $v_r$, are unknown, Blade Element Theory on its own can only provide an upper limit for the performance of a propeller by setting $v_a$ and $v_r$ to zero. To predict the performance of a propeller more accurately a relation for $v_a$ and $v_r$ is necessary, as it is provided by Momentum Theory.

### 2.2 Momentum Theory

Momentum Theory was originally developed by Rankine [14] and Froude [7] and is based on the idea that whenever a propeller produces thrust, it causes a motion of the air in opposite direction to the thrust [18]. This section is based on the textbooks by Glauert [8] and Weick [18].

In Momentum Theory the propeller is assumed to have an infinite number of blades and thus can be understood as a circular disk, also called actuator disk in this context. Further assumptions are that the flow is inviscid, incompressible and uniform over the whole disk. The Momentum Theory, as presented by Rankine and Froude, neglects any rotational motion in the slipstream and frictional drag on the propeller blades. The general concept of Momentum Theory is visualized in

![Figure 2.4. Components of the tangential velocity.](image)
Figure 2.5. Pressure and velocity in front of and behind the propeller disk (reprinted from [18]).

Figure 2.5. Far away in front of the propeller disk the flow has freestream velocity \( V_\infty \) and pressure \( p_\infty \). The airflow passes the propeller disk with velocity \( V \). The pressure is increased from \( p' \) in front of the disk to \( p' + \Delta p \) behind the disk. Far behind the disk, the flow has velocity \( V_1 \) and the pressure decreased to the initial \( p_\infty \).

Since the flow is assumed to be inviscid and incompressible, according to Bernoulli’s equation it holds

\[
p_\infty + \frac{1}{2} \rho V_\infty^2 = p' + \frac{1}{2} \rho V^2
\]

in front of the disk and

\[
p' + \Delta p + \frac{1}{2} \rho V^2 = p_\infty + \frac{1}{2} \rho V_1^2
\]

behind the disk. It follows that

\[
\Delta p = \left[ p_\infty + \frac{1}{2} \rho V_1^2 \right] - \left[ p_\infty + \frac{1}{2} \rho V_\infty^2 \right] = \frac{1}{2} \rho (V_1^2 - V_\infty^2).
\]

The assumption of incompressibility simplifies the analysis, but other unique relations between velocity and pressure, such as that of isentropic compressible flow, could be used as well. The thrust \( T \) can be expressed as

\[
T = A \Delta p = A \frac{1}{2} \rho (V_1^2 - V_\infty^2),
\]

where \( A \) denotes the area of the disk. The thrust also equals the increase in momentum in the slipstream, which is the product of the mass flow through the disk \( A \rho V \) and the overall increase in velocity

\[
T = A \rho V (V_1 - V_\infty).
\]
Equating (2.19) and (2.20)

\[ A \frac{1}{2} \rho (V_1^2 - V_\infty^2) = A \rho V (V_1 - V_\infty), \]

it follows

\[ V = \frac{V_1 + V_\infty}{2}. \]

Defining the increase in velocity at the propeller disk as the axial induced velocity \( v_a \), analogously to (2.22) it holds

\[ V = V_\infty + v_a \]

\[ V_1 = V_\infty + 2v_a \]

and thus

\[ T = 2A \rho (V_\infty + v_a) v_a. \]

In this simplified version of the Momentum Theory any rotational motion in the slipstream is neglected. However, in order to predict the performance of an aircraft propeller more accurately, it is necessary to extend the actuator disk model by a rotational component being imparted to the fluid flow as a reaction to the torque of the propeller.

Figure 2.6. Streamtube through an annular element of the propeller disk (adapted from [12]).

Figure 2.6 shows the streamtube through an annular ring of the propeller disk with radius \( r \) and width \( dr \). The rotation speed of the flow inside the streamtube, at the position of the propeller disk, is denoted by \( v_r \). The rotation speed far behind the disk is denoted by \( v_{r,1} \). The elemental torque \( dQ \) equals the increase in angular momentum in the streamtube

\[ dQ = \rho V dA v_{r,1} r, \]

where \( \rho V dA \) is the mass flow through the annular ring.
2.3. COMBINED BLADE ELEMENT MOMENTUM THEORY

It can be shown (see, e.g., [8]) that for incompressible flow the rotational velocity at the disk

\[ v_r = \frac{1}{2} v_{r,1} \]  \hspace{1cm} (2.27)

and thus

\[ dQ = \rho V dA 2v_r r = 4\pi r^2 dr \rho (V_\infty + v_a) v_r. \]  \hspace{1cm} (2.28)

Analogously, the elemental thrust produced by the annular element of the propeller disk

\[ dT = 4\pi r dr \rho (V_\infty + v_a) v_a. \]  \hspace{1cm} (2.29)

The above equations, however, have been developed under the assumption of treating the propeller as a circular disk, i.e., with an infinite number of blades. There exist solutions by Prandtl [13] and Goldstein [9] to take into account the effects that occur with a finite number of blades. Although Goldstein’s solution is more accurate, it is too complicated for general use. Therefore, Glaubert [8] proposed the following correction to Momentum Theory

\[ dT = 4\pi r dr \rho (V_\infty + v_a) v_a F \]  \hspace{1cm} (2.30)
\[ dQ = 4\pi r^2 \rho (V_\infty + v_a) v_r F, \]  \hspace{1cm} (2.31)

where

\[ F = \frac{2}{\pi} \arccos e^{-f} \]  \hspace{1cm} (2.32)
\[ f = \frac{N_b R - r}{2 r \sin \varphi}. \]  \hspace{1cm} (2.33)

As introduced in Section 2.1, \( N_b \) denotes the number of blades and \( \varphi \) is the inflow angle at the current blade element with respect to the propeller plane.

Incorporating the dependency on the angular position \( \gamma \) analogously to Section 2.1 and considering only a segment of the annular ring

\[ dT(r, \gamma) = 2r \rho (V'_r + v_a) v_a F dr d\gamma \]  \hspace{1cm} (2.34)
\[ dQ(r, \gamma) = 2r^2 \rho (V'_x + v_a) v_r F dr d\gamma. \]  \hspace{1cm} (2.35)

2.3 Combined Blade Element Momentum Theory

In the previous sections two different theories to formulate the elemental thrust \( dT \) and torque \( dQ \) have been presented. Combining Blade Element Theory and Momentum Theory allows to calculate the induced velocities, \( v_a \) and \( v_r \).

Equating the elemental thrust, (2.14) and (2.34),

\[ \frac{1}{2} \rho V_p^2 [c_L(\alpha_b) \cos \varphi - c_D(\alpha_b) \sin \varphi] c N_b dr \frac{d\gamma}{2\pi} = 2r \rho (V'_x + v_a) v_a F dr d\gamma \]  \hspace{1cm} (2.36)
CHAPTER 2. THEORY OF PROPELLER MODELLING

Figure 2.7. Definition of forces and moments on the propeller disk.

and elemental torque, (2.15) and (2.35),

\[
\frac{1}{2} \rho V_p^2 \left[c_L(\alpha_b) \sin \varphi + c_D(\alpha_b) \cos \varphi\right] cN_b r dr d\gamma = \frac{2}{\pi} \rho (V_x' + v_a) v_r F dr d\gamma, \tag{2.37}
\]

results in a system of non-linear equations

\[
f_1(v_a, v_r) = \frac{1}{2} \sigma V_p^2 \left[c_L(\alpha_b) \cos \varphi - c_D(\alpha_b) \sin \varphi\right] - 2(V_x' + v_a) v_a F = 0 \tag{2.38}
\]

\[
f_2(v_a, v_r) = \frac{1}{2} \sigma V_p^2 \left[c_L(\alpha_b) \sin \varphi + c_D(\alpha_b) \cos \varphi\right] - 2(V_x' + v_a) v_r F = 0, \tag{2.39}
\]

which can be solved for \( v_a \) and \( v_r \), e.g., with an iterative procedure like Newton’s Method. The non-dimensional factor \( \sigma = \frac{cN_b}{2\pi} \) denotes the so-called blade solidity.

After the induced velocities, \( v_a \) and \( v_r \), have been determined, the overall performance of the propeller can be calculated by integration of the elemental thrust and torque over the propeller disk

\[
T = \int_0^{2\pi} \int_{R_{Hub}}^R dT(r, \gamma) \tag{2.40}
\]

\[
Q = \int_0^{2\pi} \int_{R_{Hub}}^R dQ(r, \gamma). \tag{2.41}
\]

\( R_{Hub} \) denotes the radius of the propeller hub or the root of the blade, respectively.

The forces and moments acting on the propeller axis can be derived from the
2.4. NON-DIMENSIONAL FORMULATION

Elemental thrust and torque, as shown in Figure 2.7:

\[ dF_x = dT \quad dM_x = -dQ \]
\[ dF_y = \frac{dQ}{r} \sin \gamma \quad dM_y = dTr \sin \gamma \]
\[ dF_z = \frac{dQ}{r} \cos \gamma \quad dM_z = dTr \cos \gamma. \]

The fact that \( dM_x \) is defined in the opposite direction than \( dQ \), results from a right-handed coordinate system being used for the definition of forces and moments acting on the propeller axis, where the \( x \)-axis points in the opposite direction than the \( v_x \)-component of the freestream velocity (see Figure 2.2 and Figure 2.3). The overall forces \( F_x, F_y, F_z \), and moments \( M_x, M_y, M_z \) can be determined from the elemental values by integration, as presented in Equation (2.40) and Equation (2.41).

2.4 Non-dimensional formulation

Usually, it is not desirable to provide absolute values for the velocities when examining the performance of a propeller. Therefore, it is more common (see, e.g., [8, 18]) to use the non-dimensional rate of advance

\[ \lambda = \frac{V_\infty}{\omega R} \quad (2.43) \]

or

\[ J = \frac{V_\infty}{nD} = \lambda \pi, \quad (2.44) \]

where \( n \) is the number of revolutions per second and \( D = 2R \) is the diameter of the propeller. The other velocities can be replaced by non-dimensional expressions in the same way:

\[ \lambda_x = \frac{V'_x}{\omega R} = \lambda \cos \beta \cos \alpha \quad (2.45) \]
\[ \lambda_t = \frac{V'_t}{\omega R} = \frac{r}{R} + \lambda (\sin \beta \sin \gamma - \cos \beta \sin \alpha \cos \gamma) \quad (2.46) \]
\[ \lambda_a = \frac{v_a}{\omega R} \quad (2.47) \]
\[ \lambda_r = \frac{v_r}{\omega R} \quad (2.48) \]
\[ \lambda^2_p = (\lambda_x + \lambda_a)^2 + (\lambda_t - \lambda_r)^2. \quad (2.49) \]

Inserting the above expressions in Equation (2.38) and Equation (2.39) results in the non-dimensional formulation of the Blade Element Momentum approach

\[ f_1(\lambda_a, \lambda_r) = \frac{1}{2} \sigma \lambda^2_p [c_L(\alpha_b) \cos \varphi - c_D(\alpha_b) \sin \varphi] - 2(\lambda_x + \lambda_a)\lambda_a F = 0 \quad (2.50) \]
\[ f_2(\lambda_a, \lambda_r) = \frac{1}{2} \sigma \lambda^2_p [c_L(\alpha_b) \sin \varphi + c_D(\alpha_b) \cos \varphi] - 2(\lambda_x + \lambda_a)\lambda_r F = 0, \quad (2.51) \]
where
\[ \varphi = \tan^{-1} \frac{\lambda_x + \lambda_a}{\lambda_l - \lambda_r} \]  
(2.52)

With non-dimensional coefficients for thrust
\[ dC_T = \frac{2dT}{\rho(\omega R)^2 \pi R^2} = \frac{4r(\lambda_x + \lambda_a)\lambda_a F}{R^2 \pi} drd\gamma \]  
(2.53)
and torque
\[ dC_Q = \frac{2dQ}{\rho(\omega R)^2 \pi R^3} = \frac{4r^2(\lambda_x + \lambda_a)\lambda_a F}{R^3 \pi} drd\gamma \]  
(2.54)
the following coefficients corresponding to the elemental forces and moments acting on the propeller axis can be formulated:
\[ dC_{Fx} = dC_T \quad dC_{Mz} = -dC_Q \]
\[ dC_{Fy} = dC_Q \frac{R}{r} \sin \gamma \quad dC_{My} = dC_T \frac{r}{R} \sin \gamma \]  
(2.55)
\[ dC_{Fz} = dC_Q \frac{R}{r} \cos \gamma \quad dC_{Mz} = dC_T \frac{r}{R} \cos \gamma. \]

Analogous to Equation (2.40) and (2.41), integration of the elemental coefficients over the propeller disk yields the overall force \( C_{Fx}, C_{Fy}, C_{Fz} \) and moment coefficients \( C_{Mx}, C_{My}, C_{Mz} \).

### 2.5 Unsteady propeller model

The goal of the unsteady propeller model is to analyze variations in the force and moment coefficients as a reaction to unsteady variations in the flow field, being, for example, caused by structural vibrations of the wing on which the propeller is mounted. The model can be used to analyze the frequency dependent loads that are induced by the propeller disk on the airplane structure.

Since the variations in the \( \alpha \) and \( \beta \) angles, caused by oscillations of the wing structure, can be assumed to be small-amplitude deviations from the steady-state operation point, linearization can be applied to efficiently approximate the force and moment coefficients. The derivation will be explained on the basis of variations in \( \alpha \), but is likewise valid for variations in \( \beta \). Linearization of the governing equations of the blade element momentum theory, Equation (2.50) and (2.51), with respect to a deviation in \( \alpha \) from the operation point yields
\[ f_1(\alpha_0 + \Delta \alpha) = f_1(\alpha_0) + \frac{\partial f_1}{\partial \alpha} \bigg|_{\alpha_0} \Delta \alpha = 0 \]  
(2.56)
\[ f_2(\alpha_0 + \Delta \alpha) = f_2(\alpha_0) + \frac{\partial f_2}{\partial \alpha} \bigg|_{\alpha_0} \Delta \alpha = 0, \]  
(2.57)
2.5. UNSTEADY PROPELLER MODEL

where \( \alpha_0 \) denotes the angle of attack in the operation point. As presented in the previous sections, for the steady-state solution it holds

\[
\begin{align*}
  f_1(\alpha_0) &= 0 \\
  f_2(\alpha_0) &= 0
\end{align*}
\]

and thus it must hold

\[
\begin{align*}
  \frac{\partial f_1}{\partial \alpha} \bigg|_{\alpha_0} &= 0 \\
  \frac{\partial f_2}{\partial \alpha} \bigg|_{\alpha_0} &= 0,
\end{align*}
\] (2.58)

which results in the linear system of equations

\[
\begin{align*}
  \frac{\partial f_1}{\partial \alpha} \bigg|_{\alpha_0} &= \frac{1}{2} \sigma L^2 \partial c_t + \frac{1}{2} \sigma c_q \frac{\partial \lambda^2}{\partial \alpha} - 2 \lambda a F \frac{\partial \lambda_x}{\partial \alpha} - 2 \lambda a \frac{\partial F}{\partial \alpha} \\
  \frac{\partial f_2}{\partial \alpha} \bigg|_{\alpha_0} &= \frac{1}{2} \sigma c_q \frac{\partial c_q}{\partial \alpha} + \frac{1}{2} \sigma c_q \frac{\partial \lambda^2}{\partial \alpha} - 2 \lambda r F \frac{\partial \lambda_x}{\partial \alpha} - 2 \lambda r \frac{\partial F}{\partial \alpha} \\
  &\quad - 2(\lambda_x + \lambda_a) F \frac{\partial \lambda_x}{\partial \alpha} - 2(\lambda_x + \lambda_a) \lambda_a \frac{\partial F}{\partial \alpha} = 0
\end{align*}
\] (2.59)

(2.60)

Here,

\[
\begin{align*}
  c_t &= c_L(\alpha_b) \cos \varphi - c_D(\alpha_b) \sin \varphi \\
  c_q &= c_L(\alpha_b) \sin \varphi + c_D(\alpha_b) \cos \varphi
\end{align*}
\] (2.61)

(2.62)

which can also be expressed as

\[
\begin{align*}
  c_t &= c_Z(\alpha_b) \cos \theta - c_X(\alpha_b) \sin \theta \\
  c_q &= c_Z(\alpha_b) \sin \theta + c_X(\alpha_b) \cos \theta
\end{align*}
\] (2.63)

(2.64)

by using the normal force and axial force coefficient

\[
\begin{align*}
  c_Z &= c_L(\alpha_b) \cos \alpha_b + c_D(\alpha_b) \sin \alpha_b \\
  c_X &= -c_L(\alpha_b) \sin \alpha_b + c_D(\alpha_b) \cos \alpha_b.
\end{align*}
\] (2.65)

(2.66)

A detailed derivation of the BEM equations with respect to \( \alpha \) and \( \beta \) and the resulting linear system of equations can be found in Appendix B and C.

Solving Equation (2.59) and (2.60) for the unknowns \( \frac{\partial \lambda_a}{\partial \alpha} \) and \( \frac{\partial \lambda_r}{\partial \alpha} \), the derivatives of the elemental thrust and torque coefficient

\[
\begin{align*}
  \frac{\partial dC_T}{\partial \alpha} \bigg|_{\alpha_0} &= \frac{4r}{R^2 \pi} \left[ \lambda a F \left( \frac{\partial \lambda_x}{\partial \alpha} + \frac{\partial \lambda_a}{\partial \alpha} \right) + (\lambda_x + \lambda_a) \frac{\partial F}{\partial \alpha} \right] d\alpha d\gamma
\end{align*}
\] (2.67)
\[ \frac{\partial dC_Q}{\partial \alpha} \bigg|_{\alpha_0} = \frac{4r^2}{R^3\pi} \left[ \lambda_r F \left( \frac{\partial \lambda_x}{\partial \alpha} + \frac{\partial \lambda_a}{\partial \alpha} \right) + (\lambda_x + \lambda_a) F \frac{\partial \lambda_r}{\partial \lambda_a} \right] \\
+ (\lambda_x + \lambda_a) \lambda_r \frac{\partial F}{\partial \alpha} d\gamma \]  
(2.68)

can be determined. For the derivatives of the force and moment coefficients the same relations as in Equation (2.56) hold

\[ \frac{\partial dC_{F_\alpha}}{\partial \alpha} = \frac{\partial dC_T}{\partial \alpha}, \quad \frac{\partial dC_{M_\alpha}}{\partial \alpha} = -\frac{\partial dC_Q}{\partial \alpha} \]
(2.69)

Integration over the propeller disk yields the derivatives of the overall force and moment coefficients

\[ \frac{\partial C_{F_j}}{\partial \alpha} = \int_0^{2\pi} \int_{R_{hub}}^R \frac{\partial dC_{F_j}(r, \gamma)}{\partial \alpha} \, dr \, d\gamma, \]
(2.70)
\[ \frac{\partial C_{M_j}}{\partial \alpha} = \int_0^{2\pi} \int_{R_{hub}}^R \frac{\partial dC_{M_j}(r, \gamma)}{\partial \alpha} \, dr \, d\gamma, \]
(2.71)

which can be used to approximate the force and moment coefficients for small deviations in \( \alpha \) from the operation point

\[ C_{F_j}(\alpha_0 + \Delta \alpha) = C_{F_j}(\alpha_0) + \frac{\partial C_{F_j}}{\partial \alpha} \bigg|_{\alpha_0} \Delta \alpha \]
(2.72)
\[ C_{M_j}(\alpha_0 + \Delta \alpha) = C_{M_j}(\alpha_0) + \frac{\partial C_{M_j}}{\partial \alpha} \bigg|_{\alpha_0} \Delta \alpha. \]
(2.73)

Special treatment is necessary for the evaluation of \( \frac{\partial c_Z}{\partial \alpha} \) and \( \frac{\partial c_q}{\partial \alpha} \). The normal force coefficient \( c_Z \) does not respond instantaneously to variations in the inflow angle, but experiences a certain phase shift, which is dependent on the frequency of the oscillation. Using the same assumptions as for the steady-state case, e.g. incompressible, inviscid and two-dimensional flow around the blade section, Theodorsen’s solution for thin airfoils oscillating in incompressible flow (as presented in [2, 3]) can be applied. Hence,

\[ \frac{\partial c_Z(k)}{\partial \alpha_b} = \frac{\partial}{\partial \alpha_b} \left( c_L \cos \alpha + c_D \sin \alpha \right) \left( \frac{1}{2} ik + C(k) \right) \]
\[ = \left[ \left( \frac{\partial c_L}{\partial \alpha_b} + c_D \right) \cos \alpha + \left( \frac{\partial c_D}{\partial \alpha_b} - c_L \right) \sin \alpha \right] \left( \frac{1}{2} ik + C(k) \right). \]
(2.74)

From Equation (2.2)
\[ \alpha_b = \theta - \varphi \]
2.6. STEADY PROPELLER MODEL WITH OSCILLATORY FORCES

it follows

\[
\frac{\partial \varphi}{\partial \alpha} = -\frac{\partial \alpha_b}{\partial \alpha}
\]  

(2.75)

and thus

\[
\begin{align*}
\frac{\partial c_t}{\partial \alpha} &= \frac{\partial \varphi}{\partial \alpha} \frac{\partial c_t}{\partial \varphi} = -\frac{\partial \varphi}{\partial \alpha} \frac{\partial c_t}{\partial \alpha_b} = -\frac{\partial \varphi}{\partial \alpha} \left[ \frac{\partial c_Z(k)}{\partial \alpha_b} \cos \theta - \frac{\partial c_X}{\partial \alpha_b} \sin \theta \right], \\
\frac{\partial c_q}{\partial \alpha} &= \frac{\partial \varphi}{\partial \alpha} \frac{\partial c_q}{\partial \varphi} = -\frac{\partial \varphi}{\partial \alpha} \frac{\partial c_q}{\partial \alpha_b} = -\frac{\partial \varphi}{\partial \alpha} \left[ \frac{\partial c_Z(k)}{\partial \alpha_b} \sin \theta + \frac{\partial c_X}{\partial \alpha_b} \cos \theta \right].
\end{align*}
\]  

(2.76)

(2.77)

Here, \( C(k) \) denotes Theodorsen’s function

\[
C(k) = \frac{H_1^{(2)}(k)}{H_1^{(2)}(k) + iH_0^{(2)}(k)},
\]  

(2.78)

a complex-valued combination of Hankel functions of the second kind. The reduced frequency

\[
k = \frac{2\pi f \frac{1}{2} c}{V_p} = \frac{\pi c f}{\omega R} = \frac{\pi c f_n}{\lambda_p},
\]  

(2.79)

is a measure of the unsteadiness of the flow and also referred to as Strouhal’s number [3], where

\[
f_n = \frac{f}{\omega R},
\]  

(2.80)

is the normalized frequency of the airfoil vibration.

2.6 Steady propeller model with oscillatory forces

Whenever the propeller axis is inclined to the freestream, oscillations of the blade inflow angle occur during every blade revolution, which results in a lag of the aerodynamic forces behind the motion. This phenomena can, as well, be handled by Theodorsen’s solution for thin airfoils oscillating in incompressible flow, which has already been introduced for the unsteady propeller model in Section 2.5. In the steady-state case, however, not the instantaneous change of the aerodynamic forces is relevant, but only the resulting phase lag. In accordance with Equation (2.74) the phase shift \( \Delta \gamma \) can be determined as

\[
\Delta \gamma = \arg \left\{ \frac{1}{2} ik + C(k) \right\}.
\]  

(2.81)

Here, the reduced frequency \( k \) depends on the rotational speed of the propeller \( \omega \),

\[
k = \frac{\omega c}{2V_p} = \frac{\omega c}{\omega R} = \frac{c}{2\lambda_p R}.
\]  

(2.82)
The phase shift $\Delta \gamma$ can also be understood as displacement in angular position on the propeller disk of the elemental forces and can be incorporated in the calculation of the force and moment coefficients as follows

\begin{align*}
  dC_{F_x} &= dC_T \\
  dC_{F_y} &= dC_Q \frac{R}{r} \sin(\gamma + \Delta \gamma) \\
  dC_{F_z} &= dC_Q \frac{R}{r} \cos(\gamma + \Delta \gamma) \\
  dC_{M_x} &= -dC_Q \\
  dC_{M_y} &= dC_T \frac{r}{R} \sin(\gamma + \Delta \gamma) \\
  dC_{M_z} &= dC_T \frac{r}{R} \cos(\gamma + \Delta \gamma).
\end{align*}

(2.83)
Chapter 3

Implementation

Based on the above presented theory, a C++ software library for propeller analysis has been implemented.

The library basically consists of the following components:

- Blade Section Solver: Solution of the governing BEM equations and evaluation of section force and moment coefficients.
- Geometry and Aerodynamic Data: Representation of propeller geometry and aerodynamic data.

In the following sections further details on the implementation and the underlying algorithms will be presented. A detailed documentation of the library can be found in Appendix D.

3.1 Solution of the governing BEM equations and evaluation of section force and moment coefficients

In order to solve the BEM equations (Equation (2.50) and (2.51)) an algorithm based on Newton’s method is applied. The algorithm can be found in Algorithm 1. A detailed derivation of the Jacobian matrix $J$ used for the solution procedure can be found in Appendix A. For this two-dimensional problem, the inverse of the Jacobian matrix can be directly determined. In order to improve the global convergence behavior, line search according to Armijo’s rule is applied to determine a step size $\nu$, for which a sufficient decrease of $\|F(x)\|$ is achieved [11] (lines 11-15):

$$\|F(x + \nu d)\| < (1 - \kappa \nu)\|F(x)\|.$$  \hspace{1cm} (3.1)

In the rare case that this procedure does not succeed with a reasonable step size, a minimum step size $\nu_{\text{min}}$ is used.
CHAPTER 3. IMPLEMENTATION

Since the solution space contains a trivial solution, which will be found by the Newton approach, in some cases, when the initial value \( \lambda_{a, initial} \) is not appropriately chosen, additional logic is necessary to detect and handle this case. The trivial solution corresponds to the case

\[
\lambda_a = -\lambda_x \\
\lambda_r = \lambda_t
\]

and can be easily detected by checking if

\[
\lambda_p^2 = (\lambda_x + \lambda_a)^2 + (\lambda_t - \lambda_r)^2 < tol. 
\tag{3.2}
\]

Selecting a suitable value for \( \lambda_{a, initial} \) is, however, not trivial and has to be further discussed. One approach could be to compute an initial \( \lambda_a \) by setting \( \lambda_a = 0, \lambda_r = 0 \) and reduce \( f_1 \) to a quadratic equation, which can be directly solved for \( \lambda_a \). This procedure corresponds to approximating the thrust by means of the Blade Element Theory neglecting the induced velocities and then finding a \( \lambda_a \), which fulfills the momentum theory. Another approach could be to use an analytical solution of the BEM theory, as presented in [12], in order to have an initial guess for \( \lambda_a \). Both approaches, however, have turned out to be unsatisfactory in terms of robustness. Especially for the case when \( \lambda_x \) is so low that it has approximately the same magnitude as \( \lambda_a \) and it holds \( \lambda_{a, initial} < 0 \), the initial guess might be so close to the trivial solution that the solver will run into it.

This problem can be visualized by considering only the angle \( \varphi = \tan^{-1} \frac{\lambda_x + \lambda_a}{\lambda_t} \), where \( \lambda_r \) is neglected for simplicity, since it is generally so small that it does not have much impact on \( \varphi \). As shown in Figure 3.1 \( \varphi_{\lambda_a=0} \) corresponds to the value for \( \lambda_a = 0 \) and \( \varphi_0 = 0 \) to the trivial solution, where \( \lambda_z = -\lambda_a \). In case that \( \varphi_{ini} \), which corresponds to a \( \lambda_{a, initial} \) provided by one of the above presented approaches, is lower than zero, and \( \varphi_{sol} \), corresponding to the value for which the solution of the BEM equations is achieved, is greater than zero, it might happen that the solver will end up in the trivial case rather than finding the right solution. Of course, the same applies to the case \( \varphi_{sol} < 0, \varphi_{ini} > 0 \). If, however, it holds \( \varphi_0 < \varphi_{sol} < \varphi_{ini} \) or \( \varphi_{ini} < \varphi_{sol} < \varphi_0 \), the solver will probably find the right solution.
3.2. ADAPTIVE INTEGRATION OF FORCE AND MOMENT COEFFICIENTS

Based on these observations, a robust strategy for the selection of the $\lambda_{a,\text{initial}}$ has been developed. The initial $\lambda_a$ will be chosen, so that a high value of $\varphi$, e.g. $45^\circ$, is achieved:

$$\varphi = \tan^{-1} \frac{\lambda_x + \lambda_a}{\lambda_t} = \frac{\pi}{4}$$

$$\Leftrightarrow \tan \frac{\pi}{4} = 1 = \frac{\lambda_x + \lambda_a}{\lambda_t}$$

$$\Rightarrow \lambda_{a,\text{initial}} = \lambda_t - \lambda_x. \quad (3.3)$$

If the solver runs into the trivial solution, it is restarted using $\lambda_{a,\text{initial}} = -\lambda_{a,\text{initial}}$ (lines 18-22). In order to increase the probability of starting with the right sign, $c_t$ (see Equation (2.61)) is evaluated for $\varphi_{\lambda_a=0}$. In most cases $\varphi_{\lambda_a=0}$ is close to $\varphi_{sol}$ and, considering Equation (2.50), $c_t$ has always the same sign as $\lambda_a$.

In order to handle rare cases, for which no convergence can be achieved, as it might happen when the solver is stuck in a local minimum, a first attempt is to restart the solver with $\lambda_{a,\text{initial}} = -\lambda_{a,\text{initial}}$, as well (lines 24-33).

3.2 Adaptive integration of force and moment coefficients

In order to perform efficient numerical integration of the force and moment coefficients over the propeller disk, an adaptive algorithm based on Newton-Cotes quadrature rules, as presented in [16], is used.

In order to integrate the function $f$ in the interval $[a, b]$, the interval is subdivided in a number of smaller intervals with the nodes $t_0, t_1, t_2, \cdots, t_n$, where $t_0 = a$ and $t_n = b$. The numerical integral value $I_h$ is the sum of the subintegrals $I_{h,i}$,

$$I_h = \sum_{i=0}^{n-1} I_{h,i}, \quad (3.4)$$

where each subintegral is approximated by a quadrature rule like the Simpson rule

$$I_{h,i} = \frac{t_{i+1} - t_i}{6} [f(t_i) + 4f(t_{m,i}) + f(t_{i+1})], \quad (3.5)$$

with

$$t_{m,i} = \frac{t_{i+1} + t_i}{2}. \quad (3.6)$$

The idea of the adaptive algorithm is to refine each subintegral by further dividing it in smaller intervals until the integration error $e_i$ is lower than a given tolerance:

$$e_i = |I_{h,i} - I_i| < tol, \quad (3.7)$$

where $I_i$ is the exact value of the subintegral. This can be efficiently done by dividing the integration interval of $I_{h,i}$ in two subintervals of equal size, $I_{h,i,1}$ and $I_{h,i,2}$, since in this case all three function values $f(t_i), f(t_{m,i})$ and $f(t_{i+1})$ can be reused. Since
Algorithm 1 Solution of BEM equations

1: \( x := (\lambda_a, \lambda_r)^T \)
2: \( F(x) := (f_1(x), f_2(x))^T \)
3: \( \kappa = 1e^{-4} \) (default value in [11])
4: \( \lambda_{a,\text{initial}} = \text{sgn}(\alpha_i)(\lambda_t - \lambda_x) \)
5: \( \lambda_{r,\text{initial}} = 0.001 \)
6: \( i = 0 \)
7: \( \|F(x)\| > \epsilon \) do
8: \( \text{Compute Jacobian matrix } J(x) \)
9: \( \text{Compute search direction } d = -J(x)^{-1}F(x) \)
10: \( \nu = 1 \)
11: \( \text{while } \|F(x + \nu d)\| > (1 - \kappa \nu)\|F(x)\| \) do
12: \( \nu = \frac{\nu}{2} \)
13: \( \text{end while} \)
14: \( x = x + \nu d \)
15: \( i = i + 1 \)
16: \( \text{if } \lambda_p^2 < tol \text{ and not restarted then} \)
17: \( \lambda_a = -\lambda_{a,\text{initial}} \)
18: \( \lambda_r = \lambda_{r,\text{initial}} \)
19: \( \text{restart} \)
20: \( \text{end if} \)
21: \( \text{if } i > i_{\text{max}} \text{ then} \)
22: \( \text{if not restarted then} \)
23: \( \lambda_a = -\lambda_{a,\text{initial}} \)
24: \( \lambda_r = \lambda_{r,\text{initial}} \)
25: \( \text{restart} \)
26: \( \text{else} \)
27: \( \text{exit with error message} \)
28: \( \text{end if} \)
29: \( \text{end if} \)
30: \( \text{end while} \)
3.3. REPRESENTATION OF GEOMETRY AND AERODYNAMIC DATA

the exact integral value $I_i$ is unknown, the integration error $e_i$ is approximated by (extrapolation principle, see [16])

$$e_i \approx \frac{|I_{h,i,1} + I_{h,i,2} - I_{h,i}|}{2^p - 1},$$

(3.8)

or more specifically for the Simpson rule ($p=5$)

$$e_i \approx \frac{1}{15}|(I_{h,i,1} + I_{h,i,2}) - I_{h,i}|.$$  (3.9)

An algorithm based on this concept is presented in Algorithm 2. The algorithm subdivides the integration interval in two subintervals and then recursively refines each of these subintervals until the error $e_i$ is small enough to fulfill the abortion criterion.

The algorithm is applied to both, integration of the force and moment coefficients (see Equation (2.40) and (2.40)) and integration of the complex and frequency-dependent derivatives of the force and moment coefficients (see Equation (2.70) and (2.71)), in the same way. When integrating over the angular blade position $\gamma$ evaluation of the function value $f(\gamma)$ corresponds to integration over the blade radius for fixed $\gamma$:

$$I = \int_0^{2\pi} f(\gamma)d\gamma,$$

(3.10)

with

$$f(\gamma) = \int_{R_H}^R f(r, \gamma)dr.$$  (3.11)

### 3.3 Representation of geometry and aerodynamic data

![Figure 3.2. Airfoil sections at different radii.](image)

The geometry of a propeller blade is basically defined by the geometry of the airfoil sections and the twist angle $\tau$ with which they are arranged (see Figure 3.2).
Algorithm 2 Adaptive Simpson Rule (adapted from [16]).

// set initial values
\[ I = 0 \]
\[ t_1 = a, f_1 = f(a) \]
\[ t_m = \frac{a+b}{2}, f_m = f(t_m) \]
\[ t_2 = b, f_2 = f(b) \]
\[ h = b - a \]
\[ S = \frac{h}{6}(f_1 + 4f_m + f_2) \]

function \textsc{integral}(t_1, t_m, t_2, f_1, f_m, f_2, S, h)

// compute midpoints of left and right subinterval and function values
\[ t_l = \frac{1}{2}(t_1 + t_m), f_l = f(t_l) \]
\[ t_r = \frac{1}{2}(t_m + t_2), f_r = f(t_r) \]
\[ h = \frac{h}{2} \]

// determine integral value of left and right subintegral
\[ S_l = \frac{h}{6}(f_1 + 4f_l + f_m) \]
\[ S_r = \frac{h}{6}(f_m + 4f_r + f_2) \]
\[ S_F = S_l + S_r \]

if \( \frac{1}{15}|S_F - S| < \text{tol}(|S_F| + \text{tol}) \) then

\[ I = I + S_F \]

return

else

// refine left and right subintegral recursively
\textsc{integral}(t_1, t_l, t_m, f_1, f_l, f_m, S_l, h)
\textsc{integral}(t_m, t_r, t_2, f_m, f_r, f_2, S_r, h)

end if

end function

In order to solve the BEM equations, it is, however not necessary to represent the actual geometry of the airfoil sections, since only the chord length \( c \) and the aerodynamic data of the airfoil, described by \( c_L \) and \( c_D \)-curves, are of relevance for the calculations.

The distribution of chord length and blade twist over the blade radius is represented as a simple table. An example can be found in Listing 3.1. The first column of the chord and twist table corresponds to the non-dimensional radius

\[ ndr = \frac{r - R}{R - R_{Hub}}. \tag{3.12} \]

In order to determine values in between two data points, spline interpolation is used.

The representation of the aerodynamic data is handled in a similar way. The \( c_L \) and \( c_D \)-data of a specific airfoil geometry are stored as tabulated values in a
3.3. REPRESENTATION OF GEOMETRY AND AERODYNAMIC DATA

Listing 3.1. Example for the file format used to describe the propeller geometry

```xml
<PropGeometry bladeCount="4" hubRadius="0.316" tipRadius="1.625">
  <ChordTable>
    <row>0.0 0.20</row>
    <row>0.2 0.21</row>
    <row>0.4 0.21</row>
    <row>0.6 0.20</row>
    <row>0.8 0.16</row>
    <row>0.9 0.10</row>
    <row>1.0 0.03</row>
  </ChordTable>
  <TwistTable>
    <row>0.00000 31.36172</row>
    <row>0.05000 28.35911</row>
    <row>0.10000 25.52169</row>
    <row>0.15000 22.85904</row>
    <row>0.20000 20.37550</row>
    <row>0.25000 18.07110</row>
    <row>0.30000 15.94241</row>
    <row>0.40000 12.18660</row>
    <row>0.50000 9.04308</row>
    <row>0.60000 6.43656</row>
    <row>0.70000 4.29262</row>
    <row>0.80000 2.54333</row>
    <row>0.90000 1.12933</row>
    <row>1.00000 0.00000</row>
  </TwistTable>
</PropGeometry>

<AirfoilSet>
  <CoeffTable ndr="0.0" fileName="hs1620x.dat"/>
  <CoeffTable ndr="0.2" fileName="hs1712x.dat"/>
  <CoeffTable ndr="0.45" fileName="hs1708x.dat"/>
  <CoeffTable ndr="0.7" fileName="hs1606x.dat"/>
  <CoeffTable ndr="1.0" fileName="hs1404x.dat"/>
</AirfoilSet>
```

simple text file. Listing 3.1 (see AirfoilSet) shows how the distribution of the airfoil sections is described by defining the non-dimensional radius and the corresponding file containing the aerodynamic data for a number of sections. In order to provide values for $c_L$ and $c_D$ over the whole blade radius and range of inflow angles $\alpha_b$ spline interpolation is used here as well.
Chapter 4

Validation

In order to carry out a validation of the BEM model, results obtained with the model have been compared to experiments performed by the National Advisory Committee for Aeronautics (NACA).

4.1 Parallel inflow-conditions

The first set of experimental data is extracted from NACA Report 594 [15], where the performance of six propellers is measured over the full speed range at different blade pitch angles. Although full-size propellers have been used, the wind tunnel speed did not exceed 100 miles per hour and so, even for high advance ratios, the inflow velocity at the blade tips was with around $78 \frac{m}{s}$ far below the speed of sound.

![Thrust and power coefficient for a blade angle $\theta$ of 35° at 0.75R.](image)

**Figure 4.1.** Thrust and power coefficient for a blade angle $\theta$ of 35° at 0.75R.
Therefore, the results should not strongly be affected by compressibility effects. For the validation, the performance characteristic of Propeller Bx is used, since it reached the highest efficiency among all six tested propellers and, with a low twist angle near the blade root, has a geometry, which is most similar to modern propellers. Propeller Bx has a diameter of 10.06 ft and has three blades composed of Clark Y airfoil sections. The description of the propeller geometry is provided in the report and the aerodynamic data of the airfoil sections was generated with the software XFOIL [4].

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure42.png}
\caption{The upper curve shows the thrust coefficient of the propeller. The middle and lower curve shows the section thrust coefficient and the inflow angle of the blade section at $0.75R$. The vertical line indicates an advance ratio of 0.68, for which the stall point of the $0.75R$ section is reached. All three curves correspond to a blade angle of $35^\circ$ at $0.75R$.}
\end{figure}

Figure 4.1 shows the measured performance of the propeller in comparison to the results of the BEM model. For the representation of the thrust coefficient $C_T$...
4.1. PARALLEL INFLOW-CONDITIONS

and power coefficient $C_P$ the definition by NACA is used

$$C_T = \frac{T}{\rho n^2 D^4},$$  \hspace{1cm} (4.1)  

$$C_P = \frac{P}{\rho n^3 D^5} = \frac{2\pi n Q}{\rho n^3 D^5},$$ \hspace{1cm} (4.2)

where $\rho$ is the fluid density, $n$ the number of revolutions per second and $D = 2R$ the propeller diameter. The blade section coefficients are defined as

$$C_{j}' = \frac{dC_j}{dx},$$ \hspace{1cm} (4.3)

where  

$$x = \frac{r}{R}.$$ \hspace{1cm} (4.4)

It can be seen that the BEM model severely deviates from the experimental results for advance ratios lower than one. In this region the airfoil sections experience an inflow angle $\alpha_B$ beyond the stall point over a large part of the blade. Figure 4.2 shows the development of $\alpha_B$ for the blade section at $0.75R$. In this region of the blade the highest aerodynamic loads can usually be observed and therefore the section is representative for the performance of the whole propeller. It can be seen that the airfoil stalls for an advance ratio lower than 0.68. The low section thrust coefficient at low advance ratios corresponds to the rapidly decreasing lift coefficient $c_L$ for inflow angles exceeding the stall point (compare Figure 4.3).

The software used for the generation of the aerodynamic data, XFOIL, makes use of a zonal approach, where integral boundary layer theory is coupled with an inviscid solution of the flow field in regions away from the airfoil surface [4, 5]. The method provides good results as long as only small parts of the airfoil surface experience separated flow. In contrast, when flow separation occurs at a very early position on the upper airfoil surface, as it is the case for high inflow angles, XFOIL is not able to predict the lift coefficient very well. In terms of robustness the BEM solver requires the aerodynamic data to be defined for an inflow angle $\alpha_b$ ranging from $-90^\circ$ to $90^\circ$. XFOIL, however, usually fails to provide results for inflow angles far beyond the
stall point. The aerodynamic data for regions, where XFOIL did not provide results, have been approximated by flat plate theory (see, e.g., [1]). It is assumed that in reality the lift coefficient in the post-stall region is higher than predicted by XFOIL or flat plate theory. This would explain the large deviation of the BEM model from the experimental data at low advance ratios.

In order to further examine this behavior, in Figure 4.4 the inflow angle over the blade radius has been visualized for zero advance, where the model severely deviates from the experimental data, and an advance ratio of 1.2, which is inside a region where the model covers the performance characteristic of the propeller quite well. For zero advance, the inflow angles range in a region, where the lift coefficient is assumed to be poorly approximated, over the whole blade radius. In contrast, for an advance ratio of 1.2 the inflow angle ranges in the attached flow regime of the $c_L$-curve, where XFOIL is expected to provide good accuracy.

Figure 4.5 shows the performance of the propeller for a blade angle of 15° at 0.75R, which would be a more reasonable choice for low advance ratios. Here the overall characteristic of the thrust and power curves is covered quite well by the BEM model. Again, this can be explained by considering the distribution of the inflow angle over the blade radius (see Figure 4.6). It can be seen that for both cases the inflow angle ranges inside the attached flow regime of the $c_L$-curve, where good accuracy of the aerodynamic data can be assumed.

The fact that the BEM model underpredicts the thrust for higher advance ratios (see Figures 4.1 and 4.5) is rather unexpected. Since the Blade Element Momentum Theory is based on a simplified and idealized flow field around the blade sections,
4.1. PARALLEL INFLOW-CONDITIONS

Figure 4.5. Thrust and power coefficient for a blade angle $\theta$ of 15° at 0.75R.

Figure 4.6. Section thrust coefficient, section torque coefficient and inflow angle at $J = 0$ (left) and $J = 0.6$ (right) at a blade angle $\theta$ of 15° at 0.75R.
neglecting any three-dimensional effects, it is, in fact, expected to provide an upper limit for the thrust coefficient. One explanation could be that the presented thrust measurements might be too high. This problem was discussed by the authors of the report, who studied the influence of the shape of the engine cowling on the measurements. The test arrangement is shown in Figure 4.7. They observed that the presence of the propeller slipstream resulted in a reduction of the drag force acting on the nacelle unit. The propeller thrust $T = R + D$ was determined by measuring the axial force $R$ of the whole propeller-nacelle-unit and then adding the drag force $D$ for the corresponding air speed of the nacelle, measured with the propeller off. If the actual value for the drag force of the nacelle unit, however, is reduced due to the slipstream, this approach would result in too high values for the thrust.

![Figure 4.7. Test arrangement and cowling shapes used for the measurements (reprinted from [15]).](image)

### 4.2 Non-parallel inflow-conditions

In NACA report 1205 [10] the effect of inclination of the propeller axis to the airstream on the thrust coefficient was observed. The thrust has been measured for an inclination angle $\alpha$ of $0^\circ$, $-4.55^\circ$ and $-9.8^\circ$ at different blade sections for a number of angular blade positions $\gamma$. A 10 ft propeller with three NACA 10-(3)(08)-03 blades having NACA 16-series blade sections was used in the study. The blade geometry is provided in the report and the airfoil data was generated by XFOIL. The tests have been performed at different rotational speeds ranging from 1350 rpm to 2160 rpm. For the validation, however, only data obtained at the lowest rotational speed of 1350 rpm is considered to avoid compressibility effects as far as possible.

Figure 4.8 shows the development of the thrust coefficient with advance ratio and the corresponding inflow angle at the blade section. As before, it can be seen that
4.2. NON-PARALLEL INFLOW-CONDITIONS

Figure 4.8. Left: Section coefficient and inflow angle of the $0.6R$ blade section over advance ratio. Right: Section coefficient and inflow angle over blade radius at an advance ratio of 1.2. Both curves correspond to an inclination angle of $0^\circ$.

Figure 4.9. Lift coefficient $c_L$ over inflow angle $\alpha_B$ at the $0.60R$ blade section.

the results of the BEM model are in good agreement with the experimental data as long as the inflow angle does not exceed the stall point (see Figure 4.9). No data for the overall propeller performance is available in the report, but measurements for blade sections at different radii are presented. According to Figure 4.10 the section at $0.6R$ was chosen for the validation, since it showed less dependence on the rotational speed than sections at higher radii, which are severely affected by compressibility effects due to the higher tangential velocity. The tangential velocity of the $0.6R$ section at 1350 rpm is around $129 m/s$. On the other hand, the section is close enough to the region where the highest aerodynamic load occurs (see Figure 4.8) and thus should reflect the overall performance of the propeller quite well.

Figure 4.11 and 4.12 show the change in section thrust coefficient with respect to the value at zero inclination over the angular blade position. The definition of
the angular blade position refers to the convention used in the report, where $0^\circ$ is in direction of axis inclination, which, in this case, corresponds to the bottom center of the propeller disk. Not considering the oscillatory effects in the BEM model, the peaks occur at the $90^\circ$ and $270^\circ$ position, as it would be expected by the nature of the model. Incorporating the phase shift resulting from oscillations of the inflow angle at the blade section in the BEM model (see Section 2.6), a phase lag of $8.39^\circ$ can be determined. The phase lag of the experimental values, however, is much higher in magnitude. One reason for the discrepancy is probably that the effects of blade deformation are not included in the model, which would contribute to the phase lag. On the other hand, the authors of the report attribute some amount of the phase lag of the experimental data to be caused by the measurement equipment. This especially accounts for the difference in phase between the minimum and maximum peak at an inclination angle of $-4.55^\circ$, for which they had no other explanation.
4.2. NON-PARALLEL INFLOW-CONDITIONS

Figure 4.11. Change in section thrust coefficient of the 0.6R blade section at an advance ratio of 1.2 over the angular blade position $\gamma$ at an inclination angle $\alpha = -4.55^\circ$. The horizontal error bars indicate a measurement accuracy of $\pm 2^\circ$.

Figure 4.12. Change in section thrust coefficient of the 0.6R blade section at an advance ratio of 1.2 over the angular blade position $\gamma$ at an inclination angle $\alpha = -9.8^\circ$. The horizontal error bars indicate a measurement accuracy of $\pm 2^\circ$. 
Chapter 5

Conclusion

An aerodynamic propeller model for load analysis and the implementation in a software library has been presented.

Validation of the steady-state model showed agreement with experimental data for parallel and non-parallel inflow-conditions as long as the aerodynamic data being used provided good accuracy.

The question whether the accuracy of the model is sufficient for structural load analysis could be a topic for future work. The accuracy of the model can potentially be improved by using a compressible formulation of the Momentum Theory. For practical application this would, however, mean that the aerodynamic data must be provided for a range of Mach numbers.
Bibliography


Appendix A

Jacobian matrix for the Newton solver

Derivation of the governing non-dimensional BEM equations

\[ f_1 = \frac{1}{2} \sigma \lambda_p^2 c_t - 2|\lambda_x + \lambda_a|\lambda_a F \]
\[ = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 c_t - 2(\lambda_x + \lambda_a)\lambda_a F & \text{if } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 c_t + 2(\lambda_x + \lambda_a)\lambda_a F & \text{else} \end{cases} \]
\[ f_2 = \frac{1}{2} \sigma \lambda_p^2 c_q - 2|\lambda_x + \lambda_a|\lambda_r F \]
\[ = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 c_q - 2(\lambda_x + \lambda_a)\lambda_r F & \text{for } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 c_q + 2(\lambda_x + \lambda_a)\lambda_r F & \text{else,} \end{cases} \]

where

\[ c_t = c_Z(\alpha_b) \cos \theta - c_X(\alpha_b) \sin \theta \]
\[ c_q = c_Z(\alpha_b) \sin \theta + c_X(\alpha_b) \cos \theta \]
\[ c_Z = c_L(\alpha_b) \cos \alpha_b + c_D(\alpha_b) \sin \alpha_b \]
\[ c_X = -c_L(\alpha_b) \sin \alpha_b + c_D(\alpha_b) \cos \alpha_b, \]

with respect to \( \lambda_a \) and \( \lambda_r \) yields

\[ \frac{\partial f_1}{\partial \lambda_a} = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_t}{\partial \lambda_a} + \frac{1}{2} \sigma c_t \frac{\partial \lambda_p^2}{\partial \lambda_a} - 2\lambda_a F - 2(\lambda_x + \lambda_a)F - 2(\lambda_x + \lambda_a)\lambda_a \frac{\partial F}{\partial \lambda_a} & \text{if } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_t}{\partial \lambda_a} + \frac{1}{2} \sigma c_t \frac{\partial \lambda_p^2}{\partial \lambda_a} + 2\lambda_a F + 2(\lambda_x + \lambda_a)F + 2(\lambda_x + \lambda_a)\lambda_a \frac{\partial F}{\partial \lambda_a} & \text{else} \end{cases} \]
\[ \frac{\partial f_1}{\partial \lambda_r} = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_t}{\partial \lambda_r} + \frac{1}{2} \sigma c_t \frac{\partial \lambda_p^2}{\partial \lambda_r} - 2(\lambda_x + \lambda_r)\lambda_a \frac{\partial F}{\partial \lambda_a} & \text{if } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_t}{\partial \lambda_r} + \frac{1}{2} \sigma c_t \frac{\partial \lambda_p^2}{\partial \lambda_r} + 2(\lambda_x + \lambda_r)\lambda_a \frac{\partial F}{\partial \lambda_a} & \text{else} \end{cases} \]
\[ \frac{\partial f_2}{\partial \lambda_a} = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_q}{\partial \lambda_a} + \frac{1}{2} \sigma c_q \frac{\partial \lambda_p^2}{\partial \lambda_a} - 2\lambda_r F - 2(\lambda_x + \lambda_a)\lambda_r \frac{\partial F}{\partial \lambda_a} & \text{if } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_q}{\partial \lambda_a} + \frac{1}{2} \sigma c_q \frac{\partial \lambda_p^2}{\partial \lambda_a} + 2\lambda_r F + 2(\lambda_x + \lambda_a)\lambda_r \frac{\partial F}{\partial \lambda_a} & \text{else} \end{cases} \]
\[ \frac{\partial f_2}{\partial \lambda_r} = \begin{cases} \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_q}{\partial \lambda_r} + \frac{1}{2} \sigma c_q \frac{\partial \lambda_p^2}{\partial \lambda_r} - 2(\lambda_x + \lambda_a)F - 2(\lambda_x + \lambda_a)\lambda_r \frac{\partial F}{\partial \lambda_a} & \text{if } \lambda_x + \lambda_a \geq 0 \\ \frac{1}{2} \sigma \lambda_p^2 \frac{\partial c_q}{\partial \lambda_r} + \frac{1}{2} \sigma c_q \frac{\partial \lambda_p^2}{\partial \lambda_r} + 2(\lambda_x + \lambda_a)F + 2(\lambda_x + \lambda_a)\lambda_r \frac{\partial F}{\partial \lambda_a} & \text{else}, \end{cases} \]
with

\[ \frac{\partial \varphi}{\partial \lambda_a} = \frac{\partial}{\partial \lambda_a} \left( \frac{\lambda_x + \lambda_a}{\lambda_t - \lambda_r} \right) = \left[ 1 + \left( \frac{\lambda_x + \lambda_a}{\lambda_t - \lambda_r} \right)^2 \right]^{-1} \left( \lambda_x + \lambda_a \right) \]

\[ \frac{\partial \varphi}{\partial \lambda_r} = \frac{\partial}{\partial \lambda_r} \left( \frac{\lambda_x + \lambda_a}{\lambda_t - \lambda_r} \right) = \frac{\lambda_x + \lambda_a}{\lambda_r^2} \]

\[ \frac{\partial c_t}{\partial \lambda_a} = \frac{\partial c_t}{\partial \lambda_a} \frac{\partial \varphi}{\partial c_t} = \frac{\partial \varphi}{\partial \lambda_a} \left( \frac{\partial c_z}{\partial \alpha_b} \cos \theta - \frac{\partial c_x}{\partial \alpha_b} \sin \theta \right) \]

\[ \frac{\partial c_t}{\partial \lambda_r} = \frac{\partial c_t}{\partial \lambda_r} \frac{\partial \varphi}{\partial c_t} = \frac{\partial \varphi}{\partial \lambda_r} \left( \frac{\partial c_z}{\partial \alpha_b} \sin \theta + \frac{\partial c_x}{\partial \alpha_b} \cos \theta \right) \]

\[ \frac{\partial \lambda_a^2}{\partial \lambda_a} = \frac{\partial}{\partial \lambda_a} \left( (\lambda_x + \lambda_a)^2 + (\lambda_t - \lambda_r)^2 \right) = 2(\lambda_x + \lambda_a) \]

\[ \frac{\partial \lambda_r^2}{\partial \lambda_r} = \frac{\partial}{\partial \lambda_r} \left( (\lambda_x + \lambda_a)^2 + (\lambda_t - \lambda_r)^2 \right) = -2(\lambda_t - \lambda_r) \]

\[ \frac{\partial F}{\partial \varphi} = \frac{N_b (R - r) e^{-f} \cos \varphi}{\pi r \sqrt{1 - e^{-2f} \sin^2 \varphi}} \]

\[ \frac{\partial F}{\partial \lambda_a} = -\frac{\partial \varphi}{\partial \lambda_a} \frac{\partial F}{\partial \varphi} \]

\[ \frac{\partial F}{\partial \lambda_r} = -\frac{\partial \varphi}{\partial \lambda_r} \frac{\partial F}{\partial \varphi} \]

The Jacobian matrix \( J \) used for the solution of the BEM equations is defined as

\[ J(x) := J(\lambda_a, \lambda_r) = \begin{pmatrix} \frac{\partial f_1}{\partial \lambda_a} & \frac{\partial f_1}{\partial \lambda_r} \\ \frac{\partial f_2}{\partial \lambda_a} & \frac{\partial f_2}{\partial \lambda_r} \end{pmatrix} \]
Appendix B

Linearization of the BEM equations with respect to $\alpha$

Derivation of the non-dimensional BEM equations with respect to $\alpha$ yields

$$\frac{\partial f_1}{\partial \alpha} = \frac{1}{2} \sigma \lambda_p \frac{\partial c_t}{\partial \alpha} + \frac{1}{2} \sigma c_t \frac{\partial \lambda_p^2}{\partial \alpha} - 2 \lambda_a \frac{\partial \lambda_x}{\partial \alpha} - 2 \lambda_a F \frac{\partial \lambda_x}{\partial \alpha} - 2 (\lambda_x + \lambda_a) F \frac{\partial \lambda_a}{\partial \alpha} - 2 (\lambda_x + \lambda_a) \lambda_a \frac{\partial F}{\partial \alpha}$$

$$\frac{\partial f_2}{\partial \alpha} = \frac{1}{2} \sigma \lambda_p \frac{\partial c_q}{\partial \alpha} + \frac{1}{2} \sigma c_q \frac{\partial \lambda_p^2}{\partial \alpha} - 2 \lambda_r F \frac{\partial \lambda_x}{\partial \alpha} - 2 \lambda_r F \frac{\partial \lambda_a}{\partial \alpha} - 2 (\lambda_x + \lambda_a) F \frac{\partial \lambda_r}{\partial \alpha} - 2 (\lambda_x + \lambda_a) \lambda_r \frac{\partial F}{\partial \alpha}$$

with

$$\frac{\partial \lambda_x}{\partial \alpha} = -\lambda \cos \beta \sin \alpha$$

$$\frac{\partial \lambda_t}{\partial \alpha} = -\lambda \cos \beta \cos \gamma \cos \alpha$$

$$\frac{\partial \lambda_p^2}{\partial \alpha} = 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_x}{\partial \alpha} + 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_a}{\partial \alpha} + 2 (\lambda_t - \lambda_r) \frac{\partial \lambda_t}{\partial \alpha} + 2 (\lambda_t - \lambda_r) \frac{\partial \lambda_r}{\partial \alpha}$$

$$\frac{\partial \phi}{\partial \alpha} = \frac{1}{\lambda_p^2} \left[ (\lambda_t - \lambda_r) \frac{\partial \lambda_x}{\partial \alpha} + (\lambda_t - \lambda_r) \frac{\partial \lambda_a}{\partial \alpha} + (\lambda_x + \lambda_a) \frac{\partial \lambda_t}{\partial \alpha} + (\lambda_x + \lambda_a) \frac{\partial \lambda_r}{\partial \alpha} \right]$$

$$\frac{\partial c_t}{\partial \alpha} = \frac{\partial \phi}{\partial \alpha} \frac{\partial c_t}{\partial \phi} - \frac{\partial \phi}{\partial \phi} \frac{\partial c_t}{\partial \alpha} = -\frac{\partial \phi}{\partial \alpha} \left[ \frac{\partial c_t(k)}{\partial \alpha} \cos \theta + \frac{\partial c_X}{\partial \alpha} \sin \theta \right]$$

$$\frac{\partial c_q}{\partial \alpha} = \frac{\partial \phi}{\partial \alpha} \frac{\partial c_q}{\partial \phi} - \frac{\partial \phi}{\partial \phi} \frac{\partial c_q}{\partial \alpha} = -\frac{\partial \phi}{\partial \alpha} \left[ \frac{\partial c_t(k)}{\partial \alpha} \sin \theta + \frac{\partial c_X}{\partial \alpha} \cos \theta \right]$$

$$\frac{\partial F}{\partial \alpha} = \frac{\partial \phi}{\partial \alpha} \frac{\partial F}{\partial \phi} - \frac{\partial \phi}{\partial \phi} \frac{\partial F}{\partial \alpha} = \frac{\partial \phi}{\partial \alpha} \left[ \frac{N_b(R - r)}{e^{-2}\cos \varphi} \right]$$

In the operation point it must hold

$$\left. \frac{\partial f_1}{\partial \alpha} \right|_{\alpha_0} = 0, \quad \left. \frac{\partial f_2}{\partial \alpha} \right|_{\alpha_0} = 0.$$
APPENDIX B. LINEARIZATION OF THE BEM EQUATIONS WITH RESPECT TO $\alpha$

which results in the linear system of equations

\[
\begin{aligned}
&\left[-\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\cos \theta - \sin \theta)\right) (\lambda_t - \lambda_r) + \sigma c_t(\lambda_t + \lambda_a) - 2\lambda_a F - 2(\lambda_x + \lambda_a) F\right] \frac{\partial \lambda_a}{\partial \alpha} \\
&+ \left[\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\cos \theta - \sin \theta)\right) (\lambda_x + \lambda_a) - \sigma c_t(\lambda_t - \lambda_r)\right] \frac{\partial \lambda_a}{\partial \alpha} \\
&+ \left[\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\cos \theta - \sin \theta)\right) (\lambda_x + \lambda_a) - \sigma c_t(\lambda_t + \lambda_a) - 2\lambda_a F\right] \frac{\partial \lambda_x}{\partial \alpha} \\
&+ \left[-\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\cos \theta - \sin \theta)\right) (\lambda_t - \lambda_r) + \sigma c_t(\lambda_t - \lambda_r)\right] \frac{\partial \lambda_t}{\partial \alpha}
\end{aligned}
\]

and

\[
\begin{aligned}
&\left[-\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\sin \theta + \cos \theta)\right) (\lambda_t - \lambda_r) + \sigma c_q(\lambda_t + \lambda_a) - 2\lambda_r F\right] \frac{\partial \lambda_a}{\partial \alpha} \\
&+ \left[\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\sin \theta + \cos \theta)\right) (\lambda_t + \lambda_a) - \sigma c_q(\lambda_t - \lambda_r) - 2(\lambda_x + \lambda_a) F\right] \frac{\partial \lambda_a}{\partial \alpha} \\
&+ \left[-\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\sin \theta + \cos \theta)\right) (\lambda_t - \lambda_r) + \sigma c_q(\lambda_t + \lambda_a) - 2\lambda_r F\right] \frac{\partial \lambda_x}{\partial \alpha} \\
&+ \left[-\frac{1}{2}\sigma \left(\frac{\partial c_z}{\partial \sigma_b}(\sin \theta + \cos \theta)\right) (\lambda_t + \lambda_a) + \sigma c_q(\lambda_t - \lambda_r)\right] \frac{\partial \lambda_t}{\partial \alpha}
\end{aligned}
\]

which can be solved for the unknowns $\frac{\partial \lambda_a}{\partial \alpha}$ and $\frac{\partial \lambda_x}{\partial \alpha}$. 

42
Appendix C

Linearization of the BEM equations with respect to \( \beta \)

Derivation of the non-dimensional BEM equations with respect to \( \beta \) yields

\[
\begin{align*}
\frac{\partial f_1}{\partial \beta} &= \frac{1}{2} \alpha x p \frac{\partial c_t}{\partial \beta} + \frac{1}{2} \alpha c_t \frac{\partial \lambda_x}{\partial \beta} - 2 \alpha \lambda_x \frac{\partial \lambda_x}{\partial \beta} - 2 \alpha \lambda_a \frac{\partial \lambda_a}{\partial \beta} - 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_x}{\partial \beta} - 2 (\lambda_x + \lambda_a) \lambda_x \frac{\partial F}{\partial \beta} \\
\frac{\partial f_2}{\partial \beta} &= \frac{1}{2} \alpha x p \frac{\partial c_q}{\partial \beta} + \frac{1}{2} \alpha c_q \frac{\partial \lambda_x}{\partial \beta} - 2 \alpha \lambda_r \frac{\partial \lambda_x}{\partial \beta} - 2 \alpha \lambda_a \frac{\partial \lambda_a}{\partial \beta} - 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_x}{\partial \beta} - 2 (\lambda_x + \lambda_a) \lambda_r \frac{\partial F}{\partial \beta},
\end{align*}
\]

with

\[
\begin{align*}
\frac{\partial \lambda_x}{\partial \beta} &= -\lambda \cos \alpha \sin \beta \\
\frac{\partial \lambda_t}{\partial \beta} &= \lambda \sin \gamma \cos \beta + \lambda \sin \alpha \cos \gamma \sin \beta \\
\frac{\partial \lambda_r}{\partial \beta} &= 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_x}{\partial \beta} + 2 (\lambda_x + \lambda_a) \frac{\partial \lambda_a}{\partial \beta} + 2 (\lambda_t - \lambda_r) \frac{\partial \lambda_t}{\partial \beta} - 2 (\lambda_t - \lambda_r) \frac{\partial \lambda_r}{\partial \beta} \\
\frac{\partial \varphi}{\partial \beta} &= \frac{1}{\lambda_x^2} \left[ (\lambda_t - \lambda_r) \frac{\partial \lambda_x}{\partial \beta} + (\lambda_t - \lambda_r) \frac{\partial \lambda_a}{\partial \beta} - (\lambda_x + \lambda_a) \frac{\partial \lambda_t}{\partial \beta} - (\lambda_x + \lambda_a) \frac{\partial \lambda_r}{\partial \beta} \right] \\
\frac{\partial c_t}{\partial \beta} &= \frac{\partial \varphi}{\partial \beta} \frac{\partial c_t}{\partial \varphi} = -\frac{\partial \varphi}{\partial \beta} \frac{\partial c_t}{\partial \alpha_b} = -\frac{\partial \varphi}{\partial \beta} \left[ \frac{\partial c_z(k)}{\partial \alpha_b} \sin \theta + \frac{\partial c_X}{\partial \alpha_b} \cos \theta \right] \\
\frac{\partial c_q}{\partial \beta} &= \frac{\partial \varphi}{\partial \beta} \frac{\partial c_q}{\partial \varphi} = -\frac{\partial \varphi}{\partial \beta} \frac{\partial c_q}{\partial \alpha_b} = -\frac{\partial \varphi}{\partial \beta} \left[ \frac{\partial c_z(k)}{\partial \alpha_b} \sin \theta + \frac{\partial c_X}{\partial \alpha_b} \cos \theta \right] \\
\frac{\partial F}{\partial \beta} &= \frac{\partial \varphi}{\partial \beta} \frac{\partial F}{\partial \varphi} = -\frac{\partial \varphi}{\partial \beta} \left[ N_b \left( R - r \right) e^{-f} \cos \varphi \right] \\
\frac{\partial c_z}{\partial \beta} &= \frac{\partial \varphi}{\partial \beta} \frac{\partial c_z}{\partial \varphi} = -\frac{\partial \varphi}{\partial \beta} \left[ N_b \left( R - r \right) e^{-f} \sin \varphi \right]
\end{align*}
\]

In the operation point it must hold

\[
\frac{\partial f_1}{\partial \beta} \bigg|_{\alpha_0} = 0, \quad \frac{\partial f_2}{\partial \beta} \bigg|_{\alpha_0} = 0.
\]
which results in the linear system of equations

\[
\begin{align*}
\left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \cos \theta - \frac{\partial c_x}{\partial a} \sin \theta \right) (\lambda_t - \lambda_r) + \sigma c_t(\lambda_x + \lambda_a) - 2\lambda_a F - 2(\lambda_x + \lambda_a) F \\
+ 2(\lambda_x + \lambda_a)(\lambda_t - \lambda_r) + \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_t}{\partial \beta} \\
+ \left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \cos \theta - \frac{\partial c_x}{\partial a} \sin \theta \right) (\lambda_x + \lambda_a) - \sigma c_t(\lambda_t - \lambda_r) \\
+ 2(\lambda_x + \lambda_a)^2 \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_x}{\partial \beta} \\
+ \left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \cos \theta - \frac{\partial c_x}{\partial a} \sin \theta \right) (\lambda_t - \lambda_r) + \sigma c_t(\lambda_x + \lambda_a) - 2\lambda_a F \\
+ 2(\lambda_a + \lambda_a)(\lambda_t - \lambda_r) + \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_a}{\partial \beta}
\right] = 0
\end{align*}
\]

and

\[
\begin{align*}
\left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \sin \theta + \frac{\partial c_x}{\partial a} \cos \theta \right) (\lambda_t - \lambda_r) + \sigma c_q(\lambda_x + \lambda_a) - 2\lambda_r F \\
+ 2(\lambda_x + \lambda_a)(\lambda_t - \lambda_r) + \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_t}{\partial \beta} \\
+ \left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \sin \theta + \frac{\partial c_x}{\partial a} \cos \theta \right) (\lambda_x + \lambda_a) - \sigma c_q(\lambda_t - \lambda_r) - 2(\lambda_x + \lambda_a) F \\
+ 2(\lambda_a + \lambda_a)^2 \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_a}{\partial \beta} \\
+ \left[ -\frac{1}{2} \sigma \left( \frac{\partial c_z(k)}{\partial a} \sin \theta + \frac{\partial c_x}{\partial a} \cos \theta \right) (\lambda_t - \lambda_r) + \sigma c_q(\lambda_x + \lambda_a) - 2\lambda_r F \\
+ 2(\lambda_a + \lambda_a)(\lambda_t - \lambda_r) + \lambda_a N_b(R - r) e^{-f} \cos \varphi \right] \frac{\partial \lambda_a}{\partial \beta}
\right] = 0
\end{align*}
\]

which can be solved for the unknowns \( \frac{\partial \lambda_t}{\partial \beta} \) and \( \frac{\partial \lambda_a}{\partial \beta} \).
Appendix D

Code documentation

libprop
1.0

Generated by Doxygen 1.7.6.1

Sun Jul 8 2012 11:27:01
Contents

1 Class Index 1

2 Class Documentation 2

1.1 Class List ........................................ 2

1.1.1 Detailed Description .......................... 3

1.1.2 Member Function Documentation .............. 4

1.2 AdaptiveIntegrator Class Reference ............... 5

1.2.1 Detailed Description .......................... 6

1.2.2 Member Function Documentation .............. 7

1.3 AirfoilSet Class Reference .......................... 8

1.3.1 Detailed Description .......................... 9

1.3.2 Member Function Documentation .............. 10

1.4 AirfoilSet::CoefTable Struct Reference .......... 11

1.5 Integrator Class Reference ........................ 12

1.5.1 Detailed Description .......................... 13

1.5.2 Member Function Documentation .............. 14

1.6 NPoint Struct Reference ........................... 15

1.6.1 Detailed Description .......................... 16

1.7 Propeller Class Reference ......................... 17

1.7.1 Detailed Description .......................... 18

1.7.2 Member Function Documentation .............. 19

1.8 PropGeometry Class Reference ..................... 20

1.8.1 Detailed Description .......................... 21

1.9 ScalarSpline Class Reference ..................... 22

1.9.1 Detailed Description .......................... 23

1.10 Section Class Reference ........................... 24

1.10.1 Detailed Description .......................... 25

1.10.2 Member Function Documentation .............. 26

1.11 SectionValues Struct Reference .................. 27

1.11.1 Detailed Description .......................... 28
1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- **AdaptiveIntegrator**
  Adaptive integration of real-valued and complex functions  
  [Page 2]

- **AdaptiveIntegratorInfo**
  Information about integration performed by `AdaptiveIntegrator`  
  [Page 5]

- **AirfoilSet**
  Airfoil data for propeller blade  
  [Page 6]

- **AirfoilSet::CoefTable**
  [Page 8]

- **Integrator**
  Integration over the propeller disk  
  [Page 9]

- **NPoint**
  Non-dimensional operating point  
  [Page 10]

- **Propeller**
  Propeller analysis interface  
  [Page 11]

- **PropGeometry**
  Container for propeller geometry  
  [Page 13]

- **ScalarSpline**
  Simple cubic spline in one variable  
  [Page 14]

- **Section**
  Blade section solver  
  [Page 15]

- **SectionValues**
  Information about solution of the blade section  
  [Page 18]

2 Class Documentation

2.1 AdaptiveIntegrator Class Reference

Adaptive integration of real-valued and complex functions.
@include <adaptiveintegrator.h>

**Static Public Member Functions**

- template<class Functor , typename Type>
  static void integrate1D (Functor &f, Type &I, Real xstart, Real xend, Real hmin, 
  Real tol, AdaptiveIntegratorInfo &info)
One-dimensional integration of a scalar integrand.

- template<class Functor, typename Type>
  static void integrate1D (Functor &f, Type &I, Real xstart, Real xend, Real hmin, Real tol)

One-dimensional integration of a scalar integrand.

- template<class Functor, uint N, typename Type>
  static void integrateVec1D (Functor &f, SVector<N, Type> &I, Real xstart, Real xend, Real hmin, Real tol, bool check[], AdaptiveIntegratorInfo &info)

One-dimensional integration of an integrand vector.

- template<class Functor, uint N, typename Type>
  static void integrateVec1Drec (Functor &f, const SVector<N, Type> &S, SVector<N, Type>&I, const SVector<N, Type> &f1, const SVector<N, Type> &fm, const SVector<N, Type> &f2, Real x1, Real xm, Real x2, Real hmin, Real tol, bool check[], AdaptiveIntegratorInfo &info)

recursive call of the vector integration

- template<class Functor, typename Type>
  static void integrate1Drec (Functor &f, Type S, Type &I, Type f1, Type fm, Type f2, Real x1, Real xm, Real x2, Real hmin, Real tol, AdaptiveIntegratorInfo &info)

recursive call of the scalar integration

2.1.1 Detailed Description

Adaptive integration of real-valued and complex functions.

Adaptive integration of a real-valued or complex function \( f(x) \) in the real valued interval \([x_{start}, x_{end}]\) using Simpson’s quadrature rule. The integrand \( f(x) \) can be scalar or a vector

\[
I = \int_{x_{start}}^{x_{end}} f(x)\,dx
\]

The algorithm subdivides the integration interval \([x_{start}, x_{end}]\) in \([x_{start}, x_{mid}]\) and \([x_{mid}, x_{end}]\) and then recursively refines each of the subintervals until for each subinterval the estimated integration error is below a given tolerance \( tol \). In case that the integrand is a vector, a boolean array \( check[] \) (of the same length as the vector) has to be given, which specifies for each component of the vector, if the error tolerance must be fulfilled for this component. The integration integral will be refined, until the tolerance is fulfilled for all components, where \( check[] \) is set to true.


The integrand has to be provided as a functor, where the function value is returned by operator(). For example:
In order to retrieve information about the number of function evaluations and the minimum integration interval that have been used for the integration, the class AdaptiveIntegratorInfo can be used.

### 2.1.2 Member Function Documentation

#### 2.1.2.1 template<class Functor, typename Type>

```cpp
template<class Functor, typename Type>
static void AdaptiveIntegrator::integrate1D(Functor & f, Type & I, Real xstart, Real xend, Real hmin, Real tol, AdaptiveIntegratorInfo & info) [inline, static]
```

One-dimensional integration of a scalar integrand.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f</code></td>
<td>functor for the integrand function</td>
</tr>
<tr>
<td><code>I</code></td>
<td>integral value to be returned</td>
</tr>
<tr>
<td><code>xstart</code></td>
<td>lower bound of integration interval</td>
</tr>
<tr>
<td><code>xend</code></td>
<td>upper bound of integration interval</td>
</tr>
<tr>
<td><code>hmin</code></td>
<td>minimum stepsize that is allowed</td>
</tr>
<tr>
<td><code>tol</code></td>
<td>tolerance for the estimated integration error</td>
</tr>
<tr>
<td><code>info</code></td>
<td>information about integration</td>
</tr>
</tbody>
</table>

#### 2.1.2.2 template<class Functor, typename Type>

```cpp
template<class Functor, typename Type>
static void AdaptiveIntegrator::integrate1D(Functor & f, Type & I, Real xstart, Real xend, Real hmin, Real tol) [inline, static]
```

One-dimensional integration of a scalar integrand.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f</code></td>
<td>functor for the integrand function</td>
</tr>
<tr>
<td><code>I</code></td>
<td>integral value to be returned</td>
</tr>
<tr>
<td><code>xstart</code></td>
<td>lower bound of integration interval</td>
</tr>
<tr>
<td><code>xend</code></td>
<td>upper bound of integration interval</td>
</tr>
<tr>
<td><code>hmin</code></td>
<td>minimum stepsize that is allowed</td>
</tr>
<tr>
<td><code>tol</code></td>
<td>tolerance for the estimated integration error</td>
</tr>
</tbody>
</table>
One-dimensional integration of a integrand vector.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>functor for the integrand function</td>
</tr>
<tr>
<td>( I )</td>
<td>integral value to be returned</td>
</tr>
<tr>
<td>( x_{\text{start}} )</td>
<td>lower bound of integration interval</td>
</tr>
<tr>
<td>( x_{\text{end}} )</td>
<td>upper bound of integration interval</td>
</tr>
<tr>
<td>( h_{\text{min}} )</td>
<td>minimum stepsize that is allowed</td>
</tr>
<tr>
<td>( \text{tol} )</td>
<td>tolerance for the estimated integration error</td>
</tr>
<tr>
<td>( \text{check} )</td>
<td>specifies for each component of the vector if the tolerance has to be fulfilled</td>
</tr>
<tr>
<td>( \text{info} )</td>
<td>information about integration</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following file:

- adaptiveintegrator.h

### 2.2 AdaptiveIntegratorInfo Struct Reference

Information about integration performed by \texttt{AdaptiveIntegrator}.

```cpp
#include <adaptiveintegrator.h>
```

**Public Attributes**

- \texttt{uint numEval}
2.2.1 Detailed Description

Information about integration performed by AdaptiveIntegrator.

An AdaptiveIntegratorInfo object can be used in combination with AdaptiveIntegrator in order to retrieve information about the integration, e.g. the number of function evaluation or the minimum stepsize that has been used.

The documentation for this struct was generated from the following file:

- adaptiveintegrator.h

2.3 AirfoilSet Class Reference

Airfoil data for propeller blade.

#include <airfoilset.h>

Classes

- struct CoefTable

Public Member Functions

- AirfoilSet ()
  
  empty set
- void eval (Real ndr, Real ap, Real &CL, Real &CD) const
  
  Evaluate lift and drag at non-dimensional radius and alpha parameter.
- void evalB (Real ndr, Real ap, Real &Cz, Real &Cx) const
  
  Evaluate normal and axial force coefficient at non-dimensional radius and alpha.
- void tangent (Real ndr, Real ap, Real CL[], Real CD[]) const
  
  Determine value and first derivative with respect to alpha parameter.
- void tangentB (Real ndr, Real ap, Real CZ[], Real CX[]) const
  
  determine value and first derivative of normal and axial force coefficient with respect to alpha parameter
- Complex czak (Real ndr, Real ap, Real k) const
  
  frequency-dependent normal force derivative
- void cak (Real ndr, Real ap, Real k, Complex CZ[], Real CX[]) const
  
  determine value and first derivative of frequency-dependant normal force coefficient and axial force coefficient with respect to alpha parameter
- XmlElement toXml (bool share=false) const
create XML representation
• void fromXml (const XmlElement &xe)
  recover contents from XML representation

Private Types
• typedef std::vector< CoefTable > CoefTableArray

Private Member Functions
• void appendTable (const std::string &fileName, Real ndr, CoefTableArray &tables)
  const
  add (alpha, CL, CD) table to interpolation dataset
• void interpolateTables (const CoefTableArray &tables)
  interpolate tabulated dataset

Private Attributes
• SplineBasis m_radialBasis
  spline basis for evaluation in radial direction
• SplineBasis m_alphaBasis
  spline basis for interpolation in alpha
• Matrix m_pcl
  control grid for lift coefficient
• Matrix m_pcd
  control grid for drag coefficient

2.3.1 Detailed Description

Airfoil data for propeller blade.

AirfoilSet contains polars, that is, CL(alpha) and CD(alpha) as a smooth interpolation over both radial and alpha directions. In order to generate and initial representation, an airfoil set may be represented by an XmlElement file which references plain text files containing listings of alpha, CL and CD. Such listings must always cover the range from -90 degree to +90 degree alpha. In the case that the available data does not cover the full range, a matlab tool in the ‘polar’ folder can be used to extend polar data using flat-plate theory.

AirfoilSet uses both radial and alpha parameters instead of the actual radius and angle of attack values. The definition of these non-dimensional parameters is:

\[ \tilde{r} = \frac{r - r_H}{r_T - r_H} \]
and
\[ \tilde{\alpha} = \frac{\alpha}{\pi} + \frac{1}{2}, \]
so that both parameters are defined over the interval [0,1].
### Member Function Documentation

#### 2.3.2.1 void AirfoilSet::eval ( Real ndr, Real ap, Real & CL, Real & CD ) const

[inline]

Evaluate lift and drag at non-dimensional radius and alpha parameter.
For ap values exceeding the valid range, e.g. 0 <= ap <= 1, the values for CL and CD are extrapolated as follows:

\[
C_L(\alpha > 1.0) = C_L(1.0) + C_{L,\alpha}(1.0)(\alpha - 1.0)
\]

\[
C_L(\alpha < 0.0) = C_L(0.0) + C_{L,\alpha}(0.0)\alpha
\]

\[
C_D(\alpha > 1.0) = C_D(1.0) + C_{D,\alpha}(1.0)(\alpha - 1.0)
\]

\[
C_D(\alpha < 0.0) = C_D(0.0) + C_{D,\alpha}(0.0)\alpha.
\]

#### 2.3.2.2 void AirfoilSet::tangent ( Real ndr, Real ap, Real CL[], Real CD[] ) const

[inline]

Determine value and first derivative with respect to alpha parameter.
For ap values exceeding the valid range, e.g. 0 <= ap <= 1, the values for \(C_L, C_D, C_{L,\alpha}\) and \(C_{D,\alpha}\) are extrapolated as follows:

\[
C_L(\alpha > 1.0) = C_L(1.0) + C_{L,\alpha}(1.0)(\alpha - 1.0)
\]

\[
C_L(\alpha < 0.0) = C_L(0.0) + C_{L,\alpha}(0.0)\alpha
\]

\[
C_D(\alpha > 1.0) = C_D(1.0) + C_{D,\alpha}(1.0)(\alpha - 1.0)
\]

\[
C_D(\alpha < 0.0) = C_D(0.0) + C_{D,\alpha}(0.0)\alpha.
\]

The documentation for this class was generated from the following files:

- airfoilset.h
- airfoilset.cpp

### Public Attributes

- Vector alpha
- Vector CL
- Vector CD
- Real ndr
2.5 Integrator Class Reference

Integration over the propeller disk.

#include <integrator.h>

Static Public Member Functions

- static void solve (const AirfoilSet &afs, const PropGeometry &geo, const NPoint &opp, Real c[]) [static]
  integrate force and moment coefficients over the propeller disk for steady inflow conditions

- static void uSolveA (const AirfoilSet &afs, const PropGeometry &geo, const NPoint &opp, Real freq, Complex c[]) [static]
  integrate complex and frequency dependent derivatives of the force and moment coefficient with respect to alpha over the propeller disk for unsteady inflow conditions

- static void uSolveB (const AirfoilSet &afs, const PropGeometry &geo, const NPoint &opp, Real freq, Complex c[]) [static]
  integrate complex and frequency dependent derivatives of the force and moment coefficient with respect to beta over the propeller disk for unsteady inflow conditions

2.5.1 Detailed Description

Integration over the propeller disk.

Integration of the force and moment coefficients over the propeller disk for steady inflow conditions.

Integration of the complex and frequency dependent derivatives of the force and moment coefficient with respect to alpha or beta over the propeller disk for unsteady inflow conditions.

2.5.2 Member Function Documentation

2.5.2.1 void Integrator::solve ( const AirfoilSet & afs, const PropGeometry & geo, const NPoint & opp, Real c[]) [static]

integrate force and moment coefficients over the propeller disk for steady inflow conditions

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>afs</td>
<td>airfoil data</td>
</tr>
<tr>
<td>geo</td>
<td>propeller geometry</td>
</tr>
<tr>
<td>opp</td>
<td>non-dimensional operation point</td>
</tr>
<tr>
<td>c</td>
<td>force and moment coefficients to be returned (array of length six)</td>
</tr>
</tbody>
</table>

Generated on Sun Jul 8 2012 11:27:01 for libprop by Doxygen
2.5.2.2 void Integrator::uSolveA (const AirfoilSet & afs, const PropGeometry & geo, const NPoint & opp, Real freq, Complex c[]) [static]

Integrate complex and frequency dependent derivatives of the force and moment coefficient with respect to alpha over the propeller disk for unsteady inflow conditions.

**Parameters**

<table>
<thead>
<tr>
<th>afs</th>
<th>airfoil data</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo</td>
<td>propeller geometry</td>
</tr>
<tr>
<td>opp</td>
<td>non-dimensional operation point</td>
</tr>
<tr>
<td>freq</td>
<td>normalized frequency, which is defined as ( f_n = \frac{f \omega}{R} ).</td>
</tr>
<tr>
<td>c</td>
<td>complex-valued derivatives of force and moment coefficients to be returned (array of length six)</td>
</tr>
</tbody>
</table>

2.5.2.3 void Integrator::uSolveB (const AirfoilSet & afs, const PropGeometry & geo, const NPoint & opp, Real freq, Complex c[]) [static]

Integrate complex and frequency dependent derivatives of the force and moment coefficient with respect to beta over the propeller disk for unsteady inflow conditions.

**Parameters**

<table>
<thead>
<tr>
<th>afs</th>
<th>airfoil data</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo</td>
<td>propeller geometry</td>
</tr>
<tr>
<td>opp</td>
<td>non-dimensional operation point</td>
</tr>
<tr>
<td>freq</td>
<td>normalized frequency, which is defined as ( f_n = \frac{f \omega}{R} ).</td>
</tr>
<tr>
<td>c</td>
<td>complex-valued derivatives of force and moment coefficients to be returned (array of length six)</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following files:

- integrator.h
- integrator.cpp

2.6 NPoint Struct Reference

Non-dimensional operating point.

```cpp
#include <npoint.h>
```

**Public Member Functions**

- `NPoint()`
  - `create default point`
Public Attributes

- Real advanceRatio
  non-dimensional advance ratio
- Real bpitchAngle
  blade pitch angle
- Real alpha
  vertical inflow angle
- Real beta
  horizontal inflow angle

2.6.1 Detailed Description

Non-dimensional operating point.

The non-dimensional operation point is defined by the non-dimensional rate of advance
\( \lambda = \frac{V}{\omega R} \), the pitch angle of the blades, the vertical inflow angle \( \alpha \) and the horizontal inflow angle \( \beta \). The angles have to be given in radians.

See also

Propeller

The documentation for this struct was generated from the following file:

- npoint.h

2.7 Propeller Class Reference

Propeller analysis interface.

#include <propeller.h>

Public Member Functions

- Propeller ()
  create empty propeller object
- Propeller (const PropGeometry &geo, const AirfoilSet &afdata)
  construct from airfoil data and external geometry
- void rename (const std::string &s)
  change propeller name
- const std::string & name () const
  access propeller name
- void eval (const NPoint &op, Real c[])
  determine force and moment coefficients for given operation point
- void sweepPitch (const NPoint &op, Matrix &mcf, int np=51) const
2.7 Propeller Class Reference

**perform a sweep over blade pitch (debugging/validation)**

- Real bestBladePitch (const NPoint &op) const
determine optimal blade pitch angle for given operation point

- void createTable (uint nj, uint nalpha, Real Jmax, Real alphaMax, NArray< 3, Real > &ctab) const
  write a force interpolation table to be read into matlab

- void createTable (const Vector &Jv, const Vector &alfav, const Vector &phiv, NArray< 4, Real > &ctab) const
  write a force interpolation table to be read into matlab

- XmlElement toXml (bool share=false) const
  export propeller definition to XML

- void fromXml (const XmlElement &xe)
  recover definition from XML

**Private Attributes**

- PropGeometry m_geo
  propeller geometry, blade count etc.

- AirfoilSet m_airfoil
  airfoils used along the blade radius

- std::string m_sid
  propeller identification

**2.7.1 Detailed Description**

**Propeller** analysis interface.

Top-level interface for propeller analysis. Objects of class Propeller own geometry and airfoil data provide an interface to evaluate force and moment coefficients for a given non-dimensional operating point.

Force and moment coefficients are made non-dimensional using the following relations

\[
C_{Fx} = \frac{2F_x}{\rho (\omega R)^2 \pi R^2}
\]

\[
C_{Fy} = \frac{2F_y}{\rho (\omega R)^2 \pi R^2}
\]

\[
C_{Fz} = \frac{2F_z}{\rho (\omega R)^2 \pi R^2}
\]

\[
C_{Mx} = \frac{2M_x}{\rho (\omega R)^3 \pi R^3}
\]

\[
C_{My} = \frac{2M_y}{\rho (\omega R)^3 \pi R^3}
\]

\[
C_{Mz} = \frac{2M_z}{\rho (\omega R)^3 \pi R^3}
\]
2.8 PropGeometry Class Reference

See also

PropGeometry, AirfoilSet, NPoint

2.7.2 Member Function Documentation

2.7.2.1 void Propeller::eval ( const NPoint & op, Real c[] ) const
determine force and moment coefficients for given operation point

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>non-dimensional operation point</td>
</tr>
<tr>
<td>c</td>
<td>array of length 6 to store force and moment coefficients to be returned</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following files:

- propeller.h
- propeller.cpp

2.8 PropGeometry Class Reference

Container for propeller geometry.

#include <propgeometry.h>

Public Member Functions

- PropGeometry ()
  undefined propeller geometry
- Real solidity (Real ndr) const
  compute blade solidity fraction at non-dimensional radius r
- Real bladeRadius (Real ndr) const
  compute dimensional radius at non-dimensional coordinate ndr
- Real bladeChord (Real ndr) const
  interpolate chord at non-dimensional radius coordinate ndr
- Real bladeTwist (Real ndr) const
  interpolate twist
- Real bladeLength () const
  blade length
- uint bladeNumber () const
  number of blades
- XmlElement toXml (bool share=false) const
  return XML representation
- void fromXml (constXmlElement &xe)
  recover from XML representation
2.9 ScalarSpline Class Reference

Private Member Functions

- void tableToSpline (const XmlElement &xe, ScalarSpline &spl) const
  convert text table embedded in XML description into spline

Private Attributes

- Real m_hubRadius
  hub radius
- Real m_tipRadius
  blade tip radius
- uint m_nblade
  number of blades
- ScalarSpline m_chordSpline
  blade chord distribution
- ScalarSpline m_twistSpline
  blade twist distribution

2.8.1 Detailed Description

Container for propeller geometry.
This class holds the external geometry definitions of the propeller and stores a spline interpolation of the radial chord and twist distribution.
The documentation for this class was generated from the following files:

- propgeometry.h
- propgeometry.cpp

2.9 ScalarSpline Class Reference

Simple cubic spline in one variable.
#include <scalarspline.h>

Public Member Functions

- ScalarSpline ()
  undefined spline
- Real eval (Real t) const
  evaluate spline
- Real derive (Real t, uint k=1) const
  first derivative
- void scale (Real f)
modify control point values by scaling

• void interpolate (const Vector &x, const Vector &y)
  create spline by interpolating a set of values

• XmlElement toXml (bool share=false) const
  return XML representation

• void fromXml (const XmlElement &xe)
  recover contents from XML representation

Private Attributes

• SplineBasis m_basis
  basis

• Vector m_cp
  control point values

2.9.1 Detailed Description

Simple cubic spline in one variable.

A very simple one-variable spline object limited to cubic interpolation.

The documentation for this class was generated from the following files:

• scalarspline.h
• scalarspline.cpp

2.10 Section Class Reference

Blade section solver.

#include <section.h>

Public Member Functions

• Section (const AirfoilSet &airfoilset, const PropGeometry &propgeometry, const NPoint &oppoint)
  initialize section

• bool solve (const Real ndr, const Real gamma, Vct6 &c)
  obtain force and moment coefficient for blade section at non-dimensional radius ndr and angular blade position \( \gamma \)

• bool solve (const Real ndr, const Real gamma, Vct6 &c, SectionValues &sval)
  obtain force and moment coefficient for blade section at non-dimensional radius ndr and angular blade position \( \gamma \)

• bool dcoeffAlpha (const Real ndr, const Real gamma, const Real freq, CpxVct6 &dc, Complex &dla, Complex &dlr)
Obtain frequency-dependent derivatives of elemental force and moment coefficients with respect to alpha for given normalized frequency.

- bool dcoeffBeta (const Real ndr, const Real gamma, const Real freq, CpxVct6 &dc, Complex &dla, Complex &drl)

Obtain frequency-dependent derivatives of elemental force and moment coefficients with respect to beta for given normalized frequency.

Private Member Functions

- Real TL (const Real phi, const Real r, const Real R) const
  Prandtl tip loss factor.
- Real getInitial (Real ndr) const
  get initial value for induced velocity
- void Jacobian (Real ndr, Real va, Real vr, Real f[], Real J[], const Real h) const
  approximate Jacobian of \( F \) with central finite differences
- void F (Real ndr, Real va, Real vr, Real &f1, Real &f2) const
  Evaluate values of the governing equations of the Blade Element Momentum equations.
- void JacobianA (Real ndr, Real la, Real lr, Real f[], Real J[]) const
  evaluate \( F \) and Analytical Jacobian of \( F \)

Private Attributes

- const AirfoilSet m_airfoilset
  contains airfoil sections used along the propeller blade
- const PropGeometry m_propgeometry
  geometry of the propeller blade
- const NPoint m_oppoint
  operation point
- Real m_r
  radius of the blade section
- Real m_R
  tip radius
- Real m_theta
  theta angle (sum of blade twist angle and blade pitch angle)
- Real m_stheta
  sine of theta
- Real m_ctheta
  cosine of theta
- Real m_J
  advance ratio
- Real m_AoA
  angle of attack with respect to propeller axis
- Real m_beta
2.10 Section Class Reference

sideslip angle with respect to propeller axis

- Real $m_{\text{Jx}}$
  axial advance ratio
- Real $m_{\text{Jt}}$
  tangential advance ratio
- Real $m_{\text{sigma}}$
  blade solidity

2.10.1 Detailed Description

Blade section solver.

1) Solution of the governing Blade Element Momentum equations in order to determine the forces and moments (coefficients) that are acting on the blade section at a given radius and angular position of the blade.

2) Obtain frequency-dependent derivatives of the force and moment coefficients with respect to $\alpha$ or $\beta$ for the blade section at a given radius, angular blade position and frequency.

2.10.2 Member Function Documentation

2.10.2.1 bool Section::dcoeffAlpha(const Real ndr, const Real gamma, const Real freq, CpxVct6 & dc, Complex & dla, Complex & dlr)

Obtain frequency-dependent derivatives of elemental force and moment coefficients with respect to alpha for given normalized frequency.

Parameters

- $ndr$ non-dimensional radius $ndr = \frac{r - R_{\text{Hub}}}{R_{\text{Tip}} - R_{\text{Hub}}}$
- $gamma$ angular blade position $\gamma \ [\text{rad}]$
- $freq$ normalized frequency $fn = \frac{f}{\omega}$
- $dc$ complex-valued derivatives of force and moment coefficients to be returned
- $dla$ derivate of axial induced velocity $\frac{\partial}{\partial \alpha} \lambda_a$ to be returned
- $dlr$ derivate of tangential induced velocity $\frac{\partial}{\partial \alpha} \lambda_r$ to be returned

2.10.2.2 bool Section::dcoeffBeta(const Real ndr, const Real gamma, const Real freq, CpxVct6 & dc, Complex & dla, Complex & dlr)

Obtain frequency-dependent derivatives of elemental force and moment coefficients with respect to beta for given normalized frequency.

Parameters

- $ndr$ non-dimensional radius $ndr = \frac{r - R_{\text{Hub}}}{R_{\text{Tip}} - R_{\text{Hub}}}$
- $gamma$ angular blade position $\gamma \ [\text{rad}]$
### freq
normalized frequency \( f_n = \frac{f}{\omega} \)

### dc
complex-valued derivatives of force and moment coefficients to be returned

### dla
derivate of axial induced velocity \( \frac{\partial}{\partial \beta} \lambda_a \) to be returned

### dla
derivate of tangential induced velocity \( \frac{\partial}{\partial \beta} \lambda_r \) to be returned

#### 2.10.2.3 bool Section::solve ( const Real ndr, const Real gamma, Vct6 & c ) [inline]

obtain force and moment coefficient for blade section at non-dimensional radius \( ndr \) and angular blade position \( \gamma \)

**Parameters**

- **ndr**: non-dimensional radius \( ndr = \frac{r - R_{Hub}}{R_{Tip} - R_{Hub}} \)
- **gamma**: angular blade position \( \gamma \) [rad]
- **c**: force and moment coefficients to be returned

#### 2.10.2.4 bool Section::solve ( const Real ndr, const Real gamma, Vct6 & c, SectionValues & sval )

obtain force and moment coefficient for blade section at non-dimensional radius \( ndr \) and angular blade position \( \gamma \)

**Parameters**

- **ndr**: non-dimensional radius \( ndr = \frac{r - R_{Hub}}{R_{Tip} - R_{Hub}} \)
- **gamma**: angular blade position \( \gamma \) [rad]
- **c**: force and moment coefficients to be returned
- **sval**: detailed information about the solution

The documentation for this class was generated from the following files:

- section.h
- section.cpp

#### 2.11 SectionValues Struct Reference

Information about solution of the blade section.

#include &lt;section.h&gt;

**Public Attributes**

- **Real r**
  
radius of current section
2.11 SectionValues Struct Reference

- Real $R$
  tip radius
- Real $l$
  advance ratio $\lambda$
- Real $lx$
  axial advance ratio $\lambda_x$
- Real $lt$
  tangential advance ratio $\lambda_t$
- Real $la$
  ratio of axial induction $\lambda_a$
- Real $lr$
  ratio of tangential induction $\lambda_r$
- Real $lp^2$
  $\lambda_p^2$
- Real $phi$
  $\phi$
- Real $alphaB$
  blade inflow angle $\alpha_b$
- Real $theta$
  $\theta$
- Real $sigma$
  blade solidity $\sigma$
- Real $ct$
  thrust component $c_t$
- Real $cq$
  torque component $c_q$
- Real $F$
  tip loss factor
- Real $AoA$
  angle of attack $\alpha$
- Real $beta$
  angle of attack $\beta$

2.11.1 Detailed Description

Information about solution of the blade section.

Can be used in combination with Section in order to retrieve detailed information about the solution.

The documentation for this struct was generated from the following file:

- section.h