Westerduinweg 3
1755 LE Petten
P.O. Box 15
1755 ZG Petten
The Netherlands

www.tno.nl

T +31 88 866 50 65

TNO report

TNO 2019 R11391

Improvement of BEM and vortex-wake models

TKI WoZ VortexLoads WP4

Date	November 2019
Author(s)	K. Boorsma, M. Aman‡, C. Lindenburg∓, M. Kloosterman∓ ‡ DNV-GL ∓ LM Windpower
Copy no	
No. of copies	
Number of pages	54 (incl. appendices)
Number of appendices	0
Sponsor	TKI WoZ
Project name	TKI WoZ VortexLoads
Project number	060.33833

Acknowledgement

This project has been executed within the framework of TKI Wind op Zee.

mede mogelijk gemaakt door
TKI WIND OP ZEE
Topsector Energie

Summary

Within the framework of the TKI WoZ VortexLoads, improvements to both BEM and free vortex wake codes have successfully been implemented with the end goal to make wind turbine design load calculations more accurate. Here we can distinguish improvements to BEM type codes, which focus on engineering models and their implementation to make these more accurate. A model to account for the influence of the blade shed vorticity has been successfully implemented based on the time history of the vortices in the wake of each blade element. Application of this model was shown to bring BEM simulation results closer to the free vortex wake simulations at a relatively small extra computational expense. Another approach that has been researched is the definition of a representative streamtube wind speed in non-uniform inflow conditions for improved induction tracking. Although the first result of this approach both in sheared and turbulent inflow are very promising (in combination with shed vorticity modeling, the fatigue load difference with vortex wake codes almost disappears), more research is necessary on this topic to come to a more conclusive implementation, assuring not to cover up unintentionally other effects such as shed and trailed vorticity variation.

The improvements to free wake vortex codes focus on ways to reduce the computational effort associated with running them. Distinguishing between wake update frequency and aerodynamic sample time was shown to have a great potential in reducing CPU time. Limiting the number of free wake points and total wake length are obvious candidates to save computational time. Progressively skipping shed vortices in the far wake was shown to have a great reduction potential, with a minimal impact on the accuracy of (un)steady loading characteristics. For optimum computation speed it is vital to make use of the vector-parallel architecture in modern-day CPU's to carry out multiple floating-point operations per clock cycle. This was achieved by laying out the data contiguously in memory and by writing loops that modern compilers can vectorise, resulting in a computational effort of about 20 times the simulation time for vortex wake codes.

Contents

1 Introduction

The TKI WoZ VortexLoads project [1] considers the application of vortex wake models to design load calculations and its potential added value against traditional BEM methods. Within Work Package 4 of the project, several improvements to both BEM and vortex wake codes are considered. Chapter 2 describes the improvements to BEM type codes, focusing on engineering models and their implementation. Chapter 3 describes the improvements to vortex wake codes, focusing on ways to reduce the computational effort associated with running them.

2 Improvements to BEM type codes

2.1 Improved induction tracking in non-uniform inflow conditions (ECN.TNO)

From the work package 2 report of the TKI WoZ VortexLoads project [2], it appeared that a structural difference exists between the predicted load fluctuation amplitudes in vertical shear from vortex wake (AWSM) and BEM type codes. A parametric investigation of this observation revealed this difference to correlate with the axial induction factor, see also Figure 2.1.

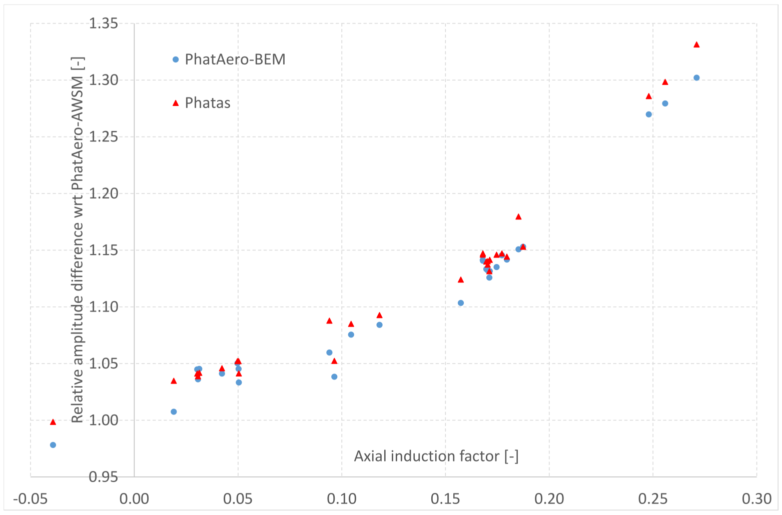


Figure 2.1: Relative difference of Mflap amplitudes of two BEM codes in shear w.r.t. PhatAero-AWSM as a function of axial induction factor [2]

Application of an engineering extension to BEM, accounting for the effect of shed vorticity variation (see also section 2.2), did not yield an explanation for this difference. Most likely the gradual inflow variations in vertical shear are not abrupt enough for the shed vorticity variation to play an important role. It was shown that almost over the full blade span the induced velocity features larger fluctuations for vortex wake models and 'track' the inflow variations better, see also Figure 2.2.

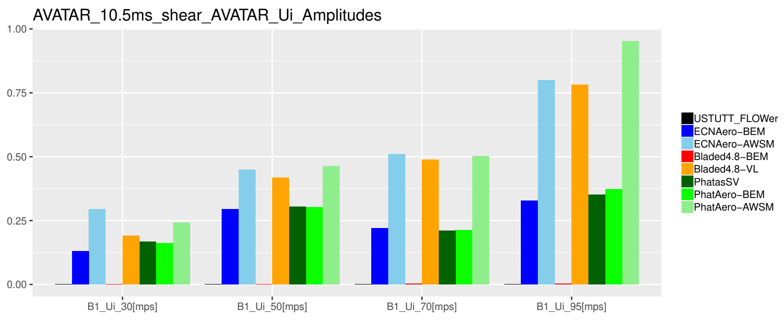


Figure 2.2: Predicted amplitudes of axial induced velocity at 30%R, 50%R, 70%R and 95%R for the 10MW AVATAR rotor in vertical shear [2]. Results are shown from BEM (ECNAero-BEM, Bladed-4.8-BEM, PhatasSV, PhatAero-BEM) and vortex wake type codes (ECNAero-AWSM, Bladed4.8-VL, PhatAero-AWSM).

As a consequence the angle of attack from the sectional velocity triangle features less fluctuations for the vortex wake code (see also the illustration in Figure 2.3) resulting in the observed lower load variation amplitude.

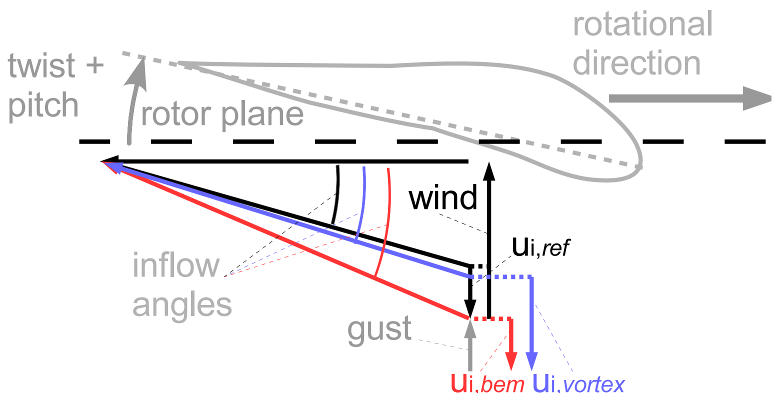


Figure 2.3: Illustration of inflow angle change due to a wind gust (grey) for a BEM (red) and vortex code (blue). The reference without gust is colored black.

In an attempt to further study the cause for the difference, results from BEM and vortex wake sheared inflow simulations on the AVATAR rotor have been post-processed to verify compliance with the axial momentum equations. The one dimensional axial momentum equations constitute a relation between the thrust coefficient Ct and the axial induction factor a at the rotor disc in the form of

$$Ct = 4a(1 - a) \tag{2.1}$$

and

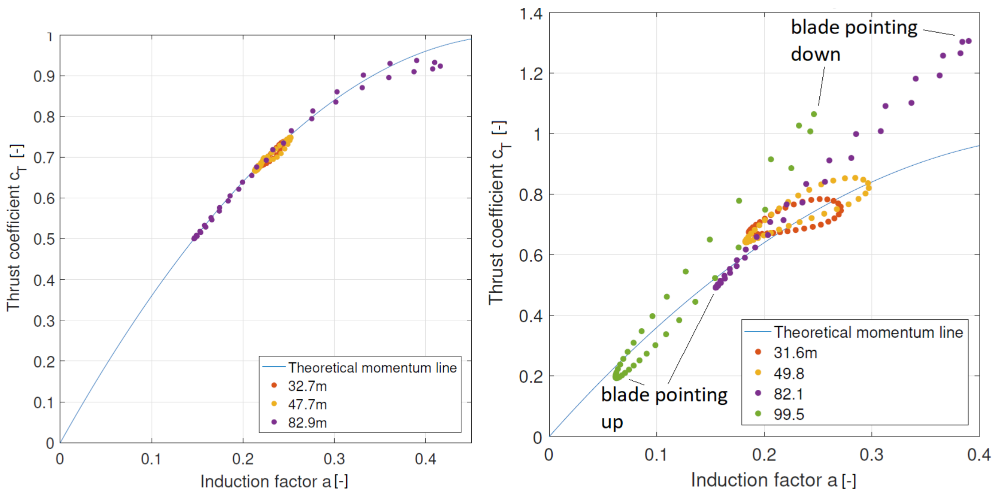
$$a = Ui/U \quad , \tag{2.2}$$

with

Ct	[-]	thrust coefficient
a	[-]	axial induction factor at the rotor disk
Ui	[m/s]	axial induced velocity
U	[m/s]	wind speed.

To be able to focus on the effect of shear a relatively large shear exponent of 0.75 at 8 m/s hub height wind speed was employed for the investigation, which is reported in more detail elsewhere [3]. The results in Figure 2.4a indicate that the BEM simulation complies with the underlying momentum equation as it is supposed to. However the vortex wake results in Figure 2.4b clearly deviate from this line depending on the azimuthal position, where especially for the outboard stations high thrust coefficients are obtained in combination with a relatively low axial induction factor (lower as would be the case for the theoretical momentum line) for a downward pointing blade featuring the lowest local inflow velocity. It can be shown that for rotors operating at higher induction, BEM theory can even predict an increase rather than decrease in axial induced velocity when the blade is pointing downward (6 o' clock position), due to the fact that for the relative high local thrust coefficient the corresponding axial induction factor increase is larger than the local wind inflow decrease (a=Ui/U). This is unlikely to be the case in reality (non-physical), which is backed up by the fact that corresponding vortex wake calculations predict the opposite trend, namely the axial induced velocities to decrease for a lower local inflow speed at the downward pointing blade position.

Acknowledging the fact that the vortex wake model does not obey the momentum equations as implemented in BEM theory, one may reflect on which shortcoming of



(a) BEM (element approach) (b) AWSM vortex wake

Figure 2.4: Comparison of post-processed AVATAR rotor simulation results in heavy shear (U=8 m/s, $\alpha = 0.75$) for several radial stations against the theoretical momentum line

BEM theory is responsible for this difference. Several assumptions are made in the derivation of BEM theory and an inventory was made which specific violations of this theory occur in sheared inflow.

- Radial independence
  The influence of neighbouring elements and of the other blades are not taken into account and each annulus is treated separately. For a spanwise uniform circulation distribution it is acknowledged that intermediate trailed vorticity effects are absent and as long as the loading differences between the blades are not significant this effect can be neglected as well. However in sheared inflow, even for a blade that is designed with uniform spanwise circulation distribution, a varying spanwise circulation distribution will result in trailed vorticity which violates the radial independence assumption.

- Axi-symmetric or uniform inflow conditions
  It is well known that BEM theory assumes steady inflow conditions, which relates to the variation of wind velocity along the longitudinal direction of the streamtube. In addition to this the derivation of the underlying one dimensional axial momentum equation assumes that the inflow conditions are axi-symmetric (or uniform) with respect to the streamtube considered. In sheared inflow conditions (or any other non-uniform inflow condition such as turbulent or waked inflow) it may be clear that this is not the case. It was shown previously that the Betz limit can be exceeded in non-uniform inflow conditions [4]. The implication of the violation of axi-symmetric or uniform inflow conditions may differ between a BEM approach that solves the momentum equations 'annulus averaged' (i.e. using one equation resulting in the same induced velocity for all the blades) or the more modern local approach that solves the momentum equation separately for each blade. In the latter case one may ask the question what the azimuthal and radial extent of the streamtube is, that balances the force exerted by a blade.

Further consideration of the second violation aspect triggered the idea to distinguish between the wind velocity used in BEM for the purpose of evaluation of the sectional force (blade element part) and the determination of induced velocity (momentum equations). See also the below displayed axial momentum equation 2.3, assuming that the tangential induction can be ignored.

$$2\mathsf{a}(1-\mathsf{a})\rho \mathsf{U}^2 2\pi r dr = \sum_B c 0.5 \rho W^2 c_l(\alpha)\cos(\phi) dr \quad , \tag{2.3}$$

where

$$\phi = atan2(W(1-\mathsf{a}), \Omega r), \quad \alpha = \phi - \epsilon \quad \text{and} \quad W = \sqrt{\mathsf{U}^2(1-\mathsf{a})^2 + (\Omega r)^2}$$

with

r	[m]	radius of element considered
c	[m]	local blade chord at radius r
ρ	[kg/m^3]	air density
W	[m/s]	effective velocity at element
c_l	[-]	lif coefficient
α	[°]	angle of attack
ϕ	[°]	inflow angle wrt rotor plane
Ω	[rad/s]	rotor speed
ϵ	[°]	twist plus pitch angle and torsion deformation

In this equation the blade element part is on the right hand side, in which the wind velocity U is included through the effective velocity term W, the inflow angle ϕ and angle of attack α. The momentum part is on the left hand side of equation 2.3. It is noted that in the so called 'annular average' BEM, the element forces are summed over the blades ($\sum_B$) and the corresponding annulus has a 360° extent, resulting in a single axial induction factor for all blades. The current 'element 'BEM implementation which is used here solves equation 2.3 for each blade separately (adjusting the annular volume correspondingly), resulting in different induced velocities for each blade element in an annulus.

Revisiting the idea to distinguish between the wind velocity used in BEM for the purpose of evaluation of the sectional force (blade element part) and the determination of induced velocity (momentum equations), it is clear that the local wind velocity acting at an element quarter or three-quarter chord point should be used for the first part. For the second part it could be argued to use a wind speed that is representative for the streamtube considered instead of a local point at the element center (as it is currently implemented). The question is how to define this streamtube and how to define a representative wind speed for it. Where a CFD or vortex wake simulation considers all spatial wind speed variations by means of a mesh, the momentum theory in BEM allows for only one. Effectively this is an inherent shortcoming of BEM and it could be argued we have arrived at a limitation that cannot be overcome. In a first attempt a streamtube is defined that considers an annular sector with azimuthal extent of 360° divided by the number of blades, symmetrically distributed around the element of consideration. A simple five point average is taken of the wind speed, the five points equally distributed in azimuthal direction at a spacing of 20° for the current example with three blades. Application of this idea to the above highlighted vortex wake simulation yields Figure 2.5, which shows an improvement in terms of agreement with the theoretical momentum line (i.e. the result lie closer to this line).

Implementing the outlined approach in a BEM code has allowed for some further testing in sheared and turbulent inflow as illustrated in Figure 2.6. In sheared inflow it is shown that induced velocity amplitudes and consequently the normal forces are more in line with the vortex wake modeling. The time trace in turbulent inflow (Figure 2.6c) clearly illustrates the improved tracking of induced velocity of the sector wind approach, again very close to the vortex wake result except for the higher frequencies.

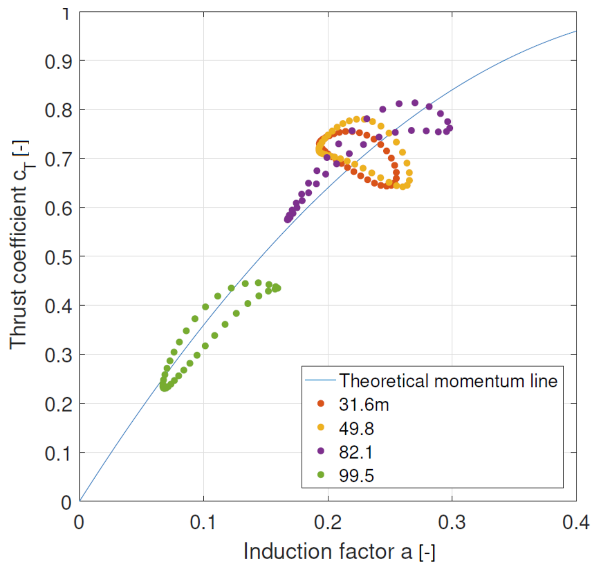
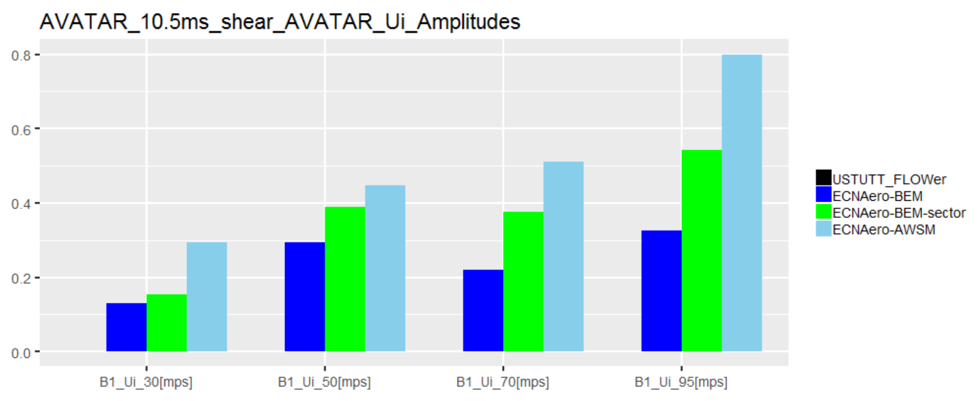


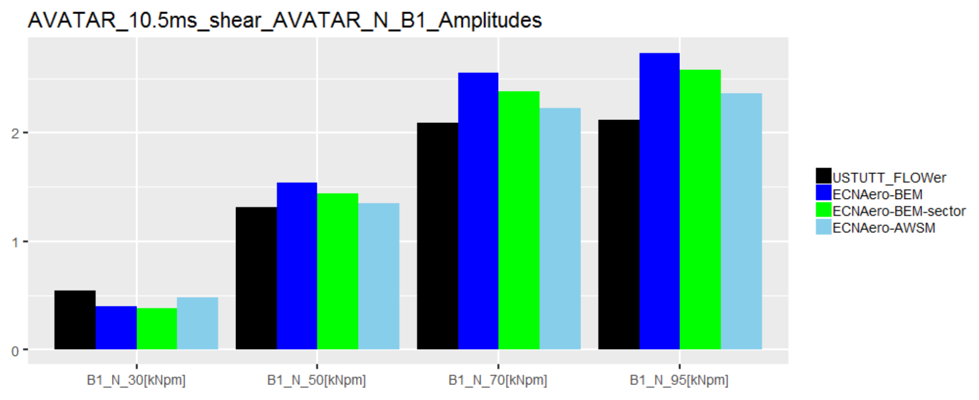
Figure 2.5: Comparison of post-processed AVATAR rotor simulation results in heavy shear (AWSM vortex wake, U=8 m/s, $\alpha = 0.75$) for several radial stations against the theoretical momentum line. The reference wind speed for non-dimensionalizing Ct and a is now taken as the sector averaged wind speed

The resulting integrated staircase plots (which are the result of the rainflow counting procedure for obtaining fatigue equivalent loads) show that application of a shed vorticity model (by means of the Beddoes Leishman model ECNAero-BEM-BL instead of the default Snel model for unsteady airfoil aerodynamics) in combination with the sector approach (ECNAero-BEM-BL-sector) results in unsteady loading characteristics matching AWSM very well for this case.

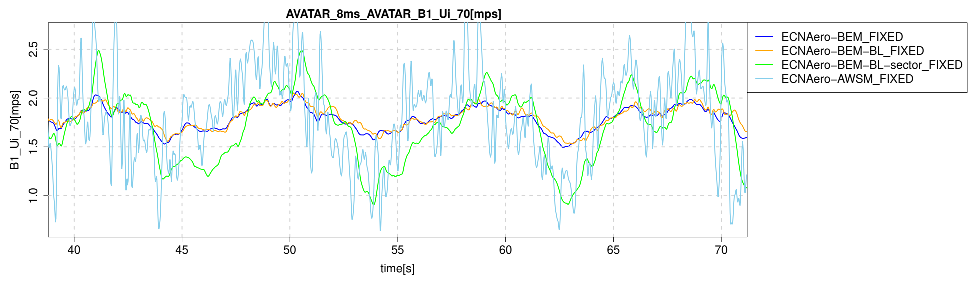
It is recommended to have a more detailed look into the definition of a representative streamtube (e.g. varying azimuthal extent and position leading/lagging, averaging procedure etc.) and run a variety of test cases (e.g. the cases were defined and ran by CFD in this project [2] and the parametric shear investigation from Figure 2.1, containing a variation in the number of blades), also to assure this procedure does not cover up unintentionally other effects such as shed and trailed vorticity variation. Reference is made to a recent publication [5] that also addresses the issue of induction tracking by means of a new approach, now solving the BEM equations on a polar grid. This approach seems to resolve the damping of local induced velocities by the dynamic inflow model by decoupling the individual blade momentum equations on a grid. In the current formulation this unwanted damping is counteracted by specifying a large number of subiterations per time step to ensure local convergence of the momentum equations.



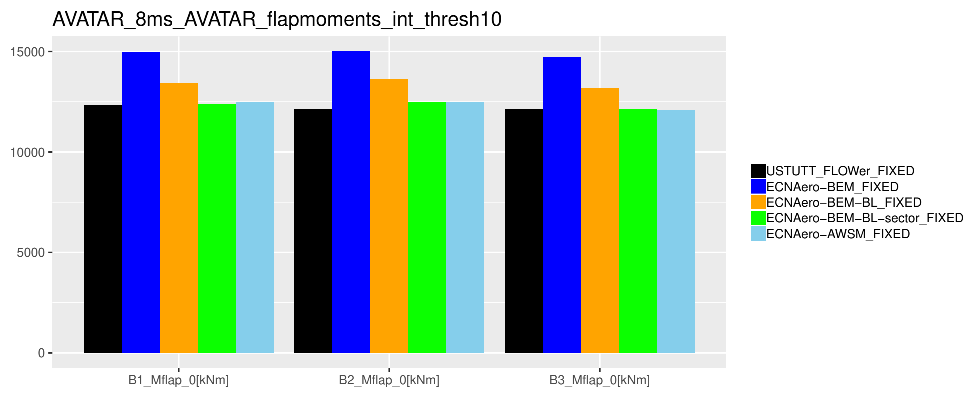
(a) Axial induced velocity amplitudes at 30%R, 50%R, 70%R and 95%R in shear (10.5 m/s)



(b) Chordnormal sectional force amplitudes at 30%R, 50%R, 70%R and 95%R in shear (10.5 m/s)



(c) Timetrace of axial induced velocity variation at 70%R in turbulent inflow (8 m/s)



(d) Integrated staircase plot values of flapwise blade root moment in turbulent inflow (8 m/s) for all three blades. The staircase values are an intermediate result of the rainflow counting procedure. The integration was performed from a threshold of 10 counts

Figure 2.6: Comparison of sector wind BEM implementation (indicated by adding 'sector' to the legend entry) to conventional BEM with Snel (default) and Beddoes Leishman (BL) modeling, vortex wake results (AWSM) and CFD (USTUTT_FLOWer) in non-uniform inflow conditions for selected AVATAR 10MW rotor simulations

2.2 Blade shed vorticity model (LM Windpower)

The program Phatas has been developed for the time-domain simulation of the structural- and aerodynamic response of a wind turbine, and is used to calculate the loads in the turbine components. This response includes the coupled unsteady aerodynamic and structural dynamic response of the rotor.

The rotor aerodynamics of the program Phatas is based on the BEM approach. In the BEM approach the influence of the rotor wake is described for each individual blade element without direct interaction of the aerodynamics of the neighbouring blade elements. The aerodynamic blade loads are calculated using tables with dimensionless coefficients for aerodynamic lift, drag, and moment coefficient as function of angle-of-attack. The algorithm for the blade aerodynamics have correction models for the effects of rotation on the lift coefficient and for dynamic stall of the flow around the airfoil.

2.2.1 Influence of blade wake

The blades of a rotor operating in unsteady conditions will have a varying lift, and likewise a varying circulation. A varying circulation involves a shed vorticity in the blade wake such that at each time the total circulation of a blade and its wake is conserved.

Some dynamic stall models such as from 'Beddoes Leishman' have a dynamic term that describes the contribution of the blade wake, which contains shed vorticity that leaves the trailing edge. In parts of the Beddoes Leishman model the contribution from blade shed vorticity is expressed as function of the derivative of the angle of attack, and is also known as the Theodorsen model.

The vortex line representations of the rotor wake such as in the program AWSM implicitly includes the effects of the blade shed vorticity on the blades itself. For this reason it would not make sense to use e.g. the complete Beddoes Leishman dynamic stall model in combination with a vortex wake model for the rotor wake. The fact that the shed wake effects are already represented in the vortex wake descriptions of the rotor aerodynamics means that the additional dynamics from shed vorticity is a property of the wake and not of the airfoil.

2.2.2 Global approach for modelling the Blade Shed Vorticity

During the current project the program Phatas is extended with a model that describes the influence of the blade shed vorticty, based on the time history of the lift coefficients and the relative velocities of the airfoils. From these time histories the vortex-structure of the shed vorticities is calculated. This vortex-structure includes the distances between the vortices, which depends on the relative flow velocities. This means that also the shed vorticity effects related with edgewise vibrations is included. The contribution of the shed vorticity on the flow on the (75% chord location of the) airfoils is calculated with the Biot-Savart expressions, by which it differs from the model described by Theodorsen.

2.2.2.1 Phatas Implementation of the model for Blade Shed Vorticity

In the program Phatas the model for blade shed vorticity uses the input variable '**blade_shed_vortices**'. The default setting for **blade_shed_vortices** is 0 for which the program Phatas directly uses the lift coefficient from the existing routines. In addition to the default value 0, the allowable values for **blade_shed_vortices** is limited between 4 and 40. In the Phatas code also a maximum is set to the azimuth-sector that is covered by the number of shed vortices. This maximum is defined as azimuth-sector-angle and amounts 72deg/Nr_blades. This azimuth-sector limitation is applied with the product of **time_increment** and **rated_rotorspeed**. This reasons for this azimuth-sector limitation are:

- The influence of the blade shed vorticity assumes a wake structure with a flat geometry in which geometry has a heliciodal-like shape.
- If the influence of a large part of the blade wake is taken into account then implicitly this represents similar effects as from the so-called 'dynamic inflow' model, which should be avoided.

2.2.3 Phatas algorithm of the blade shed vorticity

The influence of the shed vorticity is calculated as if the geometry of the blade wake is aligned with the blade chord. This approach is highly valid for moderate angles of attack.

Angle of attack range

For angles of attack between -30deg and +30deg the flow is attached or in moderate stall, for which one may assume that the flow leaves the airfoil close to the trailing edge. For this reason the influence of the blade shed vorticity is fully applied.

For angles of attack smaller than -30deg or larger than +30deg the airfoils are probably in stall, but still they will have some lift while the shed vorticity leaves airfoil approximately near the trailing edge. So for angles of attack outside of the range between -30deg and +30deg the contribution of the blade shed vorticity is reduced linear to zero at the -50deg and +50deg angle of attack of respectively.

For angles of attack outside of the range between -30deg and +30deg the contribution of the Blade Shed Vorticity is reduced linear to zero at the -50deg and +50deg angle of attack of respectively.

For angles of attack smaller than -50deg and larger than +50deg the airfoils may be in deep stall and the chord direction is far not aligned with the direction of the blade wake structure (flow direction). So for these excessive angles of attack the influence of the blade shed vorticity is not modelled.

Solution structure for the blade shed vorticity contribution

In the program Phatas the BEM equations for each blade element are solved iteratively until the induced velocity from the airfoil loads matches with the angle of attack calculated from the induced velocity. This iteration contains a CALL for a routine that returns the aerodynamic lift, drag and moment coefficients for a given of angle of attack.

Geometry of the modelled wake structure

When using the model for the Blade Shed Vorticity the CALL for the aerodynamic coefficients uses an intermediate routine that calculates the angle-of-attack including the contribution of the shed vortices in the blade wake. The contribution of the blade wake is calculated with the Biot-Savart expressions for the vortex structure that leaves the airfoil.

For each blade element the influence of the shed vortex structure is expressed for a semi infinite set of vortices that run parallel with the airfoil and lie in a plane that is aligned with the chord of the blade element. This assumption of semi-infinite vortices implies that the influence of the blade shed vorticty of neighbouring elements is not included. These parallel vortices were modelled up to the tip of the blades, where the vortex continues as trailing vortex.

The influence of the root end was not modelled because the root vortex is not strongly aligned with the blade chord and because the location of the root vorticity is not very distinct. Here it should also be noted that the loads at the inboard part of the blade are not dominant while their contribution to the bending moments in the blade is even smaller.

For simplicity the vorticity is discretised in concentrated vortices. The strength of the vortices is calculated from the time-derivative of the circulation. The circulation is calculated from the time history of lift coefficient and relative velocity of the airfoil.
$\Gamma = (\rho/2) \cdot c \cdot c_{\text{lift}} \cdot V_{\text{eff}}$,
with Γ the circulation, ρ the air density, c the blade chord, c_{lift} the aerodynamic lift coefficient, and V_{eff} the relative velocity over the airfoil.

The distance between the vortices is simply the product of time increment and relative velocity. To avoid numerical problems from the Biot Savart expression, the Phatas code for Blade Shed Vorticity has a lower bound that is set to 10% of the blade chord.

Truncation of the far wake

The influence of the blade shed vorticity is calculated for a finite number of shed vortices. In its basic form this means that for each 'new' time value one may simply omit the 'last' vortex line. In the Phatas algorithm for the influence of Blade Shed Vorticity a fraction of the strength of the 'last' vortex line is added to the second last vortex line. This fraction is such that the contribution to the 75% angle of attack is conserved.

With this approach only a moderate number of '**blade_shed_vortices**' already gives a realistic modelling of the contribution of the blade shed vorticity.

Effects for finite blade chord

The blade shed vorticity is modelled from the point where it leaves the trailing edge. The influence of the air that is surrounding the airfoil is modelled with the 'added mass' and by using the angle of attack evaluation at the 75% chord location. The 75% chord location accounts for the inflow variations by the airfoil slicing through the turbulent structures and the effects of pitch+twist variations of the airfoil.

In addition to the use of the 75% angle of attack for the table look-up of the lift, drag, and moment coefficients, a correction is applied to the moment coefficient c_{m} for the 'flow curvature'. The flow curvature is calculated by evaluations of the angle of attack at the 25% chord and at the 75% chord.
$c_{\text{m}\,curved} = c_{\text{m}\,straight} + (\pi/8) \cdot (\alpha_{25\%} - \alpha_{75\%})$.

Iterative solution of the airfoil flow

The algorithm for the contribution of the Blade Shed vorticity is modelled in a routine that is called with the angle of attack as parameter. This angle of attack gets a contributions of the Blade Shed Vorticity of the airfoil. The strength of the 'first' vortex downstream if the airfoil depends on the time derivative of the circulations, so on the lift coefficient. Because of this mutual dependency of the angle-of-attack and circulation (lift coefficient) the solution of the flow around the airfoil is solved iteratively. The criterion for this iteration is set to an angle of attack difference of 0.002deg.

3 Improvements to VL type codes

3.1 Wake update frequency (DNV-GL)

During time domain simulations, the states of the multi-body wind turbine model are time-stepped by an integrator. The states of the system can represent the degrees of freedom of the wind turbine structure. Examples of structural DOF are the blade 1st flapwise and edgewise modes. The model can also include aerodynamic states for unsteady aerodynamic effects like dynamic stall or dynamic wake.

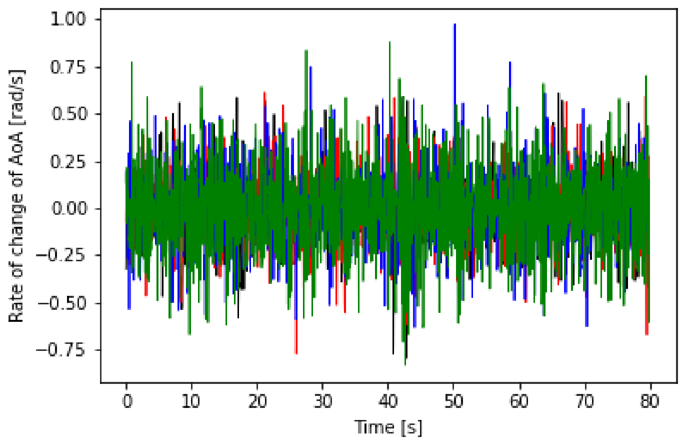
The minimum time step required to integrate the Bladed multibody model is determined by the state that has the largest rate of change. Whereas the first tower mode could be integrated at a relatively large time step ($\sim$ 1s), a smaller time step is typically required to capture higher frequency structural modes, such as the blade torsional mode ($\sim$ 5ms). Bladed provides the option of using a variable step integrator which can adapt the integrator time step to capture all the system dynamics with sufficient accuracy. The step size is continually adapted according to a normalized error relation, which follows the worst offender method. Since Bladed 4.8 there is also the option to select fixed step integrators like Newmark-Beta or Generalized-Alpha. These integrators are useful for integrating high-frequency dynamics of the structural states without requiring unfeasible small time steps.

The Vortex line wake time stepping is not coupled to the Bladed system integrator. Each free vortex node could theoretically be a state in the multibody system. However, during a Vortex line simulation, new vortex wake nodes are generated, and others are truncated from the wake as they reach the 'maximum number of wake steps' limit. The Bladed system integrator is not adapted to continuously add and remove states. Therefore, the vortex wake is integrated independently with a fixed time step.

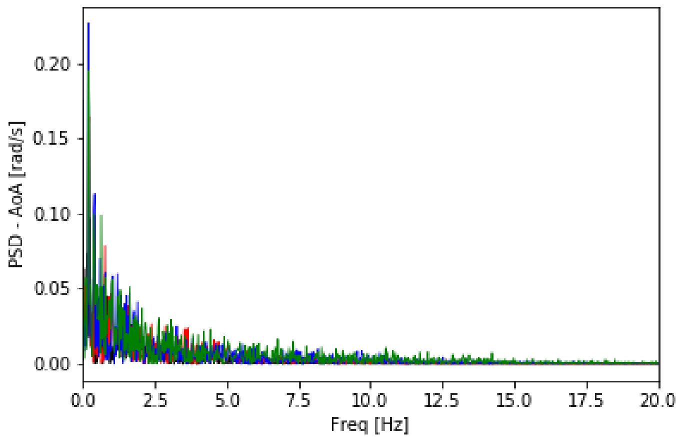
3.1.1 Motivation

Typically, the Bladed integrator uses a 5ms time step for simulating full-scale multi-MW wind turbines models. Assuming a rated rotor speed of 12rpm, this means that 1000 steps are needed to complete a single rotor revolution. Based on the grid studies conducted as part of the Bladed Vortex line validation [6], this is roughly an order of magnitude more steps than are necessary to accurately capture most aerodynamic events, but may vary depending on the environmental conditions. To demonstrate this, we take a typical large scale wind turbine in turbulent wind conditions and consider the time scales for the variation of the aerodynamic states. Figure 1 shows the time history of the rate of change in the angle of attack for a reference case simulated as part of the *VortexLoads* project, see [6] for details. This model is representative for large scale rotors in regular power production operation. It is a fixed speed rotor (9.0218rpm) and is integrated with a vortex wake time step (0.02425s) which corresponds to 1 step per degree of rotor rotation. A smaller than necessary vortex wake time step is applied here to ensure that higher frequency aerodynamic effects are captured.

Figure 3.1a shows the rate of change in the angle of attack is in the order of 1 rad/s. Figure 1b displays the power spectral density plot of the angle of attack signal at four spanwise locations. It is noted that the peak spectral energy is captured within 5Hz. A wake time-step of 0.1s should be sufficient to capture the dynamics induced by the rates of change in the angle of attack due to the slicing through the turbulence. As mentioned before, this value is specific to this model under these specific conditions. For each new test case, it is recommended that the user carry out a convergence test to verify the choice of vortex wake time step and other vortex line parameters.



(a) Rate of change of AoA at various spanwise locations for blade 1.



(b) Power spectral density of the AoA.

Figure 3.1: AVATAR 10MW locked-speed drivetrain wind turbine in turbulent wind [8ms mean with TI 10%]. 30% (black), 50% (red), 75% (blue), 90% (green) radial position.

The other aerodynamic states, which are defined as part of the dynamic stall model, need not be considered as they are integrated alongside the structure states, and not with the vortex line integrator.

3.1.2 *Tjaereborg Pitch Step*

The well-known Tjæreborg pitch step case is simulated to investigate the effects of a coarse vortex wake time step. It has been demonstrated that vortex line aerodynamics captures the settling time of the pitch step event more faithfully than BEM [6]. Here, we compare the change in rotor torque to the measurement data. It was found that the predicted steady state torque values varied for different vortex wake time steps. Therefore, all results are normalised relative to their initial steady-state values. There is a large-scale oscillation visible in the measured data signal, this is due to drivetrain oscillations which were not included in the simulated models.

To post-process the results, we have computed the peak value, peak time, settling time and steady-state error. The peak value and time are the maximum value in the torque signal and the time at which it occurs. The settling time is defined as the time between the peak value and the point at which the hub torque return to within 1% of its steady-state value. For the coarser time step simulation, this value is not reached before the simulation end time. The steady state error is defined as the relative error between the initial steady-state value, before the pitch step event, and the value at the end of the simulation.

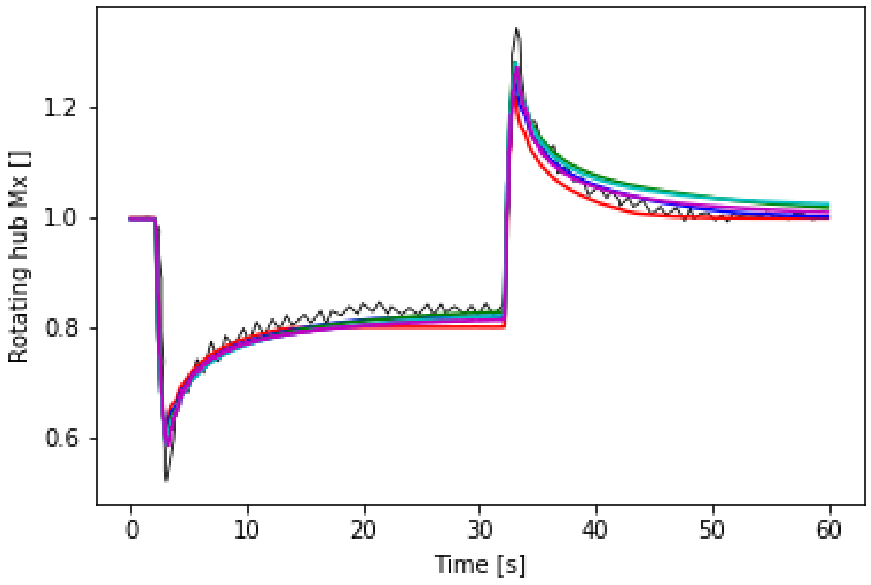


Figure 3.2: Normalised rotating hub M_x during step change in pitch. Tjaereborg pitch step test case with varying vortex wake time step. Measured data (black), 0.01s (red), 0.02s (blue), 0.04s (green), 0.08s (cyan), 0.16s (magenta).

Table 3.1: Relative error (%) in peak value, settling time and steady state value compared to measured data

	Relative error (%)			
Wake time step [s]	Peak value	Peak time [s]	Settling time [s]	Steady state error
0.01	9.09	1.78	0.13	0.01
0.02	6.34	1.78	0.63	0.25
0.04	4.69	1.86	1.31	1.86
0.08	4.78	1.93	-	2.54
0.16	5.32	2.24	-	1.03

It is surprising that the peak values are captured well by the coarse vortex wake time steps. However, the settling time and steady-state error are improved when the vortex wake time step is reduced, as can be seen in figure 3.2. This is expected, coarser time steps results in fewer trailing vortex elements in the wake, each of which have greater vorticity. This will result in an overshoot in the peak value, relative to small wake time step simulations, and a larger steady-state error after the event. Table 3.1 displays the relative error in the vortex line simulations relative to the measurement data.

It is common to normalise the vortex wake time step relative to the time taken for a single rotor revolution. However, this is only reasonable for fixed-speed rotors else this factor will vary in time. This normalisation will be applied in the following section. Based on previous studies [6], it has been shown that 1 vortex wake time step per 5° of rotor rotation produces reasonable results for large rotors at moderate wind speeds in turbulent conditions.

3.1.3 *AVATAR 10MW Model*

To understand the consequences of increasing the wake time step, it is necessary to simulate the effects for regular power production simulations. The 10MW AVATAR model is representative of modern large scale wind turbines [7]. The model used here has a 206m diameter (D) rotor with a locked-speed drivetrain operating at 9.0218rpm. Structural flexibilities have been ignored to simplify the model. The turbine runs at a fixed pitch angle (0° degrees) without external control.

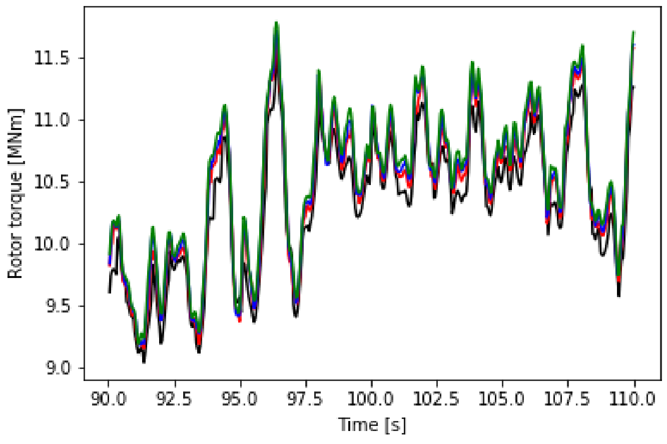
Bladed provides two other Vortex line parameters, the maximum number of wake steps and the maximum number of free wake steps. The maximum number of wake steps defines the point at which the vortex node is truncated from the wake. The maximum number of free steps is the limit after which the induction is no longer calculated, and the node propagates with the mean local wind speed and the last computed induced velocity. For these simulations, the free wake is defined as 4D and the wake length is 10D. As a result, a larger wake time will result in fewer total wake nodes as well as a slower update frequency. The rotor is given four revolutions before we begin recording the output sensors.

These simulations may not have been run on the same machine, however they were all relatively recent Intel quad-core machines with similar specifications. They were each run on 2 threads with AVX2 vectorisation enabled, and background tasks being controlled for. Software performance will have some expected variation on a run-by-run basis. Ideally, these simulations would be run multiple times so these variations can be measured and averaged out, however, given the number of simulations and time taken for each run, this was not feasible.

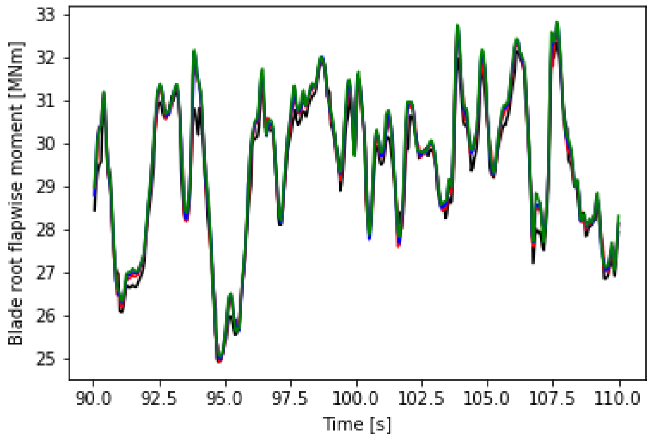
3.1.3.1 *Comparison for 10.5 m/s turbulent wind*

A wind file is generated using CFD simulations of an empty box at the rotor plane position, this is converted to Bladed format for these turbulent wind simulations. The turbulence intensity is isotropic at 10%, and there is no flow inclination or wind shear. A high fidelity Bladed run is used as a reference, this model has converged Vortex line parameters. This is compared to four Bladed Vortex line simulations with varying numbers of steps per revolution. This is a normalised measure of the wake time step. The rotor speed of 9.0218rpm corresponds to 6.6506 seconds per revolution, N steps per revolution is equivalent to a wake time step of 6.6506 / N. Here, we consider 20, 40, 60 and 120 steps per revolution. The references simulation has 240 steps per revolution.

Figure 3.3 displays time series graphs of the rotor torque and blade root flapwise loading. We observed in the same trend for all the considered runs, however there is a noticeable difference in peak loading and high-frequency variations. The longer wake time steps under-estimate both the rotor torque and the blade flapwise loading.



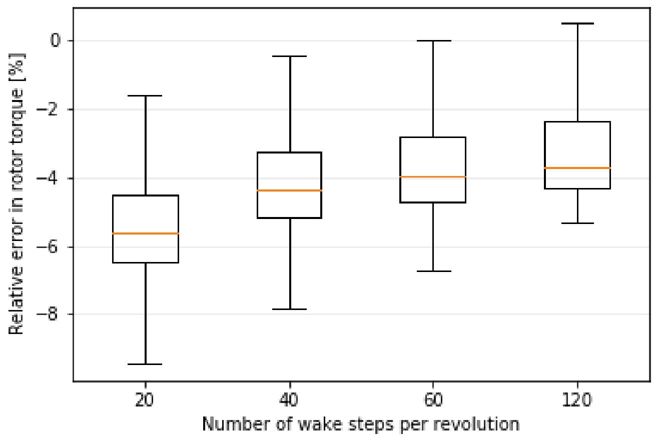
(a) Rotor torque



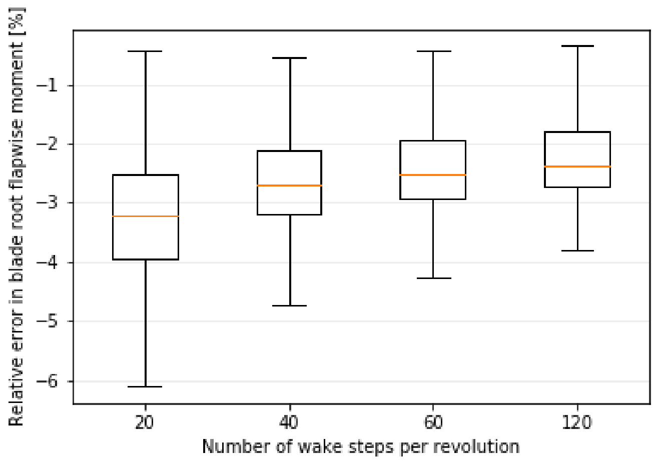
(b) Blade root flapwise moment

Figure 3.3: Time series plots for the AVATAR model in turbulent wind with varying vortex wake time step for t = [90,110]. 20 steps per revolution (black), 40 " (red), 60 " (blue), 120 " (green).

When compared to the reference run, we can see a clear convergence as we decrease wake time step size. Figure 3.4 displays these relative errors compared to the reference run, on a box-plot. The range of the error in the signals reduces with smaller wake time steps. Going from 20 steps per revolution (0.3325s) to 120 steps per revolution (0.0554s), the mean error reduces from 3.3% to 2.4%.



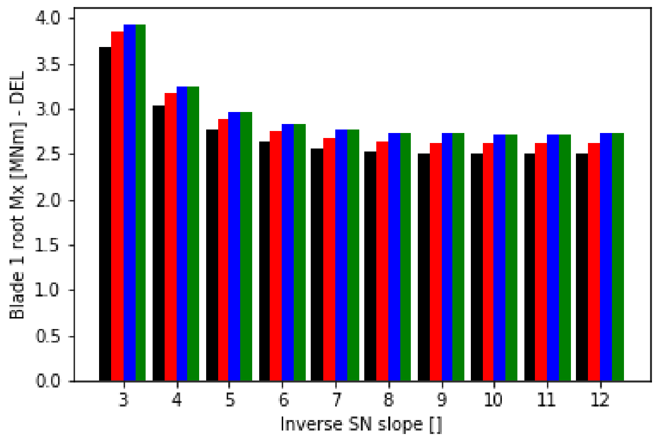
(a) Relative error in rotor torque



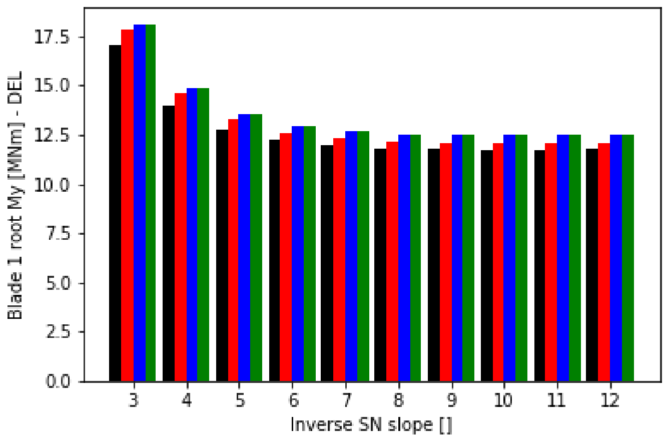
(b) Relative error in blade root flapwise moment

Figure 3.4: Box-plots for the AVATAR model in turbulent wind with varying vortex wake time step.

Figure 3.5 displays the post-processed fatigue loads for the blade root edgewise and flapwise loading. The same trend is observed in both figures 5a and 5b, the fatigue loads are underestimated as the wake time step is coarsened. This can be traced back to the rainflow exceedance ranges plots (stair-case plots not shown here). Here, we note that the maximum cycle range is underestimated by the coarser simulations, and although the other cycles are similar in range and frequency of occurrence, this effect dominates the damage equivalent loading value. This effect is amplified by the short simulations time applied in these tests, we would expect less pronounced differences for a longer time series and with seed selection variations.



(a) Blade 1 root edgewise moment



(b) Blade 1 root flapwise moment

Figure 3.5: Damage equivalent loads for AVATAR model in turbulent wind with varying vortex wake time step. 20 steps per revolution (black), 40 “ (red), 60 “ (blue), 120 “ (green).

Figure 3.6 displays the speed up factor when varying the number of steps per revolution. The speed up factor is defined as the maximum simulation time (from the set of all test cases) divided by the measured simulation time. The maximum simulation time is the 120 step per revolution simulation, so this will have a speed up factor of 1. The other runs roughly follow a cubic power law trend, with the 20 steps per revolution simulation running just over 45 times faster. Further analysis of the expected and realised simulation performance trends will be provided in the following section.

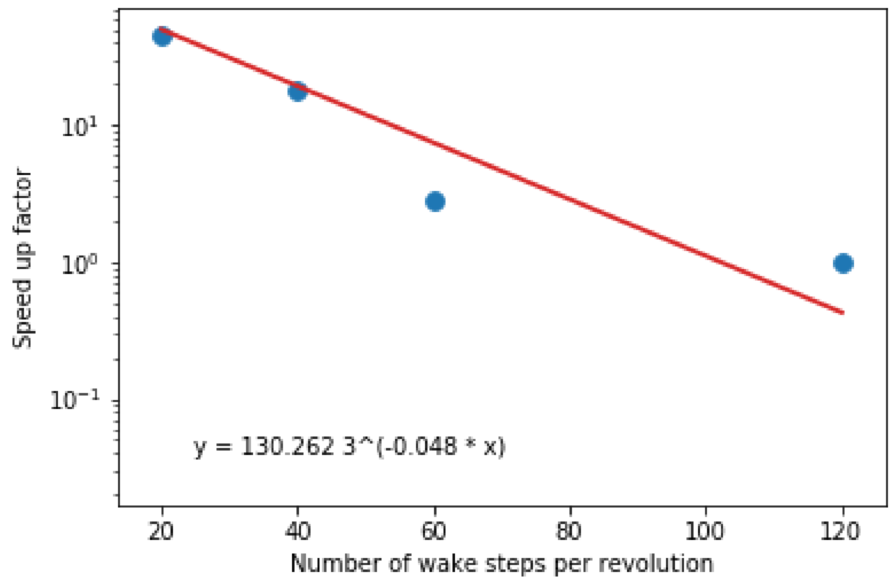
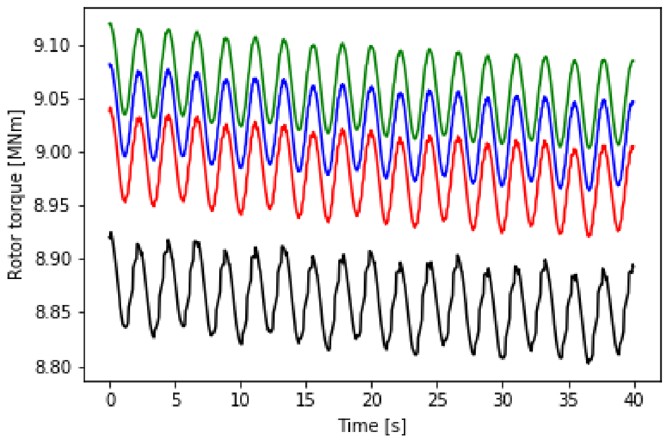


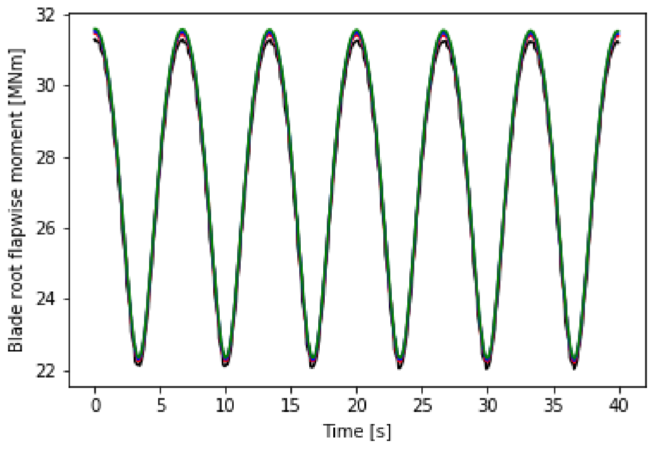
Figure 3.6: Speed up factor for the AVATAR model in turbulent wind with varying vortex wake time step.

3.1.3.2 *Comparison for 10.5 m/s vertical wind shear*

The same test was also applied to constant wind condition with an exponential vertical wind shear of 0.2. As before, there was no flow inclination and the rotor speed is fixed at 9.0218rpm. For this case, there is no reference to compare against, so we shall assume that the results tend to the true value as the number of steps per revolution increases. This is confirmed by figure 3.7, the sinusoidal rotor torque signal does appear to be converging as the number of steps increases. The results from the blade root flapwise moment signal are not as clear but do display this convergence behaviour. It should be noted that all runs start off at the same initialisation value, but then diverge as wake inductions begins to influence the velocity at the rotor plane.



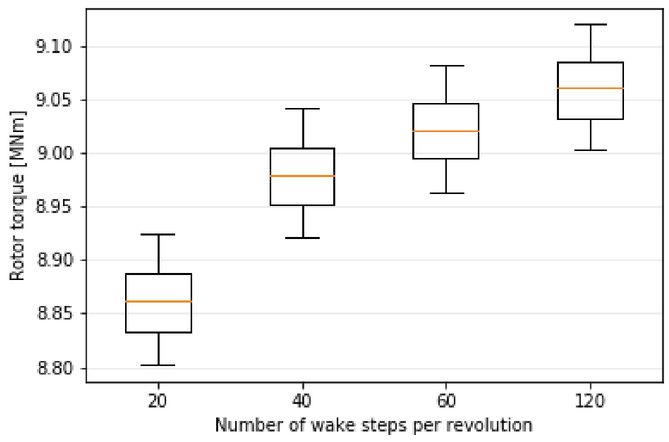
(a) Rotor torque



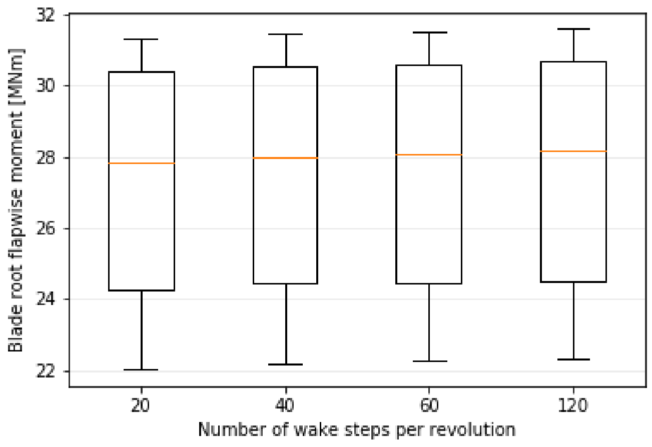
(b) Blade root flapwise moment

Figure 3.7: Time series plots for the AVATAR model in 10.5m/s vertical wind shear with varying vortex wake time step. 20 steps per revolution (black), 40 “ (red), 60 “ (blue), 120 “ (green).

From the box-plot in figure 3.8, we can see the same convergence trend more clearly. The distribution is the same for each run, it is only the mean value that changes. The plot for the blade root flapwise moment shows very little difference, there was less than 1% difference between the 20 and 120 steps per revolution runs.



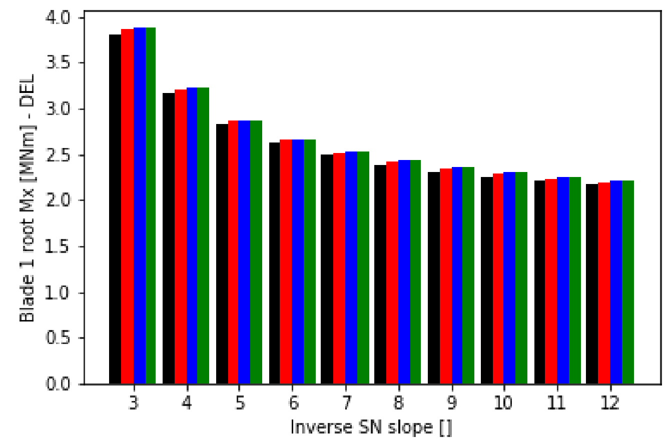
(a) Relative error in rotor torque



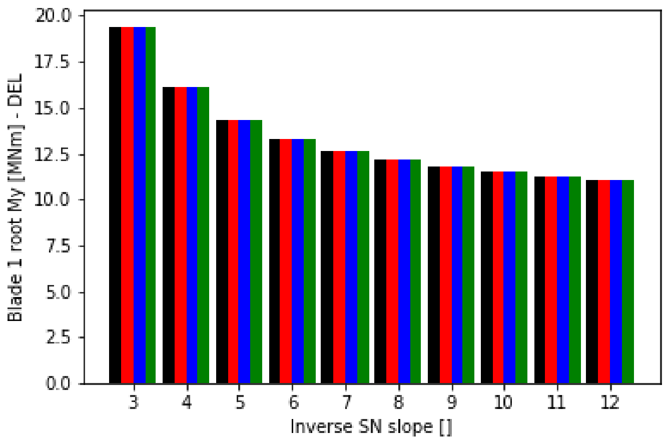
(b) Relative error in blade root flapwise moment

Figure 3.8: Box-plots for the AVATAR model in 10.5m/s vertical wind shear with varying vortex wake time step.

The damage equivalent loading for the blade root edgewise and flapwise moments in figure 3.9 show similar results for all the wake step sizes. The edgewise DELs display greater differences than the flapwise DELs, but this is not unexpected. As the edgewise loading magnitude is less than the flapwise, smaller differences in the cycle range maximum will result in larger differences in the equivalent loading signal.



(a) Blade 1 root edgewise moment



(b) Blade 1 root flapwise moment

Figure 3.9: Damage equivalent loads for AVATAR model in turbulent wind with varying vortex wake time step. 20 steps per revolution (black), 40 " (red), 60 " (blue), 120 " (green).

Figure 3.10 displays the speed up factor when varying the number of steps per revolution. The speed up factor is defined as the maximum simulation time (from the set of all test cases) divided by the measured simulation time. The maximum simulation time is the 120 step per revolution simulation, so this will have a speed up factor of 1. The other runs roughly follow a cubic power law trend, with the 20 steps per revolution simulation running just over 45 times faster. Further analysis of the expected and realised simulation performance trends will be provided in the following section.

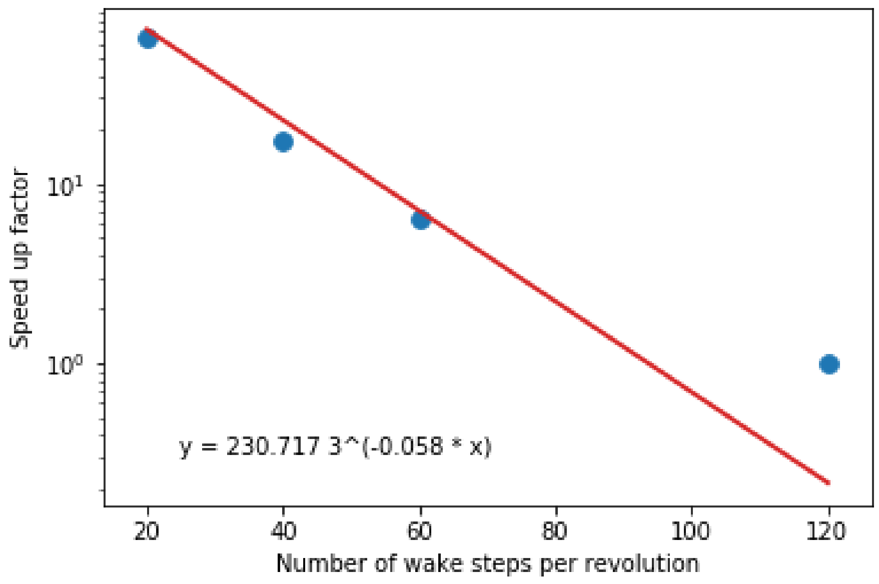
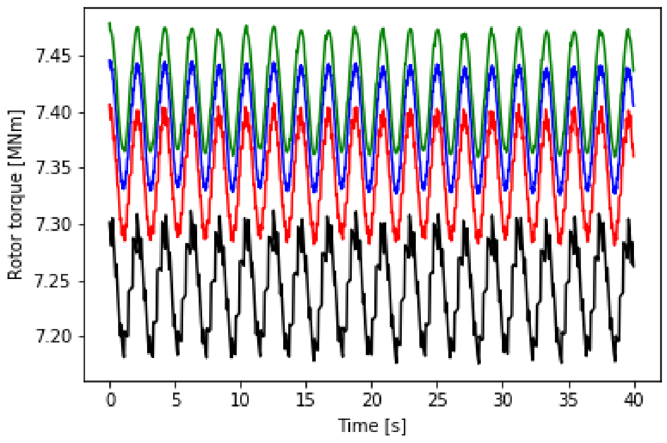


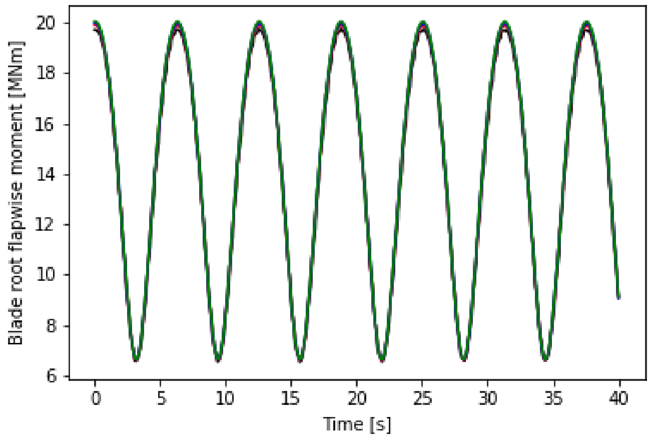
Figure 3.10: Speed up factor for the AVATAR model in turbulent wind with varying vortex wake time step.

3.1.3.3 *Comparison for 14 m/s vertical wind shear*

The same constant wind with vertical wind shear test case is run at a higher wind speed of 14m/s and a faster rotor speed of 9.6rpm (6.25 second per revolution). Once again, there is no reference run to compare against, and we assume that the results converge to a more accurate result as the number of wake steps per revolution increases. The plots in figure 3.11 show a 3% difference in mean torque between the 20 and 120 wake steps per revolutions test case and a negligible difference in blade root flapwise moments.

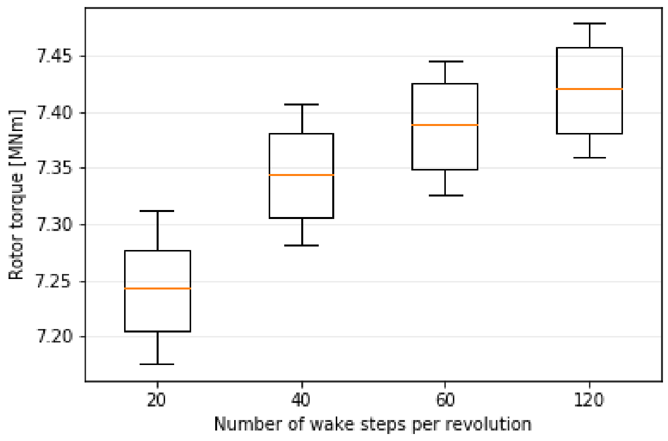


(a) Rotor torque

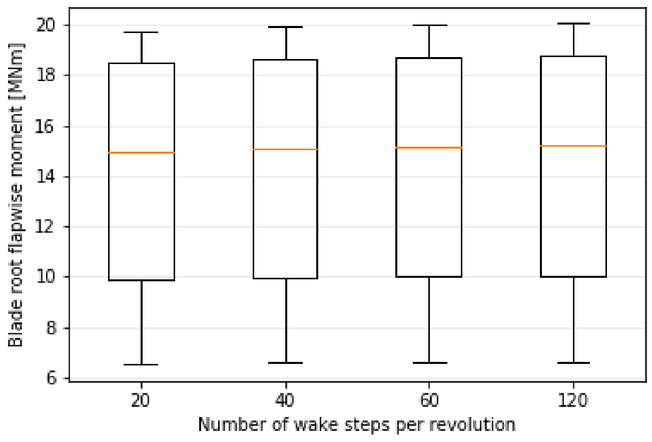


(b) Blade root flapwise moment

Figure 3.11: Time series plots for the AVATAR model in 14m/s vertical wind shear with varying vortex wake time step. 20 steps per revolution (black), 40 " (red), 60 " (blue), 120 " (green).



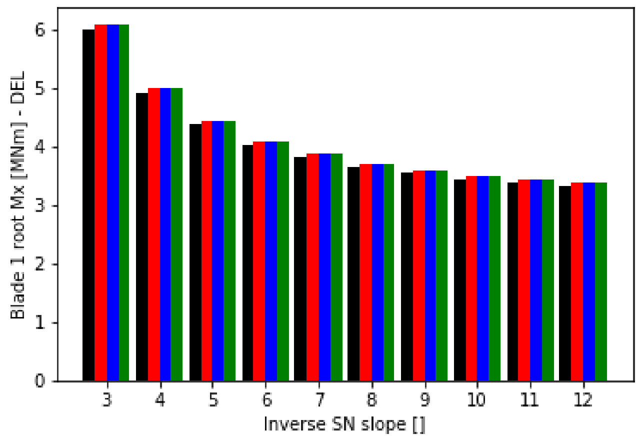
(a) Relative error in rotor torque



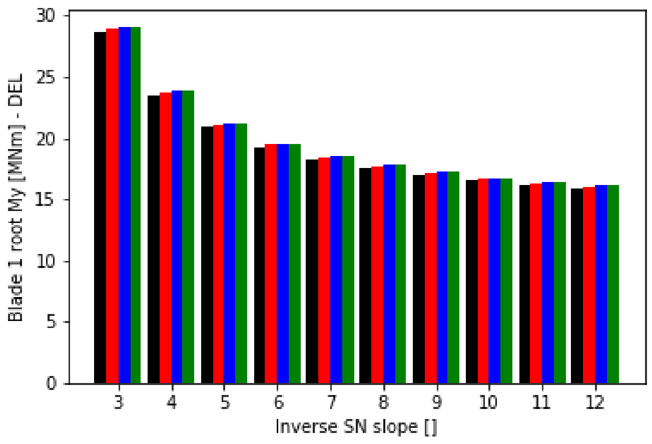
(b) Relative error in blade root flapwise moment

Figure 3.12: Box-plots for the AVATAR model in 14m/s vertical wind shear with varying vortex wake time step.

Figure 3.13 displays the post-processed damage equivalent loads. Once more, we note that the differences between all runs are small, but there is a clear convergence trend towards the 120 steps per revolution simulation.



(a) Blade 1 root edgewise moment



(b) Blade 1 root flapwise moment

Figure 3.13: Damage equivalent loads for AVATAR model in 14m/s vertical wind shear with varying vortex wake time step. 20 steps per revolution (black), 40 " (red), 60 " (blue), 120 " (green).

Again, we observe the same trend for the speed up factor. The 20 steps per revolution simulation ran 147 times faster than the reference test case.

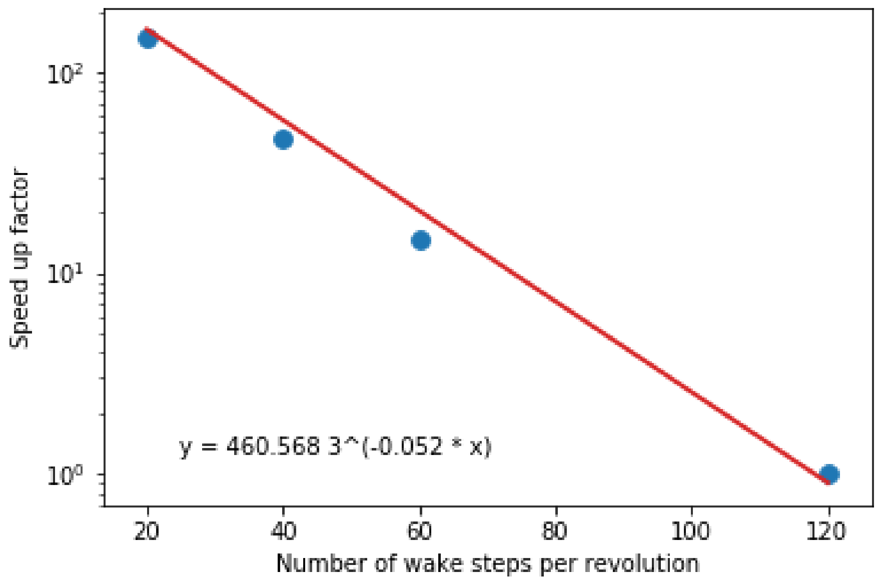


Figure 3.14: Speed of factor for the AVATAR model in 14m/s vertical wind shear with varying vortex wake time step.

3.1.4 *Performance improvements*

Changing the wake time step affects software performance in a twofold manner. First, it modifies the update frequency for the vortex line calculation, the scaling for this effect is linear. If the wake time step is doubled, for a fixed length simulation, the aerodynamics are updated half as frequently. Therefore, the total runtime spent on aerodynamic calculations decrease by half. This assumes that the complexity for each time step is the same irrespective of step size, which is the case for explicit computations. Secondly, a reduction in wake time step results in longer trailing vortex elements. If the user defines a fixed maximum wake length, rather than a maximum number of wake steps, then this will result in fewer nodes in the wake. As the complexity of the vortex wake iteration scales in proportion to $N_t^2 N_s^2$, where N_t and N_s are the number of trailing and shedding elements respectively, this effect has quadratic scaling.

The combined effect is a cubic scaling of simulation performance with respect to the wake time step. This performance scaling can be observed in figures 6,10 and 14. As we are measuring total simulation time, the scaling will not match the predicted value exactly. This is because the vortex line computation is coupled to the rest of the Bladed calculation. However, as the vortex line computation account for the majority of the simulation time, it is safe to assume that this overhead is small, but it is important to note the limitation of this assumption.

3.1.5 *Summary*

In this section, we have run the Tjaereborg pitch step case and the AVATAR model for several different environmental conditions. For the former, we recorded only the results and observed how the simulation accuracy was affected by the wake time step. The pitch step is an extreme event which does not normally occur but provides a good test case for understanding the effects of coarsening the wake time step. For the latter, the rotor torque and blade root edgewise and flapwise bending moment signals were measured and the simulation performance recorded. It is apparent from the speed up factor plots that there are significant performance gains when reducing the wake time step. There is, of course, a compromise with respect to simulation accuracy, this must be assessed for each model and environmental condition to find an acceptable trade-off. From the damage equivalent fatigue loads in our limited test set, it was observed that 40 steps per revolution provided a good comprised between simulation accuracy and performance.

3.2 Restricting number of wake nodes (DNV-GL)

With time, the number of vortex wake elements in the wake increases. As mentioned before, the complexity of the vortex line computation is proportional to $N_t^2 N_s^2$. Since the number of trailing and shedding vortex elements is incremented by the N_b and N_{b-1} respectively (where N_b is the number of blade stations) at each time step, the complexity can also be restated as being proportional to $(T/\Delta T)^4$ (where T is the current time and δT is the wake time step). This is not ideal, and it quickly becomes the limiting factor for the possible simulation length.

To get around this problem, Bladed provides the option to specify a maximum number of wake steps. This is the number of steps after which a vortex node is truncated from the wake. The quartic complexity scaling is present whilst the vortex wake is developing, but it becomes independent of time once the wake is fully developed. This is consistent with the physical model; the influence of a vortex element scales with the inverse square of its distance. Therefore, beyond a certain point, its influence on the rotor plane and near wake becomes negligible. In this context, it can be useful to define this limit in terms of the wake length rather than the number of wake steps. Generally, a value of ten rotor diameters is a conservative maximum wake length, if we are only interested in the aerodynamics at the rotor plane.

3.2.1 Free wake nodes

The details of the wake roll up are of primary importance when the wake is close to the rotor. After a user defined number of free wake time steps in Bladed, the nodes become “frozen” and convects with the mean wind speed and last computed induction velocity. There is a detailed grid study of this parameter in [6], which provides guidelines on the compromise between speed and accuracy. In term of computational complexity, where a fully free wake has a complexity scaling proportional to $N_t^2 N_s^2$, when the number of free wake nodes is restricted, the complexity becomes proportional to $N_t N_t^F N_s N_s^F$ (where N_t^F and N_s^F are the number of free trailing and shedding elements in the wake). Generally, a value of four rotor diameters is a conservative free wake length, if we are only interested in the aerodynamics at the rotor plane.

3.3 Optimisation of computational efficiency (DNV-GL)

3.3.1 Background

The key bottleneck of a free wake vortex line code is to evaluate the influence of all the vortex elements on all vortex nodes. This calculation should be done as efficiently as possible to ensure maximum computation speed.

In the first implementation, in Bladed 4.9 developer builds, an object-oriented data structure was used to store the information on the positions and velocities of all wake nodes and elements. This has the advantage of great code readability and clarity but turns out to hinder efficient computations.

For optimum computation speed it is vital make use of the vector-parallel architecture in modern-day CPU's to carry out multiple floating-point operations per clock cycle. This can be achieved if the data is laid out contiguously in memory and by writing loops that modern compilers can vectorise.

3.3.2 Estimating the Runtime for a Vortex Line Simulation

In this section we make a rough estimate of the runtime required for a typical 10 minute vortex line simulation. We base this on an estimate of the total number of floating-point operations required for a brute force implementation of the vortex line algorithm and knowledge of the peak floating-point performance of modern Intel CPUs.

3.3.2.1 Theoretical Peak Floating-Point Performance on Recent Intel CPUs

For the last 10 years, Intel CPU clock speeds have remaining unchanged at around 2 - 4GHz. Rather than increase clock speeds, Intel have instead added the following features to their CPUs to increase floating-point performance:

- Increased number of cores (Multicore)
- Widened SIMD registers to 256 and 512 bits (Vectorisation)
- Issue of multiple instruction per clock cycle (Multiple execution units per core)
- Fused operations such as fused-multiply add, FMA instructions (Two floating-point operations in one)

The theoretical floating-point performance of recent Intel CPUs (Haswell onwards) can be calculated as follows:

$$\begin{aligned}\text{Flops} = {} & \text{Number of cores} \times \text{(Register width / Float width)} \times \\ & \text{Number of floating-point execution units} \times \text{Instruction fusion ratio} \times \\ & \text{Clock speed}\end{aligned}$$

In this study, performance benchmarks of the Bladed vortex line code were run on an Intel i7-6700 processor with turbo-boost disabled. The data sheet for the Intel i7-6700 [8] provides the following data which can be used to compute the theoretical peak floating-point performance of the processor.

- Number of cores = 4
- Register width = 256 bits (AVX2)
- Float width = 32 bits (Single precision)
- Number of floating-point execution units = 2
- Instruction fusion ratio = 2 (FMA)
- Clock speed = 3.4 GHz (Base clock speed with turbo-boost disabled)

Using these figures gives a theoretical peak floating-point performance of 435.2 gigaFLOPs. To validate this theory, we ran a benchmarking code [9] yielding a peak floating-point performance of 417.2 gigaFLOPs which equates to 96% of the theoretical peak.

3.3.2.2 *Theoretical Floating Point Operation Count Required by a Brute Force Vortex Line Algorithm*

To compute the net induction at a single vortex node, the induction due to every vortex element in the wake must be computed. The induction on a single node due to a single element is computed by evaluating the Biot-Savart function. These inductions are summed to give the net induction on the vortex node. For a brute-force implementation of this computation, we can determine that:

$$\begin{aligned}\text{Single node floating-point operation count} &= \\ \text{Biot-Savart floating-point operation count} &\times \text{Number of vortex elements}\end{aligned}$$

Furthermore, to compute the induction on every node in the wake requires:

$$\begin{aligned}\text{Whole-wake floating-point operation count} = {}&\text{Biot-Savart floating-point operation count} \\ &\times \text{Number of vortex elements} \times \text{Number of vortex nodes}\end{aligned}$$

The Biot-Savart floating-point operation count depends on the precise implementation, but in the Bladed code, we obtained the values shown in table 3.2.

Table 3.2: Count of floating-point operations performed by the Biot-Savart function.

Floating-point operation	Count (Min/Max)
Add / Subtract	31-38
Multiply	38-41
Divide	3
Sqrt	3
Total	75-78

For problems of interest with a fully evolved wake, the following values are typical:

- Number of vortex nodes $\sim$ 50,000
- Number of vortex elements $\sim$ 100,000 ($\sim 2\times$ number of vortex nodes)

Using these figures, gives a whole-wake floating-point operation count of 400 x 109. From this, and our figure for peak theoretical peak floating-point performance on the i7-6700, we can compute a lower bound on the time required to compute the induction on every vortex node in the wake:

$$\begin{aligned}\text{Lower bound on whole-wake induction execution time (ms)} &= \\ 1000 * 400 \text{ x } 109 \text{ Total floating-point operation count } / \\ 435.2 \text{ gigaFLOPs Theoretical peak floating-point performance} \\ &= 919 \text{ ms}\end{aligned}$$

Finally, to estimate the runtime for a complete simulation, we must consider how many times the whole-wake induction computation is run during a simulation. Typically it is run 100 times per second of simulated time. Even running at its theoretical peak floating-point performance, this equates to 10 minutes of simulated time taking about a day to run on an i7-6700 CPU.

3.3.3 *Optimizing for Peak Floating-Point Performance*

Numerical codes will only approach the theoretical peak floating-point performance of modern Intel CPUs if they exploit all performance enhancing features of the hardware. In the Bladed vortex wake induction computation, we used the techniques shown in table 3.3.

Table 3.3: Techniques used in the Bladed vortex wake induction computation to exploit the performance enhancing features of modern Intel CPUs.

CPU Performance Feature	**Technique to Exploit Feature**
Multicore	Design an appropriate algorithm and use OpenMP.
Vectorisation	Design an appropriate data layout and algorithm, that can be vectorised by a vectorizing compiler.
Multiple execution units per-core	Rely on compiler optimisations and hyperthreading.
FMA Instructions	Select appropriate compiler settings

3.3.4 *Vectorisation Basics*

Vectorisation is the hardest feature to exploit, so it merits a brief explanation. Consider computing the square of eight 2D vectors as illustrated in table 3.4.

Table 3.4: Computing the square of eight 2D vectors. For vectorised computation, the x-values would be stored continguously in memory, as would the y-values and the s-values (result).

	0	1	2	3	4	5	6
x	0.0523	0.6836	0.6627	0.6154	0.4528	0.0276	0.1357
y	0.2844	0.7239	0.0900	0.2628	0.1767	0.9691	0.9215
s = x*x+y*y	0.0836	0.9913	0.4472	0.4478	0.2363	0.9399	0.8676

A sequential algorithm for this computation is:

Algorithm 1 Scalar algorithm

```
1: procedure SUM OF SQUARE OF TWO VECTORS
2: loop i = [1,2,3,4,5,6,7,8]:
3:     $a_s$ = load($x_i$)
4:     $a_s$ = mul($a_s$, $a_s$)
5:     $b_s$ = load($y_i$)
6:     $b_s$ = mul($b_s$, $b_s$)
7:     $a_s$ = add($a_s$, $b_s$)
8:     store($c_i$, $a_s$)
```

In the sequential algorithm, we use two scalar registers as and bs, we execute 8 x 3 = 24 scalar floating-point operations, and we perform 8 * 3 = 24 scalar loads/stores.

Using a vector width of four (meaning that the same operation acts on four values simultaneously), a vector algorithm for the computation is:

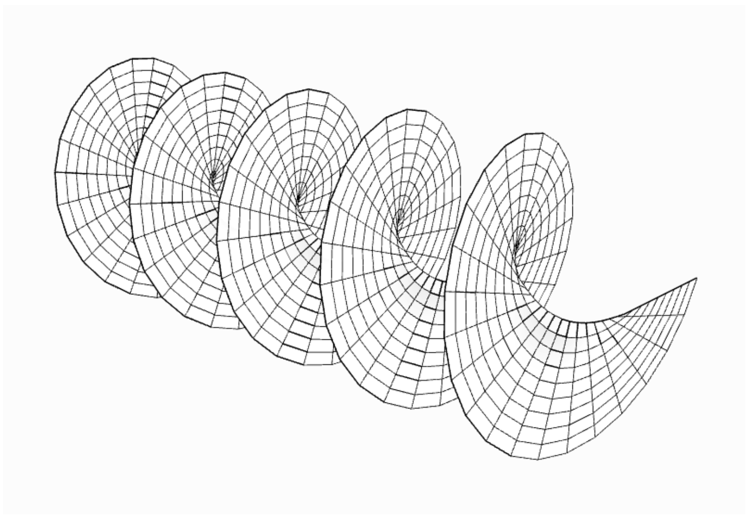
Algorithm 2 Vectorized algorithm

```
1: procedure SUM OF SQUARE OF TWO VECTORS
2: loop i = [1,4]:
3:     $a_v$ = vload($x_i$)
4:     $a_v$ = vmul($a_v$, $a_v$)
5:     $b_v$ = vload($y_i$)
6:     $b_v$ = vmul($b_v$, $b_v$)
7:     $a_v$ = vadd($a_v$, $b_v$)
8:     vstore($c_{4i,4i+3}$, $a_v$)
```

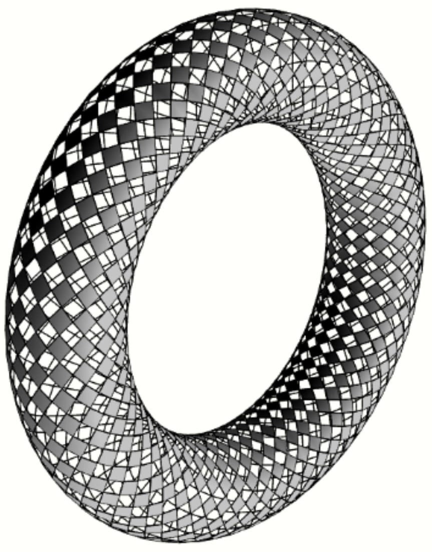
In the vector algorithm, we use two vector registers a_v and b_v, we execute 2 x 3 = 6 vector floating-point operations, and we perform 2 * 3 = 6 vector loads/stores. Thus, the vector algorithm performs $\frac{1}{4}$ of the number of operations compared to the sequential algorithm. On Intel CPUs vector loads/stores require the source/target data to be stored contiguously in memory. Understanding this point is critical when a designing a data layout and algorithm that allows a vectorizing compiler to vectorise code.

3.3.5 *Vectorisation of the Vortex Line Wake Induction Computation*

In the Bladed vortex wake code, vortex elements connect to form a 2D Cartesian grid. The position of each vortex node (or grid node) lies in 3D space. As illustrated in figures 3.15, this arrangement is flexible enough to allow complex geometries of vortex lines to be specified.



(a) Grid applied to a single blade wake



(b) Grid applied to a toroid

Figure 3.15

3.3.5.1 *Data Layout*

To permit vectorisation of the vortex line wake induction computation, we need to identify vectors of values that can be operated on simultaneously, and to store these contiguously in memory. To this end, we start by splitting the vortex elements into two sets according to the two directions in our 2D Cartesian grid. We name them:

- Shedding elements (Elements that form vortex lines shed from the trailing edge of a blade)
- Trailing elements (Elements that form vortex lines which trail from blade stations)

In term of data layout, we allocate a separate contiguous array to store each attribute of a node, shedding element or trailing element. For example, nodal (x,y,z) positions are stored as the three separate arrays

$$x = [x_0, x_1, x_2, x_3, \cdots, x_{n-1}]$$
$$y = [y_0, y_1, y_2, y_3, \cdots, y_{n-1}]$$
$$z = [z_0, z_1, z_2, z_3, \cdots, z_{n-1}]$$

3.3.6 *Indexing Scheme*

The indexing scheme illustrated in figure 3.16 is used to number the nodes and elements on a 2D Cartesian grid. The indexing scheme is defined in such a way that the index of any element can be used to compute the index of its end nodes. In order for this to be possible, some shedding elements are placed outside the boundary of the grid.

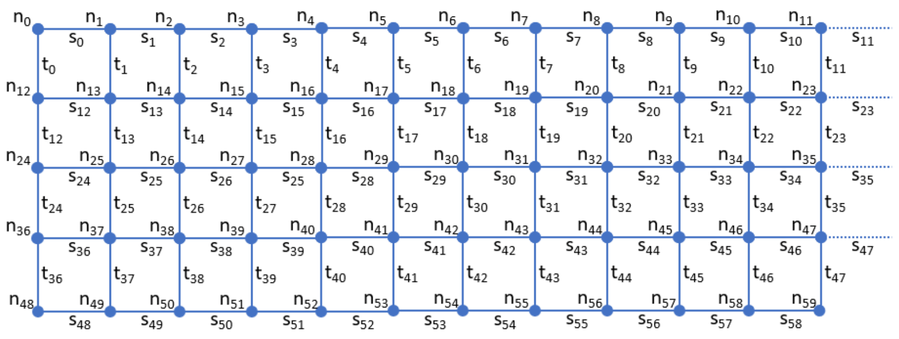

n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11
s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11
t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11
n12 n13 n14 n15 n16 n17 n18 n19 n20 n21 n22 n23
s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23
t12 t13 t14 t15 t16 t17 t18 t19 t20 t21 t22 t23
n24 n25 n26 n27 n28 n29 n30 n31 n32 n33 n34 n35
s24 s25 s26 s25 s28 s29 s30 s31 s32 s33 s34 s35
t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t34 t35
n36 n37 n38 n39 n40 n41 n42 n43 n44 n45 n46 n47
s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47
t36 t37 t38 t39 t40 t41 t42 t43 t44 t45 t46 t47
n48 n49 n50 n51 n52 n53 n54 n55 n56 n57 n58 n59
s48 s49 s50 s51 s52 s53 s54 s55 s56 s57 s58

Figure 3.16: Uniform indexing scheme defined on a 2D Cartesian grid of nodes and elements. The i-th node is labelled n_i , the i-th shedding element is labelled s_0 , and the i-th trailing element is labelled t_i . Note that shedding elements s_{11} , s_{23} , s_{35} and s_{47}, which are indicated by a dotted line, lie outside the boundary of the grid.

3.3.6.1 *Example Application of the Indexing Scheme*

The indexing scheme allows us to compute the square of the length of shedding of the i-th element as:

$$\text{Length shedding}_i^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2$$

and, to compute the square of the length of the i-th trailing element as:

$$\text{Length trailing}_i^2 = (x_{i+m} - x_i)^2 + (y_{i+m} - y_i)^2 + (z_{i+m} - z_i)^2$$

where m is the number of shedding elements per shedding vortex line. In figure 3.16, m = 12. In both cases, by taking a sequential set of element indices (i, i+1, i+2, i+3) we can compute the square of an element length entirely using vector CPU operations.

3.3.7 *Application of the Indexing Scheme to the Wake Induction Computation*

Applying the data layout and indexing scheme to the wake induction computation requires some care. The algorithm proceeds as follows:

```
1. Zero the induction for each node
2. For each node i
        For each shedding element j
            $\text{Induction}_i$ = $\text{Induction}_i$ + Biot-Savart(i,j)
3. For each trailing element j
        For each node i
            $\text{Induction}_i$ = $\text{Induction}_i$ + Biot-Savart(i,j)
```

Note the reversal of the traversal order of the two loops. This reversal is necessary to keep the inner loop vectorisable in each case.

3.3.8 *Dummy Elements*

On our Cartesian grid, the above algorithm doesn't apply to elements outside the boundary, as we don't want these elements to contribute to the wake induction. If we remove these boundary elements we break our uniform indexing scheme. If we modify our algorithm to skip these elements, our algorithm is no longer vectorisable. Instead, we introduced the concept of dummy element. A dummy element is assigned a vortex strength of zero, so makes no induction contribution to the wake.

3.3.9 *Multi-Blade Wake*

The concept of dummy element can be used to allow computation of the inductions for a multi-blade wake without changing the underlying data layout or algorithms. As shown in figure 3.17, the wake for each blade can be overlaid onto a single Cartesian grid, with dummy elements used to separate them. As the wake evolves during a simulation, the index of any given wake node remains the same, so we can grow the wake simply by increasing the memory allocated to our contiguous arrays.

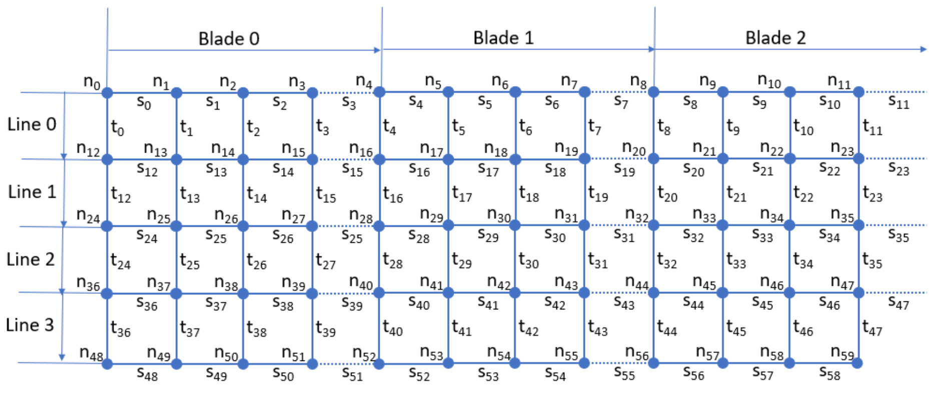


Figure 3.17: Application of the uniform indexing scheme and dummy elements to a multi-blade wake.

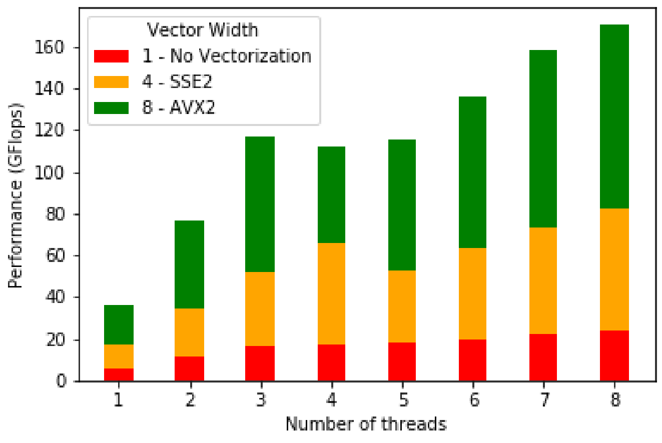
3.3.10 Toroidal Vortex Wake Induction Performance Benchmark

We created a standalone test executable for performance testing the Bladed vortex wake induction algorithm. The test executable isolated the core algorithm so that the overheads of other parts of the Bladed simulation could not influence the results. As a test case, we applied the vortex line algorithm to the toroidal geometry illustrated in figure 17b. The vortex lines form orthogonal sets of counter-winding spirals on the torus and map easily onto our 2D Cartesian grid. Arbitrarily, we chose to associate right-hand screwing vortex lines with shedding vortices, and left-hand screwing vortex lines with trailing vortices. Vortex element strengths were chosen to be constant. Due to symmetry, this test case has the property that there should be zero out-of-plane induction on the centreline of the torus. We instrumented the test to compute the total number floating-point operations executed by calls to the Biot-Savart function . The figure obtained was 10,753,064,875 operations. This figure allowed us to convert performance timings measured in ms to performance figures in gigaFLOPs.

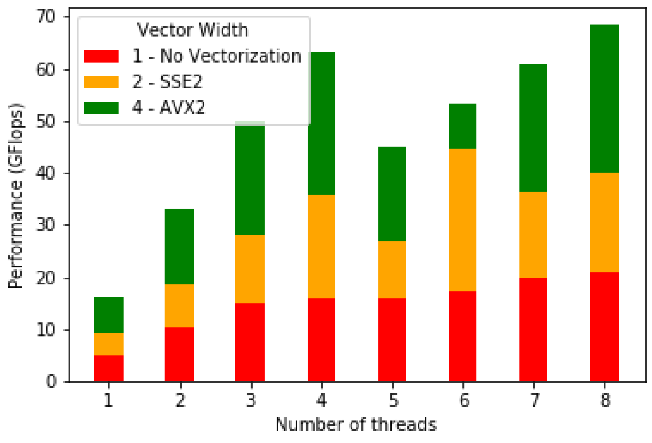
We tested all combinations of the following:

1. Vector width: 1 – No vectorisation, 2 – SSE2, 4 – AVX 2
2. Floating-point precision: Single, Double
3. Number of threads: 1, 2, 3, 4, 5, 6, 7, 8

We averaged runtimes over 20 runs. The slowest run, took 2112ms (5.1 gigaFLOPs) for double precision / 1 thread / no vectorisation, and the fastest run took 63 ms (170.7 gigaFLOPs) for single precision / 8 threads / AVX2. The speedup ratio between these runs was 33.5. The fastest run acheived 40% of the theoretical peak flops for the Intel i7-6700. Further results are charted in figure 3.18.



(a) Toroidal vortex wake induction performance benchmark. Single-precision on Intel i7-6700 at 3.4GHz



(b) Toroidal vortex wake induction performance benchmark. Double-precision on Intel i7-6700 at 3.4GHz

Figure 3.18

3.3.10.1 Performance Scaling with Number of Threads

From figure 3.18, we see that floating-point performance scales approximately linearly with the number of threads up to 4 threads. Stepping from 4 to 5 threads mostly causes a drop of performance. An upward trend is then recovered up to 8 threads. This behavior for 5-8 threads is likely caused by the hyperthreading feature of the Intel CPUs. Hyper threads share the same arithmetic unit on a CPU core. The Intel i7-6700 has 4 cores, each with 2-way hyperthreading. When running on 5 threads, one core must be hyper-threading. If those two threads are competing for use of the arithmetic unit of the core, then their combined throughput may be less than twice that of the cores that are not hyperthreading. If work is shared equally between all 5 threads, then the net effect is to reduce the overall performance.

3.3.10.2 Performance Scaling with Vector Width

Discounting the figures for 5-8 threads, we see that performance increases approximately linearly with vector width, as we would expect.

3.3.10.3 *Performance Scaling with Floating Point Precision*

Again, we see approximately linear scaling. Single-precision runs twice as fast as double-precision.

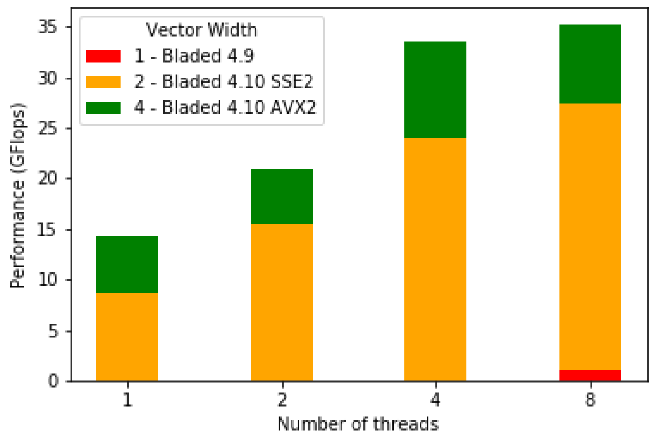
3.3.11 *AVATAR 10MW N-120 Performance Benchmark*

In order to test the performance of our vortex wake induction algorithm when integrated into Bladed, we chose to simulate the AVATAR 10MW model. The simulation duration was two minutes, while the vortex wake timestep was chosen such that wake updates occurred every 30° of blade rotation.

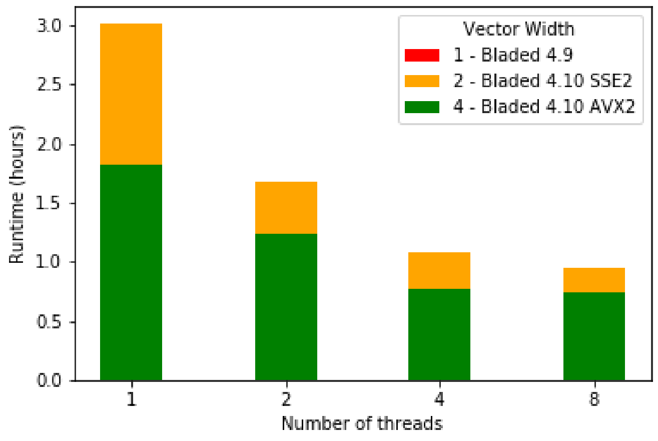
We ran the simulation using development versions of Bladed 4.9 and Bladed 4.10. Bladed 4.9 includes our naively parallelised vortex wake induction algorithm. Bladed 4.10 implements the new vector-parallel vortex wake induction algorithm. We measured the runtime for the whole simulation, which included the overhead of the rest of the Bladed computation. We recorded the number of Biot-Savart calls to allow the runtimes to be coverted to floating-point performance figures. Single-precision results are not available because the simulations failed. The results are presented in table 3.5 and figure 3.19.

Table 3.5: 10 MW performance benchmark. Total simulation runtime and floating-point performance (Vortex line floating-point operations only). Double-precision on Intel i7-7440 at 3.4GHz.

	1 thread	2 threads	4 threads	8 threads
Bladed 4.9	No data	No data	No data	27h 00m (0.96 gigaFLOPs)
Bladed 4.10 SSE2	3h 01m (8.61 gigaFLOPs)	1h 41m (15.5 gigaFLOPs)	1h 05m (24.0 gigaFLOPs)	0h 57m (27.4 gigaFLOPs)
Bladed 4.10 AVX2	1h 49m (14.2 gigaFLOPs)	1h 14m (20.9 gigaFLOPs)	0h 46m (33.6 gigaFLOPs)	0h 44m (35.2 gigaFLOPs)



(a) 10 MW performance benchmark. Double-precision on Intel i7-6700 at 3.4GHz



(b) 10 MW performance benchmark. Double-precision on Intel i7-6700 at 3.4GHz.

Figure 3.19

Firstly, we note that our new algorithm out-performed our old one by a factor of 37. We expect the AVX2 vectorisation to give us a factor of 4 performance boost. The additional speedup, we assume arises from the efficient uniform memory access patterns of our new algorithm, when compared the mostly random memory access performed by the old algorithm. The performance results for the AVATAR 10 MW model are harder to analyze than the results for the toroidal case because the figures include the overhead of the rest of the Bladed computation. Broadly, the trends are the same. Multi-threading performance scales well up to 4 threads, but then tails off. AVX2 gives better performance than SSE2. It is worth noting that running 8 threads and AVX2, our simulation achieved a floating-point performance of 35.2 gigaFLOPs versus a theoretical double-precision peak of 217.6 gigaFLOPs.

3.3.12 Conclusions

We have demonstrated that a carefully designed vector-parallel vortex wake induction algorithm can run at speeds that approach the peak theoretical floating-point performance of modern Intel CPUs. Our implementation achieves 40% of peak theoretical floating-point performance on the Intel i7-6700 and out-performed our naively implemented parallel algorithm by a factor of 37. The AVATAR 10 MW turbine performance test, which used typical input parameters for a vortex wake induction simulation, took 44 minutes for 2 minutes of simulated time. This runtime would be tolerable for users of Bladed.

3.3.12.1 Further work - GPU Vortex Wake Induction Computation

Our vortex wake induction algorithm would translate easily to OpenCL allowing it to be executed on GPUs. The peak floating-point performance of modern GPUs, such as the NVidia Titan RTX, is as much as 30 times greater than the peak floating-point performance of the Intel i7-6700 used for our benchmarks. Whilst this performance scaling might be achievable for the pure execution of the algorithm, it is important to understand that there is performance overhead associated with data transfer to/from the GPU. Careful analysis of this overhead would be required before concluding that a GPU could be used to accelerate a vortex wake induction calculation. In practice, there may be be a trade-off point, in terms of problem size, where switching to GPU computation yields better performance than CPU computation.

3.3.12.2 Benchmark against BEM

It is important to understand the performance penalty for using vortex line relative to BEM aerodynamics. When averaged across the many simulations run as part of the VortexLoads project, BEM simulation run at approximately real-time i.e. 1 minute of simulation time correspond to 1 minute of real time. Vortex line with double precision on 8 threads and AVX2 vectorisation requires 20x the simulation time. This conversion factor is important when designing hybrid load calculations, which will be discussed in the WP5 report.

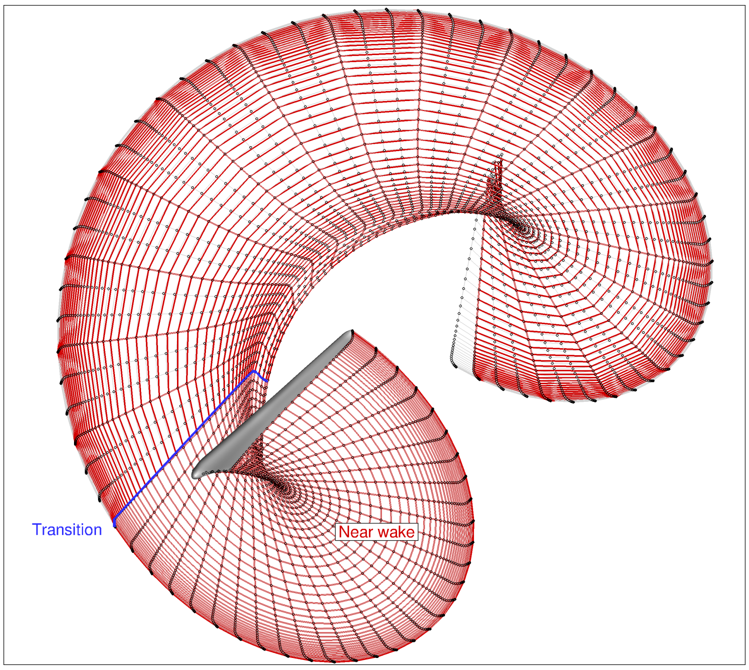
3.4 Far wake grid coarsening (ECN.TNO)

Vortex wake methods rely on the discretization of the wake into a network of shed and trailing vortices and on free-wake algorithms to align the rotor wake with the local induced flowfield. This nonlinear iterative process, acknowledging the fact that a suitable wake length behind the turbine has to be taken into account, has a severe impact on the memory and CPU-time requirements of the simulation. A CPU-time saving can be achieved by decreasing the total number of wake points through coarsening grid resolution in streamwise direction after the near wake region. An example of this technique is shown in Figure 3.20a, where, for clarity, the transition zone is started after half a rotor revolution, skipping two streamwise wake points as depicted in red (the fine wake grid being in black). During blade revolution, at each time step a new sheet is shed into the wake. The wake reduction technique allows to uncouple the time discretization of the problem from the streamwise spatial resolution of the wake. Indeed, after the transition line (in blue in Figure 3.20a) the effective time step for wake sheets convection is increased (triple in this example) whilst the time discretization for the numerical solution of the problem remains unchanged. Hence, due to the reduction, in this example two shed vorticity lines are discarded and the trailing vorticity is distributed along a straight path spanning a larger azimuth step. In this way, the vorticity shedding process from one lattice element to the next one in streamwise direction is applied to wake elements of increasing azimuthal dimension. In practical applications, the transition zone can be further downstream (e.g. after one rotation, like illustrated in Figure 3.20b).

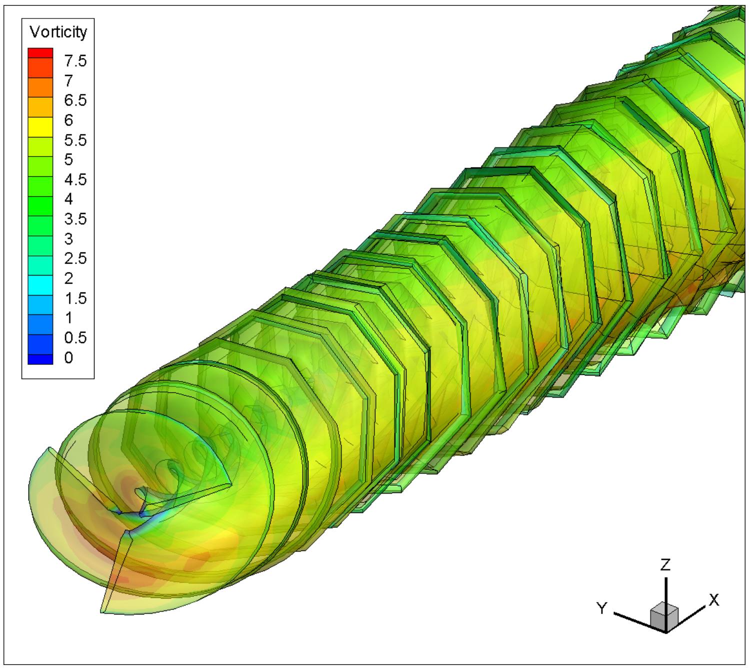
3.4.1 Implementation and validation strategy

The accuracy and CPU-time saving of the proposed wake model is assessed by implementing them into lifting line free vortex wake code AWSM [10] as implemented in the ECN Aero Module [11, 12] and applying the model to a number of load cases. In addition to that the technique was also implemented in the CNR-INSEAN FUNAERO panel code [13, 14] and verified for a number of cases. The streamwise position of the start(s) of the coarsening and the degree of coarsening itself are varied, which can be compared to a reference without coarsening. To enable a proper comparison, the full wake length in terms of convected distance was kept the same between reference and coarsened simulations. BEM results are calculated as well allowing to identify the added value of the vortex wake methods. To give the comparison a physical reference, measured load cases from the New Mexico wind tunnel campaign [15] have been used where possible. This test features the fully instrumented 4.5 m diameter three bladed Mexico rotor with variable speed and pitch control, tested in the large open jet facility of the German Dutch Windtunnels DNW. A comparison is made using the sectional loads obtained from the fast pressure sensors instrumented at five different spanwise sections (25%, 35%, 60%, 82% and 92%R). Since the measured values at the 60%R station were suspect (outlier), these were replaced using linearly interpolated values from the nearest stations. Chord normal and tangential sectional forces are obtained by integrating the measured pressure distributions along the local chord. Rotor integral variables such as axial force and torque are obtained by integrating these along the span. To prevent differences due to the limited number of sensors, the experimental resolution in spanwise direction is used to obtain these from the simulations. An experimental uncertainty is estimated assuming an uncertainty in the pressure sensors of 35 Pa. The solid aluminium blades are considered rigid, hence the simulations featuring the Mexico rotor do not include aeroelastic interaction.

The AWSM wake length has been defined to extend approximately three rotor diameters downstream of the rotor. Unless otherwise indicated the time step of the simulations is set to equal 10 $^\circ$ rotor azimuth. The applied wake reduction technique in AWSM consisted of one transition zone from where the shed vorticity lines were



(a) Reduced wake starting after half a revolution (red lines) compared to fine wake grid (black lines)



(b) Wake shape prediction (3 shed vortices skipped after one revolution)

Figure 3.20: Illustration of the applied rotor wake reduction technique

coarsened, usually applied after one rotation. The AWSM code was ran on a computer that allowed parallelization over 40 cores.

3.4.1.1 Load cases

For uniform axial inflow conditions, a traverse through the operational regime is performed from very low thrust values (featuring a high local angle of attack and separated flow), to design conditions and to high loading in the turbulent wake state.

Several representative unsteady load cases were also selected: three cases in 30 ° yawed flow, a hypothetical case featuring large vertical wind shear (power law exponent $\alpha = 1$) and a dynamic inflow case featuring two fast pitch steps. Finally, a normal production load case in partial load is subject of investigation, featuring a rigid version of the 200 m AVATAR 10MW rotor in turbulent inflow. Here time varying pitch and rotational speed have been prescribed as obtained from a BEM simulation with controller to ensure realistic local inflow conditions. An overview of the cases is given in Table 3.6.

Table 3.6: Load case overview

Name	Nr. cases	Rotor$^{\pm}$	U_∞	Pitch angle	Rot. speed	TSR
	[-]		[m/s]	[°]	[rpm]	[-]
Axial flow	9	M	7-30	-2.3	425	3-13
Yawed flow (30 °)	3	M	10-24	-2.3	425	4-10
Sheared flow	1	M	14.9$^{\mp}$	-2.3	425	6.7
Dynamic inflow	1	M	10.0	steps	425	10.0
Turbulent inflow	1	A	8.0‡	0.0	var*	var*

$^{\pm}$ M = Mexico, A = AVATAR † Attached flow cases only $^{\mp}$ Power law exp. $\alpha = 1$
‡ Time averaged hub height wind speed, class A NTM stochastic field
* Prescribed temporal variation based on BEM simulation with controller

3.4.2 *Results*

3.4.2.1 *Axial flow*

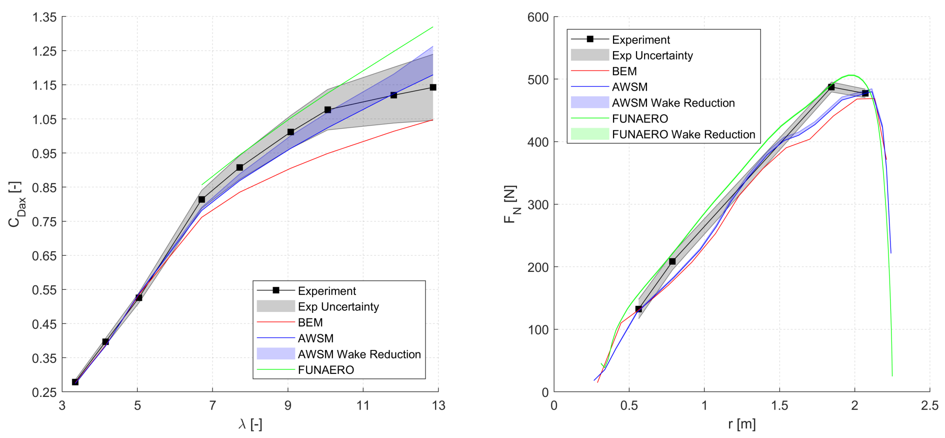
Since computational time and its saving mainly depend on the used number of wake points, they can be regarded irrespective of the load case under consideration. For axial flow, a full tip speed ratio (λ) traverse was simulated both by the panel and lifting line codes. The design load case was taken to study the effect on the computational effort of the proposed wake models. A summary of the results is given in Table 3.7. Progressively coarsening the wake geometry resolution after one wake revolution results in up to 79% CPU-time reduction at a comparable expense in accuracy. The

Table 3.7: Result of wake reduction techniques for New Mexico design load case (U_∞=14.93 m/s, λ=6.7, pitch=-2.3 °).

Code	Revolutions before transition	Skip shed vortices	Tot. shed vortices	ΔCPU*	ΔP^{*}	$\Delta F_{ax}{}^{*}$
	[revs]	[-]	[-]	[%]	[%]	[%]
AWSM	-	0	541	-	-	-
AWSM	1	1	288	-58	0.8	0.3
AWSM	1	2	204	-71	1.8	0.7
AWSM	1	3	162	-79	2.9	1.1
BEM	-	-	-	-99	-3.4	-2.3

* Δ with respect to reference (0)

same wake reduction configurations as reported have been used for all the load cases herein considered.



(a) Rotor axial force coefficient as a function of λ

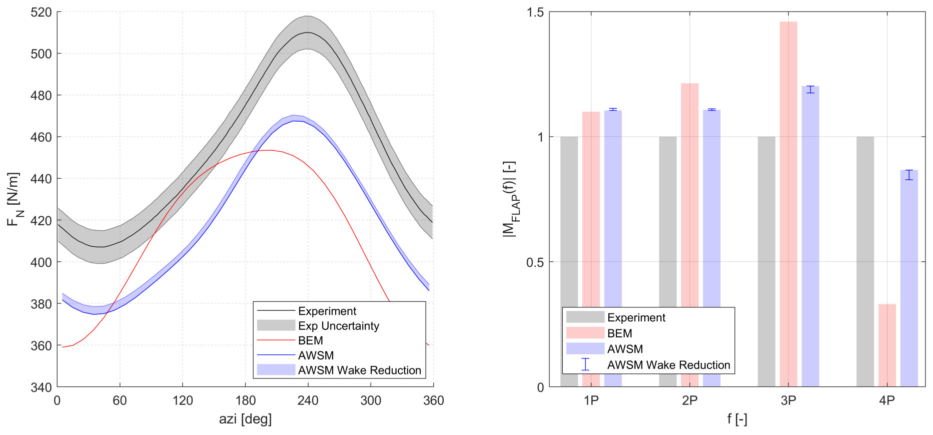
(b) Blade distribution of chord normal force (λ=6.7)

Figure 3.21: Effect of wake reduction on loading in axial inflow for the Mexico rotor

A visualization of the impact of this technique on the load prediction is given in Figure 3.21. The effect of the wake reduction technique is indicated by the lighter colored band for the configurations outlined in Table 3.7, where the reference is indicated by the darker colored line. BEM results are shown as well to identify the added value of the vortex wake methods. For the low thrust cases up to the design load case (λ=6.7), the effect of reducing the far wake grid resolution is small judging by the axial force coefficient predicted by AWSM in Figure 3.21a. Above this condition this effect progressively increases the predicted loading. A look into the wake geometry (not shown here) reveals the transition zone to be relatively close to the rotor due to the high rotor induction. Hence, rather than specifying the start of the transition zone in terms of rotor revolutions, it is better to define it in terms of downstream distance (approximately half a diameter when using a 10° azimuth time step). Moreover, it can be shown that including multiple transition zones (such as done in the FUNAERO code) reduces the sensitivity of predicted loads to wake reduction. The agreement between experiment and simulations is very good up to design load conditions. Inspecting the spanwise distribution of normal force in design load conditions (Figure 3.21b) also reveals a generally good agreement. However axial force results seem to slightly diverge for high tip speed ratios towards the turbulent wake state. It is hypothesized that BEM starts to suffer from not accounting for radial expansion effects and the lack of physics included in the turbulent wake state model. The difference between the two vortex wake models is attributed to the usage of airfoil data, which seems to differ from the panel method calculated pressure distributions. However the approximation of viscous effects in the latter can also add to the uncertainty. It is also noted that these conditions feature relatively low tunnel speeds and consequently low dynamic pressures utilizing only a small fraction of the measurement range (plus the fact that absolute differences are non-dimensionalized with a lower velocity enlarging differences in C_{Dax}). For a more detailed comparison between experiment and simulations the reader is referred to the final report of Mexnext Phase III [16].

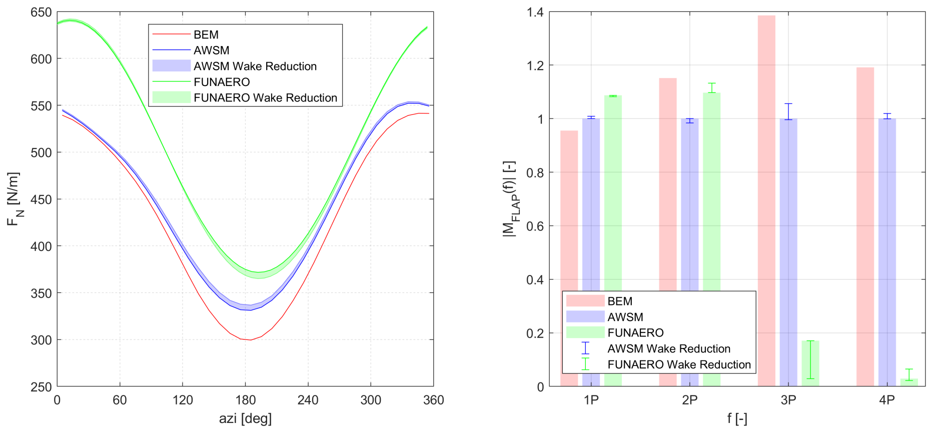
3.4.2.2 *Yawed and sheared inflow*

The results of the design load cases in *yaw* are illustrated in Figure 3.22. The wake reduction approach results in a small increase of the steady F_N load component, similar to the axial inflow case. The load variation (trend) is hardly influenced as depicted by the first 4 blade passage frequencies (i-P) magnitude shown in Figure 3.22b. The other two load cases in yaw (not shown here) reveal a similar behavior. It can also be observed that the AWSM trend is very much in line with the experiment, where BEM clearly falls short. The level offset between experiment and simulations is in line with the axial flow case at 82%R and is attributed to the difference between the used airfoil data and real sectional performance. Similar conclusions can be drawn for the other blade forces and moments and at different blade sections.



(a) Blade chord normal force variation at 82%R (b) 1, 2, 3 and 4P magnitudes of the flapwise blade root moment, referenced by experimental value

Figure 3.22: Effect of wake reduction for yawed flow (30 °), Mexico rotor, λ=6.7.



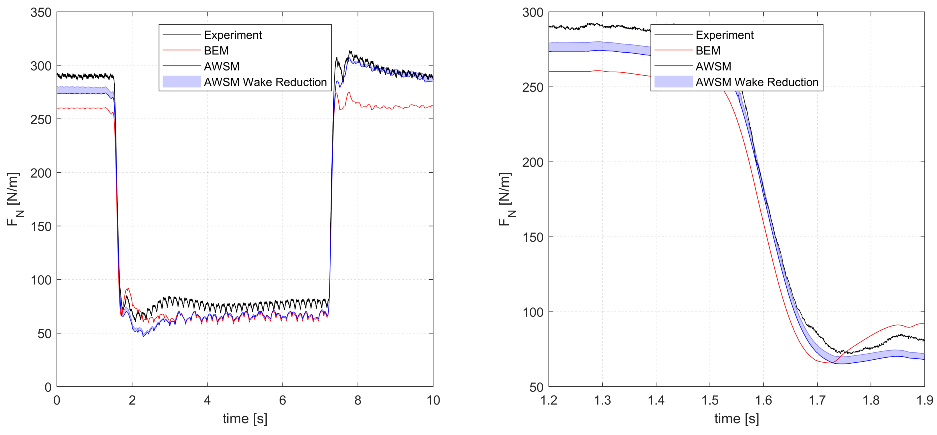
(a) Blade chord normal force variation at 82%R (b) 1, 2, 3 and 4P magnitudes of the flapwise blade root moment, referenced by AWSM value

Figure 3.23: Effect of wake reduction for extreme sheared inflow ($\alpha = 1$), Mexico rotor, λ=6.7

Although no experimental data is available for the *shear* case, the comparison to the FUNAERO code is provided. From Figure 3.23a there is a small positive offset in loading due to the application of wake reduction for the AWSM code, in line with the previous load cases. The FUNAERO result shows a small load decrease when the blade is in the downward position (180 ° azimuth). The harmonics analysis in Figure 3.23b indicates a small effect on the dynamic loading characteristics, most noticeable for the higher harmonics. The offset between AWSM and FUNAERO are in line with the previous comparisons for the high magnitude harmonics (1 and 2P), whilst increasing discrepancies arise at 3 and 4P where unsteady flow separation phenomena (not modelled in the panel code) can affect those harmonics magnitudes that, however, are very small in terms of absolute value. Also, Figure 3.23 clearly shows that BEM overestimates the load fluctuation in comparison to AWSM, which is due to the fact that induction variations in BEM are smoothed out over the annular streamtube. Although this difference may seem small it can be regarded as a canonical case for turbulent inflow conditions, imagining a blade slicing through turbulent eddies, which dominates blade fatigue loading.

3.4.2.3 Dynamic inflow

Coarsening the wake resolution is expected to have an impact on dynamic loading in case of sudden wake geometry changes. A *dynamic inflow* case featuring two pitch steps, first from -2.3° to 5° and then back to -2.3° again, is illustrated here. For this case the Mexico turbine is operating at a relatively high thrust (λ=10) and the time step is set to approximately 6° azimuth. Figure 3.24 indicates that, apart from the small level offset, both the initial load overshoot as well as the damping caused by the wake inertia are preserved. The latter effect is seen to be modeled superiorly by AWSM in comparison to the poor agreement with the experiment for BEM, which returns to the equilibrium value too soon.

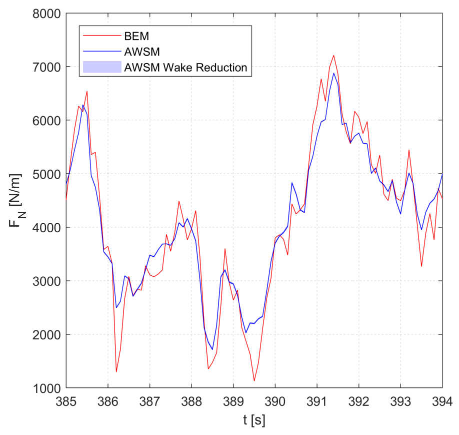


(a) Time trace of blade chord normal force at 82%R (b) F_N time trace during pitch step up from -2.3° to 5°

Figure 3.24: Effect of wake reduction for pitch step case (-2.3° $\rightarrow 5^\circ \rightarrow -2.3^\circ$), Mexico rotor, λ=10.

3.4.2.4 Turbulent inflow

A rigid version of the AVATAR rotor is used with a *turbulent* windfield as defined in Table 3.6. The time step is set to approximately 4° azimuth, depending on the prescribed rotor speed variations. Fatigue loads are determined over a 4 minute time serie. The tabulated results in Figure 3.25 indicate that the applied wake reduction configurations hardly influence the fatigue loads, also illustrated by the time trace of normal force at 82%R. This can be explained by the fact that the AVATAR rotor operates at a relatively low thrust coefficient, together with the fact that the time step in terms of azimuth angle is smaller than for the other load cases (i.e. the wake description suffers less from linearization of its curved shape). In agreement with the sheared inflow case, the BEM results feature larger load fluctuations resulting in a 7.4% higher damage equivalent load for the blade root flapwise moment.



(a) Time trace of chord normal force at 82%R

Configuration		$\Delta M^{\dagger}_{flap,eql}$
Revs before transition [rev]	Skip shed vortices [-]	[%]
1	1	-0.01
1	2	0.08
1	3	0.00
BEM		7.40

† Difference with respect to AWSM reference case

Figure 3.25: Effect of wake reduction for turbulent inflow load case, AVATAR rotor.

3.4.3 *Conclusions*

A cost-effective free-wake model has been investigated that requires limited programming effort to existing codes and can be considered as an engineering method making approximations to free vortex wake theory. The resolution of the far wake is reduced by progressively skipping shed vortices. The effect of this approach on loading characteristics has been investigated for a variety of load cases, from steady axial inflow to yawed flow, a dynamic pitch step and a turbulent inflow case. Results indicate that generally speaking dynamic and time averaged loading characteristics are preserved when the mid to far wake grid resolution is reduced, whilst CPU-time reductions of 60% to 90% are obtained. An offset in the average load levels can be found if the grid coarsening is initialized too close to the rotor plane. Gradually coarsening the wake grid resolution by using multiple transition zones is more efficient. Also, a preferred approach would not only skip shed vorticity lines but also mid-span trailing vorticity. Alternatively an analytical formulation representing a cylindrical wake may be used to account for the far wake from a certain downstream distance. In the end the preferred wake reduction configuration depends on the application and the accuracy needed. Concluding it can be stated that the applied wake engineering model is a promising technique bringing us desktop design load calculations using vortex wake methods. More details can be found in the dedicated publication on this topic [17].

Acknowledgments

The present work has been supported by the Project ”FARWING - Fast and Advanced Rotor Wake modellING” within the framework of the EU funded IRPWind Mobility Programme. In addition to that, the EU AVATAR project and TKI Wind Op Zee are acknowledged for their support.

4 Conclusions and recommendations

Within the framework of the TKI WoZ VortexLoads, improvements to both BEM and free vortex wake codes have successfully been implemented with the end goal to make wind turbine design load calculations more accurate. Here we can distinguish improvements to BEM type codes, which focus on engineering models and their implementation to make these more accurate. A model to account for the influence of the blade shed vorticity has been successfully implemented based on the time history of the vortices in the wake of each blade element. Application of this model was shown to bring BEM simulation results closer to the free vortex wake simulations at a relatively small extra computational expense. Another approach that has been researched is the definition of a representative streamtube wind speed in non-uniform inflow conditions for improved induction tracking. Although the first result of this approach both in sheared and turbulent inflow are very promising (in combination with shed vorticity modeling, the fatigue load difference with vortex wake codes almost disappears), more research is necessary on this topic to come to a more conclusive implementation, assuring not to cover up unintentionally other effects such as shed and trailed vorticity variation.

The improvements to free wake vortex codes focus on ways to reduce the computational effort associated with running them. Distinguishing between wake update frequency and aerodynamic sample time was shown to have a great potential in reducing CPU time. Limiting the number of free wake points and total wake length are obvious candidates to save computational time. Progressively skipping shed vortices in the far wake was shown to have a great reduction potential, with a minimal impact on the accuracy of (un)steady loading characteristics. For optimum computation speed it is vital to make use of the vector-parallel architecture in modern-day CPU's to carry out multiple floating-point operations per clock cycle. This was achieved by laying out the data contiguously in memory and by writing loops that modern compilers can vectorise, resulting in a computational effort of about 20 times the simulation time for vortex wake codes.

5 References

[1] K. Boorsma and F. Wenz and M. Aman and C. Lindenburg and M. Kloosterman. TKI WoZ VortexLoads Final report. Technical Report TNO 2019 R11388, TNO, September 2019. http://publications.tno.nl/publication/34634923/tbIASC/TNO-2019-R11388.pdf.

[2] K. Boorsma and M. Aman and C. Lindenburg and F. Wenz. Validation of BEM and Vortex-wake models with numerical tunnel data, TKI WoZ Vortexloads WP2. Technical Report TNO 2019 R11389, TNO, September 2019. http://publications.tno.nl/publication/34634924/jVk0uF/TNO-2019-R11389.pdf.

[3] O. Boeckmann. The effect of different bem implementations for wind turbines in sheared inflow conditions. ECN wind memo 2018-16, ECN, Petten, the Netherlands, 2018.

[4] L.P. Chamorro and R.E.A Arndt. Non-uniform velocity distribution effect on the Betz–Joukowsky limit. *Wind Energy*, 16:279–282, 2013.

[5] H.A. Madsen, T.J. Larsen, G.R. Pirrung, A. Li, and F. Zahle. Implementation of the blade element momentum model on a polar grid and its aeroelastic load impact. *Wind Energy Science Discussions*, pages 1–43, 08 2019.

[6] Menno Kloosterman. Verification and validation of vortex line code in bladed. Technical report, DNV GL, feb 2019. Verification and validation of Vortex line.

[7] AVATAR: Home, 2019.

[8] Intel® Core™ i7-6700 Processor. https://ark.intel.com/content/www/us/en/ark/products/88196/intel-core-i7-6700-processor-8m-cache-up-to-4-00-ghz.html. Accessed: 2019-05-19.

[9] Flops how many flops can you achieve? https://github.com/Mysticial/Flops. Accessed: 2019-05-19.

[10] A. van Garrel. Development of a wind turbine aerodynamics simulation module. Technical Report ECN-C–03-079, ECN, 2003.

[11] K. Boorsma, M. Hartvelt, and L.M. Orsi. Application of the lifting line vortex wake method to dynamic load case simulations. *Journal of Physics: Conference Series*, 753(2):022030, 2016.

[12] K. Boorsma, F. Grasso, and J.G. Holierhoek. Enhanced approach for simulation of rotor aerodynamic loads. Technical Report ECN-M–12-003, ECN, presented at EWEA Offshore 2011, Amsterdam, 29 November 2011 - 1 December 2011, 2011.

[13] L. Greco, R. Muscari, C. Testa, and A. Di Mascio. Marine Propellers Performance and Flow-Field Features Prediction by a Free-Wake Panel Method. *Journal of Hydrodynamics, Ser. B (English Ed.)*, 26(5):780–795, 2014.

[14] A. Calabretta, M. Molica Colella, L. Greco, and M. Gennaretti. Assessment of a comprehensive aeroelastic tool for horizontal-axis wind turbine rotor analysis. *Wind Energy*, 19(12):2301–2319, December 2016.

[15] K. Boorsma and J.G. Schepers. Rotor experiments in controlled conditions continued: New Mexico. *Journal of Physics: Conference Series*, 753(2):022004, 2016.

[16] J.G. Schepers and K. Boorsma et al. Final report of IEA Task 29: Mexnext (Phase 3). ECN-E-18-003, Energy Research Center of the Netherlands, 2018.

[17] K. Boorsma, L. Greco, and G. Bedon. Rotor wake engineering models for aeroelastic applications. *Journal of Physics: Conference Series*, 1037(6):062013, 2018.