Linearised CFD Models for Wakes

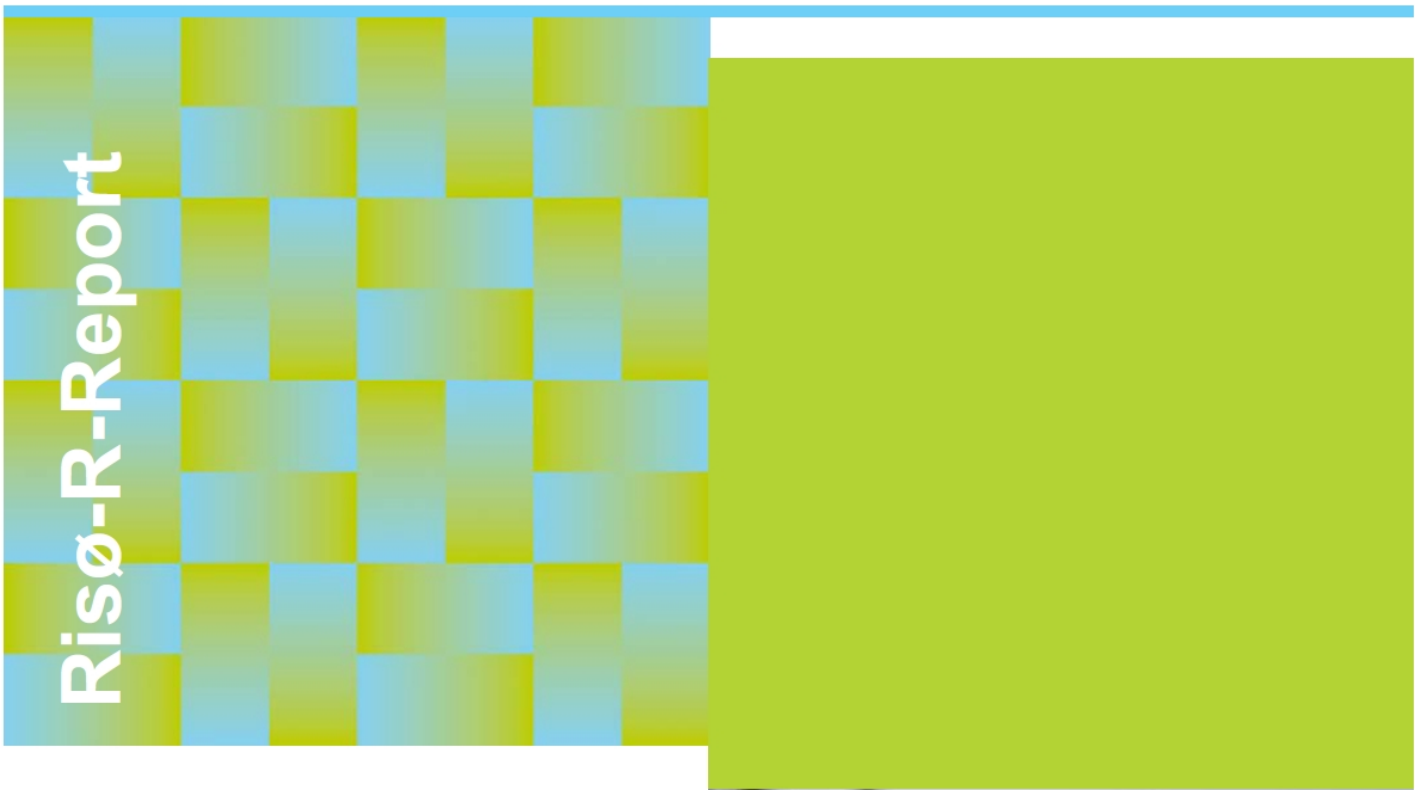
Ott, Søren; Berg, Jacob; Nielsen, Morten

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Risø–R–1772(EN)

Linearised CFD Models for Wakes



Søren Ott, Jacob Berg and Morten Nielsen

Risø National Laboratory, Roskilde, Denmark
December 2011

Abstract

This report describes the development of a fast and reasonably accurate model for the prediction of energy production in offshore wind farms taking wake effects into account. The model has been implemented as a windows application called Fuga which can run in batch mode or as a graphical user interface. Fuga is briefly described. The model is based on alinearization technique which is described in some detail, and linearized, governing equations are derived and written in a standard form based on a mixed–spectral formulation. A new solution method is used to solve the equations which involves intensive use of look–up tables for storage of intermediate results. Due to the linearity of the model, multiple wakes from many turbines can be constructed from the wake of a single, solitary turbine. These are in turn constructed from Fourier components by a fast Fourier integral transform of results derived from generic look–up tables. Three different models, based on three different closures, are examined:

- the 'simple closure' using an unperturbed eddy viscosity $\kappa u_* z$
- the mixing length closure
- the E–ε closure

Model results are evaluated against offshore wind farm production data from Horns Rev I and the Nysted wind farm, and a comparison with direct wake measurements in an onshore turbine (Nibe B) is also made. A very satisfactory agreement with data is found for the simple closure. The exception is the near wake, just behind the rotor, where all three linearized models fail. The mixing length closure underestimates wake effects in all cases. The E–ε closure overestimates wake losses in the offshore farms while it predicts a too shallow and too wide the wake in the onshore case. The simple closure performs distinctly better than the other two. Wind speed data from the the Horns rev met masts are used to further validate Fuga results with the 'simple' closure.

Finally, Rødsand 1 and 2 are used as an example to illustrate the interaction between wind farms.

Approved by: Hans E. Jørgensen

Checked by: Søren E. Larsen

Print: Pitney Bowes Management Services Denmark A/S · 2011

Contents

1 Introduction

Wake effects is an important issue in the planning of large offshore wind farms. Loss of energy production due to the wakes of upwind turbines, or even the combined wake from a nearby wind farm, has to be taken into consideration. Wakes that hit downwind rotors can also be a major cause of fatigue loads. A large number of flow models exist to assist in computing the various aspects of wakes and the interaction of wakes with rotors, see Vermeer, Sørensen and Crespo (2003) and Frandsen, Jørgensen, Barthelmie, Rathmann, Badger, Hansen, Ott, Rethore, Larsen and Jensen (2009) for reviews of wake models and experimental data. Their complexity and demand for computer resources range from simple, analytical models that run in virtually no time, over models based on the Reynolds Averaged Navier-Stokes equations (referred to here as RANS or CFD models) that may take several hours to run, to high resolution, time resolved large eddy simulations (LES models), where a computation may take several weeks to complete even on a large cluster. The computational costs of direct numerical simulation (DNS) is even more excessive, currently prohibiting its use in industrial applications, but the other model types could prove useful. Each type of model serves a purpose. Thus a time resolved LES model can be used for detailed studies of the interaction of turbulent flow with a rotor blade or the interactions between wakes. In this way LES calculations could serve as a source of validation or calibration data for simpler models such as RANS models. It is, however, difficult to use LES to compute the flow field in a whole wind farm and impossible to use it for annual energy production (AEP) estimates. In order to estimate the AEP of a large offshore wind farm we need the power production for a range of wind speeds combined with a range of wind directions and possibly even a range of atmospheric stabilities. This easily amounts to several hundreds of flow cases. On top of that we may even want to repeat the whole exercise a large number of times, e.g. in order to optimize the configurations of turbine positions. We need a fast model to do such things.

This report describes a particularly promising approach, linearized CFD models. A linearized CFD model is a 'cheap' version of a CFD model. It mimics the full CFD model's behaviour very well in regions where the perturbations are small, i.e. near the ground and in the far field of wakes. Very close to the rotor the behaviour is not entirely correct because the linearization spoils the momentum budget. It would therefore be natural to expect that the spoiled momentum balance in the near field behind the rotor would spoil the whole wake. However, experience shows that there is a tendency for the linearized wake to repair the damage done to the momentum balance in the near field and return to reasonable momentum deficits in the far field. This very convenient feature seems to make linearized models relatively accurate. At the same time they are also very fast. Depending on the size of the problem a linearized model can be 10^4 or 10^5 times faster than the corresponding CFD model. Another advantage of linearized models is that they can be solved without using a computational grid. This entirely eliminates numerical diffusion, which can otherwise be a problem, especially for large computational domains. A related problem is the generation of spurious mean pressure gradients caused by systematic errors in the momentum balance. This problem is also entirely eliminated in the linearized models. There is a price to pay for these advantages in terms of other types of numerical difficulties, and finding a useful solver for the linearized equations is the main problem. It took many failed attempts before the orthogonal chasing method was discovered. It is a new method and therefore described in some details below.

The linearization technique is completely general, and can be applied to any set of governing equations, but we shall concentrate on only three different closures: the 'simple closure', the mixing length closure and the E–ε closure (often called k–ε, but we have other uses for the letter k). The closures are defined and shortly discussed in section 3. In section 4 the principles of the derivation of the linearized model equations are demonstrated using the simple closure as example. When the work with linearized CFD was initiated it was not clear

which CFD model was the best and the choice was delayed. Since then more experience has been gained with CFD (and LES) applied to wakes, but without pointing out a specific CFD model as the best one. Instead there has been a growing concern for ability of present day CFD models to cope with wakes. This does not necessarily make linearized CFD a bad idea; but we just may have to wait for the proper CFD model to be invented. The parallel use of three closures reflects this state of affairs. The validation exercise concluding this report gives preference to the simple closure which clearly beats the other two. In the simple closure any influence the wake may have on the eddy viscosity is ignored. The other two closures have such a feed back, but the result turns out to be worse than no feed back at all. This is quite an embarrassment for turbulence theory.

The report contains parts that can be read separately. Section 2 describes the Fuga model. Sections 3–5.6 describe the mathematical details of linearized CFD models and solution methods, and these sections can be skipped or read separately from the rest of the report. It should also be possible to jump directly to the validation in section 6 or to section 7.2 on inter–farm shadow effects. Conclusions are drawn in section 8.

2 Fuga

The methods presented here were developed in the WINDSHADOW and TOPFARM projects and via internal funding. Carbon Trust project sponsored the implementation of the method in a PC based application as well as a series of validation exercises. An important outcome of the project is the PC application which we have called Fuga. Fuga is a tool that can be used to estimate wake effects on the power production in a cluster of wind turbines. It applies to offshore sites or, more generally, to homogeneous terrain. Fuga can be adapted to all the three closures studied here.

In this report we concentrate on validations of model results made by Fuga. Prior to Fuga the model results were produced on a more or less ad hoc basis by adapting various Mathematica notebooks. The risk of making errors was quite large, and many were actually made. The new code, made in Fortran, C++ and Delphi, has been subject to intense debugging and we believe that it produces bug free results. This is more than we will claim for results made with the old notebooks. Therefore we have chosen to validate the three linearized models exclusively with results made with Fuga (version 1.3).

The model has been optimized for fast computations on an ordinary PC. One of the ways to achieve computational speed is to use a system of look–up tables, where some are general and some are turbine specific. These tables are used to construct velocity fields behind a single turbine, and the combined effect of multiple turbines is evaluated as the sum of all velocity perturbations. Wind conditions at turbine sites inside the wind farm will depend on wakes from neighbour turbines. For a given wind direction the turbines sites are therefore first sorted according to their upwind distance, and then the local wind speed and thrust coefficient is evaluated starting with undisturbed upwind turbines sites and progressively evaluating sites in the downwind direction. When the thrust of all turbines is know, we can evaluate the combined wake velocity deficit anywhere. Estimates of annual energy production are calculated as probability-weighted integrals of power output at all wind speeds and directions.

Calculations are done by a suite of programs operating on the file system described in Ott and Nielsen (2010). All programs can be called from a Windows command line, but it is often easier to work with Fuga, which has a graphical user interface. Fuga is responsible for combining wakes from multiple turbines and presenting the results graphically. It also maintains the file structure, see figure 1, and calls the two programs, Preludium and Trafalgar, whenever new lookup tables and single-turbine wake velocity fields are needed. New tables are needed for each turbine type and combinations of boundary-layer height and surface roughness. Wake velocity deficits can, however, be scaled for variable turbine thrust

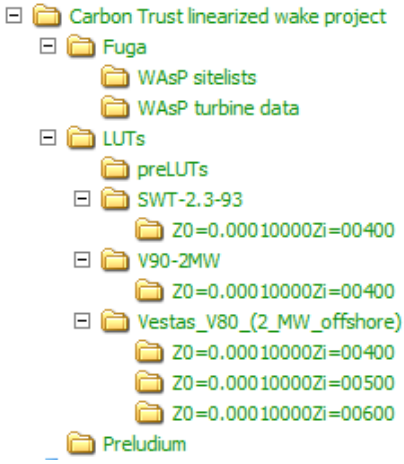


Figure 1. Fuga file structure. Subfolders to turbine type folders contain look-up tables for different atmospheric boundary-layer cases.

and thereby for variable wind speed.

The ambition is to integrate Fuga in WAsP as a supplement to the traditional wake model, and the model input and calculations are consequently organized in nearly the same way as in WAsP . Here, annual energy production is predicted by a flow model translating observed wind climates at reference sites to wind climates at individual turbine sites. In preparation for Fuga, the user must first set up a WAsP project; do a WAsP wind-farm calculations; export a so-called windfarm file (*.wwf) containing turbine positions and site-specific wind climates; and finally import this file into Fuga. Turbine data are read from a wind turbine generator file (*.wtg) as in WAsP . A sample collection of turbine data is available and new ones can be generated by the WAsP Turbine editor.

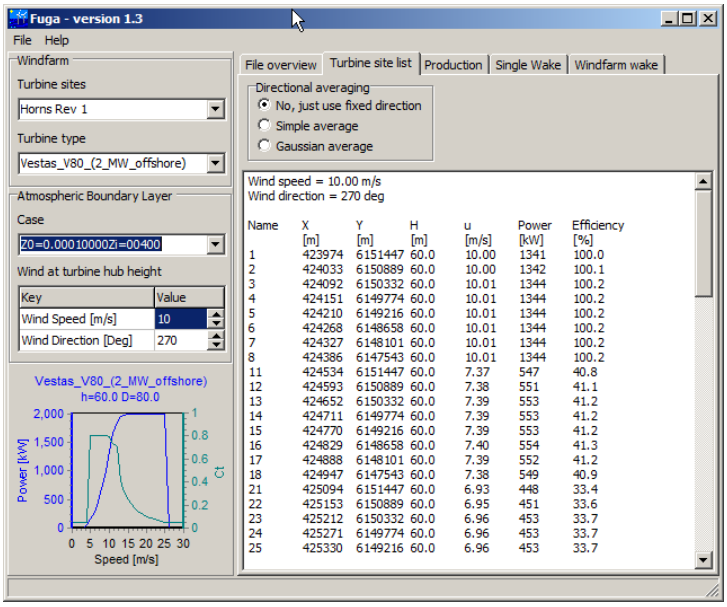


Figure 2. The Fuga graphical interface

Perhaps the best way to get acquainted with Fuga is to run it and use the help facility. Here you will find instructions on how to set up a project and how to generate various kinds of output. Figure 2 shows the Fuga user interface. The top-left frame contains two drop-down lists with available WAsP windfarm files and WAsP turbine files, which together

with the below boundary-layer case list defines a project. The free-stream wind speed and direction are chosen in the table below the boundary-layer selector. The right-hand side of program shows various pages:

- An overview of look-up tables;
- A list of turbine sites with sheltered wind speeds, power production and efficiency for the current free-stream wind;
- A list of annual energy productions including wake effects for individual turbine sites - either calculated by Fuga or imported from WASP . These results can also be shown for individual wind sectors;
- A display of the single-wake velocity field;
- A display of the combined wind-farm wake velocity field.

Initial comparison indicated narrower model wakes with lower centre-line velocities than could be observed in the field. We believe this to be an effect of variable and non-uniform wind direction causing wake meandering, as will be further discussed in section 6 of this report. The effect is not included in the present version of the model, but what we can do is to average the results for a range of wind speeds. This option is shown above the turbine site list.

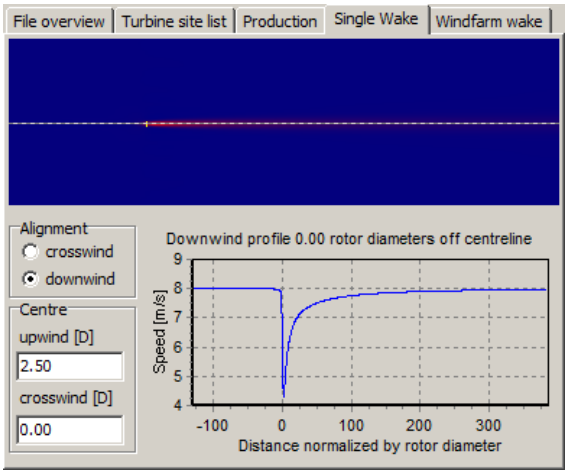


Figure 3. Fuga display of a single wake

Figure 3 shows the single wake velocity field at the selected wind speed. Local wind speeds are shown at the status bar at the bottom of the program window when the mouse cursor is moved over the wake display. Clicking on the wake field will change the position of the below cross profile, which is also indicated by a dotted line in the wake field display. A more precise positioning of the profile is done by the edit boxes left of the profile view.

Figure 4 shows the wind farm wake at the selected wind speed and direction. It is possible to adjust the main plot scales and change the reference turbine or reference position for the below charts. Two chart types are available - a power efficiency plot and a wind profile along a transect line. The first chart type shows power efficiency or local speed reduction as function of wind direction. These values are normalized by the undisturbed wind speed or the production of a turbine without wake effects. The chart can both be shown for a specific turbine or for the entire wind farm. The second chart type shows the wind speed along a transect line with downwind, crosswind or flexible alignment. The centre position is selected by clicking on the wind-farm wake plot.

Fuga has several data export facilities, which are invoked from its main menu. Graphics and data used for charts in the user interface can be exported and, in addition, you can

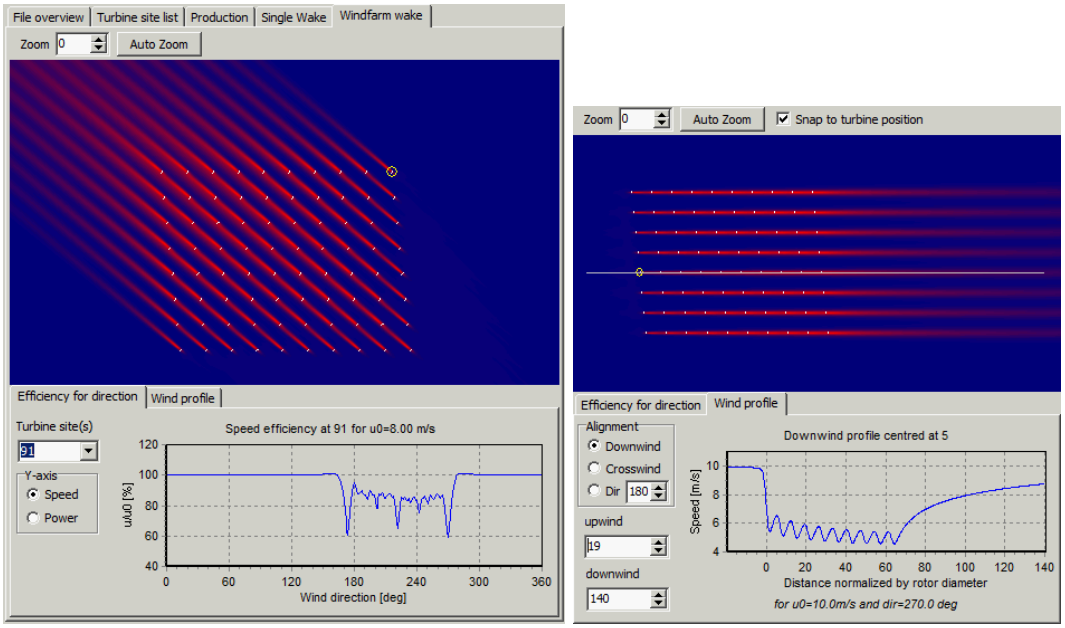


Figure 4. Fuga display of the combined wind–farm wakes: a) efficiency as a function of wind direction and b) wind speed along a transect line.

construct a result table with results from a range of wind speeds and wind directions, either for a single turbine site, for all the sites, or accumulated for the entire wind farm. For some computations, like systematic testing of a range of wind conditions, it might be more practical to call the scriptable FugaBatch program, which has no graphical user interface. A practical working method would be to 1) set up a WAsP project with reliable climate statistics and export the windfarm file, 2) load that file and WAsP turbine data into the interactive Fuga program, 3) generate look–up tables for appropriate boundary-layer cases, 4) execute FugaBatch calculations from a script and 5) analyse the results in programs like Excel or Mathematica.

A Fuga project is currently limited to wind farms with a single turbine type and uniform hub height, where the same single-turbine wake can be used for all sites and information is needed only at turbine hub height. In the near future Fuga will be able to work with 3D wakes and complex WAsP projects including a mixture of turbines in cluster of wind farms, like in section 7.2 of this report. Later on an attempt to model wake meandering will also be made. A Fuga limitation is that surface roughness and wind field have no horizontal variation. It could be argued that wakes should follow the curved flow modified by non–uniform terrain. However, since this is a second order effect, the wakes produced by the linearized flow model in Fuga only follow the undisturbed, horizontally homogeneous, flow field. This is no problem for an offshore site, but we recommend that the present version of the model is not used in complex terrain. On the other hand, the model works with site–specific wind climates calculated by WAsP which do depend on the specific terrain, and in this way the model takes the influence of coast lines into account.

3 The governing equations

The starting point is the Reynolds averaged Navier-Stoke's (RANS) equation for a quasi–steady flow. Setting $\partial u/\partial t = 0$ we have[1]

$$u_j \frac{\partial u_i}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial p}{\partial x_i} + f_i + \nu \nabla^2 u_i \tag{1}$$

where $\boldsymbol{u}$ is the mean velocity, $p = P/\rho$ is the pressure scaled with the (uniform) density ρ and $f_i = F_i/\rho$ is the bulk force (i.e. drag force field from turbines) also scaled with ρ. We are neglecting buoyancy and can therefore only model neutral stratification. The Coriolis force has also been neglected. The Reynolds stress tensor τ_{ij} needs to be determined by some kind of closure. The ones we will consider all model τ_{ij} via an eddy viscosity K:

$$\tau_{ij} = K \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \frac{1}{3} \delta_{ij} \tau_{kk} \tag{2}$$

Equation (1) is solved together with the continuity equation for incompressible fluid:

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{3}$$

Inserting (2) into (1) leaves a term $\frac{\partial \frac{1}{3} \tau_{kk}}{\partial x_i}$ which is the gradient of a scalar and can be absorbed into the pressure gradient (we will still call it p). $K \gg \nu$ so the real viscous term can be neglected.

3.1 Closures

Next point is to specify K. We consider three possibilities:

Simple closure

$$K = \kappa \, u_* \, z \tag{4}$$

This closure is about as simple as it can get. K is just taken as the free stream value without considering any effects from the perturbed flow field in the wake.

Mixing length closure

$$K = l_{\mathrm{m}}^2 \sqrt{2 S_{ij} S_{ij}} \tag{5}$$

where

$$l_{\mathrm{m}} = \kappa \, z \tag{6}$$

is the mixing length and

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{7}$$

is the rate–of–strain tensor.

The mixing length closure is based on classical turbulence theory of Prandtl (1925). It is used intensively in atmospheric applications.

[1] Here and in the following a summation over repeated subscripts is understood.

$E-\varepsilon$ closure

$$K = C_\mu \frac{E^2}{\varepsilon} \tag{8}$$

Where C_μ is a constant, E is the turbulent kinetic energy (TKE) and ε is the dissipation of TKE[2]. Two additional transport equations govern the two extra variables

$$\frac{\partial}{\partial x_j} v_j E = \frac{\partial}{\partial x_j} \frac{\nu}{\sigma_E} \frac{\partial E}{\partial x_j} + \Pi - \varepsilon \tag{9}$$

$$\frac{\partial}{\partial x_j} v_j \varepsilon = \frac{\partial}{\partial x_j} \frac{\nu}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_i} + \frac{\varepsilon}{E} \left(C_{\varepsilon 1} \Pi - C_{\varepsilon 2} \varepsilon \right) \tag{10}$$

where σ_E, σ_ε, $C_{\varepsilon 1}$ and $C_{\varepsilon 2}$ are model constants and Π is the TKE production given by

$$\Pi = \nu \frac{\partial v_i}{\partial x_j} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) = \frac{\nu}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right)^2 \tag{11}$$

With a body force (the turbine drag) present we might expect an extra production term in (9) or even an extra term in (10). Such extensions of the $E-\varepsilon$ model have been discussed by Kasmi and Masson (2008) and for canopy drag also by Sanz (2003) and Sogachev and Panferov (2006).

The E-ε model and similar so called one–and–a–half order closures are very popular for industrial applications, see Wilcox (2002) for details. It should be noted that the model constants are not supposed to be free parameters that can be used to fit specific data. The intension is that the model constants should reflect universal properties of turbulent flow and hence a single set of constants should fit all possible flows. Comparison with data from a variety of experimental flows has resulted in a set of 'standard' constants:

$$C_\mu = 0.09 \quad \sigma_E = 1 \quad \sigma_\varepsilon = 10/9 \quad C_{\varepsilon 1} = 1.44 \quad C_{\varepsilon 2} = 1.92 \tag{12}$$

We are not supposed to deviate from these constants unless we have good arguments. A re–calibration of the constants should involve new as well as old data.

It should be mentioned that Yakhot, Orszag, Thangam, Gatski and Speziale (1992) managed to derive the $E-\varepsilon$ model equations more or less from scratch using renormalization group (RNG) theory, a technique originally developed for quantum field theory. The derivation yields a theoretical estimate of the values of the constants[3] which are not far from the standard values. It is worth noting that the RNG derivation involves the assumption that the $E-\varepsilon$ model actually works. If a particular flow is poorly represented by the RNG (or standard) model results, it could therefore be taken as an indication that the $E-\varepsilon$ model just does not work for that flow, so that re–adjustment of model constants would be in vain or even misleading. Unfortunately, this seems to be the case for wakes where the standard model makes a very poor performance. This observation has been made by Kasmi and Masson (2008), Rados, Prospathopoulos, Stefanatos, Politis, Chaviaropoulos and Zervos (2009), Réthoré (2009) and many others. The possible causes for this failure were discussed in Réthoré (2009) and Réthoré, Sørensen and Bechmann (2010) where the usefulness of the eddy viscosity concept is seriously questioned. One of the problems is that a wake in the atmosphere is influenced by turbulence on a very wide range of scales. The small scale turbulence, i.e. eddies with sizes less than the wake width, tend to mix the wake with the surroundings in an essentially diffusive manner which is well described by an eddy viscosity. However, in the atmosphere a large portion of the turbulent kinetic energy resides on scales that are larger than the wake width, and these large eddies tend to move the wake as a whole without causing much mixing. The effect of large eddies is therefore to produce meandering motion, which leads to an uncertainty in the side–to–side wake position, but does not lead

[2] 'Specific dissipation' is more correct - ε is measured in W/kg=m^{2}s^{-3}

[3] $C_{\varepsilon 2}$ is actually given by an expression involving the rate of strain ($\sqrt{2S_{ij}S_j}$) and a constant β which must be determined from experiments

to decay of the wake by mixing. The best we can hope for using eddy viscosity is therefore to model the influence of the small scale turbulence on the mean flow. If we want also to represent the large scale part of the turbulent kinetic energy, we must of course incorporate the mechanisms responsible for it. These are specific, atmospheric processes such as convective rolls, gravity waves, the occasional, spontaneous creation of large descending jets, differential heating by passing clouds and possibly more mechanisms. In order to include these processes we would have to use a time resolved model that includes thermal effects and the Coriolis force. The conventional RANS/CFD framework is inadequate, but it could be done with Large Eddy Simulations that include temperature. However, only at rather extreme computational costs. The sad conclusion is that simple CFD models lack the features necessary to represent meandering.

The logarithmic profile

In flat terrain all these closures yield the usual logarithmic profile

$$u^0(z) = \frac{u_*}{\kappa} \log z/z_0 \tag{13}$$

where u_* is the friction velocity, z_0 is the surface roughness and $\kappa = 0.4$ is the von Karman constant. We will regard κ as fixed which poses the following constraint on the E–ε model constants

$$C_\mu^{1/4}\sigma_\varepsilon^{1/2}(C_{\varepsilon 2} - C_{\varepsilon 1})^{1/2} = \kappa \tag{14}$$

Since the model does not include the Coriolis force, the surface layer fills the whole boundary layer. The logarithmic wind profile, which is a consequence of surface layer scaling, is therefore valid all the way up to the upper boundary.

Over land the surface roughness is often assumed to be a constant independent of the wind speed. Over water the roughness is created by waves and hence z_0 depends on the wind speed as well as other local factors that determine the local wave characteristics. These other factors include fetch (distance to shore) and the orography of the seabed. The surface roughness increases with the wind speed and is often assumed to follow the Charnock (1955) relation

$$\frac{gz_0}{u_*^2} = A_{\mathrm{C}} \tag{15}$$

where A_{C} is the Charnock parameter. The relation is limited to 'normal' wind speeds, neither too low or too high. At very low wind speeds z_0 approaches a value appropriate for a smooth boundary, and at very large wind speeds, where water droplets can have a kind of lubricating effect, z_0 even decreases with increasing wind speed, thus contradicting (15). However, turbines usually operate in the 'normal' range of wind speeds where Charnock's relation can be applied. The relation only yields a rough estimate of z_0, a fact reflected in a wide range of experimental values of $A_{\mathrm{C}} \sim 0.012$–$0.035$. It should also be noted that swell can cause relatively large waves to inject momentum into the air rather than extracting it. In other words, the waves push the air forward and act as a kind of negative friction. In order to representing this in framework of (13) would require $u_* < 0$ and $z_0 \gtrsim z_h$. This is rather bizarre, so we have to disregard such special situations. Perhaps this could amount to about 1% of the cases, but mainly low wind speeds.

For a fixed wind speed at hub height, $U_h = u^0(z_h)$, we can use (13) and (15) to determine u_* and z_0. Using the empirical relation $\sigma_u \approx 2.5u_*$ (Panofsky and Dutton 1984) we then have fixed the turbulence intensity $\mathrm{Ti} \equiv \sigma_u/U_h$ at hub height, in fact we have

$$\mathrm{Ti} = \frac{\sigma_u}{U_h} \approx \frac{2.5\kappa}{\log z_h/z_0} \approx \frac{1.0}{\log z_h/z_0} \tag{16}$$

Real data always show some spread of Ti even for constant U and neutral conditions. We could therefore turn the argumentation around and determine z_0 by means of (16) using U_h and Ti from data. The wide range of experimental values of A_C justifies this procedure.

3.2 Actuator disk model

The actuator disk model is used for the forcing term $\boldsymbol{f}$. The drag force is determined by the free wind i.e. the wind speed at hub height determined after removing the turbine. For a given free wind U_{free} along with the x-axis and the hub placed at $(x_{\text{h}}, y_{\text{h}}, z_{\text{h}})$ the drag force can be written as

$$f_1 = -\frac{1}{2} C_{\text{T}} U_{\text{free}}^2 \, \delta(x - x_{\text{h}}) \, \Theta(R^2 - (y - y_{\text{h}})^2 - (z - z_{\text{h}})^2) \tag{17}$$

where δ is a Dirac δ–function and Θ is a step function (zero for negative arguments, 1 for positive). In calculations the force is always smeared out a bit for numerical reasons.

3.3 Boundary conditions

The flow will be assumed to be *lid driven* with vanishing mean pressure gradient. At the upper boundary $z = z_i$, the velocity is assumed to be fixed. The lower boundary is imposed at $z = z_0$ where we $\boldsymbol{u} = 0$. This gives 6 boundary conditions in all:

$$\begin{aligned} u(x, y, z_0) &= 0 \\ v(x, y, z_0) &= 0 \\ w(x, y, z_0) &= 0 \\ u(x, y, z_i) &= u^0(z_i) \\ v(z_i) &= 0 \\ w(z_i) &= 0 \end{aligned} \tag{18}$$

This is sufficient both for the simple closure and for the mixing length closure, but the E–ε model has two more variables and needs two more conditions. These are:

$$\begin{aligned} E'(x, y, z_0) &= 0 \\ \varepsilon(z_0)\, z_0 &= C_\varepsilon E^{3/2}(z_0) \\ E'(z_i) &= 0 \\ z_i \varepsilon'(z_i) + \varepsilon(z_i) &= 0 \end{aligned} \tag{19}$$

where $E' = \partial E / \partial z$ and $C_\varepsilon = C_\mu^{3/4} / \kappa$.

These boundary conditions describe a lid driven flow where momentum is injected at the top. This can be realized by means of various different upper boundary boundary conditions: constant velocity, constant momentum flux, constant energy flux and combinations of the three, but in practice it makes little difference. A consequence of lid driven flow is that wind farms get more efficient the shallower the boundary layer is. One might expect a blocking effect, which perhaps would be more realistic for a real boundary layer, but such an effect is dominated by the efficiency gain caused by the 'engine' in the lid moving closer to the rotors. It is a small effect, however, with almost no dependence on z_i. The real atmospheric boundary layer is driven by a pressure gradient and the Coriolis force. They cancel each other at the top, so the flow is anything but lid driven. We could of course include the Coriolis force and model the boundary layer in a more realistic manner, but the Coriolis parameter introduces an additional dimension in the look–up tables (see Section 5.3) which makes everything more complicated. An alternative would be to replace the lid driven flow with

one driven by a pressure gradient without Coriolis force but with a given, fixed boundary layer height. However, this possibility has not been pursued.

4 Linearization

In the section we describe how the equations are linearized and set up the problem in a standard form. In the following we use the simple closure as example. The principles are the same for other closures even if the actual equations may differ. Linearization can mean different things. Sometimes it just means 'throwing away terms until the problem becomes linear with an analytical solution'. Belcher, Jarram and Hunt (2003) divided the flow into three regimes (layers) each with its own set of linear equations. These were then solved analytically and pieces together by the asymptotic matching technique. The present model uses a different strategy. We make a formal perturbation expansion and keep all terms, large or small, and numerical rather than analytical solutions are sought. The advantage is that we do not have to worry about asymptotic matching or how to define the regimes, and it becomes less problematic to treat higher orders. Below we describe the perturbation expansion and phrase the problem in a standard form. We use the simple closure as an example.

4.1 The perturbation expansion

With the simplifications described in the previous section, the momentum equation can be written as

$$u_j \frac{\partial u_i}{\partial x_j} = \frac{\partial}{\partial x_j} \kappa\, u_* z \left\{ \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right\} - \frac{\partial p}{\partial x_i} + \xi\, f_i \tag{20}$$

Note that an extra factor ξ appears on the forcing. The idea is to regard $\boldsymbol{u}$ and p as functions of the parameter ξ. If we set $\xi = 0$ there is no drag, and the solution is given by (13) and a constant pressure e.g. $p^0 = 0$. For finite ξ a perturbation expansion is made using the external drag force as a 'small' term. Formally, this is done by writing all variables as Taylor series in ξ, vis.

$$\begin{aligned} u_i &= u_i^0 + u_i^1\,\xi + u_i^2\,\xi^2 + \ldots \\ p &= p^0 + u_i^1\,\xi + p^2\,\xi^2 + \ldots \\ E &= E^0 + E^1\,\xi + E^2\,\xi^2 + \ldots \\ \varepsilon &= \varepsilon^0 + \varepsilon^1\,\xi + \varepsilon^2\,\xi^2 + \ldots \\ f_i &= f_i^1\,\xi + f_i^2\,\xi^2 + \ldots \end{aligned} \tag{21}$$

When this is inserted into (20) we get an equation for each power ξ^n. For $n = 0$ we get the zeroth order equation which is just (20) with $f_i = 0$. This is a non–linear equation, but not difficult. For $n = 1$ we get the following first order equations

$$\begin{aligned} u^0 \frac{\partial u^1}{\partial x} + w^1 \frac{\partial u^0}{\partial z} &= \frac{\partial}{\partial x_j} K \left(\frac{\partial u^1}{\partial x_j} + \frac{\partial u_j^1}{\partial x} \right) - \frac{\partial p^1}{\partial x} + f_1^1 \\ u^0 \frac{\partial v^1}{\partial x} &= \frac{\partial}{\partial x_j} K \left(\frac{\partial v^1}{\partial x_j} + \frac{\partial u_j^1}{\partial y} \right) - \frac{\partial p^1}{\partial y} \\ u^0 \frac{\partial w^1}{\partial x} &= \frac{\partial}{\partial x_j} K \left(\frac{\partial w^1}{\partial x_j} + \frac{\partial u_j^1}{\partial z} \right) - \frac{\partial p^1}{\partial z} \\ \frac{\partial u^1}{\partial x} + \frac{\partial v^1}{\partial y} + \frac{\partial w^1}{\partial z} &= 0 \end{aligned} \tag{22}$$

K is given by (4) and the fourth equation comes from the continuity equation. It has been assumed that the forcing is longitudinal, i.e. parallel with the incoming flow so that $f_2 = 0$. This corresponds to an ideal situation with no yaw error. It is possible to take a (known or assumed) yaw error into account by considering transverse forcing, where $f_1^1 = 0$ and $f_2^1 \neq 0$, and obtain the appropriate solution by superposition. This is allowed because the equations are linear in the variables (u^1, v^1, w^1, p^1). In the following we concentrate on longitudinal forcing since the treatment of transverse forcing is very similar. We also limit the number of turbines to just one because the solution for a cluster of turbines can be obtained by superimposing the wakes (velocity deficits) calculated for each turbine separately. The drag force is proportional to $C_\mathrm{T}\, U_\mathrm{free}^2$, and, due to linearity, the first order solution can simply be scaled with this factor. We can therefore set the factor equal to 1, solve the equations, and scale the solution later with the correct factor before the wakes are superimposed. We start with the most upwind turbine which is not in the wake of any of the others. The free stream velocity is therefore equal to the velocity at hub height for the approach flow. Then we proceed downwind to the next turbine, which might be in the wake of the first one. If so we calculate U_free (and the corresponding C_T) taking the wake of the first turbine into account, and so forth. In principle the procedure should be iterated on order account for the upstream influence, but in practice convergence is reached after the first iteration.

We will not study higher orders here, but just mention that all the higher order equations are linear in the variables (u^n, v^n, w^n, p^n) and only differ by a source term which is non–linear in the lower order variables. All higher orders can therefore be solved in succession using the same linear solver.

4.2 The mixed-spectral formulation

The governing first order equations equations (22) are simplified considerably in the mixed–spectral formulation. This is effectuated by Fourier transformations in the two horizontal coordinates x and y. In other words we introduce mixed–spectral variables, e.g.

$$\hat{u}(k_1, k_2, z) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \mathrm{d}x \int_{-\infty}^{\infty} \mathrm{d}y \; u(x, y, z)\, \mathrm{e}^{-i(xk_1 + yk_2)} \tag{23}$$

with the inverse transformation

$$u(y, x, z) = \int_{-\infty}^{\infty} \mathrm{d}k_1 \int_{-\infty}^{\infty} \mathrm{d}k_2 \; \hat{u}(k_1, k_2, z)\, \mathrm{e}^{i(xk_1 + yk_2)} \tag{24}$$

The mixed–spectral equations corresponding to (22) read

$$u^0\, ik_1 u^1 + w^1 \frac{\partial u^0}{\partial z} = -\boldsymbol{k}^2\, K\, u^1 + \frac{\partial}{\partial z} K \frac{\partial u^1}{\partial z} + \frac{\partial K}{\partial z}\, ik_1\, w^1 - ik_1\, p^1 + f_1^1 \tag{25}$$

$$u^0\, ik_1\, v^1 = -\boldsymbol{k}^2\, K\, v^1 + \frac{\partial}{\partial z} K \frac{\partial v^1}{\partial z} + \frac{\partial K}{\partial z}\, ik_2\, w^1 - ik_2\, p^1 \tag{26}$$

$$u^0\, ik_1 w^1 = -\boldsymbol{k}^2\, K\, w^1 + \frac{\partial}{\partial z} K \frac{\partial w^1}{\partial z} + \frac{\partial K}{\partial z} \frac{\partial w^1}{\partial z} - \frac{\partial p^1}{\partial z} \tag{27}$$

$$ik_1\, u^1 + ik_2\, v^1 + \frac{\partial w^1}{\partial z} = 0 \tag{28}$$

Where $\boldsymbol{k}^2 = k_1^2 + k_2^2$. We have not bothered to put hats on the variables; u^1 really should have been $\hat{u}^1(\boldsymbol{k}, z)$. Note that most terms can be written as a product of two functions, one that depends on $\boldsymbol{k}$ and one that depends on z. The only exception is the forcing $f_1^1(\boldsymbol{k}, z)$, but we can approximate it to any desired accuracy as a sum of such terms: $f_1^1(\boldsymbol{k}, z) = \sum_n f_{n,\boldsymbol{k}} \phi_n(z)$. The solution can then be obtained as the sum of the responses to each of the terms $f_{n,\boldsymbol{k}}\, \phi_n(z)$, or even better, we can find the response to $\phi_n(z)$ and multiply by $f_{n,\boldsymbol{k}}$ afterwards. The advantage of this procedure is that $f_{n,\boldsymbol{k}}$, which contains the information

of turbine geometry and thrust coefficient, does not have to be known when the equations are solved. The functions $\phi_n(z)$ can be chosen in many ways. We choose a set of chapeau functions of the form

$$\Delta_n(z) = \begin{cases} \frac{z-z_{n-1}}{z_n-z_{n-1}} & \text{for } z_{n-1} < z < z_n \\ \frac{z-z_{n+1}}{z_n-z_{n+1}} & \text{for } z_n < z < z_{n+1} \\ 0 & \text{elsewhere} \end{cases} \tag{29}$$

where $\{z_n\}$ is a set of sufficiently closely spaced levels. We set $z_n = z_0\,\mathrm{e}^{nds}$, where $ds = 0.1$ seems to be a good choice. Note that $\Delta_n(z_m) = 0$ for $n \neq m$ while $\Delta_n(z_n) = 1$. In other words, $\Delta_n(z)$ is a sort of piecewise linear version of a δ function from which we can construct any piecewise linear function with elbows points in $\{z_n\}$.

The big advantage of this formulation is that variables for different wave vectors $\boldsymbol{k}$ do not mix. We therefore only have to solve four coupled ordinary differential equations, belonging to a given $\boldsymbol{k}$, at a time. The mixed–spectral formulation therefore allows calculations to be broken down into relatively simple, independent sub–problems that can be handled by a PC or by a cluster working in parallel. This is in contrast to (22), which involves partial differential equations, and where a representation of the field by values on a grid results in one, very large, set of coupled, algebraic equations.

Some more manipulations are made to (25–28):

1. Use (28) to eliminate $\frac{\partial w^1}{\partial z}$ in (27).
2. Define a new independent variable $s = kz$.
3. Set $k_1 = k\cos\beta$ and $k_2 = k\sin\beta$.
4. Define new dependent variables $\tilde{u} = u^1\,u_*\,k/f_{n,\boldsymbol{k}}$,$\tilde{v} = v^1\,u_*\,k/f_{n,\boldsymbol{k}}$, $\tilde{w} = w^1\,u_*\,k/f_{n,\boldsymbol{k}}$ and $\tilde{q} = p^1\,k/(f_{n,\boldsymbol{k}}s)$
5. Isolate $\frac{\partial^2\tilde{u}}{\partial s^2}$ on the left hand side of (25)
6. Isolate $\frac{\partial^2\tilde{v}}{\partial s^2}$ on the left hand side of (26)
7. Isolate $\frac{\partial\tilde{p}}{\partial s}$ on the left hand side of (27)
8. Isolate $\frac{\partial\tilde{w}}{\partial s}$ on the left hand side of (28)

This leads to the following equations

$$\begin{aligned}
\frac{\partial^2\tilde{u}}{\partial s^2} &= \left(1 + \frac{i\cos\beta\log\frac{s}{kz_0}}{\kappa^2 s}\right)\tilde{u} - \frac{1}{s}\frac{\partial\tilde{u}}{\partial s} + \left(\frac{1}{(\kappa s)^2} - \frac{i\cos\beta}{s}\right)\tilde{w} \\
&\quad + \frac{i\cos\beta}{\kappa}\tilde{q} - \frac{\Delta_n(s)}{\kappa s} && (30) \\
\frac{\partial^2\tilde{v}}{\partial s^2} &= \left(1 + \frac{i\cos\beta\log\frac{s}{kz_0}}{\kappa^2 s}\right)\tilde{v} - \frac{1}{s}\frac{\partial\tilde{v}}{\partial s} - \frac{i\sin\beta}{s}\tilde{w} + \frac{i\sin\beta}{\kappa}\tilde{q} && (31) \\
\frac{\partial\tilde{w}}{\partial s} &= -i\cos\beta\,\tilde{u} - i\sin\beta\,\tilde{v} && (32) \\
\frac{\partial\tilde{q}}{\partial s} &= -\frac{2i\kappa}{s}\cos\beta\,\tilde{u} - i\kappa\cos\beta\frac{\partial\tilde{u}}{\partial s} - \frac{2i\kappa}{s}\sin\beta\,\tilde{v} - i\kappa\sin\beta\frac{\partial\tilde{v}}{\partial s} \\
&\quad - \left(\kappa + \frac{i\cos\beta\,\log\frac{s}{kz_0}}{\kappa s}\right)\tilde{w} - \frac{1}{s}\tilde{q} && (33)
\end{aligned}$$

where $\Delta_n(s)$ is a chapeau function with top point $s_n = k\,z_n$. These equations are well suited for the upper part where $s > 1$, but for small s they are very unstable. We therefore use an alternative formulation for $s < 1$. It is derived in the same way except that $t = \log\frac{z}{z_0}$ is used as independent variable and $\tilde{p} = p^1/(kf_{n,\boldsymbol{k}})$ is used instead of $\tilde{q}$. The result is

$$\frac{\partial^2\tilde{u}}{\partial t^2} = \frac{\mathrm{e}^{2t}k^2z_0^2 + ikz_0t\mathrm{e}^t\cos\beta}{\kappa^2}\tilde{u} + \left(\frac{1}{\kappa^2} - i\mathrm{e}^t kz_0\cos\beta\right)\tilde{w} + \frac{i\mathrm{e}^t kz_0\cos\beta}{\kappa}\tilde{p} - \frac{\mathrm{e}^{2t}\Delta_n}{\kappa} \quad (34)$$

$$\frac{\partial^2\tilde{v}}{\partial t^2} = \frac{\mathrm{e}^{2t}k^2z_0^2 + itkz_0\mathrm{e}^t\cos\beta}{\kappa^2}\tilde{v} - i\mathrm{e}^t kz_0\sin\beta\tilde{w} + \frac{i\mathrm{e}^t kz_0\sin\beta}{\kappa}\tilde{p} \quad (35)$$

$$\frac{\partial\tilde{w}}{\partial t} = -ikz_0\mathrm{e}^t\cos\beta\,\tilde{u} - ikz_0\mathrm{e}^t\sin\beta\,\tilde{v} \quad (36)$$

$$\frac{\partial\tilde{p}}{\partial t} = -2i\mathrm{e}^t kz_0\kappa\cos\beta\,\tilde{u} - i\mathrm{e}^t kz_0\kappa\cos\beta\,\frac{\partial\tilde{u}}{\partial t} - 2i\mathrm{e}^t kz_0\kappa\sin\beta\,\tilde{v}$$
$$-i\mathrm{e}^t kz_0\kappa\sin\beta\,\frac{\partial\tilde{v}}{\partial t} - \frac{\mathrm{e}^{2t}(\kappa\,kz_0)^2 + it\mathrm{e}^t kz_0\kappa\cos\beta}{\kappa}\tilde{w} \quad (37)$$

These equations work much better than (30–33) near the ground, which can be judged from the singular value decomposition. The most extreme singular values are obtained for $kz_0 = 1$, i.e. $s = 1$ or $t = -\log kz_0$, and this point is therefore used to switch between the two formulations.

Note that k and z_0 only appears in the combination kz_0. We can therefore solve the equations without knowing k and z_0 separately. Apart from the independent variable (s or t), the solution only depends on four parameters: n, β, kz_0 and kz_i. In the next section we show that kz_i can be eliminated.

5 Numerical solution method

During his PhD study Corbett (2007) discovered that the linearized equations can be hard to solve. He was solving a linearized model for flow in complex terrain, but linearized wake models present the same difficulties. One of the difficulties is that it is a *two–point problem* with boundary conditions at both ends. *Initial value* problems are usually more benign and can be solved using a standard method such as Runge–Kutta integration. Corbett observed that the problem tends to become numerically ill–posed for small values of kz_0 where extremely high working precision had to be used. In some cases, which can be conceived to be encountered in practice, a working precision of as much as 50 decimals is needed, which is more than most conventional programming languages can deal with. Mathematica can, but at at a very high cost in terms of computation time. Corbett (2007), see also Corbett, Ott and Landberg (2008), used a shooting technique where solutions obtained by integration from below and from above were matched at a suitably chosen midpoint. Small values of kz_0 are relevant for offshore wind farms because wakes are long and the roughness at sea is small, hence there was a need for a more powerful numerical method. All existing methods we have tried fail, including the state–of–the–art algorithms of Mathematica, and it therefore was necessary to invent a new method, which will be described in some detail in this section.

5.1 Standard form

Note that $\tilde{u}$, $\partial\tilde{u}/\partial s$, $\tilde{v}$, $\partial\tilde{v}/\partial s$, $\tilde{w}$ and $\tilde{q}$ (or $\tilde{p}$) appear on the right hand sides of (30–33) or (34–37). The equations are second order in $\tilde{u}$ and $\tilde{v}$ and first order in $\tilde{w}$ and $\tilde{q}$ (or $\tilde{p}$). First order equations are more practical so we define two new variables $\tilde{u}'$ and $\tilde{v}'$ and two more equations

$$\frac{\partial\tilde{u}}{\partial s} = \tilde{u}'$$

$$\frac{\partial\tilde{v}}{\partial s} = \tilde{v}'$$

Defining $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5, x_6) = (\tilde{u}, \tilde{u}', \tilde{v}, \tilde{v}', \tilde{w}, \tilde{q}$ or $\tilde{p})$, the linearized equations (30–33) or (34–37) can then be cast in the general form

$$\frac{\mathrm{d}x_i}{\mathrm{d}s} = A_{ij}\,x_j + g_i \tag{38}$$

where both A_{ij} and g_i are functions of s. In addition there are boundary conditions at $s = s_a$ (the ground at $z = z_0$) and $s = s_b$ (the upper boundary at $z = z_i$).

Equation (38) is inhomogeneous, which is impractical, so we use a trick to turn it into a homogeneous problem. First rewrite (38) in the form

$$\frac{\mathrm{d}x_i}{\mathrm{d}s} = g_i\,x_0 + A_{ij}\,x_j \tag{39}$$

which is the same as (38) provided that $x_0 = 1$, which in turn can be expressed by means of the differential equation with the solution $x_0 = 1$:

$$\frac{\mathrm{d}x_0}{\mathrm{d}s} = 0 \tag{40}$$

with the boundary condition

$$x_0(s_0) = 1 \tag{41}$$

In (38) j runs from 1 to n, say, but we can include x_0 as a variable and write (39) and (40) as

$$\frac{\mathrm{d}x_i}{\mathrm{d}s} = A_{ij}\,x_j \tag{42}$$

where we i and j now run from 0 to n. This only requires that we enlarge A_{ij} by defining an extra (upper) row and an extra (left) column

$$\begin{aligned} A_{0j} &= 0 \qquad j = 0 \ldots n \\ A_{i0} &= g_i \qquad i = 1 \ldots n \end{aligned} \tag{43}$$

Using shorthand vector/matrix notation we can write (42) as

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}s} = \boldsymbol{A}\boldsymbol{x} \tag{44}$$

The vector valued function $\boldsymbol{x}(s) = (x_0(s), \ldots, x_n(s))$ can be thought of as a map from the interval $[s_a, s_b]$ into the complex linear space $\mathbb{C}^{n+1}$. Here we define the usual inner product

$$\langle \boldsymbol{x} | \boldsymbol{y} \rangle \equiv \boldsymbol{x}^\dagger \boldsymbol{y} = x_i^*\,y_i \tag{45}$$

where the dagger (†) denotes the conjugate transpose and the asterisk denotes complex conjugation. A boundary conditions at $s = s_a$, say, can then be expressed as

$$\langle \boldsymbol{b} | \boldsymbol{x}(s_a) \rangle = c \tag{46}$$

where $c = 0$ for all the original boundary conditions while $c = 1$ for the 'extra' boundary condition $x_0(s_a) = 1$.

5.2 Linear chasing

Consider now the *dual* problem[4]

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}s} = -\boldsymbol{A}^\dagger \boldsymbol{y} \tag{47}$$

where $\boldsymbol{A}^\dagger$ is the Hermitian adjoint of $\boldsymbol{A}$, i.e $A^\dagger_{ij} = A^*_{ji}$. The dual problem is interesting because solutions to it can be used to move boundary conditions. Suppose that $\boldsymbol{y}$ is a solution to (47) with the $n+1$ boundary conditions $y(s_a) = c$ and $\boldsymbol{x}$ is a solution to (44) with (46) as one of its boundary conditions. Then we have

$$\frac{\mathrm{d}\langle \boldsymbol{y} | \boldsymbol{x} \rangle}{\mathrm{d}s} = \langle -\boldsymbol{A}^\dagger \boldsymbol{y} | \boldsymbol{x} \rangle + \langle \boldsymbol{y} | \boldsymbol{A}\boldsymbol{x} \rangle = \langle \boldsymbol{y} | - \boldsymbol{A}\boldsymbol{x} \rangle + \langle \boldsymbol{y} | \boldsymbol{A}\boldsymbol{x} \rangle = 0 \tag{48}$$

[4]The term *auxiliary* problem is often used, but we prefer *dual*.

and it follows that

$$\langle \boldsymbol{y}(s) | \boldsymbol{x}(s) \rangle = c \tag{49}$$

holds for all value of s. Setting $s = s_b$ we therefore get a boundary condition on $\boldsymbol{x}(s_b)$ equivalent to the original boundary condition (46) at s_a. Note that solving (47) is relatively easy because it is an initial value problem (i.e. all boundary conditions are at the same point). Moving all boundary conditions to s_b we end up with an *initial value* problem for $\boldsymbol{x}$ starting from s_b. This is the linear chasing method, which is the default method for solving linear two–point problems with NDSolve in Mathematica. A non–linear method similar to a method proposed by Gel'fand and Lokutsiyevskii in an unpublished paper is also offered. Unfortunately the documentation of the method found in the Mathematics help system does not allow us to figure out exactly what NDSolve does when this method is enforced.

Linear chasing works well in many cases, but not always. After moving all boundary conditions to s_b we need to solve a set of linear equations in order to determine the initial value $\boldsymbol{x}(s_b)$, and that may turn out to be numerically impossible. This can happen if two or more boundary conditions at s_a effectively translate into the same boundary condition at s_b and extraordinarily high working precision is required to distinguish the small differences. Even if an initial value $\boldsymbol{x}(s_b)$ can be determined the numerical solution may end up violating the boundary conditions at s_a after integrating down from s_b. This is due to the accumulation of numerical inaccuracies which tends to wipe out the 'memory' of the lower boundary conditions. Solutions to (44) may increase or decrease fast for $s \to \infty$ and we are generally interested solutions that decrease when approaching a boundary. However, when the equations are integrated upwards from s_a small numerical errors tend to excite the increasing the solutions which soon begin to dominate. We can therefore end in a situation where different boundary conditions effectively translate into only one condition because we do not have enough working precision to distinguish them. Integrating from above will excite the decreasing solutions, which is more healthy, at least for as long as s is large. But near the lower boundary the solutions once again tend to explode, and it becomes difficult to get the lower boundary conditions right. The effect of the upper boundary conditions just drown in numerical noise, and the solution seriously violate the original lower boundary conditions.

5.3 Orthogonal chasing

In section 5.2 we saw how a dual equation can be used move boundary conditions from one boundary to another. We can also use them to move boundary conditions to positions in between. If all boundary conditions are moved to the same point s we have enough conditions to determine $\boldsymbol{x}(s)$. So the idea is to determine $\boldsymbol{x}(s)$ from the conditions imposed by the dual equations. This is better than solving the original equation, as in the linear chasing methods, because the boundary conditions are enforced directly and not via 'memory' kept through long integrations.

It is most practical to work with inhomogeneous equations, hence we drop the trick with the extra variable $x_0(s)$ and go back to (38)

$$\frac{\mathrm{d}x_i}{\mathrm{d}s} = A_{ij}\, x_j + g_i \tag{50}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$.

We define dual variables $\boldsymbol{y}^p(s) = (y_1^p(s), y_2^p(s), \ldots, y_n^p(s)$ where $p = 1, 2, \ldots, n$ and impose the following equation on them:

$$\frac{\mathrm{d}\boldsymbol{y}^p}{\mathrm{d}s} = -\boldsymbol{A}^\dagger \boldsymbol{y}^p \tag{51}$$

Defining the square matrix

$$Y_{ij} \equiv y_i^j \tag{52}$$

we may also write (51) as

$$\frac{\mathrm{d}}{\mathrm{d}s}\boldsymbol{Y} = -\boldsymbol{A}^\dagger \boldsymbol{Y} \tag{53}$$

For each p, $\boldsymbol{y}^p(s)$ is used to specify a (boundary) condition on $\boldsymbol{x}$:

$$\langle \boldsymbol{y}^p(s)|\boldsymbol{x}(s)\rangle = c_p(s) \tag{54}$$

At the lower boundary $s = s_a$ the first m conditions (for $p \leq m$) should coincide with the lower boundary conditions. Thus the boundary condition $u(z_0) = 0$ can be written as $x_1(s_a) = 0$ corresponding to $\boldsymbol{y}(s_a) = (1, 0, 0, 0, 0, 0)$ and $c(s_a) = 0$. Without loss of generality we can assume orthogonality:

$$\langle \boldsymbol{y}^p(s_a)|\boldsymbol{y}^q(s_a)\rangle = \delta_{pq} \tag{55}$$

For $m < p \leq n$ it would seem natural to choose initial values at s_b so that $\boldsymbol{y}^p(s_b)$ is given by the upper boundary conditions, and integrate down from s_b to s_a. This would ensure n conditions at every s from which $\boldsymbol{x}(s)$ could be determined. However, in practice the n conditions tend to degenerate and sometimes a numerical solution cannot be obtained. Instead we choose some lower boundary conditions with $c_p(s_a) = 0$ for $p > m$ so that the orthogonality condition (55) extends to the whole set of $\boldsymbol{y}$s. In other words, $Y_{ij}(s_a)$ is a unitary matrix. With these initial conditions we can integrate (53) upwards and determine $\boldsymbol{Y}(s)$ up to a value of s which is larger than any s_b we can think of. Note that we do not have to know s_b or the form of the upper boundary conditions or the functional form of the forcing $\boldsymbol{f}$ (which determines $\boldsymbol{g}$) to obtain $\boldsymbol{Y}$. This makes $\boldsymbol{Y}(s)$ very versatile because it only depends only on the two parameters $s_a = kz_0$ and β while it is independent of s_b and the wind turbine type which dictates $\boldsymbol{f}$. More calculations need to be done before we can determine a specific $\boldsymbol{x}$, but these calculation are relatively fast compared to calculating $\boldsymbol{Y}$, and therefore it is advantageous to store $\boldsymbol{Y}$ as a look–up–table (LUT). Actually $\boldsymbol{x}(s)$ is used to construct the wake of a solitary turbine which is also used as a look–up–table. The tables containing $\boldsymbol{Y}$ are therefore very general look–up tables from which more specific look-up–tables can be constructed. We will refer to the $\boldsymbol{Y}$ tables as preLUTs - preliminary look–up tables. The preLUTs are so general that they can be made once and for all at installation.

Once $\boldsymbol{Y}(s)$ is known we find $\boldsymbol{x}(s)$ by first finding $\boldsymbol{c}$ and then solving (54), which can also be written as

$$\boldsymbol{Y}^\dagger(s)\boldsymbol{x}(s) = \boldsymbol{c}(s) \tag{56}$$

The equation for $\boldsymbol{c}$ is simply

$$\frac{\mathrm{d}c_p}{\mathrm{d}s} = \langle \boldsymbol{y}^p|\boldsymbol{g}\rangle \tag{57}$$

For a given turbine type the right hand side is known and the integration is fast and easy. We need an initial value which we only have for $p \leq m$ where $c_p(s_a)$ is dictated by the lower boundary conditions. We therefore integrate these equations from s_a to s_b to get $\boldsymbol{y}^p(s)$ for $p \leq m$. When we arrive at s_b we have m conditions in addition to the $n - m$ upper boundary conditions, which is enough to determine $\boldsymbol{x}(s_b)$ and subsequently calculate $\boldsymbol{c}(s_b) = \boldsymbol{Y}^\dagger(s_b)\boldsymbol{x}(s_b)$. Knowing $c_p(s_b)$ we can then integrate the remaining equations for $p > m$ down from s_b to s_a. This yields $\boldsymbol{c}(s)$ and we finally solve (56) to get $\boldsymbol{x}(s)$.

The algorithm outlined above illustrates the main principles, most notably the use of preLUTs which only depend on kz_0 and β, but some additional tricks are necessary to eliminate the problem with rapidly growing modes so that the matrix $\boldsymbol{Y}$ becomes (almost) singular. To avoid this we use the set $\{s_j\}$ as 'stations' to divide the interval of integration $[s_a, s_b]$ into sub–intervals $[s_j, s_{j+1}]$. The same set of points was used to define chapeau functions for the forcing term. The idea is to avoid that $\boldsymbol{y}$s grow too much from one station to the next. At each station we then make a Gram–Schmidt orthonormalization and restart the integration with $\boldsymbol{y}$s that can be clearly distinguished numerically. Formally this is achieved by a so called QR decomposition

$$\boldsymbol{Y} = \hat{\boldsymbol{Y}}\,\boldsymbol{T}^\dagger \tag{58}$$

where $\hat{\boldsymbol{Y}}$ is a unitary matrix and $\boldsymbol{T}^\dagger$ is an upper triangular matrix. The integration is restarted with the value $\boldsymbol{Y} = \hat{\boldsymbol{Y}}$ at each station. The advantage of changing to new $\boldsymbol{y}$s every

now and then is that we avoid that the $\boldsymbol{y}$s all take off in the direction of the fastest growing mode. Because $\boldsymbol{T}^\dagger$ is upper triangular, the first jth column of $\boldsymbol{Y}$ span the same subspace as the first j columns of $\hat{\boldsymbol{Y}}$. Thus $\boldsymbol{y}^1$ is just normalized at each station and will tend to grow in the direction of the fastest growing mode. When the rest of the $\boldsymbol{y}$s are forced to be perpendicular to $\boldsymbol{y}^1$ this tendency is reduced for them. Note also that we can still solve $\boldsymbol{y}^p$ for $p = 1, \ldots, m$ separately because they never mix with the remaining $\boldsymbol{y}^p$ with $p > m$.

The condition $\boldsymbol{Y}^\dagger \boldsymbol{x} = \boldsymbol{c}$ translates into the condition

$$\hat{\boldsymbol{Y}}^\dagger \boldsymbol{x} = \hat{\boldsymbol{c}} \tag{59}$$

where

$$\hat{\boldsymbol{c}} = \boldsymbol{T}^{-1}\, \boldsymbol{c} \tag{60}$$

Note that because $\hat{\boldsymbol{Y}}^\dagger$ is unitary the solution to (59) is straight forward

$$\boldsymbol{x} = \hat{\boldsymbol{Y}}\hat{\boldsymbol{c}} \tag{61}$$

The preLUTs contain $\hat{\boldsymbol{Y}}(s_j)$, $\boldsymbol{T}^{-1}(s_j)$ and $\boldsymbol{T}(s_j)$ for every station s_j. In addition there are results from the integration of (57) with four different forcings: either constant or proportional to s and either longitudinal or transversal. This allows us to model any forcing which is a chapeau function with elbow points at three consecutive stations. Furthermore, in our case the boundary conditions are homogeneous in the sense that $\boldsymbol{c}(s_a) = 0$ and $\boldsymbol{c}(s_b) = 0$. If s_j is the 'central station' (i.e. $f(s_j) \neq 0$) we have $c^p(s) = 0$ for $s < s_{j-1}$ allowing us to start the integration at s_{j-1}. Integrating up to $s = s_j$ gives the response to half as chapeau function. The second half is obtained from the integrations on $[s_j, s_{j+1}]$. No more integration is needed because $c^p(s)$ remains constant for $s > s_{j+1}$. It is therefore sufficient to find the results of the integrations just from one station to the next, and store them in the preLUTs. The preLUTs therefore contain three matrices and four vectors for each station and further are indexed by the two parameters β and kz_0.

We mentioned that it is advantageous to use a logarithmic coordinate for $s < 1$. The procedure is therefore slightly modified and some care should be taken when crossing $s = 1$. This is straight forward and we will spare the reader from further details.

5.4 Numerical integration scheme

Explicit Runge–Kutta methods with built in step size control were used for the numerical integration of (53) and (56). At first the fifth–order Runge–Kutta–Fehlberg (Cash and Karp 1990) used by Corbett et al. (2008) was tried, but found to be rather slow. A simpler third–order method described in Press, Teukolsky, Vetterling and Flannery (1992) turned out to be much faster for the same accuracy. It therefore seems as if slow convergence of the Taylor series prevents us from taking advantage of high order methods. This suggests that the Burlirsh–Stoer method (Press et al. 1992), which is based on a clever extrapolation of results from a low order scheme, could be efficient for this problem, but it was not tried since the third order scheme already worked satisfactory.

5.5 FFIT

A conventional Fast Fourier Transform (FFT) can be used to construct the wind and pressure fields from its Fourier components. FFT is a discrete transform resulting in periodic fields. The calculated wake is therefore the response to an infinite, periodic array of turbines. We want the response from a single, solitary turbine which can be approximated if we make the periodic domain large enough. In practice this means extremely large because the drag from the turbine should be negligible compared to the friction of the surface of the domain, and water is very smooth. It would be better to use a Fourier Integral Transforms, which can be regarded as the limit as the domain size goes to infinity. The problem is that we need to do a numerical Fourier integral for each point where we want output and that takes very

long time. Fortunately there is a technique where all the numerical Fourier integrals can be made in one batch which makes use of the FFT. The result is the fast Fourier integral transform (FFIT) which we now describe in more detail. For simplicity we will restrict the explanations to the 1D case which can easily be generalized to the 2D case we are interested in here. Thus the problem is to find

$$f(x) = \int_{\infty}^{\infty} \hat{f}(k)\, \mathrm{e}^{ikx}\, dx \tag{62}$$

for a representative set of x–values. Numerical Recipes (Press et al. 1992) offers a method which is intended for cases where $\hat{f}$ is rather smooth and has compact support (so that the integration goes over a finite interval). We can get compact support if we restrict the calculation to the set of points $x_j = x_0 + a\, j$, where $j = 0, 1, \ldots, n-1$ and x_0 is an offset. Then we can rewrite (62) as

$$f(x_j) = \int_{0}^{2\pi/a} \hat{F}(k)\, \mathrm{e}^{ikx_j}\, dk \tag{63}$$

where

$$\hat{F}(k) = \sum_{p=-\infty}^{\infty} \hat{f}(k + 2p\pi/a) \tag{64}$$

where it in practice is sufficient to sum a finite number of terms.

We evaluate the integral as a sum using the rectangular rule. Thus we choose n equally spaced points on the k–axis $k_0, k_1, \ldots, k_{n-1}$ with $k_q = k_0 + b\, q$ and evaluate $f(x)$ as

$$f(x) \approx b \sum_{q=0}^{n-1} \hat{F}(k_q)\, \mathrm{e}^{ik_q x} \tag{65}$$

For $x_j = x_0 + a\, j$ we can write (65) as

$$f(x_j) \approx b\, \mathrm{e}^{ik_0 aj} \sum_{q=0}^{n-1} \left\{ \hat{F}(k_q)\, \mathrm{e}^{ik_q x_0} \right\} \mathrm{e}^{iabqj} \tag{66}$$

Taking a look at the sum we note it is of the form

$$\sum_{q=0}^{n-1} g_q\, \mathrm{e}^{iabqj} \tag{67}$$

This in fact is a discrete Fourier transform for certain choices of a, b and n, namely those where

$$ab = 2\pi/n \tag{68}$$

Assuming this restriction holds we can use FFT to evaluate *all* the integrals in one go and it appears that we have a fast method. We should of course make sure that we cover a decent range of k–values and the spacing b should also be small enough to ensure an accurate evaluation of the integral. Because of (68) the only way to get b small enough is to make na large and that is the main problem with the method. Experience shows that increasing n for fixed a has little effect, hence a good resolution on the on the k–axis is obtained at the expense of a poor resolution on the x–axis and visa versa.

According to Numerical Recipes (65) is a really lousy approximation when na is not large enough. A better approximation can be made if it can be assumed that $\hat{F}$ is slowly varying so that it can be approximated by a piecewise linear function (or more generally a higher order spline). In that case we can write $\hat{F}$ as

$$\hat{F}(k) \approx \sum_{q} \hat{F}(k_q)\, \phi(k - k_q) \tag{69}$$

where ϕ is a triangular *chapeau* functions: $\phi(k) = 1 - |k/b|$ for $|k| < b$ and zero for $|k| > b$. The point is that with this approximation we can do (63) analytically:

$$f(x_j) \approx \sum_{q=0}^{n} \hat{F}(k_q) \int_0^{2\pi/a} \phi(k - k_q)\, \mathrm{e}^{ikx_j}\, dk = bW(bx_j) \sum_{q=0}^{n-1} \hat{F}(k_q)\, \mathrm{e}^{ikx_0}\, \mathrm{e}^{i2\pi qj/n} \tag{70}$$

Here we notice that the sum is a Fourier sum that can be done with FFT. The so called attenuation factor W is given by

$$W(s) = \frac{2(1 - \cos s)}{s^2} \tag{71}$$

Our experience with attenuation factors is that they do not cure the main problem, which arises from the reciprocal relation (68) between a and b. Fortunately, there exists a method that eliminates this restriction. It is based on a trick due to Bailey and Swarztrauber (1991), see also Bailey and Swarztrauber (1994) and Inverarity (2002). First note that qj can be written as

$$qj = \frac{1}{2}(q^2 + j^2 - (q - j)^2) \tag{72}$$

so that (66) can be written as

$$f(x_j) \approx b\, \mathrm{e}^{ik_0 aj + \frac{1}{2}iabj^2} \sum_{q=0}^{m-1} \left\{ \hat{f}(k_q)\, \mathrm{e}^{ik_q x_0 + \frac{1}{2}iabq^2} \right\} \mathrm{e}^{-\frac{1}{2}iab(j-q)^2} \tag{73}$$

The sum looks like a circular convolution of two vectors of length $2n$.

$$\sum_{q=0}^{2n-1} g_q\, h_{j-q} \tag{74}$$

It *is* in fact a convolution if we define

$$g_q = \begin{cases} \hat{f}(k_q)\, \mathrm{e}^{ik_q x_0 + \frac{1}{2}iabq^2} & \text{for } 0 \le q < n \\ 0 & \text{for } n \le q < 2n \end{cases} \tag{75}$$

and

$$h_j = \begin{cases} \mathrm{e}^{-\frac{1}{2}iabj^2} & \text{for } 0 \le j < n \\ \mathrm{e}^{-\frac{1}{2}iab(2n-j)^2} & \text{for } n \le j < 2n \end{cases} \tag{76}$$

With the sum written in this form we can make use of FFT to do the convolution all in one go. Simply apply FFT to g and h separately, multiply and take the inverse transform. We need $2n$ points instead of n, but that's a reasonably small price to pay for the freedom to choose a and b independently.

It should be emphasized that FFIT is an approximation and that there are a number of choices to make. Approximating $\hat{F}$ with piecewise cubic functions (instead of piecewise linear) leads to a different attenuation function. More fancy integration schemes have also been suggested, e.g. Simpson's rule (Simonen and Olkkonen 1985). FFIT can of course be used for the inverse transformation but unlike FFT we do not return to where we started when we transform back and forth. Worst of all is the lack of error estimates which means that we have little guidance in choosing a, b and n. Firmer FFIT theory is badly needed.

5.6 Code verification and accuracy issues

The numerical solver was implemented in two different programs, Preludium and Trafalgar, and a graphical user interface called Fuga was also made. Fuga takes input from the user and from WAsP files containing wind turbine data and wind farm layouts and controls the execution of Preludium and Trafalgar. Fuga also calculates the wake from a whole wind farm on the basis of a solitary turbine wake. The solitary wake is made by the C program Trafalgar, which performs a FFIT on LUT data. The LUTs, containing $\boldsymbol{x}(z_j)$ for a range of wave vectors $\boldsymbol{k}$ are made by the Fortran program Preludium which can also make preLUTs.

The complexity of the calculations and the fact that intermediate results can only be checked at a late stage in a long series of manipulations makes debugging very hard. The situation changed when an analytical solution was found for a special case. It is for the LUT with $\beta = \pi/2$. It represents the response to a vertically chapeau shaped and horizontally harmonic forcing and the analytical solutions is for the $u_1(s_c)$ where s_c is the level where the chapeau function has its maximum. For $\beta = \pi/2$ the problem reduces to a solvable Sturm–Liouville problem. When the forcing is a chapeau function with elbows at s_{c-1}, s_c and s_{c+1} we find

$$
\begin{aligned}
u(s_c) &= \frac{K_0(s_c)I_0(s_b) - I_0(s_c)K_0(s_b)}{2(s_{c-1} - s_c)(K_0(s_a)I_0(s_b) - I_0(s_a)K_0(s_b))} \\
&\times \big(I_0(s_a)\big(-\pi s_{c-1}^2 \mathrm{L}_{-1}(s_{c-1})K_0(s_{c-1}) + \pi s_{c-1}s_c \mathrm{L}_{-1}(s_c)K_0(s_c) + s_c K_1(s_c)(\pi s_{c-1}\mathrm{L}_0(s_c) + 2) \\
&\quad -s_{c-1}(\pi s_{c-1}\mathrm{L}_0(s_{c-1}) + 2)K_1(s_{c-1})) \\
&\quad +K_0(s_a)\big(s_{c-1}^2(\pi \mathrm{L}_1(s_{c-1}) + 2)I_0(s_{c-1}) - s_{c-1}s_c(\pi \mathrm{L}_1(s_c) + 2)I_0(s_c) + s_c I_1(s_c)(\pi s_{c-1}\mathrm{L}_0(s_c) + 2)\big) \\
&\quad -s_{c-1}K_0(s_a)(\pi s_{c-1}\mathrm{L}_0(s_{c-1}) + 2)I_1(s_{c-1})) \\
&+ \frac{K_0(s_a)I_0(s_c) - I_0(s_a)K_0(s_c)}{2(s_{c+1} - s_c)(K_0(s_a)I_0(s_b) - I_0(s_a)K_0(s_b))} \\
&\times \big(I_0(s_b)\big(-\pi s_{c+1}^2 \mathrm{L}_{-1}(s_{c+1})K_0(s_{c+1}) + \pi s_{c+1}s_c \mathrm{L}_{-1}(s_c)K_0(s_c) + s_c K_1(s_c)(\pi s_{c+1}\mathrm{L}_0(s_c) + 2) \\
&\quad -s_{c+1}(\pi s_{c+1}\mathrm{L}_0(s_{c+1}) + 2)K_1(s_{c+1})) \\
&\quad +K_0(s_b)\big(s_{c+1}^2(\pi \mathrm{L}_1(s_{c+1}) + 2)I_0(s_{c+1}) - s_{c+1}s_c(\pi \mathrm{L}_1(s_c) + 2)I_0(s_c) + s_c I_1(s_c)(\pi s_{c+1}\mathrm{L}_0(s_c) + 2)\big) \\
&\quad -s_{c+1}(\pi s_{c+1}\mathrm{L}_0(s_{c+1}) + 2)I_1(s_{c+1})K_0(s_b)) \qquad (77)
\end{aligned}
$$

where I_n and K_n are modified Bessel functions of the first and second kind, respectively and L_n is a modified Struve function. An analytical expression for $u(s)$ with $c \neq s_c$ exists, but it is too complicated to write down here. The solution was found with the aid of Mathematica, and Mathematica can also be used to evaluate the expression. This is a bit tricky because some of the terms can get very large while almost cancel each other. Mathematica therefore sometimes give up due to lack of precision, but it was possible to get representative values to compare with numerical values generated from preLUTs. The comparison of numerical and analytical results is very satisfactory indeed, not worse than a relative accuracy of 10^{-6}, i.e. 6 significant digits. This is much better than expected from the conservative error estimates used for error control. The agreement is so substantial that it must be safe to declare major parts of the code bug free. We note that a similar analytical solution can be found for the mixing length closure and a similar good accuracy of the numerical solution is found. We did not manage to find analytical solutions for the $E - \varepsilon$ closure.

Figure 5 shows a 3D 'blanket plot' of $\mathrm{Re}\, u^1(s_a)$. The analytical solution is the intersection between the blanket and the right, vertical face. The curve on the left, vertical face was obtained from a (very long) analytical expression representing the limit $kz \to 0$. The fact that the numerical solution connects smoothly to this theoretical limit is a further check of the correctness of the numerical solution.

Although there are strong indications that the look–up tables are accurate, the accuracy of the final result also depends other factors as well. Inaccuracies sneak in when we make interpolations in the tables and when the FFIT is applied. The interpolation errors are probably most severe. This could be the cause of the tiny, but spurious, oscillations seen in the calculated velocity profiles. Perhaps the simple bi–linear interpolation scheme used by the Trafalgar module in Fuga should be replaced with something better.

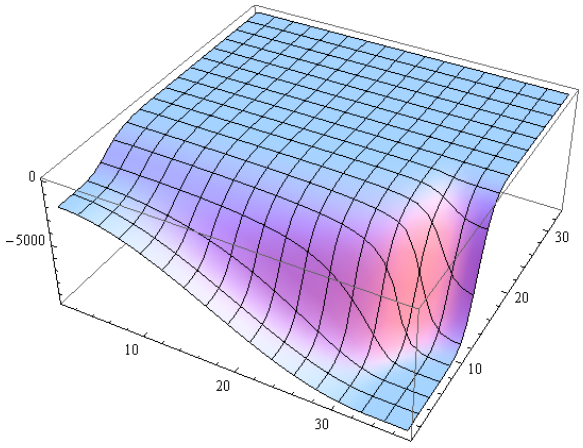


Figure 5. Numerical results for $\mathrm{Re}\, u^1(s_c)$*. The text explains further.*

6 Validation against data

In this section we present validations of the model (Fuga version 1.3) against data using three different datasets. Two of the sets are production data from the offshore wind farms Horns Rev I and Nysted and a third contains velocity measurements in the wake of an onshore turbine placed in homogeneous terrain.

The offshore data were extracted from the database generated within the UpWind project (Hansen, Barthelmie, Cabezon and Politis 2008, Barthelmie, Frandsen, Rathmann, Hansen, Politis, Prospathopoulos, Schepers, Rados, Cabezn, Schlecz, Neubert and Heath 2011). They have also been used within the TOPFARM project (Hansen 2009). The data consist of production data for all turbines in the park averaged over a 1m/s wide free wind speed interval in five degree wind direction bins. The wind direction bins were centrered around a 'critical' direction, where the wind is parallel with the rows. This happens for direction 278° at Nysted and 270° at Horns Rev I. The extracted data set is based on raw data measured up to November 2009, thus including more raw data than e.g. Barthelmie, Rathmann, Frandsen, Hansen, Politis, Prospathopoulos, Rados, Cabezon, Schlez, Phillips, Neubert, Schepers and van der Piljl (2007). For each turbine care has been taken only to select periods where all relevant upwind turbines are producing. Extra selection criteria were imposed requiring the preceding 10 minute period to fall within the same wind speed and wind direction bins (Hansen 2009). This was done in order to filter out data for situations where the wind was changing too much in time or space.

We need the surface roughness z_0 and the inversion (lid) height z_i as input to the model. For the offshore cases $z_0 = 0.1$mm was used. The value is based on (15) using $A_{\mathrm{Ch}} = 0.012$ as determined by Peña and Gryning (2008) for Horns Rev 1 met mast data. Via (16) this corresponds to Ti=7.4%. The same value was used for Nysted, but $z_0 = 0.2$mm could also be justified. In all cases the inversion height was set to $z_i = 400$m. This is an average offshore value, but choosing $z_i = 1000$m does not change the results very much and would not lead to any changes of conclusions.

Very narrow wind directions bins, only two degrees wide, have been used for production data, e.g. Barthelmie et al. (2007) and Frandsen et al. (2009). The intension has been to use the multiple wake situation as a model test case. There are several problems with this. Firstly, the experimental data series is not stationary, hence it could be relevant to represent a ten minutes series as a mixture of realizations with different mean values. If so the only way to compare a model with data is to make several runs for a range of mean directions and make an appropriate, weighed average of the results. A theory that could tell us how to make such an average is unfortunately lacking. Secondly, we feed the models with a horizontally

homogeneous free wind, but in reality conditions are not strictly homogeneous over a 5–6 km wide wind park. The stationarity selection criterion is an attempt to compensate for this. We don't know how successful this compensation is, but it seems unlikely that it can guarantee uniformity to within two degrees. Finally, for Horns Rev the wind direction was taken from the yaw angle of one of the western turbines. This was done because the wind vane at met mast M2 was not working while the vanes on M6 and M7 were disqualified because they were located in the wake. It is therefore possible that narrow wind direction bins correspond to fairly wide model input bins, and that the experimental data contain very few multiple wake situations. Due to these uncertainties we will only use wide bins covering 15, 25 and 35 degrees. This should make a more fair comparison because relatively fewer wind directions are 'misplaced' in the wide bins. The averages were taken from model results with wind directions in one degree steps, which is close enough to represent a continuum.

Inhomogeneity seems to be a worse problem for Nysted than for Horns Rev. Nysted is located about 5 km to the South of Lolland and with more islands further to the East. In westerly winds it is characteristic that the turbines generally produce less energy the nearer they are to the cost. The wind speed bins were selected according to the production of one of the turbine in the front row. This turbine therefore served as anemometer. It might have been better to estimate free stream velocities in a way that takes spatial variations across the farm into account, but that is easy to say in hindsight. Horns Rev I is located about 12km to the West of Jutland and with the whole North Sea acting as fetch for westerly winds. This should result in more homogeneous flow field and a much better agreement between productions from turbines in the first, undisturbed row is in fact seen. This might explain a somewhat better model performance for Horns Rev I than for Nysted.

Three linearized models were tested. They are based on 1) the simple closure, 2) the mixing length closure and 3) the E–ε closure. The following set of model constants for the E–ε model were suggested by Gribben (2010)

$$C_\mu = 0.0293 \quad \sigma_E = 1 \quad \sigma_\varepsilon = 0.8 \quad C_{\varepsilon 1} = 1.44 \quad C_{\varepsilon 2} = 2.67 \tag{78}$$

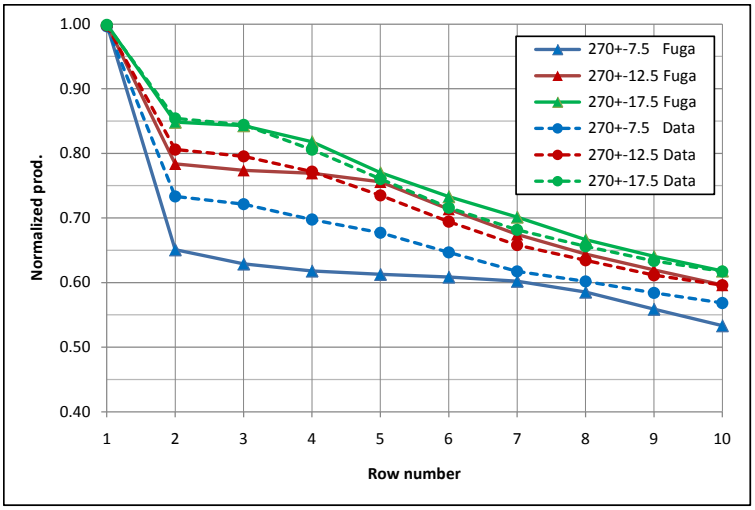
Figure 6a–c shows that the three closures give quite different results. The results for the simple closure are very close to the measured productions for the two widest bins, while they fall below the measurements for the 15 degree bin. It could be a bin size effect, but it could also be a genuine flaw of the model. The mixing length closure is less successful, consistently over–predicting the production. The E–ε closure consistently under–predicts the production. The same trends can be seen in the comparisons with data from the Nysted wind farm in figure 7. The simple closure again performs best, and this time the results for the 15 degree bin look better. The mixing length closure again consistently over–predicts the production. The E–ε closure reproduces the measured energy productional in the second row almost exactly, but it vastly under—predicts the energy production in the downwind parts of the farm.

A comparison was also made with wake data for the Nibe B turbine published by Taylor (1990). The turbine is placed onshore 1km to the East of a cost line extending N–S. Data were taken for southerly winds where the upwind fetch is flat, rural terrain. The case is interesting because of the larger roughness. According to the data report it is as large as 10cm, while Hassan, Taylor and Garrad (1988) has z_0=4cm. Both estimates appear to be based on fits to velocity profile data. The reported turbulence intensity around 10–15% would suggest a lower value around z_0=1cm. For wake calculations it is more important to reproduce the correct turbulence level than the correct velocity profile, because the decay of the wake is governed by turbulent mixing. We have therefore chosen z_0=1cm.

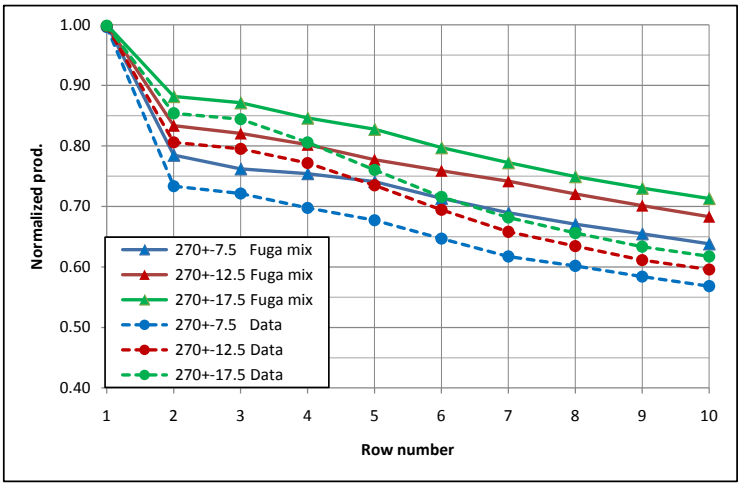
Figure 8a–c shows the comparison. It appears that none of the closures capture the near wake at $x = 2.5D$. We cannot expect linearized models to do that. However, at the two longer distances $x = 4D$ and $x = 7.5D$ the results tend to be more reasonable. All the models under–predict the deficit, but the simple model makes the best performance, both with respect to the width and the depth of the wake. The mixing length closure predicts practically the same (slightly too narrow) width as the simple model, but the wake is more

shallow than the data. Finally the E–ε closure over–predicts the width and under–predicts the depth. For higher values of z_0 all the closures would yield even more shallow velocity deficits and all would make poorer performances, but the simple closure would still be best.

a



b



c

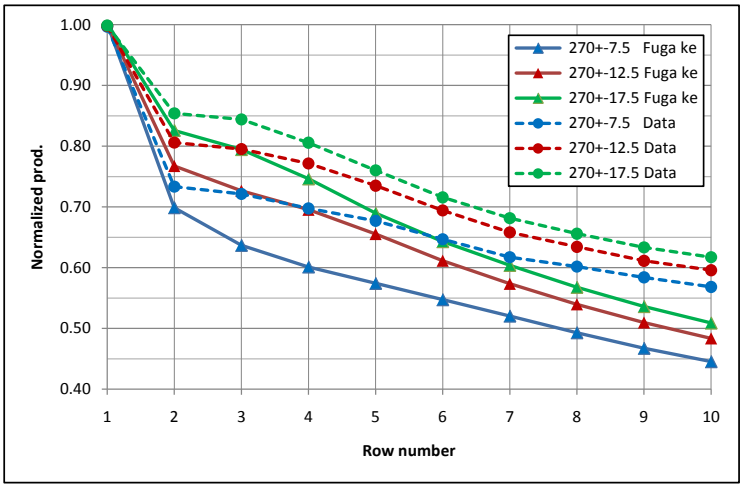
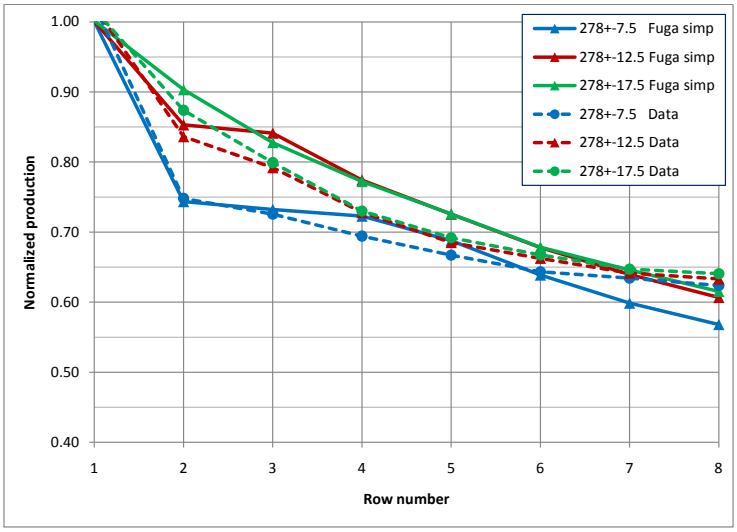
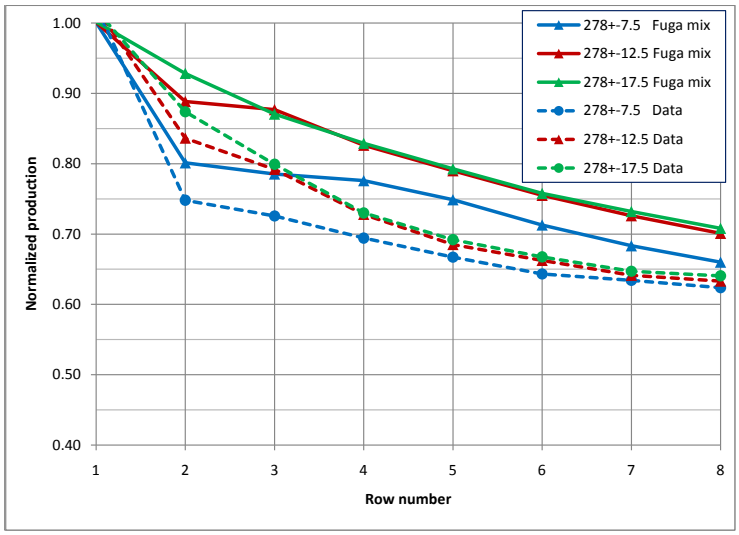


Figure 6. Comparison of Fuga results with data from the Horns Rev I wind farm. U=10m/s, z_0=0.1mm, a: simple closure. b: mixing length closure. c: E–ε closure.

a



b



c

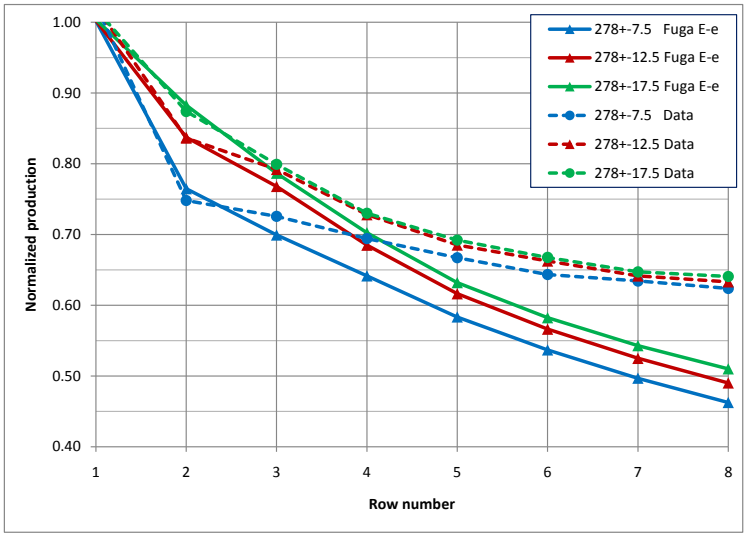
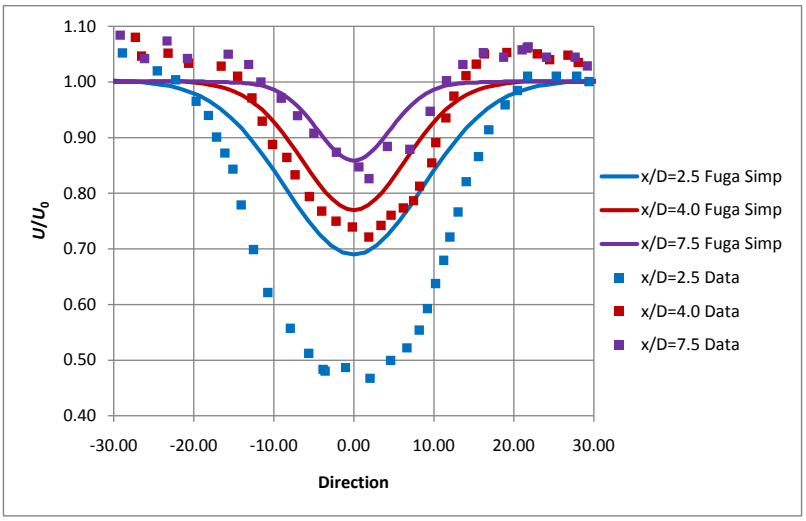
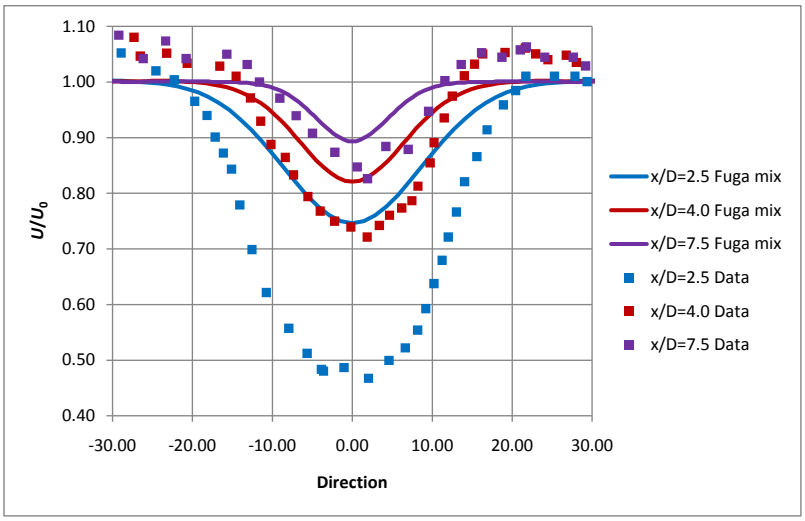


Figure 7. Comparison of Fuga results with data from the Nysted wind farm. $U = 8m/s$, z_0=0.1mm, a: simple closure. b: mixing length closure. c: E–ε closure.

a



b



c

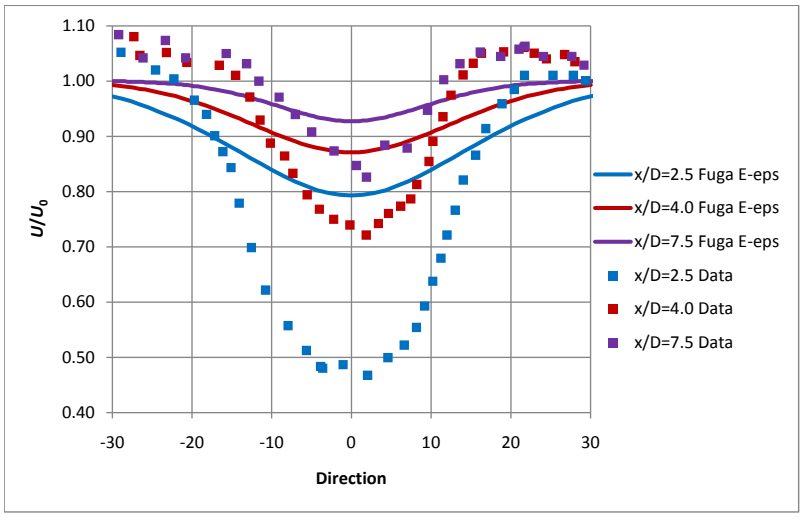


Figure 8. Comparison of Fuga results with data for a solitary Nibe B turbine. $U = 8.55$ m/s, $z_0 = 1$ cm, $C_T = 0.82$, a:simple closure, b: mixing length closure, c: E–ε closure.

7 Inter farm shadow effects

7.1 Farm wake validation

The long–range behaviour of the model is obviously important if the model is to be used to estimate shadow effects between wind farms. Not much data has been available for validations, but the met mast data form Horns Rev 1 can be used. For westerly winds met masts M6 and M7 are in lee of the farm and the reduction of wind speed has been measured, see figure 9. Data were selected from the same periods as in the validation of production predictions in Section 6. Thus wind direction data were inferred from the yaw angle of turbine 07 (green dot in figure 9). The yaw angle is related to the wind direction through the action of the control system in a complicated way, so the method adds an unknown amount of uncertainty to the wind direction data. The bin size of five degree is therefore probably too small so that many measurements may have ended up in the 'wrong' bin.

Data for free wind speeds of 8 m/s and 10m/s are shown in figure 10. M6 is located 2 km behind the last row in the farm. For M7 the distance is 6 km. At M6 the model yields a pronounced peak around 270 degree which is hard to see in the data. This is probably due to the uncertainty in the binning of the data. The reason why the peak is positive is that the met masts are not located exactly behind an West-East row of turbines, but between two rows. The wakes from the two turbine rows therefore tend to miss M6 when the wind direction is exactly 270. The model therefore predicts more shadow effect for the two neightbouring bins at 265 and 275 where the mast is hit more directly by individual wakes. It is likely that the model exaggerates the peak because in reality meandering will blur the picture. At M7 the peak has disappeared leaving an almost perfect agreement between model and measurements. This demonstrated the model's ability to predict wind farm wakes, at least out to distances out to about 6 km.

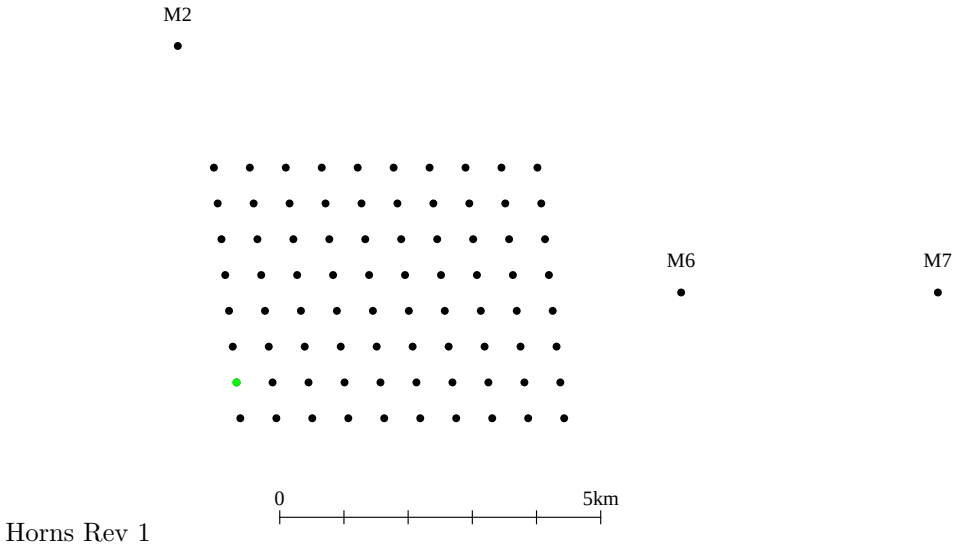


Figure 9. Layout of Horns Rev 1 showing wind turbines and met masts. Production and yaw angle from turbine 07 (green dot) was used to estimate wind speed and wind direction.

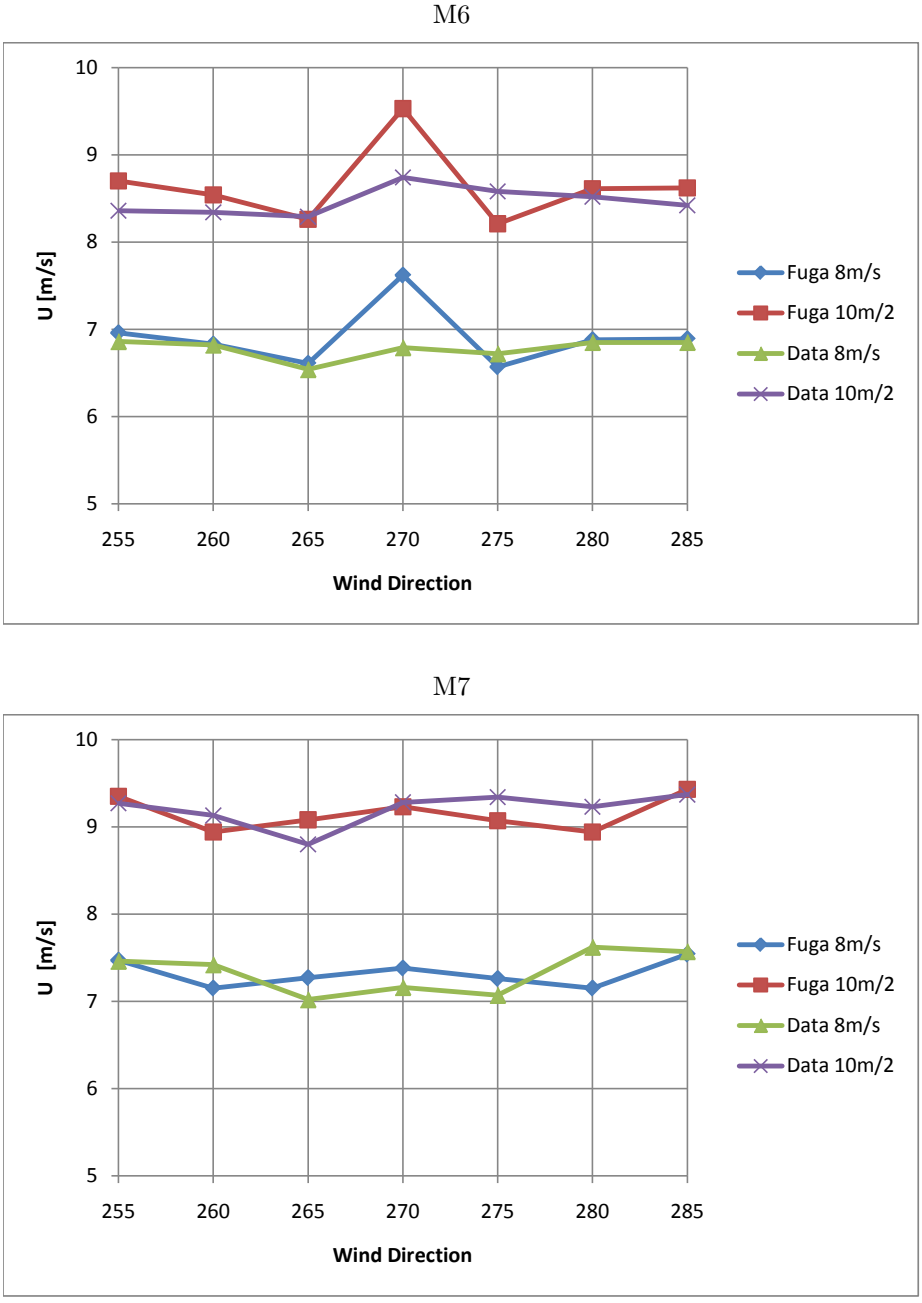


Figure 10. Measured and predicted wind speeds at the two met masts in the lee of Horns Rev 1 for westerly winds. Data and predictions were collected in 5 degree bins.

7.2 A case study: Rødsand 1 and 2

The ability of the model to predict farm–to–farm shadow effects is illustrated below by way of an example. We have chosen the Nysted and Rødsand 2 wind farms, which are two offshore farms located relatively close to each other, see figure 11. Nysted wind farm was finished in December 2003 and consists of 72 Bonus 2.3 MW turbines with 80 m rotor diameter and 69 m hub height. It is placed in an area called Rødsand (with red sand) and has therefore sometimes been referred to as Rødsand 1. Rødsand 2 consists of 90 Siemens 2.3 MW turbines with 93 m rotor diameter and 68.5 m hub height. It started operating in August 2010.

Production data for the period where both farms have been operating are, unfortunately, not yet available. Therefore we cannot make a validation exercise at this point, but must suffice with certain predictions.

The present (June 2011) version of the Fuga program does not allow for different turbine types. Predictions of shadow effects between the two farms can therefore only be made if the same turbine type is assumed for both farms. However, an implementation in the form of a Mathematica notebook exists which can cope with different turbine types, and it is this version that has been used to produce the results that follow.

The distance between the two wind farms is only 3 km, and judging from the wind rose (figure 12), Rødsand 1 should often be in the wake of Rødsand 2. The shadow effect of Rødsand 2 is illustrated in figure 13 which shows the net energy production of Rødsand 1 with and without Rødsand 2 as a function of wind speed. The model was run for 360 directions and results were subsequently collected in 5 degree bins. In the calculations the free windspeed at hub height is fixed at 10 m/s and the roughness is set to 0.1 mm. There is not much shadow effect except for westerly winds. Here the net power reduction amounts to as much as 30% of the free stream value. Figure 14 shows similar graphs for an WestEast line of turbines in Rødsand 1. The line is shown with red dots in figure 11. The most severe power reduction is found for a5 located closest to Rødsand 2. The worst case for a5 is a normalized loss of 60%. The production of a5 averaged over the sector $270° \pm 15°$ is reduced by 42%. For the same sector h5, the most easterly turbine in the line, suffers a reduction of only 14%, much less than the 38% at h5 caused by the other Rødsand 1 turbines, but still not negligible. In all other 30 degree sectors the production losses are rather small and the total loss of annual power production from Rødsand 1 is less than 5%.

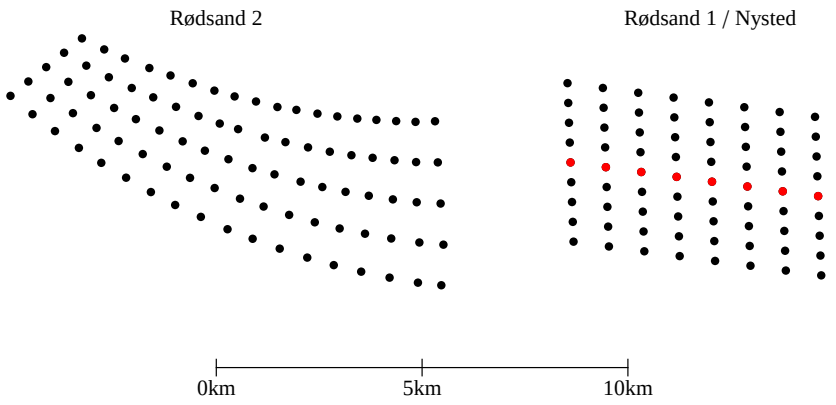


Figure 11. Layout and positioning of Nysted/Rødsand 1 and Rødsand 2 wind farms. The red dots are the turbines used for figure 14

.

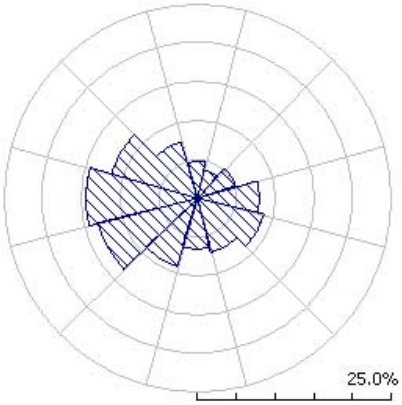


Figure 12. Wind rose for Rødsand.

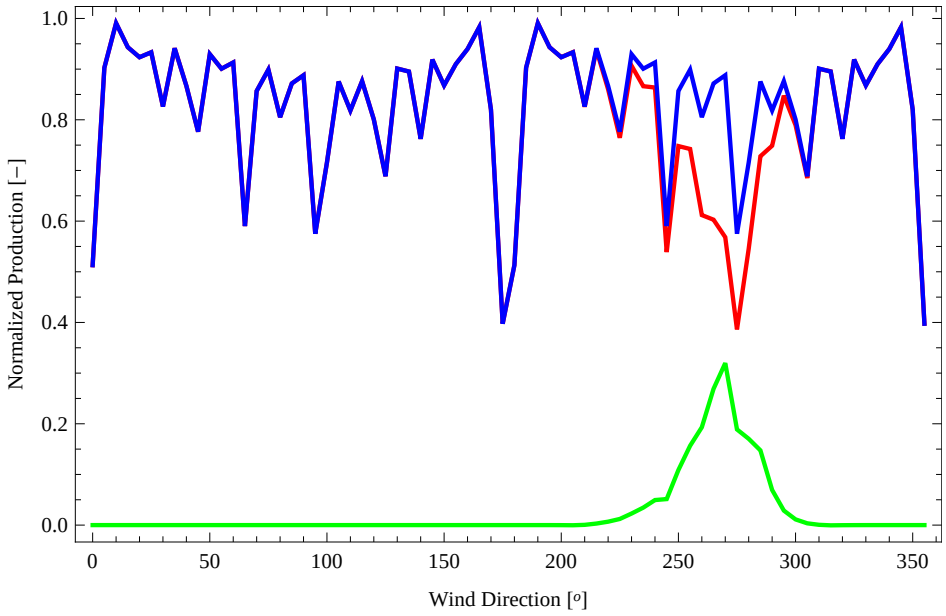


Figure 13. Net production of Rødsand 1 at 10 m/s normalized with free stream value as a function of wind direction. Blue line: without Rødsand 2. Red line: with Rødsand 2). Green line: Normalized power reduction.

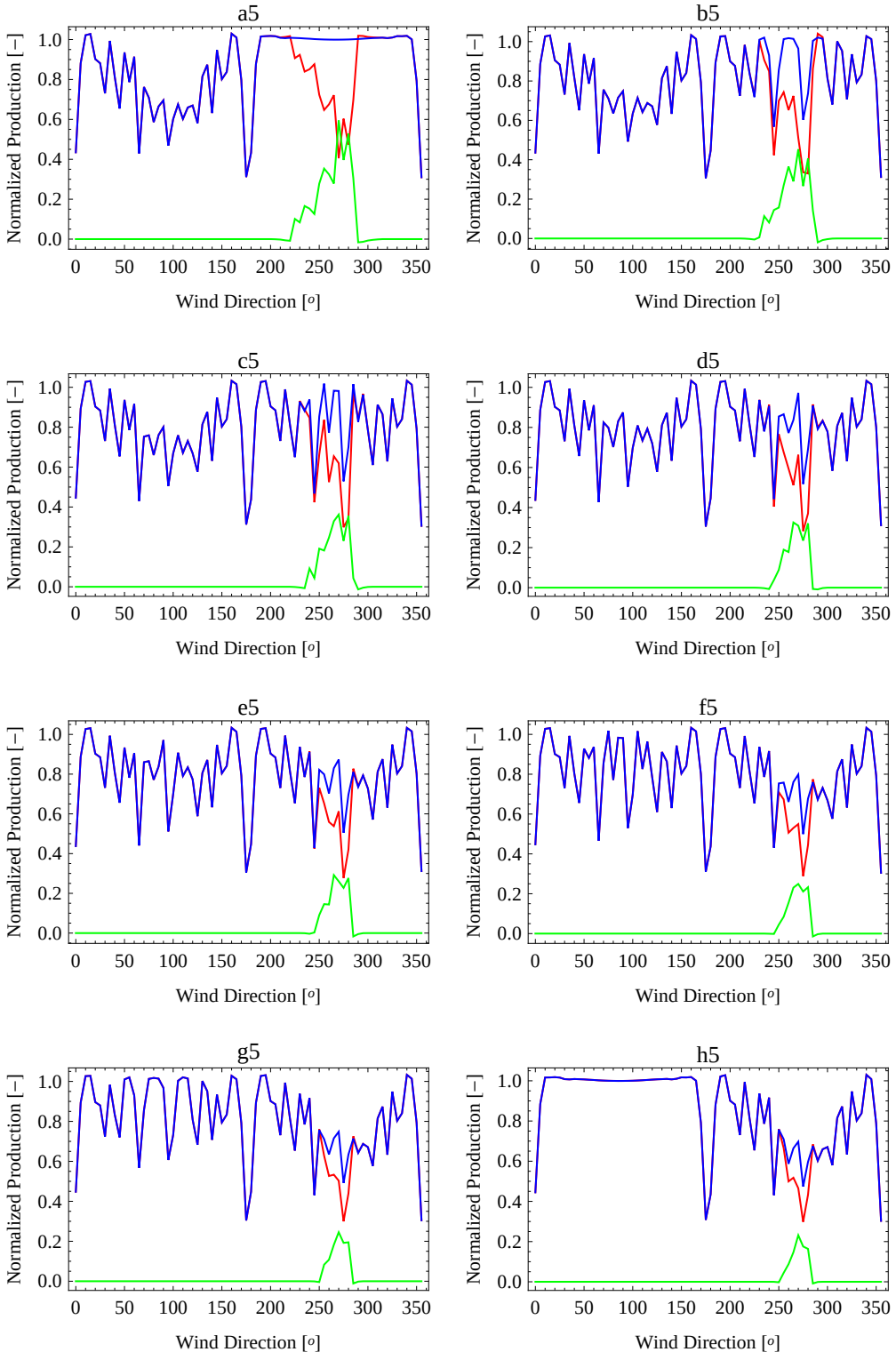


Figure 14. As in fig. 13 except for the line of individual turbines marked with red dots in fig. 11. Turbine identifiers listed from West to East are: a5, b5, c5, d5, e5, f5, g5 and h5.

8 Conclusions

A general linearization procedure has been suggested that applies to any set of CFD equations. The resulting linear equations are further simplified by using a mixed–spectral formulation. This breaks the problem up into independent sets of coupled, ordinary differential equations for which a new solution technique is suggested. The method involves the use of pre–calculated look–up tables which considerably speeds up the solution. The linearized model is typically about 10^6 times faster than the CFD model which it was constructed from.

Three different closures have tried: the 'simple', the mixing length and $E-\varepsilon$. A validation exercise with production data from the Horns Rev 1 and Nysted wind farms and wake measurements from the Nibe turbines shows that the simple closure is clearly superior to the two other closures. This closure is therefore suggested for future, commercial versions of Fuga. The simple closure further validated against direct measurements in the wake of Horns Rev 1 taken with met masts M6 and M7 in the lee of the farm. The predictions are in almost perfect agreement with the measurements.

The agreement with data collected in narrow wind direction bins is, however, not very good. The problem arises when the wind comes straight down turbine rows. In such situations the model tends to exaggerate the wake losses caused by multiple wakes that accumulate momentum deficit from several turbines. At the same time the data are probably not binned very accurately because the wind direction is inferred from the yaw angle of one of the turbines (wind vane data were not reliable). This leads to less pronounced variations with the (ten minutes average) wind direction in the production data than in the model results. Imperfect data is part of the problem, but it is not the whole explanation. It is most likely that meandering should be allowed to smear out the model results. The wakes produced by the model are very straight in contrast to real wakes. The wakes made in LES simulations by Calaf, Meneveau and Meyers (2010), which are adjusted to offshore roughness, show considerable meandering, and it appears to be very unlikely to have more than double wakes. In other words, the wake from a turbine from time to time hits the next downwind turbine to form a double wake, but the double wake only very seldom hits the next turbine to form a triple wake, even if the mean wind points in that direction. This behaviour is caused by large scale eddies, larger than the wake cross section, that twist and turn the wake centreline. The RANS models we have linearized, on the other hand, only have one length scale and therefore cannot distinguish between large scale eddies that produce meandering, and small scale eddies that produce mixing. In the 'simple' closure the eddy viscosity is set equal to the unperturbed value in a statistically homogeneous and stationary background flow field. The eddy viscosity is therefore essentially determined by the mean vertical momentum flux, which is not very much influenced by meandering, since meandering is dominated by side–to–side motion. Large scale eddies that cause meandering are therefore not present in the model universe and meandering effects will have to be added in order to interpret model results in a real world context. We should allow for some random variations added to the straight centrelines of the calculated wakes.

Fuga is useful for the prediction of power production. The model's neglect of meandering makes predictions more sensitive to the wind direction than data can account for. However, the annual production is not so much affected by this. Multiple wake situations are, after all, not very common, and although the model probably predicts too much penalty in multiples wake situations, it also predicts too little penalty in situations where the wakes are not predicted to hit downwind rotors, but meandering in reality sometimes makes it happen from time to time.

The inclusion of meandering effects will be the next step in the development of Fuga. The influence of atmospheric stability and non–uniform terrain are other aspects that will be subject to future work.

9 Acknowledgements

This work was funded by

- Danish Public Service Obligation (PSO) as part of the WindShadow project (Energinet.dk 10086)
- The European Commission in the framework of the Non Nuclear Energy Programme Sixth Framework, Contract REN07/FP6EN/S07.73680/038641 (TOPFARM - Next Generation Design Tool for Optimization of Wind Farm Topology and Operation)
- Carbon Trust as part of the *Offshore Wind Accelerator* project

Notation

Variable	Explanation
A_{Ch}	Charnock constant
A_{ij}	Matrix in the standard form the linearized equations
C_{T}	Thrust coefficient
$C_{\varepsilon 1}$	E–ε model constant
$C_{\varepsilon 2}$	E–ε model constant
C_μ	E–ε model constant
C_ε	E–ε model constant $= \sqrt{\frac{C_\mu}{\sigma_\varepsilon (C_{\varepsilon 2} - C_{\varepsilon 1})}}$
D	Rotor diameter
F	Drag force $= -T$
f	$= F/\rho$
K	Eddy viscosity
k	Horizontal wave number $= \lvert\boldsymbol{k}\rvert$
$\boldsymbol{k}$	Wave number vector $= (k_1, k_2, 0) = k\,(\cos\beta, \sin\beta, 0)$
l_{m}	Mixing length
P	Pressure
p	$= P/\rho - \frac{1}{3}\tau_{jj}$
Q^*	The complex conjugate of the quantity Q
$Q^\dagger$	Complex conjugate transpose of the matrix Q. $(Q^\dagger)_{ij} = Q^*_{ji}$.
R	Rotor radius
s	kz
S_{ij}	Rate of strain tensor, cfr. (7)
t	$\log \frac{z}{z_0}$
T	Thrust
Ti	Turbulence intensity $= \frac{\sigma_u}{U}$
U_{free}	Free stream velocity at hub height
u_*	Friction velocity
$\boldsymbol{u}$	Mean velocity field $= (u, v, w) = (u_1, u_2, u_3)$
$\boldsymbol{u}^0$	Unperturbed velocity $= (u^0(z), 0, 0)$
$\boldsymbol{u}^1$	First order perturbed velocity $= (u^1, v^1, w^1) = (u^1_1, u^1_2, u^1_3)$
(x, y, z)	Coordinates: x–axis downwind, z–axis upwards
$\boldsymbol{x}$	Linear variables = e.g. (u, u', v, v', w, p)
$\boldsymbol{y}^q$	Dependent variable in the dual equation for the qth boundary condition
$\boldsymbol{Y}$	$Y_{ij} = {y_i}^j$
z_{h}	Hub height
z_0	Surface Roughness
β	Angle from $\boldsymbol{u}^0$ to $\boldsymbol{k}$
κ	von Karman constant $= 0.4$
ν	Molecular viscosity
ρ	Air density
σ_E	E–ε model constant
σ_ε	E–ε model constant
τ_{ij}	Reynolds stress tensor $= -\langle u'_i u'_j \rangle$
ξ	Expansion parameter for perturbation series

References

Bailey, D. H. and Swarztrauber, P. N.: 1991, The fractional fourier transform and applications, *SIAM Review* **33**, 389–404.

Bailey, D. H. and Swarztrauber, P. N.: 1994, A fast method for the numerical evaluation of continuous fourier and laplace transforms, *SIAM J. Sci. Comput.* **15**, 1105–1110.

Barthelmie, R., Frandsen, S. T., Rathmann, O., Hansen, K. S., Politis, E. S., Prospathopoulos, J., Schepers, J. G., Rados, K., Cabezn, D., Schlecz, W., Neubert, A. and Heath, M.: 2011, Flow and wakes in large wind farms:final report for upwind wp8, *Technical Report Risø–R–1765(EN)*, Risø-DTU.

Barthelmie, R. J., Rathmann, O., Frandsen, S. T., Hansen, K. S., Politis, E., Prospathopoulos, J., Rados, K., Cabezon, D., Schlez, W., Phillips, J., Neubert, A., Schepers, J. G. and van der Piljl, S.: 2007, Modelling and measurements of wakes in large offshore wind farms, *Conference on the science of making torque from wind*, Journal of Physics Conference Series, Vol. 75, 012049., Danish Technical University, p. 8 p.

Belcher, S. E., Jarram, N. and Hunt, J. C. R.: 2003, Adjustment of a turbulent boundary layer to a canopy of roughness elements, *J. Fluid Mech.* **488**, 369–398.

Calaf, M., Meneveau, C. and Meyers, J.: 2010, Large eddy simulations study of fully developed wind-turbine array boundary layers., *Physics of Fluids* **22**, 015110.

Cash, J. R. and Karp, A. H.: 1990, A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides, *Transactions of Mathematical Software* **16**, 201–222.

Charnock, H.: 1955, Wind stress on a water surface, *QJRMS* **81**, 639–640.

Corbett, J.-F.: 2007, Ramsim: A fast computer model for mean wind flow over hills., *Technical Report Risø–PhD–17*, Ph. D. thesis from Copenhagen University, Denmark. Available from Risø National Laboratory.

Corbett, J.-F., Ott, S. and Landberg, L.: 2008, A mixed spectral–integration model for neutral mean flow over hills, *Boundary-Layer Meteorol.* **128**, 229–254.

Frandsen, S. T., Jørgensen, H. E., Barthelmie, R. J., Rathmann, O., Badger, J., Hansen, K., Ott, S., Rethore, P.-E., Larsen, S. E. and Jensen, L.: 2009, The making of a second-generation wind farm efficiency model-complex, *Wind Energy* **12**, 445–458.

Gribben, B.: 2010. Private communication. The constants were suggested by Carbon Trust in connection with the Offshore Wind Accelerator project.

Hansen, K. S.: 2009, Analysed data for the topfarm project. Private communication.

Hansen, K. S., Barthelmie, R. J., Cabezon, D. and Politis, E.: 2008, Wake measurements used in the model evaluation. Deliverable D8.1 of the UpWind project.
URL: *http://www.upwind.eu/Shared%20Documents/WP8%20-%20Flow/D8.1%20-%20Wake%20measurements%20used%20in%20the%20model%20evaluation.pdf*

Hassan, U., Taylor, G. J. and Garrad, A. D.: 1988, The dynamic response of wind turbines operating in a wake flow, *Journal of Wind Engineering and Industrial Aerodynamics* **27**, 113–126.

Inverarity, G. W.: 2002, Fast computation of multidimensional fourier integrals., *SIAM J. SCI. COMPUT.* **24**, 645–651.

Kasmi, A. E. and Masson, C.: 2008, An extended k–ε model for turbulent flow through horizontal–axis wind turbines, *J. of Wind Eng. and Ind. Aerodyn.* **96**, 103–122.

Ott, S. and Nielsen, M.: 2010, Fuga files and scripts. Note on Fuga input and output files and how to run batch mode. Fuga documentation file FilesAndScripts.pdf.

Panofsky, H. A. and Dutton, J. A.: 1984, *Atmospheric Turbulence*, John Wiley & Sons, New York.

Peña, A. and Gryning, S.-E.: 2008, Charnock's roughness length model and non-dimensional wind profiles over the sea, *Boundary-Layer Meteorology* **128**, 191–203.

Prandtl, L.: 1925, Über die ausgebildete Turbulenz, *Zeitschrift für Angewandte Mathematik und Mechanik* **5**, 136–139.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: 1992, *Numerical Recipes*, 2nd edn, Cambridge University Press.

Rados, K., Prospathopoulos, J., Stefanatos, N., Politis, E., Chaviaropoulos, P. and Zervos, A.: 2009, CFD modeling issues of wind turbine wakes under stable atmospheric conditions, *EWEC 2009*.

Réthoré, P.-E.: 2009, Wind turbine wake in atmospheric turbulence., *Technical Report AAU-DCE Thesis No. 22*, Ph. D. thesis from Aalborg University & Risø–DTU, Denmark. Available from Risø National Laboratory: RIS-PhD–53(EN).
URL: *http://windenergyresearch.org/?p=115H*

Réthoré, P.-E., Sørensen, N. and Bechmann, A.: 2010, Modelling issues with wind turbine wake and atmospheric turbulence., *TORQUE Conference proceedings.*
URL: *http://windenergyresearch.org/?p=76H*

Sanz, C.: 2003, A note on $k - \varepsilon$ modelling of vegetation canopy air-flows, *Boundary–Layer Meteorology* **108**, 191–197.

Simonen, P. and Olkkonen, H.: 1985, Fast method for computing the fourier integral transform via simpsons numerical integration, *J. Biomed. Eng.* **7**, 337–340.

Sogachev, A. and Panferov, O.: 2006, Modification of two-equation models to account for plant drag, *Boundary-Layer Meteorology* **121**, 229–266.

Taylor, G. J.: 1990, Wake measurements on the nibe wind turbines in denmark, *Technical Report ETSU WN 5020*, Department of Energy, National Power.

Vermeer, L. J., Sørensen, N. J. and Crespo, A.: 2003, Wind turbine wake aerodynamics, *Progress in Aerospace Sciences* **39**, 467–510.

Wilcox, D. C.: 2002, *Turbulence Modeling for CFD*, 2nd edn, DCW Industries.

Yakhot, V., Orszag, S. A., Thangam, S., Gatski, T. B. and Speziale, C. G.: 1992, Development of turbulence models for shear flows by a double expansion technique, *Physics of Fluids A* **7**, 1510–1520.

Bibliographic Data Sheet **Risø–R–1772(EN)**

Title and author(s)

Linearised CFD Models for Wakes

Søren Ott, Jacob Berg and Morten Nielsen

ISBN 978-87-550-3892-9

ISSN

Dept. or group Wind Energy Department

Date December 9, 2011

Groups own reg. number(s)

Project/contract No. 1120183-01

Pages	Tables	Illustrations	References
39			

Abstract (Max. 2000 char.)

This report describes the development of a fast and reasonably accurate model for the prediction of energy production in offshore wind farms taking wake effects into account. The model has been implemented as a windows application called Fuga which can run in batch mode or as a graphical user interface. Fuga is briefly described. The model is based on alinearization technique which is described in some detail, and linearized, governing equations are derived and written in a standard form based on a mixed–spectral formulation. A new solution method is used to solve the equations which involves intensive use of look–up tables for storage of intermediate results. Due to the linearity of the model, multiple wakes from many turbines can be constructed from the wake of a single, solitary turbine. These are in turn constructed from Fourier components by a fast Fourier integral transform of results derived from generic look–up tables. Three different models, based on three different closures, are examined:

- the 'simple closure' using an unperturbed eddy viscosity $\kappa u_* z$
- the mixing length closure
- the E–ε closure

Model results are evaluated against offshore wind farm production data from Horns Rev I and the Nysted wind farm, and a comparison with direct wake measurements in an onshore turbine (Nibe B) is also made. A very satisfactory agreement with data is found for the simple closure. The exception is the near wake, just behind the rotor, where all three linearized models fail. The mixing length closure underestimates wake effects in all cases. The E–ε closure overestimates wake losses in the offshore farms while it predicts a too shallow and too wide the wake in the onshore case. The simple closure performs distinctly better than the other two. Wind speed data from the the Horns rev met masts are used to further validate Fuga results with the 'simple' closure.
Finally, Rødsand 1 and 2 are used as an example to illustrate the interaction between wind farms.

Descriptors

Available on request from:
Information Service Department, Risø National Laboratory
(Afdelingen for Informationsservice, Forskningscenter Risø)

P.O. Box 49, DK–4000 Roskilde, Denmark
Phone (+45) 46 77 46 77, ext. 4004/4005 · Fax (+45) 46 77 40 13
E-mail: risoe@risoe.dk