

Requirements for a Wind Turbine Aerodynamics Simulation Module

Version 1

A. van Garrel

Abstract

In this report requirements are formulated for a new wind turbine aerodynamics simulation module. The new module will more accurately model rotor wake physics, thus reducing the number of uncertainties that accompany current blade-element-momentum methods. The requirements are subdivided into functional, non-functional and future requirements and will serve as a guide during the development of the new aerodynamics simulation module.

Acknowledgement This report is part of the project “Aerodynamic Windturbine Simulation Module”, In this project a new windturbine aerodynamics simulation code will be developed.

The project is partly funded by NOVEM, the Netherlands Agency for Energy and the Environment, contract number 224.312-0001. Additional funding is provided by ECN, project number 7.4318.

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Objective	1
1.3	Strategy	1
1.4	Overview	1
2	SYSTEM REQUIREMENTS	3
2.1	Functional Requirements	3
2.1.1	Allowable flows and geometries	3
2.1.2	Computed output quantities	5
2.2	Non-functional Requirements	5
2.3	Future Requirements	7
3	CONCLUSIONS	9

1 INTRODUCTION

1.1 Motivation

Current computer codes for the simulation of the aerodynamic behavior of wind turbines are all based on the relatively simple blade-element-momentum (BEM) theory. In this theory a steady and homogeneous onset flow is assumed. Moreover, forces are supposed to be distributed over independently operating annuli. In order to extend the range applicability of BEM theory, numerous empirical and semi-empirical correction formulae have been introduced. For example, there are correction formulae that try to take into account the effects of yaw misalignment, dynamic inflow, dynamic stall, blade tips, tower influence, the finite number of blades, blade cone angle and the influence of rotation on the sectional aerodynamic characteristics.

The fact that these correction formulae are largely based on (semi-)empirical considerations implies the introduction of substantial uncertainties. Moreover, because correction factors are tuned on the basis of few experimental results, the reliability of these correction factors is questionable in dissimilar situations. This reduction of these uncertainties become more and more important now wind turbines are getting larger and are placed offshore. The reduction of the number of (semi-)empirical correction formulae through a more advanced model of the physics involved is the goal of this project.

1.2 Objective

As a first step in the development of a future system for the reliable simulation of the aeroelastic behavior of wind turbines a new time-accurate aerodynamics module will be developed. This module will be based on non-linear vortex-line theory in which the shape and strength of the wake of the blades will develop in time. The aerodynamic lift-, drag-, and pitching-moment characteristics of the blade cross-sections are assumed to be known and corrected for the effects of blade rotation. Generally speaking, more accurate predictions can be expected in situations where local aerodynamic characteristics strongly vary in time and where dynamic wake effects play a significant role.

A general interface will be developed that in the future enables an easy coupling with a module for the structural dynamic behavior of the windturbine.

1.3 Strategy

The development of the new windturbine aerodynamics simulation module will be done in a number of phases. In the first phase the system requirements will be formulated that define the required functionality of the final computer program. The requirements serve as a guide during the development of the system and as a reference during the final assessment. In the next phase the mathematical, numerical and software models of the aerodynamics and fluid-structure-interaction (FSI) algorithms are constructed. This is followed by the implementation of the design in actual programming language. In the final phase, the resulting computer program will be subjected to some selected verification and validation test cases.

1.4 Overview

In the current report the requirements are formulated which the resulting computer program has to meet. A subdivision will be made into three sections: the functional-, the non-functional- and

the future requirements. The functional requirements describe what the computer program is supposed to do, as seen from the standpoint of the user. The specification of the non-functional requirements define the framework in which the software system should deliver the specified functionality and addresses concerns such as performance and platform. In a separate section the specification of anticipated future requirements is dealt with. As a result, during software design allowance can be made for possible future code enhancements.

It should be noted that the underlying report is intended to be a “life” document that serves as a guide during the software design process. This flexibility allows new insights and ideas to be adopted in the course of the project.

2 SYSTEM REQUIREMENTS

2.1 Functional Requirements

In this section the functional requirements are described that state the behavior that the wind turbine aerodynamics simulation module must possess to perform its operational role. A technical description of the capabilities of the system is given as seen from the standpoint of the user. Where possible, functionality that will *not* be incorporated in the first code release will be mentioned explicitly.

2.1.1 Allowable flows and geometries

1. The flow simulation of a wind turbine rotor configuration is based on generalized lifting line theory. A leading assumption in this theory is that for each cross-section of a lifting

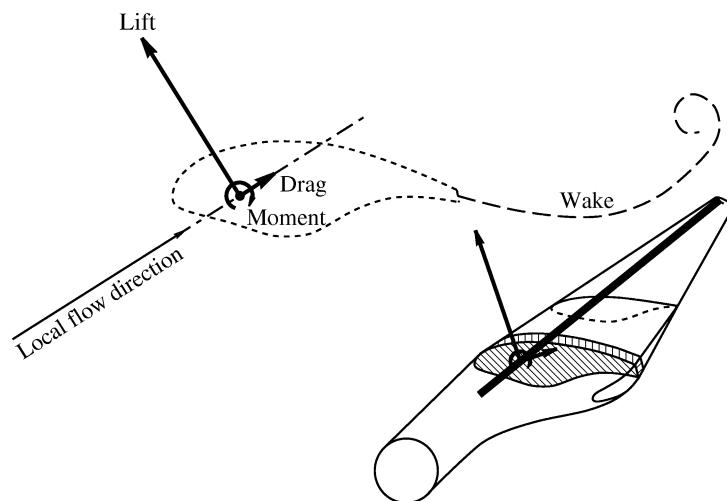


Figure 1: Lifting line representation

surface the generated lift is determined by a local onset flow direction (figure 1) assumed to be aligned with the plane of the cross-section. This restricts the flow simulation to slender and planar or slightly curved blade geometries that do not show strong radial flow interactions. As yet, the flow simulation of multiple (interacting) wind turbines will *not* be supported.

2. The effects of viscosity are taken into account through the user-specified non-linear relations between local flow direction and local lift (figure 2), drag and pitching moment coefficients. It is assumed that the range of flow directions in the user-supplied force and moment coefficient tables covers all flow-directions that occur in the simulation. A runtime error will result when a local flow direction is outside this user-supplied range. No restrictions are placed on the allowable range of locally occurring flow directions; a full 360 degree range is possible. Locally occurring onset flow velocities are assumed to be much smaller than the speed-of-sound so that incompressible flow can be assumed. Effects of non-lifting components like tower, nacelle and base surface are *not* included.
3. Solid body motions, deformations and deformation rates typically vary in time and from blade to blade. As a result of this generality, the exploitation of geometric periodicity

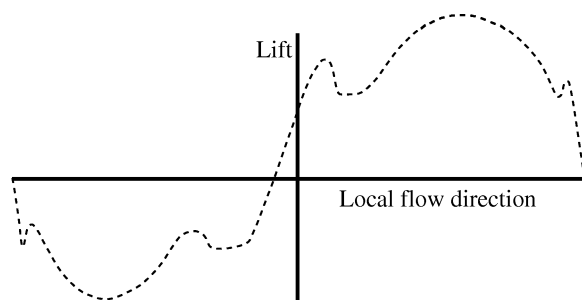


Figure 2: Non-linear relation between local lift and flow direction

is *not* pursued. Displacement velocities and solid body motions of components will be specified in order of the geometric hierarchy *rotor* → *blade* → *segment* (figure 3). Specifying rotor motion will be felt by the blades and their defining segments. However,

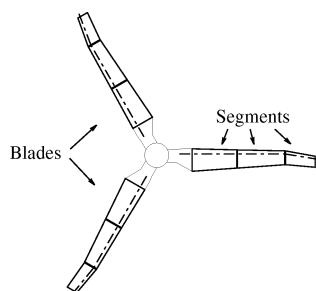


Figure 3: Wind turbine rotor components

specifying the motion of a segment (for example a tip-brake pitch action) will have no impact on the prescribed motion of the complete rotor.

4. The geometry of a rotor consists of an arbitrary combination of blades. Each blade carries a set of lift-generating vortex-line elements of possibly unequal length (figure 4). The geometry of the blades is user-specified. The coordinates of the vortex-line elements

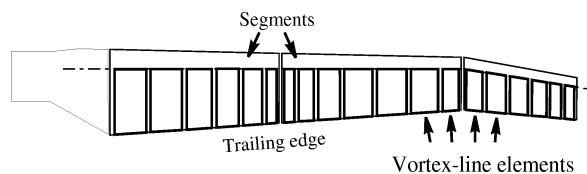


Figure 4: Rotor blade vortex-line elements

are extracted from this specification.

5. Vortex shedding occurs at the trailing edge of the blades (figure 4). Vortices are shed by the vortex-line elements each computational time-step and are subsequently convected in the local flow direction.

6. Onset wind is typically time and space dependent. As a consequence aerodynamic periodicity (in combination with geometric periodicity) will *not* be exploited. As yet, the option to take the influence of stochastic wind fields into account will *not* be included.
7. Stand-alone operation is possible with prescribed time-dependent data for (homogeneous) onset wind and rotor motion and deformation velocities. Multiple simulations can be performed in one run for a specific rotor. Input files for the program are provided by the user. Geometry modelling capabilities are considered to be outside the scope of this project.
8. An optional time-consistent coupling with a structural dynamics module makes sure that, at each (discrete) instance of time, a coherent set of data is shared by the aerodynamics and structural dynamics modules.

2.1.2 Computed output quantities

1. Time-dependent forces and moments acting on the complete wind turbine rotor and its defining blades, segments and vortex-line elements in local and global coordinate systems.
2. Local angle-of-attack occurring at the blade vortex-line elements and the three components of the local velocity vector.
3. Rotor wake geometry and vorticity strength.
4. Numerical convergence and/or accuracy information.

2.2 Non-functional Requirements

In the non-functional requirements the properties of the system are expressed in terms of desired performance, platform characteristics and other operational needs.

1. The anticipated hardware platform is a year 2002 PC with an estimated performance equal to a single processor 2000 Mhz Pentium IV CPU with 512 Mb RAM and 100 Gb harddisk storage capacity. No specific operating system is selected.
2. A 10 minute windturbine flow simulation should run within a day on the anticipated hardware. This requirement will have a very strong impact on the development of an efficient rotor wake evolution algorithm.

To illustrate the importance of algorithm efficiency, assume there are N_b rotor blades each carrying N_v vortex-line elements. Every time-step an extra $N_b N_v$ number of vortices are shed from the trailing edges. At time-step N_t there are a total of $N = N_b N_v N_t$ vortex-line elements shed into the flow domain. With a straightforward implementation the work W to calculate the mutual influence of all N vortices at time-step N_t scales quadratically: $W = C_q N^2 = C_q N_b^2 N_v^2 N_t^2$. The total work done, including all previous timesteps, therefore amounts to

$$\begin{aligned}
 W &= C_q N_b^2 N_v^2 \left[N_t^2 + (N_t - 1)^2 + (N_t - 2)^2 + \dots + 1 \right] \\
 &= C_q N_b^2 N_v^2 \frac{1}{6} \left[2N_t^3 + 3N_t^2 + N_t \right]
 \end{aligned}$$

For a large number of time-steps the total required amount of work can be approximated by

$$W \approx \frac{1}{3} C_q N_b^2 N_v^2 N_t^3$$

This formula shows us that doubling the number of vortex-line elements on the blades will cause the work (and thus computational time) to increase by a factor of 4. Extending a simulation period from 10 minutes to 30 minutes will increase the work by a factor of 27.

As a thought experiment, let us replace the straightforward implementation by an ideal one that scales linearly with the total number of vortices: $W = C_l N = C_l N_b N_v N_t$. Now the total work done, including all previous timesteps, amounts to

$$\begin{aligned} W &= C_l N_b N_v [N_t + (N_t - 1) + (N_t - 2) + \dots + 1] \\ &= C_l N_b N_v \frac{1}{2} (N_t + 1) N_t \end{aligned}$$

For a large number of time-steps the total required amount of work can now be approximated by

$$W \approx \frac{1}{2} C_l N_b N_v N_t^2$$

Now, increasing the number of vortex-line elements by a factor of 2 increases the work by the same factor. A three times longer simulation period will now increase the required amount of work by a factor of 9.

(NB. For a 10 minute simulation the total number of time-steps is in the order of $N_t \approx 10^5$)

3. Input files will be unambiguous and self-describing. Missing data will cause the program to fail; no hard-coded defaults are assumed. Specifying an input value twice will generate a runtime error.
4. During a typical simulation a vast amount of data is generated. The amount of data in the output files and their specific contents can be controlled by the user through the specification of output frequency and -granularity.
5. Software maintainability and extensibility are of highest priority. As a consequence even computational performance will be sacrificed in favor of this requirement. In the implementation phase coding guidelines will be adopted. Software documentation will be written in a clear and concise style with the emphasis on global module composition, code intent and algorithm functioning. This will facilitate the possible transfer of code from one developer to another.
6. Program robustness, i.e. the ability to produce results for given valid input data, is of high importance. This will have its effect on the selection of the numerical algorithms. Robust but slower converging algorithms will be preferred in cases where the solution seems to diverge. A well known example in this respect is the difference between the quadratically converging Newton-Raphson root finding algorithm and the more robust but slower (linear) converging bisection algorithm.
7. The software development language will be Fortran 90. Some observations on Fortran are:

- (a) For scientific computations Fortran compilers produce the fastest executables.

- (b) A large database of implemented scientific algorithms is freely available in Fortran.
 - (c) In the scientific community most programming experience is in Fortran.
 - (d) Most wind turbine aerodynamics and structural dynamics legacy codes are written in Fortran. Coupling new code with existing programs is less error-prone when using one programming language only.
 - (e) Fortran 90 includes features like dynamic memory allocation and pointers. This makes the use of tree-based data structures possible; a prerequisite for the efficient implementation of a vortex-wake dynamics algorithm.
 - (f) Fortran 90 possesses some of the characteristics of modern Object Oriented programming languages like Java and C++ such as data hiding, data composition and function encapsulation. These features enhance code maintainability and extensibility and stimulate code reuse. However, the lack of inheritance capabilities in Fortran 90 precludes subtyping and most instances of polymorphism; features that increase code maintainability.
 - (g) Programs written in Fortran are platform-dependent. This will increase the cost of code maintenance in case of multiple runtime platforms.
8. If possible, use will be made of automated unit test cases. This will be extremely useful in regression tests which assure that inclusion of new functionality does not introduce software bugs in existing code.
 9. The program will have logging facilities for debugging purposes.
 10. User documentation will be concise and directed towards practical applications.

2.3 Future Requirements

By an early specification of possible future requirements the implementation of these features in the software development phase can be anticipated. Obviously the future requirements can also be seen as a specification of what the system most likely will *not* be able to do in its first release. Possible future requirements are:

1. Taking into account the influence of stochastic wind fields.
2. Implementation of a (simple) model for the controller unit using a prescribed relation between (aerodynamically generated) rotor torque and the rotational speed.
3. Coupling the aerodynamics model with an existing structural dynamics windturbine simulation program.
4. Input preparation and/or job run through a utility program with a graphical user interface (GUI).
5. Computation of velocity vectors at user specified points in space.
6. Modelling the effect of thick non-lifting bodies like tower and nacelle. However, because non-lifting bodies are often associated with large regions of separated flow at their downstream side, it is questionable if this feature can be adequately simulated without a model for the trailing wakes.

7. Introduction of an (inviscid) model for thick lifting elements. With this option the detailed pressure distribution on the rotor blades can be studied. Moreover, the same inviscid model can be used to simulate the flow about "non-lifting" bodies exhibiting large regions of separated flow. An essential further development is the modelling of viscous

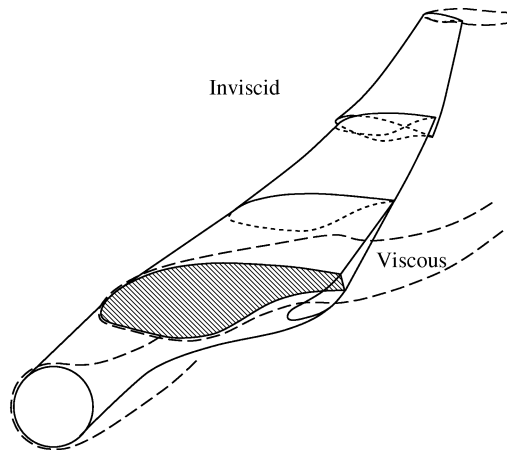


Figure 5: Viscous-inviscid domain decomposition

effects on the rotor blades by a 3D boundary-layer method in strong interaction with the inviscid outer flow (figure 5).

8. Computation of base surface effects on the aerodynamic characteristics of the rotor. This ranges from the influence of level solid ground through the effects of complex shaped terrain and time-dependent moving seawaves.
9. Computation of internal flows. This option enables the flow simulation of a rotor in a closed windtunnel. A necessary condition is the ability to model thick lifting elements.
10. Taking into account the actual deformation of the rotor blades.
11. Parallelization of the computational algorithms. In the ideal situation computer time can decrease linearly with the number of processors.
12. Allowing the simulation of multiple (interacting) wind turbines.

3 CONCLUSIONS

To avoid many of the uncertainties that accompany current blade-element-momentum theory based windturbine aerodynamics simulation codes a new aerodynamics simulation module will be developed. This module will be based on non-linear vortex-line theory in which rotor wake geometry and vorticity distribution will develop in time. As a first step in the development of this new module the system requirements are formulated.

The capabilities of the system as seen from a user's point of view are described in the functional requirements section. Topics like desired performance, software maintainability and development environment are dealt with in the description of the non-functional requirements. Of all non-functional requirements listed, the one stating run-time performance will have a significant impact on the selection of an efficient rotor-wake evolution algorithm. Additionally a list of possible future code enhancements and extensions is given.

The underlying report is intended to be a "life" document that serves as a guide during the software development process and as a reference during the final assessment.