NTNU

Norwegian University of
Science and Technology

Numerical Simulations of Flow Past a Truss Tower with an Evaluation of Tower Shadow Models for Wind Turbines

Torbjørn Ruud Hagen

Master of Science in Physics and Mathematics

Submission date: June 2011

Supervisor: Johan Skule Høye, IFY

Co-supervisor: Michael Muskulus, Institutt for bygg, anlegg og transport

Norwegian University of Science and Technology
Department of Physics

Abstract

The performance of steady-state tower shadow models for a wind turbine truss tower have been evaluated. The Reynolds-Averaged-Navier-Stokes (RANS) approach, in conjunction with the $k-\omega$ Shear-Stress-Transport (SST) model, was used to simulate transient flows past cross sections of a truss tower. The objective was to compare numerical results with Powles', Blevins' and Schlichting's tower shadow models and evaluate their performance on a multimember structure. Parameters for each model have been estimated. It will be shown that the RANS model was able to reproduce realistic results when used in transient simulations on high Reynolds number flows (supercritical regime). The importance of considering unsteady motion when calculating the turbulence intensity, using RANS with transient simulations, will be explained. The multimember extension used for the tower shadow models reproduces the mean velocity profiles quite well, and by using a suitable estimation method, global parameters were found for all models. Additionally, turbulent inflow has been implemented with a user-defined function in Fluent. The results have been evaluated, and show that such such sophisticated inflow modeling is not necessary to predict realistic mean velocity profiles.

Project description

A truss tower could be a wind turbine tower option to the more common monopile. By further considering downwind turbines, there is a need for a multimember tower shadow model. Powles' model have a multimember extension, but this is not validated. Using a computational fluid dynamics (CFD) approach, the numerical results can be evaluated with the performance of the tower shadow model.

Acknowledgements

I would like to thank my supervisor, Dr. Michael Muskulus at the Department of Civil and Transport Engineering, who has been a great teacher and mentor through this last year at NTNU. In addition to have to many projects going on at the same time, he have helped me publish two conference papers (OMAE-2011 and ISOPE-2011) from this research. A special thanks to Marit Reiso for great collaboration and valuable discussions, and who made it possible to present the ISOPE-paper in Maui, Hawaii. I would also like to thank everybody in the 'basement' at Department of Civil and Transport Engineering, for great discussions and serving me unhealthy food every friday.

At the Department of Physics, I would like to thank my internal supervisor Professor Johan Skule Høye.

Finally I would like to thank all my classmates at NTNU, Tindegruppa for the necessary ingredient (climbing) after hours in front of the computer, REKA FK for many happy hours, everybody who has helped me avoid embarrassing mistakes in the these, and of course my Lina. I could not have done this without you.

Contents

List of Figures

List of Tables

Key-results

- The Reynolds-Averaged-Navier-Stokes (RANS) approach is able to give accurate results for transient simulations of flow around two dimensional structures.
- When doing transient simulations with RANS it is important to consider that turbulence intensity has contribution both from the sub-grid parametrization of turbulence and the unsteady fluid motion, and needs to be properly calculated.
- Tower shadow models give a good representation of the mean velocity profile behind multi-member structures.
- When finding tower shadow model parameters for numerous cases simultaneously, the best optimization method is minimizing the largest maximum error for a single profile.
- There are global tower shadow parameters that can accurately predict the mean velocity profile for a number of cross sections simultaneously.
- Turbulent inflow is not necessary to predict realistic tower shadow effects behind a truss tower.

Chapter 1

Introduction

Renewable energy is high on the political agenda and large companies are starting to invest in wind energy. As there are several issues placing large wind turbines onshore, like size, noise, transport, etc., the offshore environment is an interesting option. However, moving wind turbines offshore gives a high increase in the overall cost. Transport, maintenance and installations are only a few of the expenses that will increase offshore. This challenges the planning and accurate analyzes are necessary to find cheaper ways of expanding wind energy production. The material cost can be reduced by changing the more common monopile with a truss tower [17]. Usually, offshore wind turbines are installed on bottom-fixed sub-structures, such as tripod sub-structures, gravity based foundations or jacket (truss) structures [6]. An interesting alternative is to build a complete bottom-fixed wind turbine with a truss tower all the way to the nacelle. As the truss tower is angled and has a wider tower diameter than the monopile, tower clearance then becomes an issue. It is therefore interesting to look at a wind turbine with a downwind rotor. The blades will bend away from the tower, reducing the risk of hitting the tower, and it is possible to further decrease manufacturing cost of the blades by making them more flexible (implying cheaper and lighter blades). A downwind rotor will also have a simpler yawing mechanism and will therefore be more robust. However, a downwind rotor will experience a tower shadow effect much greater than the tower shadow for an upwind rotor. Since a truss tower is more transparent than a monopile, impact loads on the blades from the tower shadow will be reduced in a downwind configuration, when using a truss tower [25].

When analyzing damaging loads on complete wind turbines, the core issue is reliability. Accurately predicting these loads is important and good numerical models are needed. For now, numerous tower shadow models have been proposed and used for a monopile. Most of them simply model the mean wind field behind a circular cylinder, as a representation of a two dimensional cross section of a monopile. Powles' tower shadow model has been used as an analytical steady-state tower shadow model in the commercialized software package GH Bladed [5], which simulates a complete wind turbine setup. Recently, this model have been extended to be applicable for truss towers. However, there is little literature regarding the choice of model parameters. With incorrect parameters the tower shadow model could predict mean velocity profiles with large discrepancies, which again could result in over- and/or underpredicitons when used in a fatigue analysis of a wind turbine. This important aspect is one of the key-issues in this report.

In chapter 2: *Theoretical aspects* the fundamentals of this study is presented. Then, in chapter 3: *Numerial simulations*, a numerical study using the Reynolds-Averaged-Navier-Stokes (RANS) approach, on flow past two dimensional cross sections of wind turbine towers, are presented. Here, the mean velocity profiles, turbulence intensity profiles and power spectrums, are analyzed. Further, the tower shadow models are introduced in chapter 4: *Tower shadow models*. Their performance will be compared with the numerical results and evaluated. A way to find the best tower shadow model parameters are introduced and a recommended set of global parameters will be presented. The RANS model is used for transient simulations, which means there is only parameterized turbulence. In chapter 5: *Simulations with turbulent inflow*, the same numerical approach have been used with additional turbulent inflow, to see if this affects the tower shadow. A summary is found in chapter 6: *Aftermath*, where the key-results are discussed.

Chapter 2

Theoretical aspects

A fluid in motion involves a lot of complexity, and this has for several decades been a challenge both for scientists and for industrial applications. For the time being there is no exact analytical solution to the general equations, given by Navier-Stokes (NS) (2.3), of fluid dynamics, but with simplifications practical problems can be solved. In short, one could say that fluid dynamics is a discipline that consider the behaviour of fluid in motion, and can be used to solve practical problems by calculating properties of the flow i.e., velocity, pressure, density and temperature, that varies in space and time.

2.1 Fundamental equations

A fluid in motion is usually characterized using a non-dimensional quantity called the Reynolds number, which is the ratio of the inertial to the viscous forces,

$$Re = \frac{\text{inertial forces}}{\text{viscous forces}} = \frac{\rho DU}{\mu} = \frac{DU}{\nu}, \tag{2.1}$$

where D is a characteristic length, e.g. the diameter of a circular cylinder, which the fluid is passing, and U is the flow velocity. The density of the fluid is ρ, μ is the dynamic viscosity and ν is the kinematic viscosity. When this concept was popularized by Osborne Reynolds in 1883 [26] it opened up the opportunity to compare different fluids. For the same Reynolds number their behaviour will be (almost) the same, and this gave way to a more coordinated study of fluids in motion.

Zdravkovich [39] divided the qualitative flow features into different flow regimes depending on the Reynolds number, summarizing results from a large experimental study of flows past circular cylinders. The flow is laminar for very low Reynolds number flows ($0 < Re < 200$), i.e., the fluid smoothly flows around the cylinder. For Reynolds numbers above ~ 5, the flow separates (at different distances due to the Reynolds number) and a von Karman vortex street develops. These vortices occur with a certain frequencies f, which are described by the non-dimensional Strouhal number, defined by

$$St = \frac{fD}{U} \ . \tag{2.2}$$

For a circular cylinder the Strouhal number will typically be between 0.2-0.6, depending on the Reynolds number (and the surface roughness). By using mass, momentum and energy conservation together with Reynolds transport theorem [37], it is possible to give a complete description of the fluid in motion, that captures these features. These equations, together with the stress tensor for Newtonian viscous fluid $\sigma_{ij} = -pI + \tau_{ij}$[1] and the continuity equation, is usually referred to as the Navier-Stokes equations,

$$\frac{\partial(\rho\vec{u})}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}\vec{u}) = -\vec{\nabla}p + \mu\nabla^2\vec{u} + \rho\vec{f_b}, \tag{2.3}$$

and the continuity equation

$$\frac{\partial\rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}) = 0\ . \tag{2.4}$$

Here, ρ is the fluid density, the fluid velocity is denoted by $\vec{u}$, p is the pressure and $\vec{f_b}$ is the volumetric forces acting on the fluid. The viscous stresses are represented by the term $\mu\nabla^2\vec{u}$, where μ is the dynamic viscosity. On the left side of equation (2.3), the term $\frac{\partial(\rho\vec{u})}{\partial t}$ describes the velocity change in time and $\nabla\cdot(\rho\vec{u}\vec{u})$ is the convective acceleration, which makes the equations non-linear. This non-linearity is the major cause of turbulent behaviour. Turbulent flows vary significantly in space and time. This unsteady motion, or irregularity, is random, and together with diffusivity the turbulence change a non-uniform system into a homogenous system when sufficient energy is added.

Dissipation is another important feature in turbulent motion and it is a cascade process. If no external energy is added to the turbulent system, all the kinetic energy dissipates eventually, due to viscous shear stress, i.e., large eddies (or vortices) dissipates to smaller eddies before they transform into internal energy (or heat). The latter happens at the Kolmogorov scale which is the smallest turbulent scale [22, 32]. These complex features makes it necessary to use a numerical approach to solve most practical problems.

2.2 Numerical models

Numerically predicting a flow behavior is always a challenge. There are several different methods which all have advantages and disadvantages. A common approach is the Finite Element Method (FEM), which divides the domain(s) of interests into elements (or nodes), and the partial differential equations (PDE) (of interest) are solved by simple local approximations for each element [8]. This means that more elements requires more computational time. Another possibility is to use the Finite Difference Method (FDM) [8], but this is not pursued here.

Direct Numerical Solution (DNS) solves the Navier-Stokes equations without any approximations. Unfortunately, the computational cost required for simulating flows with large Reynolds numbers, is prohibitively high. The number of elements needed for DNS is

[1] I is the identity matrix and τ_{ij} is the Reynolds stresses

$$N_{nodes} \sim (Re)^{\frac{9}{4}}, \tag{2.5}$$

i.e., this accurate numerical method can only be used for low Reynolds numbers, for the time being.

Another computational demanding numerical model is the Large Eddy Simulation (LES). The turbulent length scale can be divided into three scales: large eddies, the cascade range where the large eddies transform into smaller eddies and the dissipation range, where the eddies dissipates into viscous stresses. LES models the dissipation range with sub-grid parametrization, and solve the equations explicitly for the large scales. LES is much faster than DNS, but it still requires high computational cost [22].

2.2.1 Reynolds-Averaged-Navier-Stokes(RANS) approach

A more popular way, but less accurate, is to model the dissipation and cascade range. Several different models do this and all of them are based on the Reynolds-Averaged-Navier-Stokes (RANS). The RANS model does not provide a completely accurate description of turbulent behaviour in time, but results in a time-averaged solution of the Navier-Stokes equations. Therefore, RANS is usually used for steady-state simulations. The governing equations are derived from the Navier-Stokes equations using Reynolds decomposition,

$$\vec{u} = \vec{U} + \vec{u}', \tag{2.6}$$

where $\vec{U}$ is the mean velocity and $\vec{u}'$ is a perturbation around this mean. The RANS equations are then given by

$$\rho\frac{\partial U_i}{\partial t} + \rho U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) - \rho\overline{u_i u_j}\right), \tag{2.7}$$

where the turbulent motions are represented by only $-\rho\overline{u_i u_j}$, which are known as the Reynolds stresses. The Reynolds stresses are not explicitly provided by the equations and additional assumptions are necessary. Boussinesq assumed in 1877 [32, 37] that turbulence could be modeled with eddy viscosity,

$$-\rho\overline{u_i u_j} = \mu_t\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right) + \frac{2}{3}\rho k\delta_{ij}\ , \tag{2.8}$$

where k is the kinetic energy. Here, the turbulent dynamic viscosity μ_t is introduced. This turbulent-viscosity hypothesis introduces six additional independent equations (2.8), resulting in too many unknown variables, and a closure of the set of the equations became a problem. Jones and Launders introduced in 1972 [15] a two-equation model, commonly known as the $k-\epsilon$ model, where they related the turbulent eddy viscosity to the turbulent kinetic energy k and the turbulent dissipation rate ϵ.

2.2.2 Two-equation models

The two additional transport equations gave way to a new direction of solving fluids in motion. One equation for the kinetic energy

$$\rho\frac{Dk}{Dt} = \nabla\cdot(\Gamma_k\nabla k) + G_k - \rho\epsilon, \tag{2.9}$$

and one for the dissipation rate

$$\rho\frac{D\epsilon}{Dt} = \nabla\cdot(\Gamma_\epsilon\nabla\epsilon) + C_{1\epsilon}\frac{\epsilon}{k}G_k - C_{2\epsilon}\rho\frac{\epsilon^2}{k}, \tag{2.10}$$

where

$$\Gamma_{k,\epsilon} = \mu + \frac{\mu_t}{\sigma_{k,\epsilon}}\ . \tag{2.11}$$

Here, μ_t is the turbulent dynamic viscosity and $\sigma_{k,\epsilon}$ the turbulent Prandtl numbers for k and ϵ. Note, that $\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u}\cdot\vec{\nabla}$ is the material derivative. The dissipation rate is given by ϵ, k is the turbulent kinetic energy and ρ is the fluid density. Production of k is represented by G_k. The model constants are found in table 2.1.

Table 2.1: Parameters used in the standard $k-\epsilon$ viscosity model.

Parameters	$C_{1\epsilon}$	$C_{2\epsilon}$	C_μ	σ_k	σ_ϵ
Values	1.44	1.92	0.09	1.0	1.3

The standard $k-\epsilon$ model is popular for its simplicity and is usually accurate for simple flows. For more complex flows, on the other hand, flow patterns can be qualitatively incorrect. It also needs to be improved in order to apply in the viscous near-wall region. Kolomogorov had already suggested a different approach in 1942, which was refined by Wilcox in 1993 [22]. To better handle transitions in the near-wall region, the turbulent kinematic viscosity ν_t was specified with the specific dissipation rate ω,

$$\nu_t = \frac{\mu_t}{\rho} = C_\mu\frac{k^2}{\epsilon} = C_\mu\frac{k}{\omega}\ . \tag{2.12}$$

This model is identical to the $k-\epsilon$ model for homogenous flows, but has an additional term for inhomogeneous flows. The major difference is that the $k-\omega$ model accounts for the effects of streamwise pressure gradients and it has a superior way to treat the viscous near-wall region. Menter refined this model further in 1994 [19], to yield the best behaviour of the $k-\epsilon$ at a free-stream boundary and the $k-\omega$ in the near-wall region.

k-ω Shear-Stress-Transport (SST) model

This model combines the free-stream independence of the k-ϵ model and the near-wall treatment of the k-ω model, by multiplying each model with a blending function and then adding the models together. The blending function reassures that the appropriate behavior is predicted in the near-wall region and the far-field zones. This makes the model a good choice for this study. The two transport equations for the $k-\omega$ SST model are given by,

$$\rho\frac{Dk}{Dt} = \nabla\cdot(\Gamma_k\nabla k) + \tilde{G}_k - Y_k, \tag{2.13}$$

and

$$\rho\frac{D\omega}{Dt} = \nabla\cdot(\Gamma_\omega\nabla\omega) + \tilde{G}_\omega - Y_\omega + D_\omega\ . \tag{2.14}$$

Here, the dissipation of the turbulent kinetic energy k and the specific dissipation rate ω is represented by Y_k and Y_ω. The generation of k and ω is represented by $\tilde{G}_k$ and $\tilde{G}_\omega$, respectively and D_ω represents the cross-diffusion term. The quantities Γ_k and Γ_ω are the effective diffusitivities for k and ω. See ANSYS Fluent Theory Manual [3] for further details.

Flow past circular cylinders

The basic object of this study can be described as flow past circular cylinders, or more accurate; flow past circular cross sections. This classical problem has been studied for a long time and it is widely used in order to solve practical problems. Four regions characterize flow past circular cylinders: (a) The stagnation region in front of the cylinder, (b) the boundary layers which is attached to the cylinder surface, (c) the speed-up region to the side of the cylinder and (d) the cylinder wake, which is the focus of this study.

Flow past a cylinder can be divided into different flow regions, depending on the Reynolds number. For very low Reynolds number (Re<200) the flow is laminar and closely attached to the cylinder, and vortices are produced in the far-wake region[2]. For higher Reynolds numbers, a transition from laminar to turbulent state happens in different regions behind the cylinder. The transitions happens in the far-wake for $Re \sim 200$, and as the Reynolds number increase even higher, the transitions move towards the cylinder until the boundary layer region becomes turbulent (at about $Re \sim 10^5$). When the Reynolds number is sufficiently high ($Re \sim 10^6$), the flow reach the supercritical Reynolds number regime. This regime was earlier believed to be the ultimate state for all flows [39], but later research found that earlier features reappear for even higher Reynolds numbers [27]. This is not discussed further here, but an in-depth study is found in Zdravkovich [39, 40]. In this study, a flow in the supercritical Reynolds number regime has been simulated numerically.

[2]When Re <5 there is no separation in the wake and therefore no vortices either.

Chapter 3

Numerical simulations and wake development

Numerical simulations are a good option to avoid large experimental setups, which, in addition to be time consuming and expensive, can give uncertain instrumental and measurement errors. Numerical simulations also have drawbacks, but despite the challenges, i.e., computational capacity and numerical instabilities, they can give reliable results in a short period of time. As the computer capacity has increased the last decades, several new and fast methods in computational fluid dynamics (CFD) have been developed (see chapter 2). Many of these models are widely used in industrial applications, and their importance continue to expand as they become even more accurate. In this study a numerical approach to measure wind fields behind a wind turbine tower has been used.

Looking at a global scale, the wind is caused by pressure differences across the earth surface. By considering the four atmospheric forces: the Coriolis force due to earth rotation, pressure forces, inertial forces due to large scale motion and frictional forces at earth's surface, one can make a simple model of the winds behavior.

The Reynolds-Averaged-Navier-Stokes (RANS), in conjunction with the $k-\omega$ Shear-Stress-Transport (SST) viscosity model, has been used to predict mean wind fields and unsteady fluctuations behind two dimensional cross sections of a wind turbine tower. Usually, RANS is not used for transient simulations. Applying this means there will be an additional contribution from the fluctuations to the turbulence intensity, and this needs to be accounted for (see section 3.5.2). Only one wind speed (12m/s) is considered, and turbulence intensity (10%) is added to the inflow from the $k-\omega$ SST sub-grid parametrization.

The two dimensional cross section of the monopile is represented as a single circular cylinder. When looking at two dimensional cross sections, important features like three dimensional effects [38] are lost. This was necessary to reduce the computational time, but the major features are still captured. Results from simulating flow past this cylinder was compared with earlier literature, in order to validate the $k-\omega$ SST model. An arrangement of cylinders represent the cross section of the truss tower. The truss tower is angled 0, 22.5 and 45 degrees with respect to the inflow angle, in order to simulate different wind directions. The 22.5 degree case will

be more representative to a real flow, where as the 0 and 45 degree towers, with their aligned members (with respect to the inflow), are more extreme cases.

Note that it is common practice in computational fluid dynamics (CFD) to non-dimensionalize all parameters. However, to be more consistent in the field of wind energy, dimensions have been kept for all parameters in this study.

3.1 Geometry and mesh

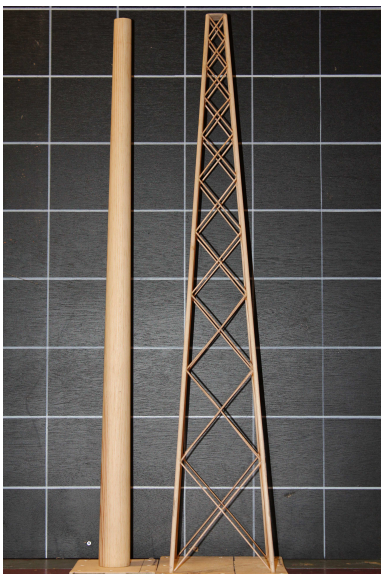


Figure 3.1: Monopile (left) and the truss tower (right)

There are currently two tower options for the wind turbines, the monopile and the truss tower (Fig. 3.1). Studied here is only the two dimensional cross sections of these towers. The cross section of the monopile is represented by a single cylinder with a diameter of D=4m.

Four cross sections of the truss tower have been looked at (Fig. 3.2). Each of them are represented as a varied arrangement of cylinders, with main cylinders representing the main legs and smaller cylinders representing the X-brace. The main cylinders has a diameter of d_1=0.9m and are separated by a distance of l_1=10.8m for all cases. To represent the X-brace there are several smaller cylinders, with a diameter of d_2=0.36m, in different arrangements. All cases use the four main cylinders, and in addition has Case-A four smaller cylinders and Case-B,C,D eight smaller cylinders (Fig. 3.3).

Case-A (Fig. 3.3a) represents the joint section of the X-brace, (top line in Fig. 3.2). Case-B (Fig. 3.3b) has two smaller cylinders close to each other at the center between the main cylinders (second line from above in Fig. 3.2). The third cross section, Case-C (Fig. 3.3c), has two small cylinders at a larger distance from the center (third line from above in Fig. 3.2). Finally, the

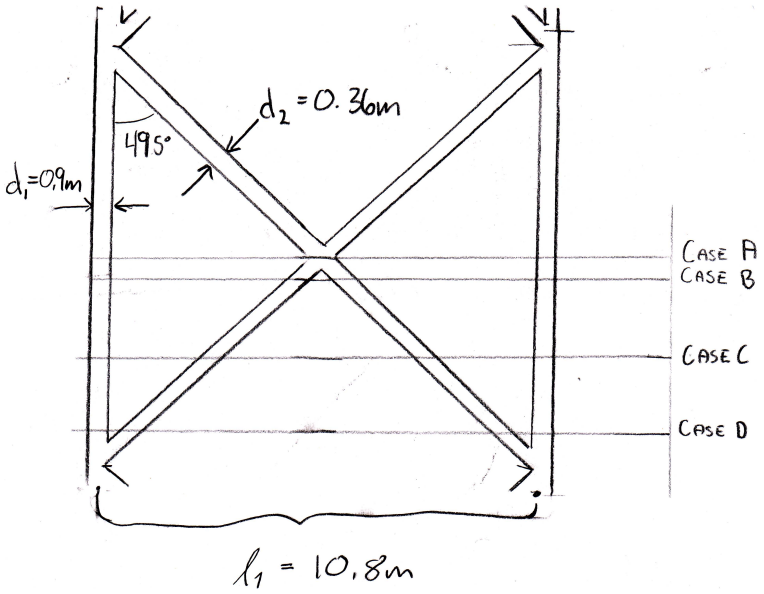


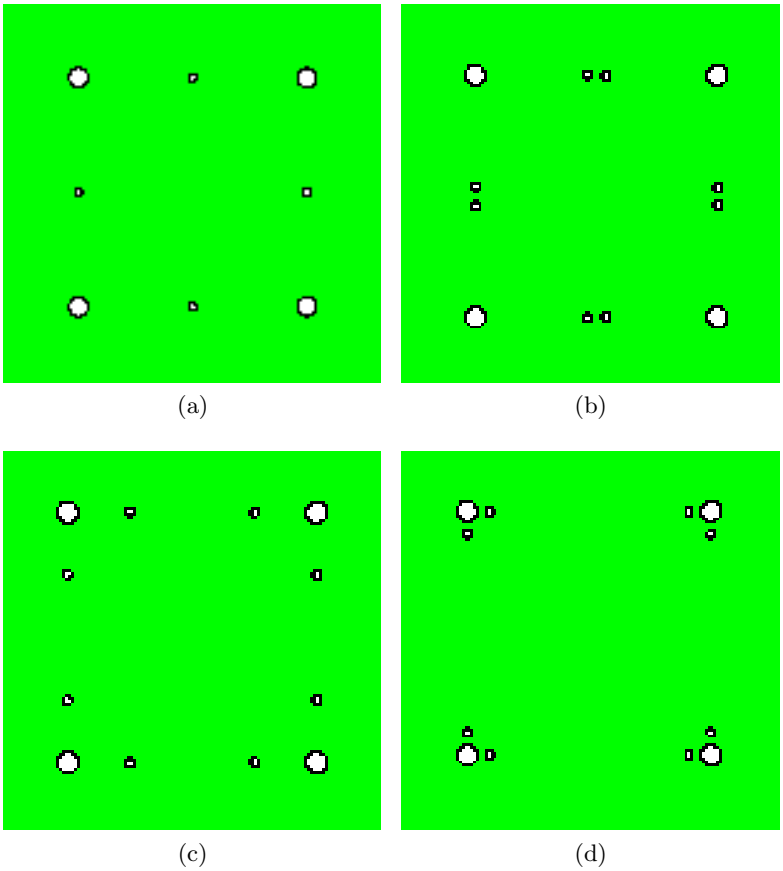
Figure 3.2: X-brace section of the truss tower. The four different cross sections that are looked at are Case A through Case D.

fourth cross section, Case-D (Fig. 3.3d), is close to where the X-brace connects with the main cylinders (bottom line in Fig. 3.2).

Both the main legs and the intersecting X-brace is angled at, respectively, 84.3 degrees and 49.5 degrees with respect to the horizontal. The cross section will therefore give an ensemble of ellipsoids, but for simplicity an approximation using circular cylinders, have been made. The main cylinders coordinates should also have a slight adjustment for the different cases, but for consistency the coordinates are kept the same.

The truss tower is also simulated at 22.5 degrees and 45 degrees angle transverse to the in-flow direction, to realize different wind directions. Figures can be found in appendix A, (Fig. A.1 and Fig. A.2).

The two dimensional geometry of the cross sections are implemented in ANSYS Gambit (Version 2.4.6; ANSYS Inc., Cantonburg USA). Each cross section is inserted in a domain spanning 17.5D, where D=4m is the diameter of the monopile, in x-direction (free-stream flow direction), and 10D in y-direction (Fig. 3.5). An additional validation case for the monopile, where a larger domain was used, is also included. Here, the domain is 27D in x-direction and 14D in y-direction.
The left boundary of the domain is defined as the velocity-inlet. To give a free-stream behavior of the flow, sidewalls with periodic boundary conditions were used. The outflow boundary was implemented as a pressure-outlet with a reference pressure of 1atm, and the cylinder surface was chosen to be wall.



(a) (b)

(c) (d)

Figure 3.3: Case1: 0 degree truss tower cases embedded in mesh. a) Case1-A. b) Case1-B. c) Case1-C. d) Case1-D

Table 3.1: Number of grid cells for the different cases

Cell number	Monopile	Truss tower			
	-	A	B	C	D
Monopile	$8.7 \cdot 10^4$	-	-	-	-
Case1	-	$3.0 \cdot 10^5$	$3.3 \cdot 10^5$	$2.5 \cdot 10^5$	$3.2 \cdot 10^5$
Case2	-	$2.1 \cdot 10^5$	$2.8 \cdot 10^5$	$2.2 \cdot 10^5$	$2.6 \cdot 10^5$
Case3	-	$2.8 \cdot 10^5$	$2.9 \cdot 10^5$	$2.1 \cdot 10^5$	$2.8 \cdot 10^5$

To minimize numerical diffusion a quadrilateral mesh was used. This also give the opportunity to have a large aspect ratio between the cells, that works well with flows that are in alignment with the cells. In the preparations for this thesis [13], where a control surface encircled the whole ensemble, there were a large area with unstructured mesh. This highly increased the computational cost. To save time, the amount of unstructured mesh in the domain was reduced. Control surfaces were added around each cylinder, and this highly reduced the number of skewed cells. Since there was no prior knowledge of where the transition would occur on the cylinder surface, unstructured mesh encircled whole cylinders.

Table **??** shows each case with total number of nodes. There number of cells are somewhat different, because the mesh needed to be adapted for the individual complex geometry.

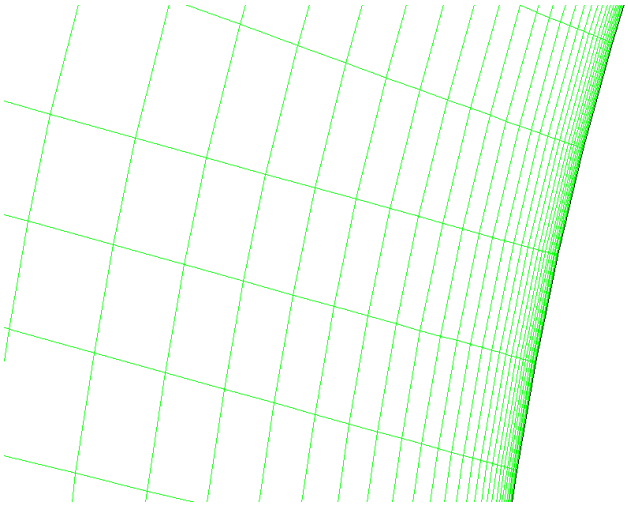


Figure 3.4: Boundary layer outside the cylinder wall. The boundary layer contains 30 layers with inner element size 0.0001 and a growth factor of 1.2.

The Reynolds number of this flow is in the supercritical regime. This means, it is a turbulent boundary layer, very close to the cylinder wall, that need careful treatment. To capture this important feature, a mesh boundary layer was added to each of the circular members. The boundary layer has 30 layers with an inner element size of 0.0001m and a growth factor of 1.2

(Fig. 3.4).

3.2 Numerical method

The geometry and mesh is implemented in the commercial software package ANSYS FLUENT (Version 12.1.4; ANSYS Inc., Cantonburg, USA). This is a software used in a wide variety of industrial applications, but here its computational fluid dynamic (CFD) capabilities are used.

In the preparations for this study [13], it was found difficult to accurately predict the flow behavior in the supercritical Reynolds number regime. The necessary ingredient was a proper initialization of the flow, which "triggered" the flow into shedding. Another essential implementation needed to capture the flow features, was the mesh boundary layer that was added around the circumference of the cylinders. The initialization of the numerical simulations followed this recipe:

1. Choose a double precision solver. This should always be the case if accuracy is of importance.
2. Read the mesh and check if it is applicable.
3. Choose the pressure-based transient solver.
4. Set velocity-inlet to 12m/s, and make sure the fluid material is air.
5. The boundary conditions should be set in GAMBIT, but it is always important to check if they are correct. Velocity-inlet for the inflow boundary, pressure-outlet for the outflow boundary, periodic for the sides and wall for cylinders.
6. Compute the reference values from the velocity-inlet. Note: Area(m2) should be set to the diameter of the cylinder in order to calculate the correct drag. The rest of the parameters are as default. Also note that the reference values only affects the calculated values and not the CFD simulation numerics.
7. Initialize the flow with the laminar model and the default solution method (The SIMPLE Pressure-velocity coupling scheme, and first order upwind spatial discretization. Standard pressure and Least Squares Cell Based Gradient discretization).
8. Run calculation with 2000 time-steps and a time-step size of 0.005s with 30 iterations for each time-step.
9. Next step is initializing with k-ω SST viscosity model. Choose this model, and set the turbulence specification method to Intensity and Length scale. Set the turbulent intensity to 10% and the turbulent length scale to 1m.
10. Change the viscosity (μ) in the order of 10^{-2} and compute the reference values again.
11. Change pressure-velocity coupling to PISO, the spatial discretizations to PRESTO! for pressure and second order upwind for momentum, turbulent kinetic energy and specific dissipation rate. Set the transient formulation to second order implicit. Gradient stays default.

12. Run calculation with 2000 time-steps and a time-step size of 0.005s with 30 iterations for each time-step.

13. Repeat from step 10 with the k-ω SST viscosity model once, only changing the viscosity to the correct value for air, μ=1.789·10^{-5}, and simulate with 6000 time-steps.

The measurement lines were added after the initialization. All lines were placed at multiples of the monopile diameter D=4m from the center of the tower. Respectively, there are lines and rake lines (401 points) transverse to the free stream flow at every multiple from 6D to 12D, and every half multiple from -5D to 5.5D, from the tower center (Fig. 3.5).

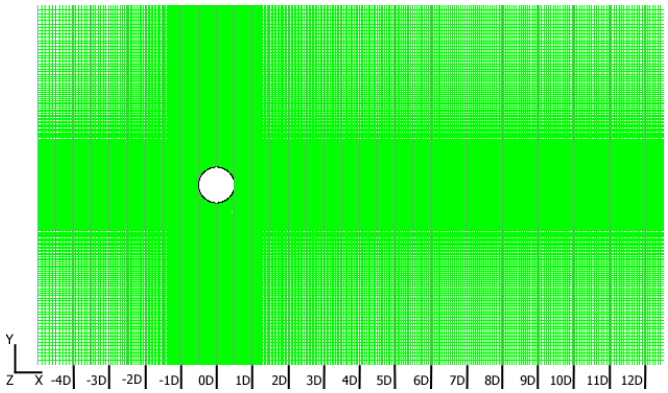


Figure 3.5: Monopile in a domain spanning 10D x 17.5D. Measuring lines are placed every multiple from -5D to 12D and every half multiple -5D to 5.5D. Inflow is on the left hand side and outflow is on right hand side. Periodic boundaries on each side.

Total pressure, velocity magnitude, x-velocity, y-velocity, turbulence intensity and production of turbulent kinetic energy (tke) are outputs from the simulations. There are also data from the pressure coefficient and y^+ on the cylinder wall, together with lift and drag coefficients. All data outputs are time series resulting in statistical data. Note that simulating with several of these outputs should be done in the text-interface version of ANSYS Fluent.

3.3 Postprocessing

The unsteady flow was simulated with a time-step size of 0.005s with 6000 time-steps, resulting in a real flow time of 30s. The time-step size was chosen in order to capture at least 25 measurements for each vortex shedding. The vortex shedding frequency is given by

$$f = \frac{StU}{D}, \tag{3.1}$$

where the Strouhal number value used is St=0.2, which is the normal value for a circular cylinder [4]. The free-stream velocity U=12m/s and D is the characteristic diameter of the cylinder. For the monopile, with a diameter of D=4m, the predicted shedding frequency becomes f=0.6=$\frac{1}{\Delta t}$

and gives a shedding each Δt=1.67s. To measure at least 25 times for each cycle, gives the time-step size of 0.05s. However, the smallest members of the truss tower has a diameter of d_2=0.36m, which gives a shedding each Δt=0.15s, which results in a measuring time-step size of 0.005s. For consistency, this time-step size have been chosen for all simulations.

As explained in section 3.2: *Numerical method*, the measuring lines are in fact rake lines with 401 points along the y-axis (transverse to the flow) and are found at distances of every half multiple from -5D to 5.5D and every multiple from 6D-12D[1] from the tower center. This means that e.g., 3D behind the tower is in reality 12m from the center of the tower. Outputs were recorded every time-step, at each rake line.
Postprocessing and analysis were performed using the last 5000 time-steps, with the statistical computing software R [24], using the MASS library [35] and custom written functions in addition. Appendix H contains details about the scripts used in the postprocessing.

3.4 Validating the model

The model was validated in a high Reynolds number regime (Re=$3.3{\cdot}10^6$). After initialization, the k-ω Shear-Stress-Transport (SST) viscosity model was able to reproduce the von Karman vortex street (Fig. 3.6), if the time-step size was chosen to be low enough (Δt=0.005s).

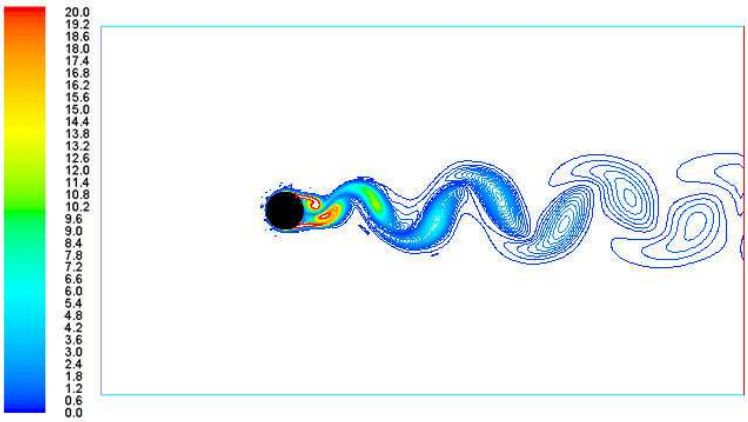
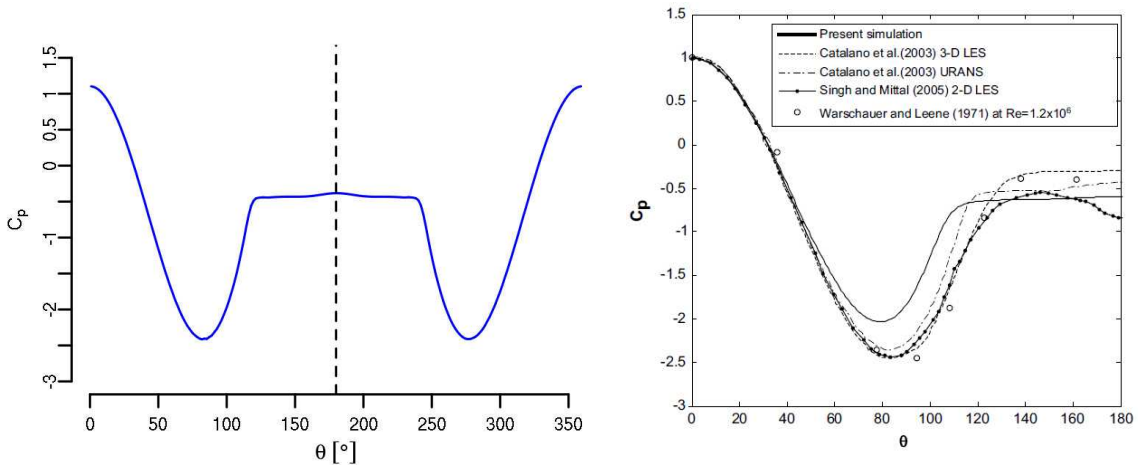


Figure 3.6: Vortex shedding behind the monopile. The colorbar to the left shows vorticity magnitude from 0-20$\frac{1}{s}$.

The pressure coefficient is calculated around the circumference of the monopile (Fig. 3.7a). Comparing with data from Ong et al. [21] (Fig.3.7b), the pressure coefficient is very well predicted in this validation.
At the stagnation point upstream of the cylinder, $\theta = 0°$, C_p is close to 1 and successfully drops to $C_p = -2.5$ at $\theta = 80°$, due to the flow separation. Then C_p rise to -0.5 before stabilizing at $\theta = 120°$. Experimental data from Warshauer and Leene [36] and Achenbach [1] confirms

[1]This varies from the monopile to the different truss tower angles, as the lines were placed from the center of the tower. E.g., the 0 degree truss tower has lines every half multiple from -6D to 5.5D and every multiple from 6D to 11D.



(a) (b)

Figure 3.7: Pressure coefficient measured clockwise around the circumference of the monopile. 0 degrees is the stagnation point in front of the cylinder. a) Simulated pressure coefficient from the monopile. b) Experimental and simulated pressure coefficients (from Ong et al.[21])

this pattern. The Reynolds number in the referred literature is not exactly the same as in the present study, but since they are in the same Reynolds number regime is it possible to qualitatively compare the results [39]. The drag coefficient C_D=0.37 is within the expected range, 0.21-0.6 (Ong et al.[21]).

It is important to capture transitions in the viscous sublayer, and to check that the computational wall y^+ was sufficiently employed. This is a non-dimensional quantity that uses viscous length units to measure the length of the boundary layer [22]. The value of y^+ should be below 5 in this Reynolds number regime, and results showed a y^+_{max}=3.5, which should be accurate enough for industrial applications [28].
At the sides of the domain the expected value is 12m/s from the velocity inlet, but Fig. 3.8 shows that the free-stream velocity is slightly higher at the sides for both domains. By simulating in a larger domain, 14D in y-direction and 27D in x-direction, it was hoped that this numerical blockage would be reduced. Only a slight reduction was seen. This indicates that expanding the computational domain would probably reduce the blockage effect, but the domain needs to be very much larger for a sufficient reduction. The larger domain will substantially increase the computational demand and is therefore not pursued further. Also, the drag coefficient (C_d=0.36) is still in the expected region of 0.21-0.6 and almost the same as in the smaller domain.

3.5 Results

The results have been divided into three different sections: Mean velocity profiles, turbulence intensity and power spectral density analysis.
Flow past a cross section of a monopile, represented as a single cylinder, and cross sections of a

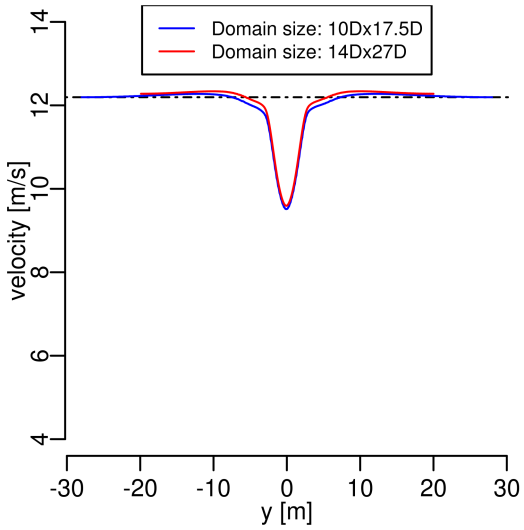


Figure 3.8: Velocity profiles from the large and small domain used for the monopile

truss tower have been studied. The truss tower is represented at four different cross sections depending on the height from the X-brace. Case A, has four main cylinders of diameter d_1=0.9m, representing the main legs of the truss tower and four smaller cylinders with diameter d_2=0.36m, representing the joint section of the X-brace. Case B,C,D the same main cylinders as Case A and eight smaller cylinders (still with diameter d_2=0.36m) representing the cross section further down the X-brace (Fig. 3.2). This is explained in detail in section 3.1: *Geometry and mesh*.

The truss tower is also angled 0 degrees, 22.5 degrees and 45 degrees with respect to incoming wind direction, respectively called Case1, Case2 and Case3. This results in a total of twelve different truss tower cases.

3.5.1 Mean velocity profiles

The features of the flow vary with changing Reynolds number [39]. In this study the Reynolds number lies in the supercritical Reynolds number regime, and is $7.4{\cdot}10^5$ for the truss tower, calculated with the diameter of the main cylinders d_1=0.9m and incoming wind speed of 12m/s. The Reynolds number for the monopile (with D=4m) is $3.3{\cdot}10^6$.
When flow passes a circular cylinder the pressure increases at the stagnation point in front (upstream) of the cylinder. Here, the velocity drops and the flow exhibits a speed-up to the sides of the cylinder. If the Reynolds number is high enough, that is higher than about 400, the flow separates from the cylinder surface, which again results in eddies (Fig. 3.9). The separation points move towards the front of the cylinder as the Reynolds number increase. The velocity field behind the cylinder changes and a velocity deficit can be observed. The deficit is due to unsteady motion, or turbulent behavior, which are eddies that grow larger downstream. These eddies, or vortices, are the famous von Karman vortex street (Fig. 3.9). At some distance downstream of the cylinder the velocity deficit will be minimal and the wake fully developed.
For the truss tower, which includes more than one cylinder, the wake from each cylinder could

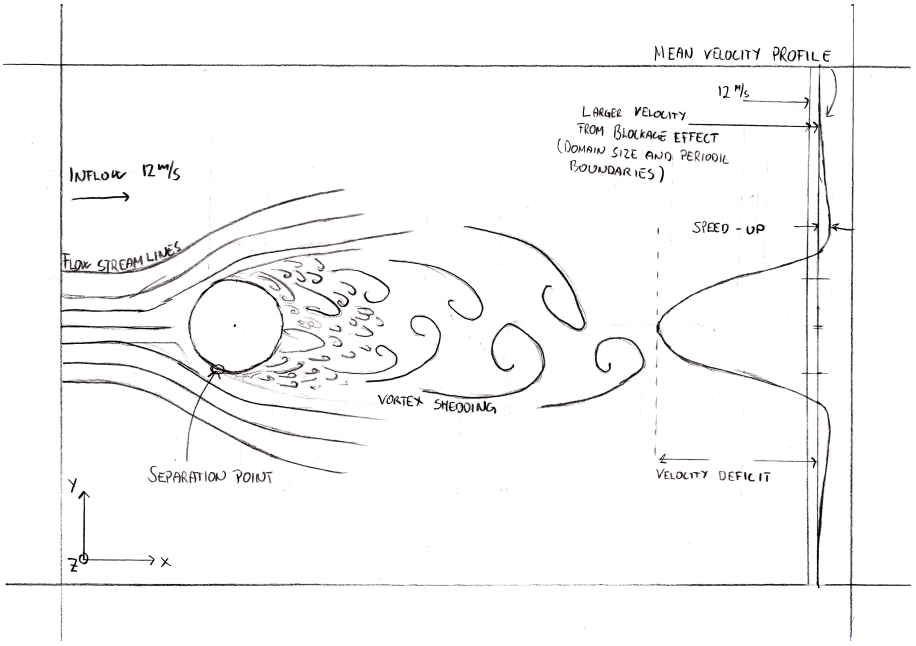


Figure 3.9: Authors drawing of flow past a single circular cylinder. An example of the mean velocity profile downstream of the cylinder is shown to the right in the drawing.

interact, depending on the spacing between the cylinders [16, 18]. This is also expected to have an influence on the wake development.

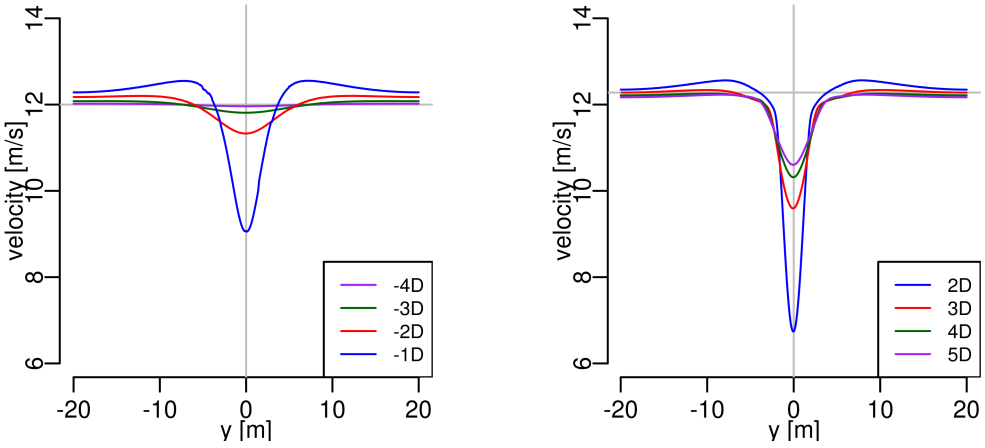
In this section, the numerical results of the mean velocity profiles are analyzed. The velocity deficits have been measured at certain distances behind the towers. The region for distances from the cylinder wall up to 5D downstream the cylinder (where D=4m is the monopile diameter), will be called the near-wake region. This region is expected to have the largest velocity deficit. The region from 5D to 9D will be called the far-wake region. Note, the wind turbine blade for a downwind rotor will pass through the tower shadow region at about 3D behind the tower. However, both near- and far-wake region have been studied in order to understand the wake development.

Mean velocity profiles for the monopile

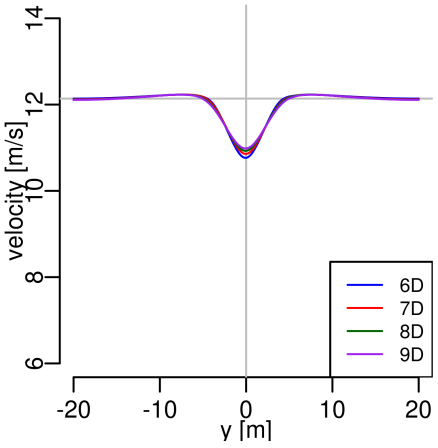
The first noticeable feature upstream of the cylinder is the high velocity magnitude at the side of the domain at y=±20m (Fig. 3.10a). The velocity is expected to be similar to the inflow velocity of 12m/s, but due to periodic boundaries and insufficient domainsize, this is somewhat higher. This numerical blockage effect is already slightly visible at 4D upstream of the cylinder, and at 1D upstream the velocity magnitude is 12.3 m/s.

The numerical model is able to capture the expected speed-up to the side of the cylinder. The speed-up is about 2% at 2D downstream, and it is still slightly visible further downstream.

A single 'dip', which is symmetric at the centerline (y=0m), represents the time-average ve-



(a) (b)



(c)

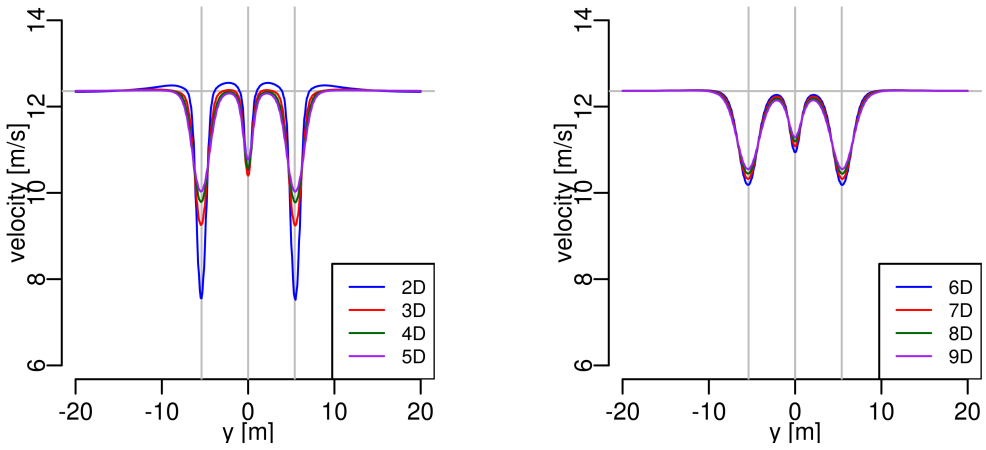
Figure 3.10: Mean velocity profiles for the monopile. The vertical line indicates the centerline. a) Upstream of the cylinder. b) Near-wake region, 2D-5D downstream of the cylinder. c) Far-wake region, 6D-9D downstream of the cylinder.

locity deficit up- and downstream of the cylinder (Fig. 3.10). As expected, the deficit is largest close to the cylinder. Upstream at the centerline, the velocity deficit dip is first seen at a distance 3D from the cylinder, where the velocity is 11.8m/s. From 3D to 2D upstream, the velocity deficit increase, about 4%, and another 20% from 2D to 1D. At 1D upstream at the centerline, the velocity is found to be 9m/s. This means that the cylinder already affects the flow upstream.

At 2D downstream, the velocity is 6.8m/s at the centerline. This is a reduction of almost 50% from the inflow of 12m/s. Note that the velocity at the domain sides (y=±20m) is higher than 12m/s due to limited domain size and periodic boundary conditions, as mentioned above. From 2D to 3D the minimum velocity magnitude increase about 30%, from 6.8m/s to 9.6m/s and another increase of 8% from 3D to 4D. Moving towards the far-wake region, where the wake is almost fully developed, there is only 2% increase in the minimum velocity magnitude from 4D to 5D. In the far-wake region the increase is minimal, about 2% change from 5D to 9D.

Mean velocity profiles for truss tower - Case1

The setup of the truss tower cross sections were designed at four different heights, giving the smaller cylinders different coordinates (Fig. 3.2 and Fig. 3.3 from section 3.1 *Geometry and mesh*). The main cylinders were fixed for all cases. The velocity profiles show several dips, respectively behind each of the cylinders.



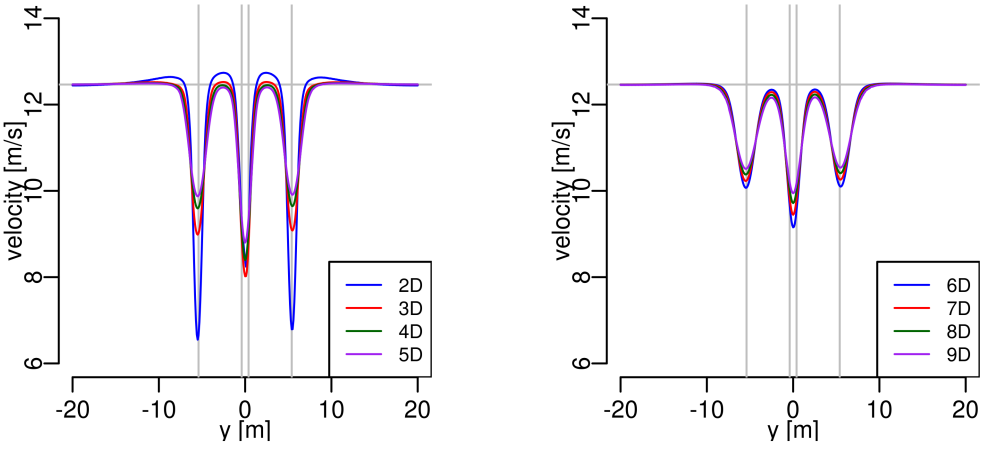
(a) (b)

Figure 3.11: Mean velocity profiles for the truss tower Case1-A. The vertical lines indicates the center of each cylinder. a) Near-wake region, 2D-5D downstream of the cylinders. b) Far-wake region, 6D-9D downstream of the cylinders.

In truss tower Case1-A, were the tower is angled 0 degrees transverse to the inflow there are three tandem configurations (Fig. 3.3a). One with two small cylinders (in the middle) and two with two main and one small cylinder (left and right). This gives three distinct dips in the

velocity profile (Fig. 3.11).
As expected, is the velocity deficit behind the main cylinders larger compared with the deficit behind the small cylinders. At 2D downstream, the minimum velocity magnitude is 7.6m/s behind the main cylinders and 10.5m/s behind the smaller cylinders. This is a difference of about 28%.
The wake downstream of the small cylinders seems fully developed already at 2D, and does not change much further downstream[2]. The velocity deficit behind the small cylinders decrease about 4% from 2D to 9D.

At 3D downstream of the main cylinders, the minimum velocity is 9.2m/s, which is an increase of 18% from 2D. Another 6% increase from 3D to 4D and 3% from 4D to 5D. Here, the wake is fully developed, with a velocity of 10.5m/s in the far-wake region, corresponding to a total velocity deficit decrease of about 2%.



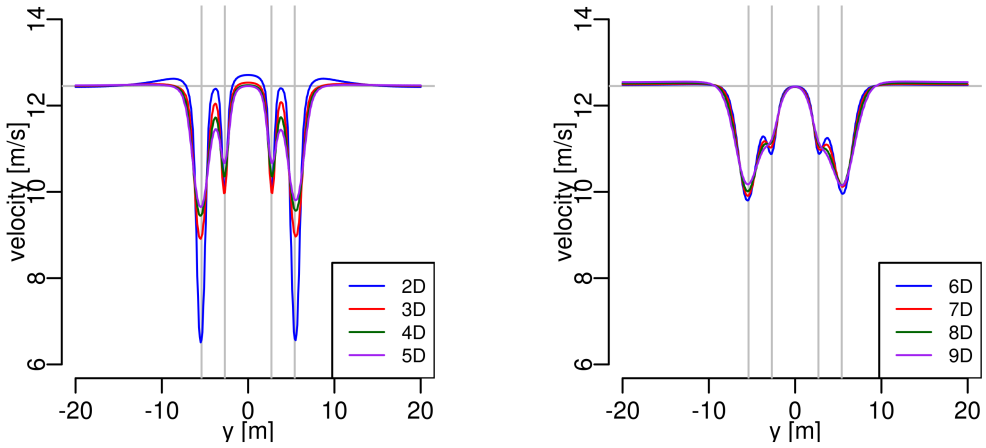
(a) (b)

Figure 3.12: Mean velocity profiles for the truss tower Case1-B. The vertical lines indicates the center of each cylinder. a) Near-wake region, 2D-5D downstream of the cylinders. b) Far-wake region, 6D-9D downstream of the cylinders.

Case1-B,C,D have in total eight smaller cylinders in addition to the four main cylinders (Fig. 3.3). There are now two tandem configurations including two main cylinders and two small cylinders, and additional two tandem configurations including only two small cylinders.

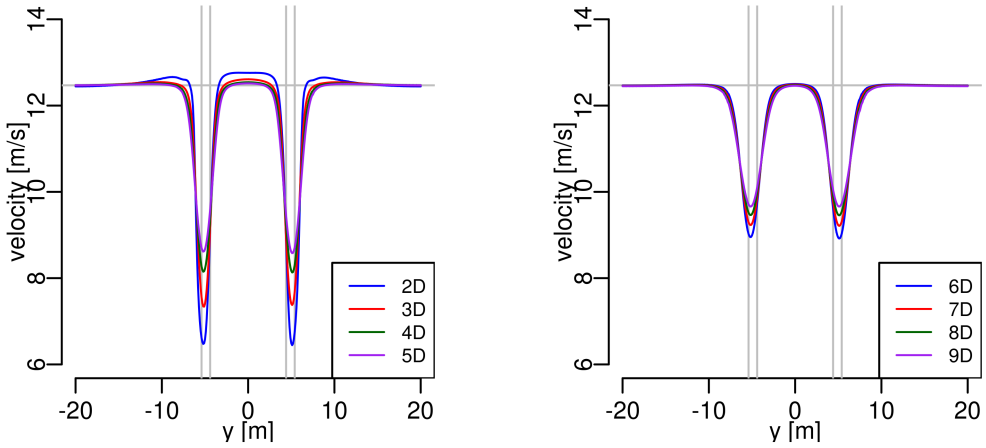
The velocity deficit at 2D downstream of the main cylinders, have a minimum velocity magnitude of about 6.4m/s (Fig. 3.12a, Fig. 3.13a and Fig. 3.14a). It seems that the additional small

[2]The diameter of the small cylinders is 1/10 the size of the monopile. Which would make 2D a far wake region for the small cylinders. For consistency the near- and far-wake regime are kept as a multiple of the monopile diameter D.



(a) (b)

Figure 3.13: Mean velocity profiles for the truss tower Case1-C. The vertical lines indicates the center of each cylinder. a) Near-wake region, 2D-5D downstream of the cylinders. b) Far-wake region, 6D-9D downstream of the cylinders.



(a) (b)

Figure 3.14: Mean velocity profiles for the truss tower Case1-D. The vertical lines indicates the center of each cylinder. a) Near-wake region, 2D-5D downstream of the cylinders. b) Far-wake region, 6D-9D downstream of the cylinders.

cylinders, between the main cylinders, give larger velocity deficits, compared to Case1-A (about 15% difference).

Case1-B has, in addition to the tandem arrangements, two smaller cylinders in a side-by-side arrangement at a distance $1.2d_2$ from center to center (d_2=0.36m is the diameter of the small cylinder). The velocity deficit dip in Fig. 3.12a) is 8.3m/s at 2D and, interestingly, its even lower at 3D, about 8.0m/s. Further downstream, the velocity magnitude only has about 3-4% increase at each diameter distance from 4D to 9D. This means that the wake behind the close side-by-side arrangement has developed earlier, compared with the wake from the single small cylinder in Case1-A. For Case1-B, the small cylinder wake also give a larger deficit at 3D and further downstream, compared behind the main cylinders. It is likely to believe that this effect is caused by the wake interactions [18].

Behind the main cylinders, 2D to 3D, it is a velocity increase of 30%, from 6.4m/s to 9.0m/s. Another increase to about 7% from 3D to 4D, and 4% from 4D to 5D. In the far-wake region the wake is almost fully developed, and the total change from 5D to 9D is only about 4%, which is similar to Case1-A.

Case1-C (Fig. 3.13) shows a very similar wake pattern behind the small cylinders, compared to the single small cylinder in Case1-A (Fig. 3.11). This means there are sufficient distances, which is 5 cylinder diameters according to Zdravkovich [41], between the small cylinders to avoid interference between the wakes. The largest difference for this case, as mentioned above, is the velocity deficit which is 15% lower behind the main cylinders compared with Case1-A. However, the wake behind the smaller cylinders seems to affect the wakes behind the main cylinders further downstream, making the deficit slightly larger (about 4%) compared with Case1-A. This is probably because the eddies grow larger in the far-wake region and absorbs the smaller eddies.

For Case1-D (Fig. 3.14) only two distinct dips were expected, because the small cylinders are placed very close (with a gap of $1d_2$) to the main legs [2, 12]. Note that experiments in the earlier literature have a different Reynolds numbers. A similar behavior is, however, expected. Interestingly, no significant differences for the minimum velocity magnitude behind these clusters are visible, compared with Case1-B,C. The wake behind the side-by-side arrangements and the tandem arrangements behaves very differently. The tandem and side-by-side arrangements exhibit a larger velocity deficit, but the wake develops earlier for the side-by-side configuration, resulting in a large deficit through out the wake.

From 2D to 3D for Case1-D, the minimum velocity magnitude changes from 6.4m/s to 7.4m/s, which is only 14% compared to Case1-B and Case1-C, which has a change of 30%. An additional 9% change from 3D to 4D and 4% change from 4D to 5D. The wake already seems fully developed and change about 2% every diameter from 5D to 9D (8.5m/s to 9.7m/s).

Mean velocity profiles for truss tower - Case2 and Case3

The results from Case2 and Case3 will not be discussed thoroughly here, but the mean velocity profiles for all cases can be found in appendix B.

After analyzing the data from the more extreme Case1, it is clear that cylinder wake interactions have a great influence on the mean velocity profile. Flow past the truss tower Case2 would perhaps be the most representative case, since it is not expected that the wind will affect exactly aligned members. The wake behind Case2 exhibits more wake interactions, because the members are in a staggered arrangement (Fig. 3.15). Interestingly, the velocity deficit differences between the near- and far-wake region for Case2, are not very large. It is also overall lower compared with Case1 and the monopile. It seems that by avoiding tandem configurations, the velocity deficit is reduced to a large extent, especially in the near-wake region. Case3 is somewhat similar to Case1, since both have members aligned in tandem configurations to the flow direction. However, there are more tandem configurations and thereby more deficit dips, which behaves similarly to Case1.

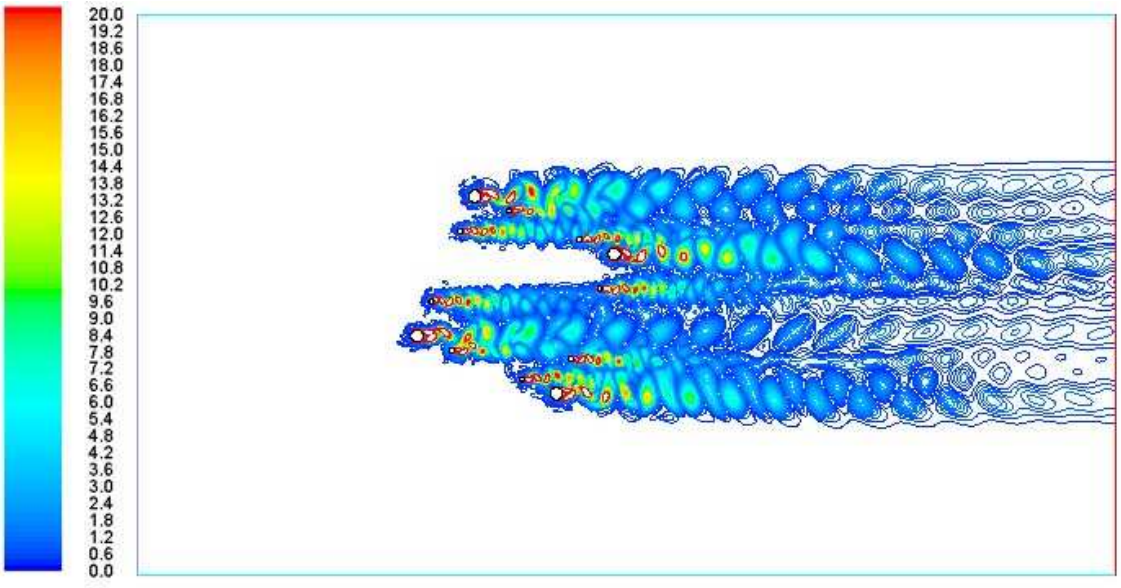


Figure 3.15: Vorticity magnitude[1/s], Case2-C. There are clearly wake interactions downstream of the tower.

3.5.2 Turbulence intensity

Offshore, the wind speed has effect on surface roughness [10] and the turbulence intensity is a function of the wind speed [14]. In an offshore environment, where the wave height increases with higher wind, the turbulence intensity in the wind could be high. The turbulence intensity TI, is given by the ratio between velocity fluctuations (u') and the mean free-stream velocity U_0,

$$TI = \frac{u'}{U_0} \ . \tag{3.2}$$

For a wind speed at U_0=12m/s, which is used in this study, the turbulence intensity could be about 6-10% depending on the height above sea level [33]. The turbulence intensity added to the inflow has therefore been chosen to be 10%, with a turbulent length scale of 1m. The latter should be chosen to be of the same scale as the expected size of the vortices that are exhibited behind the structure. For wind turbines, such turbulence intensities could result in damaging fatigue loading to e.g., the wind turbine blades. When further using a downwind turbine, the wind will exhibit additional unsteady motion because of the tower. This will probably induce even more damage to the blades.

In this study two tower options have been tested for the downwind turbine, by simulating two dimensional cross sections of a monopile and truss tower. The turbulence intensity behind the towers is analyzed, without including wind turbine blades.

The turbulent flow can be split into three components: mean flow, unsteady motion and high-frequency turbulence. Usually, Reynolds-Averaged-Navier-Stokes (RANS) is used for steady simulations, which means there is no additional contribution fra fluctuations. In this study, on the other hand, the RANS approach have been used to do transient simulations. This means that TI is only calculated (in Fluent) from the parameterized turbulence, which gives a substantially lower turbulence intensity, compared to reality. It is therefore important that the contribution from the fluctuations is added to the calculated TI.

The high-frequency turbulence is actually represented by a sub-grid parameterization. This means that the mean velocity, $\overline{U}_i$, in each cell for each time-step i is actually an ensemble velocity mean $< U_i >$,

$$\overline{U}_i =< U_i >= \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} U_i^{(j)}, \tag{3.3}$$

where j is the ensemble index. The turbulence fluctuations at time-step i from the sub-grid parametrization can therefore be given as:

$$\sigma_{\text{ensemble,i}} = \sqrt{< U_i^2 > - \overline{U_i}^2} \ . \tag{3.4}$$

Here, U_i is a random velocity at time-step $i = 1, 2, ..., N$. Note, that this is the turbulence

intensity calculated by Fluent.

The other contribution is from the unsteady motion or the fluctuations in the flow. The turbulence intensity from this component is given by

$$\sigma_{\text{unsteady}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\overline{U_i}^2 - \overline{U}^2}, \tag{3.5}$$

where the mean velocity is

$$\overline{U} = \frac{1}{n}\sum_{i=1}^{N}\overline{U}_i, \tag{3.6}$$

and $\overline{U_i}$ is the ensemble mean velocity at time-step i. With both these contributions the total turbulence intensity is

$$TI_{(tot)} = \frac{\sqrt{\sigma_{\text{unsteady}} + (\frac{1}{N}\sum_{i=1}^{N}\sigma_{\text{ensemble},i})}}{U_0} \tag{3.7}$$

$$= \frac{\sqrt{\sigma^2}}{U_0}\ . \tag{3.8}$$

A detailed proof is given in appendix G.

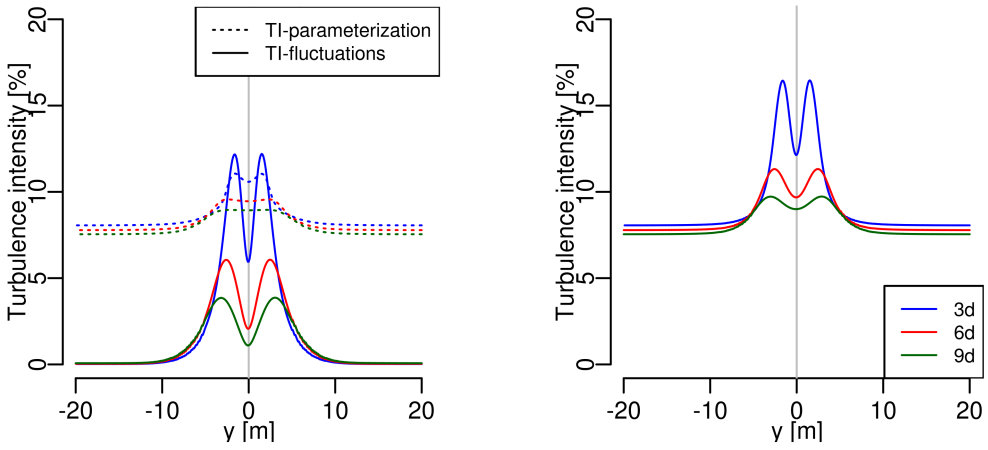
The turbulence intensity will be studied at 3D, 6D and 9D downstream of the tower. As before, the near-wake region is regarded as 2D-5D and the far-wake region is from 5D and downstream.

Turbulence intensity for the monopile

The turbulence intensity from the fluctuations clearly results in the largest contribution behind the monopile (Fig. 3.16). This is as expected because of the vortex shedding that develops. As the vortices shift from left to right there are two distinct peaks, or twin peaks at approximate $y = \pm 2$m. At 3D downstream of the monopile, the turbulence intensity is highest, about 12% where the vortices are, but just 6% at the centerline (y=0). In the far-wake region, at 6D, this changes to 6% for the highest peak and 2.5% for the lowest.

The parameterized turbulence intensity give similar, but smaller, peaks at $y = \pm 2$m. This means there are contributions, both from the turbulent kinetic energy and from the fluctuations behind the structure. At the sides of the domain, only the parameterized turbulence contributes to the turbulence intensity. From the inflow this was set to be 10%, but at 3D downstream this has dropped to 8%, due to dissipation. At 6D and 9D an additional drop from 8% to 7.5% is visible (Fig. 3.16).

As expected, the turbulence intensity is highest close to the cylinder, 3D downstream. Here, the



(a) (b)

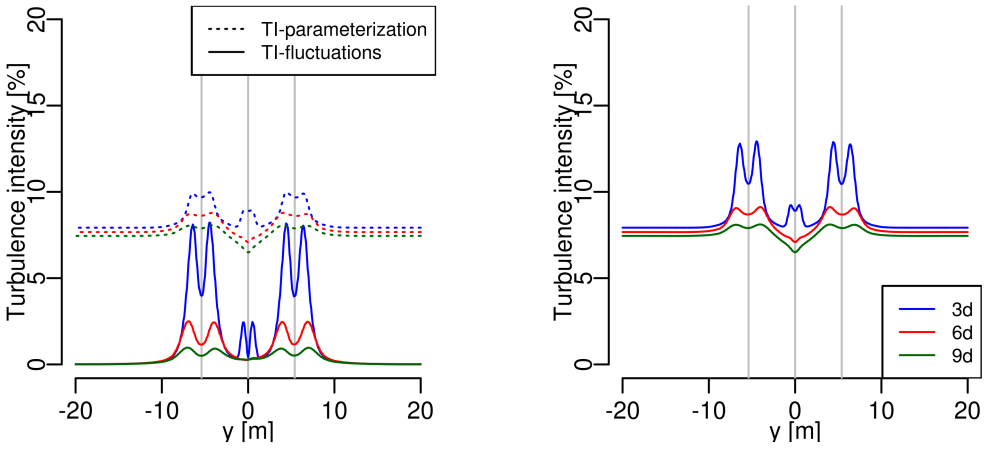
Figure 3.16: Turbulence intensity for the monopile. a) The two different components. b) Total turbulence intensity

largest total turbulence intensity is 15%, and is found as two twin-peaks behind the monopile. The total turbulence intensity between the peaks drops to 12.5%. At the sides, the total turbulence intensity is 8%, because it only has contributions from the parameterization. Further downstream, the total turbulence intensity is maximum 12% at 6D, with a dip to 9.5% between the two twin-peaks. The twin-peaks are less distinct at 9D, with a maximum intensity of 9%.

It is clear that parameterized turbulence contributes more to the total turbulence intensity at the sides of the cylinders, and that unsteady motions gives the largest contribution behind the monopile. This shows that it is very important to consider both contribution and that it needs to be properly calculated. Considering that a tower shadow model can represent the mean velocity profiles, this could result in large underpredictions in a fatigue analysis. This is discussed in chapter 4: *Tower shadow models.*

Turbulence intensity for truss tower - Case1

From what was seen behind the monopile, a twin-peak is expected behind each individual circular member in the cross section of the truss tower, with the dip behind the center of the cylinders (y=0m). This is visible for truss tower Case1-A (Fig. 3.17). The twin-peaks have their main contribution from the fluctuations. At 3D downstream, the maximum total turbulence intensity is 12.5% with the dip at 10%. At the sides (y=±20m), the turbulence intensity is the same, 8%, as for the monopile. This should be expected because the inflow is the same, and no fluctuations generated from the cylinders should affect this region significantly. The same values are found behind the main cylinders in Case1-B and Case1-C (Fig. 3.18 and Fig. 3.19).
Behind the smaller members in Case1-A, the maximum intensity at 3D downstream is 9%. Its



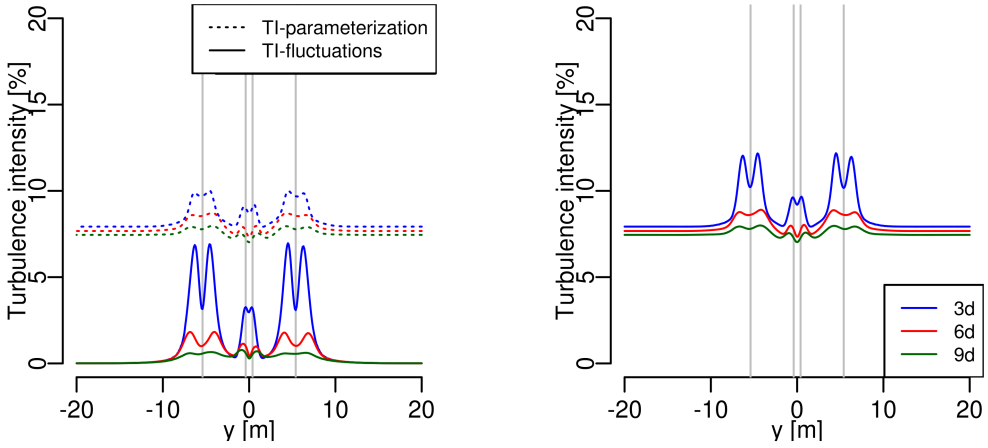
(a) (b)

Figure 3.17: Turbulence intensity for the Case1-A. The vertical lines indicate the center of each cylinder. a) The two different components. b) Total turbulence intensity

interesting that the intensity is lower behind the smaller cylinders, at 6D and 9D, than the initial inflow intensity. The vortex shedding from the cylinders have decayed and reduced the total turbulence intensity to be lower here than at the sides. This is also seen for Case1-B (Fig. 3.18), but here some shedding is visible at 6D (and even 9D). From the mean velocity profiles (see section 3.5.1) it was understood that a side-by-side arrangement gave a slower decay for the vortex shedding and this is also seen here. Note, however, that in the far-wake regime have the same total intensity behind the small member and at the domain sides.
The features discovered in Case1-C (Fig. 3.19) is very similar to Case1-A. But it is interesting to see that the shedding generated behind the small members is absorbed by the large shedding from the main cylinders, without increasing the turbulence intensity behind the main cylinders. The most interesting feature is seen behind the clusters in Case1-D (Fig. 3.20). It seems that maximum total intensity is still 12.5%, but that one of the twin-peaks is one percent lower. This means it is not much increase in turbulence intensity due to the interacting wakes. At 6D and 9D, the differences are small compared with the other cases. It seems that wake interactions from the clusters do not increase the turbulence intensity. A side-by-side arrangement could, however, make the total turbulence intensity higher further downstream.

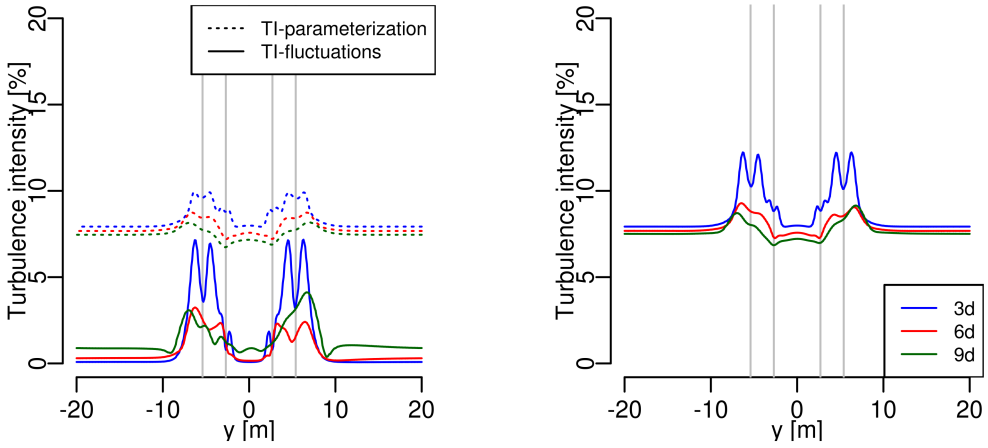
Turbulence intensity for truss tower - Case2 and Case3

The turbulence intensity behaves similarly for Case2 and Case3 as for Case1. The largest difference is that the last cylinder affected by the flow, contributes to higher turbulence intensity. The main reason is that the measuring lines are from the center of the tower and are now closer to the last cylinder. This is most evident for Case3, where the last cylinder is closest to the measuring lines.



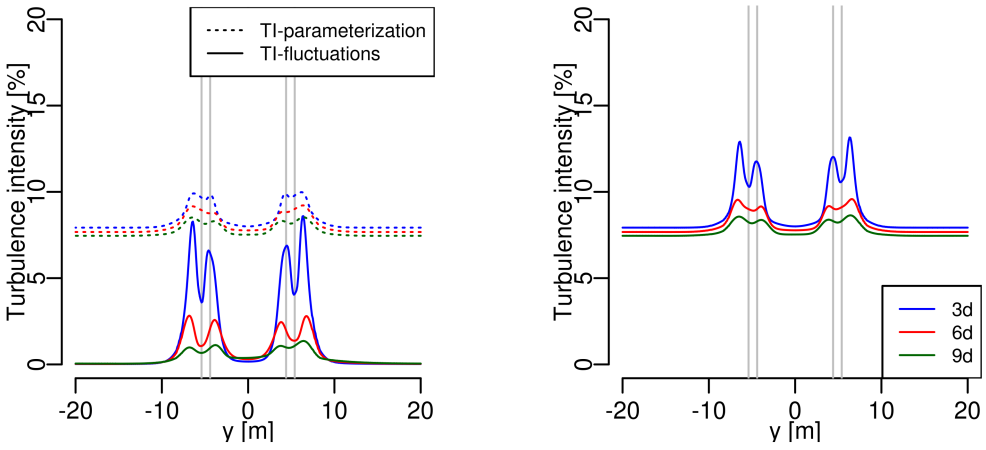
(a) (b)

Figure 3.18: Turbulence intensity for the Case1-B. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



(a) (b)

Figure 3.19: Turbulence intensity for the Case1-C. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



(a) (b)

Figure 3.20: Turbulence intensity for the Case1-D. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulent intensity

The same features is, however, seen. There is only a small contribution from the small fluctuations and there is highest turbulence intensity behind the main cylinders. The tandem configurations in Case1 and Case3 seems to give more contribution to the intensity, compared with the staggered arrangement in Case2. All plots for Case2 and Case3 can be found in appendix C.

3.5.3 Power spectral analysis

When the wind is passing a structure like a monopile or a truss tower, an unsteady motion will develop. This unsteady motion is also experienced at the area where the blades of a downwind wind turbine rotate, and could induce damaging fatigue loads on the blades. In a two dimensional environment, which in this study means flow around cylinders, the fluctuations are mainly seen as a von Karman street. This vortex shedding will occur with a certain frequency. However, the flow will also exhibit other frequencies due to transient behaviour.

For flow past a circular cylinder, the expected frequency of the vortex shedding is made non-dimensional by the Strouhal number. The Strouhal number is usually found to be about 0.2 for a circular cylinder, but having a smooth surface on the cylinder makes this larger. In this study the cylinder surface is smooth and will therefore give a Strouhal number between 0.2-0.47 (Fig. 3.21).

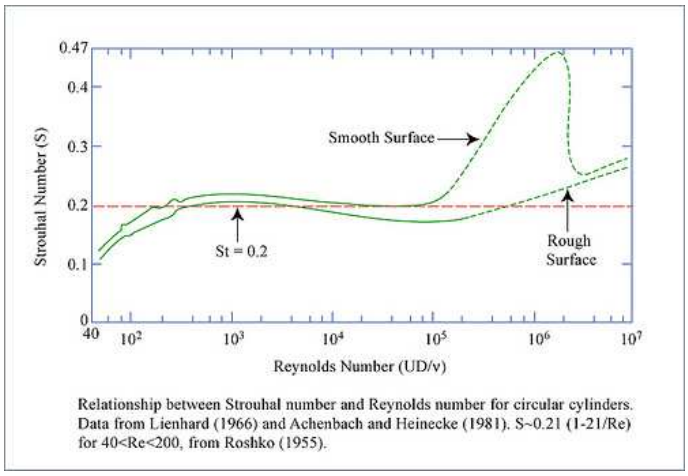
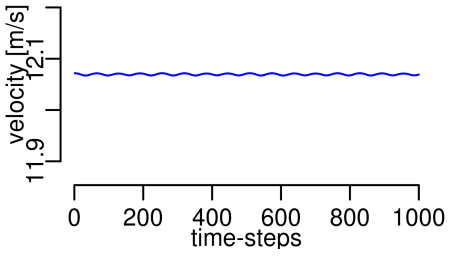


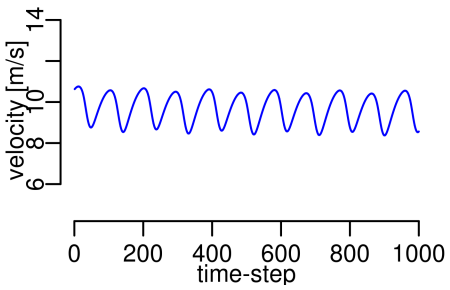
Figure 3.21: Strouhal number vs Reynolds number (from [31])

As the frequency is calculated from the time-series, they would give indications of what to expect in spectral analysis. In the time-series from the inflow (Fig. 3.22a), measured very close to the sides of the domain 3D in front of the towers, there are not much oscillations. It could be expected that some high frequencies will be found, but the fluctuation magnitude would be negligible.

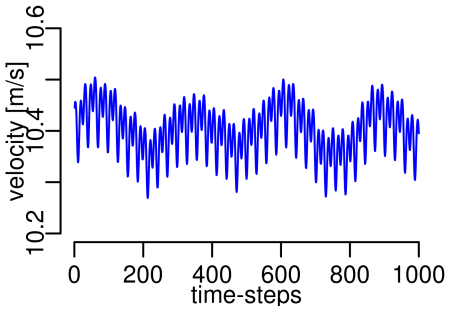
Behind the monopile, the situation is very different (Fig. 3.22b). The time-series at the centerline, 3D downstream of the cylinder, exhibits regular fluctuations and the velocity changes with a certain frequency. This is a result from the single von Karman street, that is expected behind a single cylinder. The shedding is fully developed as the oscillations follow the same pattern during a period of time.
The vortex shedding behind the truss tower Case1-A is also fully developed (Fig. 3.22c). There is a clear pattern, but the regular single oscillation is not visible anymore. Instead, it seems that there are (at least) two major frequencies. One high frequency that is modulated on a lower



(a)



(b)



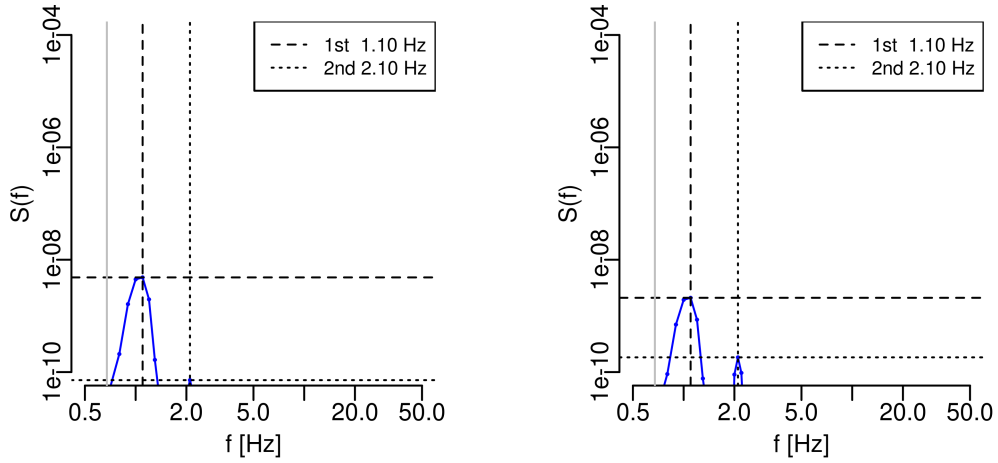
(c)

Figure 3.22: Time series for the inflow, a) 3D upstream of the monopile measured at the centerline(y=0m). b) 3D downstream of the monopile measured at the centerline. c) 3D downstream of the truss tower measured at the centerline

frequency. Comparing the time-series behind the monopile and the truss tower, it is expected that flow behind the different cross sections will exhibit different frequency content.

The shedding frequency is calculated from the velocity magnitude fluctuations using the Fast Fourier transform (FFT) algorithm without any smoothing, to capture distinct peaks.

Monopile



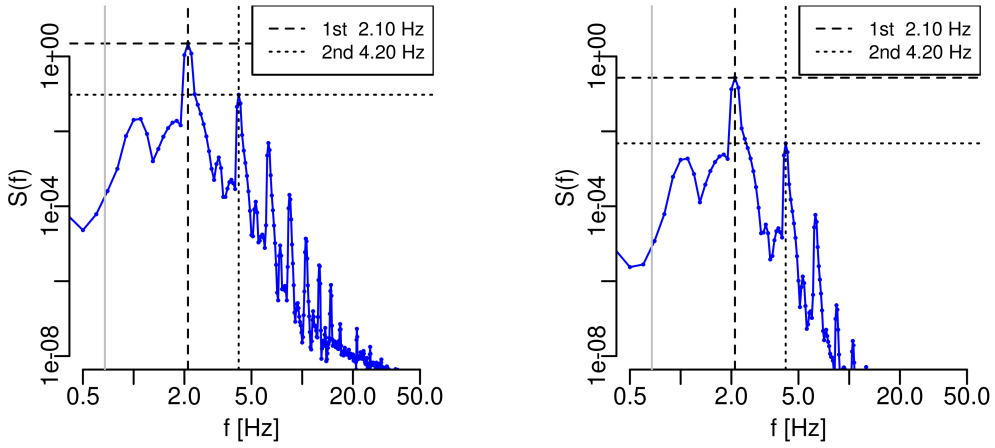
(a) (b)

Figure 3.23: Power spectral density, 3D upstream of the monopile. a) Measured at y=+20m. b) Measured at the centerline(y=0m)

For a circular cylinder with a diameter $D = 4$m the Strouhal number is expected to be between 0.2-0.47 (Fig. 3.21). This relates to a shedding frequency behind the monopile between 0.6 and 1.41Hz.

At the centerline (y=0) in front of the monopile only two frequency peaks at 1.10Hz and 2.10Hz are visible (Fig. 3.23b). They have a low power magnitude, about 10^{-9}. This is close to the detecting range and could be numerical instabilities. The same peaks are also found to the side upstream, but with the same low magnitude (Fig. 3.23a).
The same frequencies are found at 3D downstream, but with higher magnitude, close the order of 10^0 (Fig. 3.24a). This probably mean that they are not numerical instabilities upstream, but inherited from the unsteady motions downstream. None of these frequencies are in the expected vortex shedding frequency range. However, there is an additional peak at 1.2Hz, which could be from the vortex shedding that develops behind the monopile, resulting in a Strouhal number of 0.4. This have a power density in the order of 10^{-1}.

At 6D the frequency situation is very similar. The only difference is that the power density



(a) (b)

Figure 3.24: Power spectral density downstream of the monopile. a) 3D downstream. b) 6D downstream

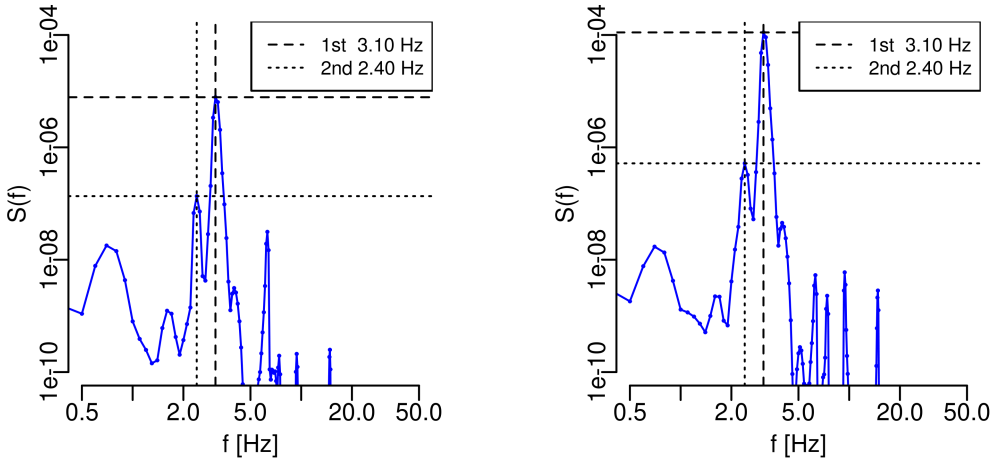
is overall lower than at 3D. Now the peak at 2.10Hz has the density of 10^{-1} and the peak at 4.20Hz has a density of 10^{-2} (Fig. 3.24b). This implies that vortex shedding still occur, but less prominent.

Truss tower - Case1

There are two different power spectra upstream of the truss tower Case1-A (Fig. 3.25). The one measured at the side has a power density in the order of 10^{-6}, which is considered very low (close to the detecting range) (Fig. 3.23). The two highest peaks are in the order of 10^{-5}, one at 0.70Hz and one at 3.10Hz. The latter peak is also found with a higher power density, in the order of 10^{-4}, at the centerline.
The frequencies change behind the truss tower (Fig. 3.26). As expected, there is an increase in power density and the peaks are found in the order of 10^{-2}. The two main peaks are found at 0.70Hz and 13.90Hz. Interestingly, is seems that the 0.70Hz frequency is inherited from the upstream motions, but with a higher density. The 3.10Hz peak is also visible in the spectra, but with low power density.

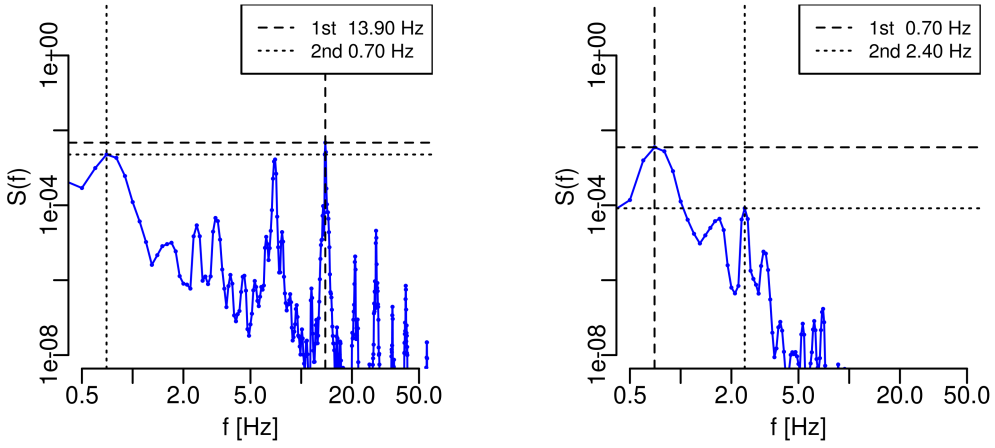
A frequency of 13.90Hz give a Strouhal number of 0.417 for the smaller cylinders, with diameter d_2=0.36. This makes sense in the supercritical Reynolds number regime (Fig. 3.21). A third peak is found at about 6Hz. Knowing that the main cylinder diameter is d_1=0.9, the resulting Strouhal number will be 0.45, which also seems reasonable.

At 6D downstream, the frequencies from 3D have disappeared. The largest power magnitude, in the order of 10^{-2}, is found at 0.70Hz. Another frequency peak is found at 2.40Hz has a power



(a) (b)

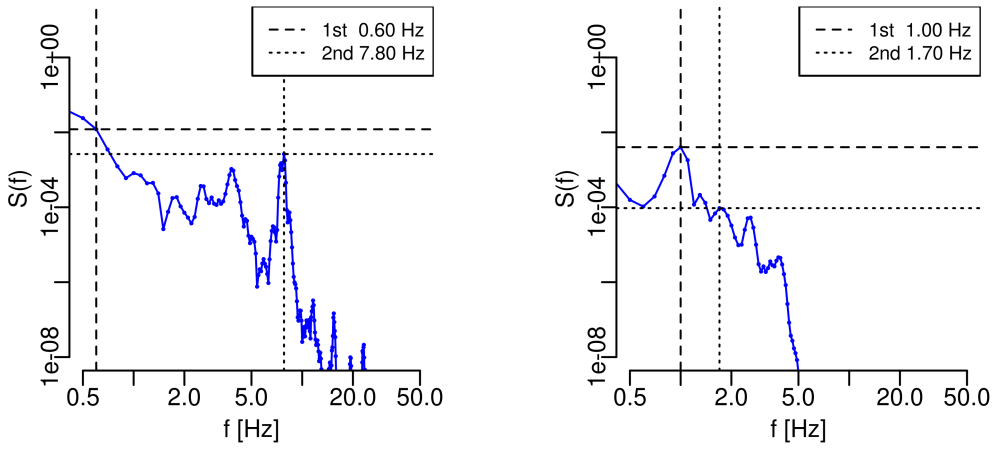
Figure 3.25: Power spectral density, 3D upstream of the truss tower, Case1-A. a) Measured close to the domain sides. b) Measured at the centerline.



(a) (b)

Figure 3.26: Power spectral density downstream of the truss tower, Case1-A. a) 3D downstream. b) 6D downstream

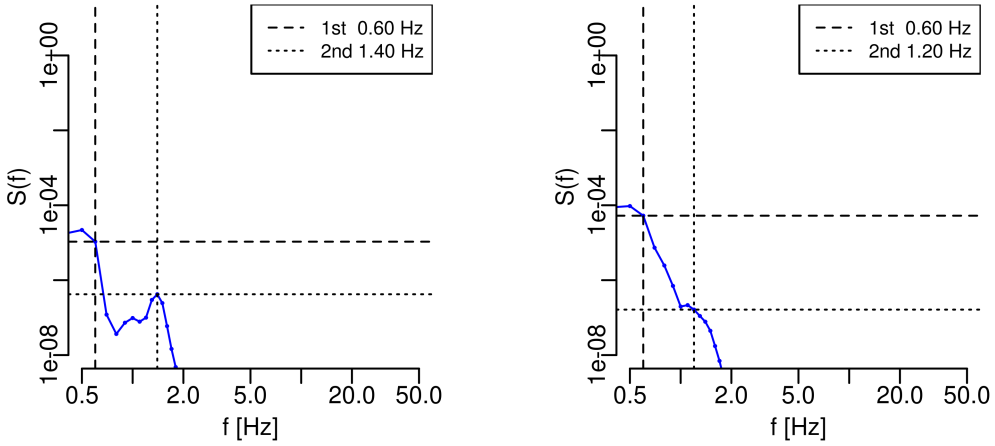
density in the order of 10^{-4}. The latter could be a result from wake interactions and that some of the shedding is less prominent, which results in a different frequency.



(a) (b)

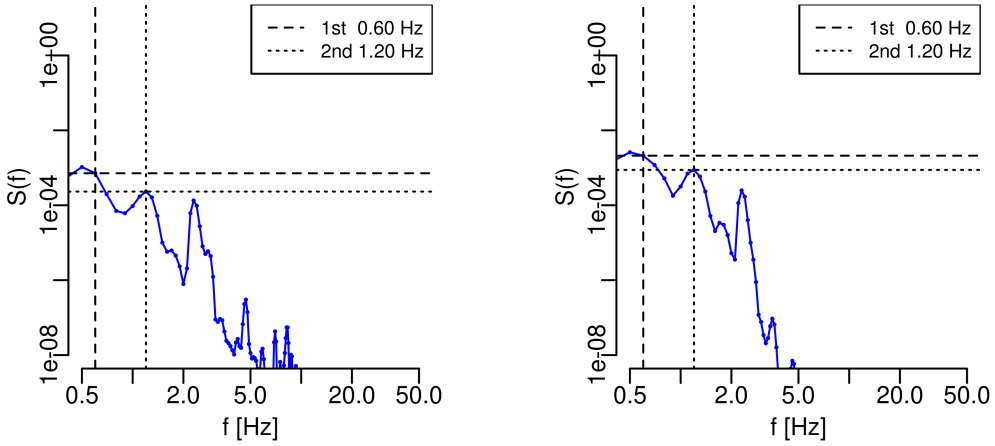
Figure 3.27: Power spectral density downstream of the truss tower, Case1-B. a) 3D downstream. b) 6D downstream

The two small cylinders in a side-by-side arrangement, in Case1-B, seems to change the frequency (Fig. 3.27a). The peak is found at 7.80Hz with a density in the order of 10^{-2}. If the same Strouhal number of 0.417 was to be expected for the vortex shedding behind the side-by-side arrangements, that arrangement will correspond to a cylinder with diameter of 0.65m. Each cylinder has in reality a diameter of 0.36m and they are separated with distance of 0.8m.
Another peak at about 4Hz could be from the shedding exhibited behind the main cylinders. This would then result in a Strouhal number of 0.3. Not only one main cylinder results in the vortex shedding, but a tandem structure involving two main and two smaller cylinders. This explains the frequency difference compared with one cylinder [2]. The smaller vortex sheddings have dissipated at 6D and the peak at 4Hz now has lower power density (Fig. 3.27b). Here, 1Hz is the dominant frequency which is inherited from the inflow.
The power density for Case1-C seems very low already at 3D, compared with the other results (Fig. 3.28a). The reasons for this is unknown. Interestingly, the power density is about the same at 6D downstream of the tower (Fig. 3.28b). Both frequency peaks are found at 0.60Hz and 1.40Hz with a power density in the order of 10^{-4}.
The clusters in Case1-D exhibit frequencies in the order of 10^{-3} (Fig. 3.29). One frequency at 1.20Hz and another one at 2.5Hz. The latter would result in a Strouhal number of 0.1875 using the main cylinder diameter of $d_1 = 0.9$m. At 6D downstream there are similar frequencies. The three peaks are still the dominant, and interestingly, they keep the power densities. The reason could be that the wakes develop earlier and the same shedding are found in both near- and far-wake region (See section 3.5.1: *Mean velocity profiles*).



(a) (b)

Figure 3.28: Power spectral density downstream of the truss tower, Case1-C. a) 3D downstream. b) 6D downstream



(a) (b)

Figure 3.29: Power spectral density downstream of the truss tower, Case1-D. a) 3D downstream. b) 6D downstream

Truss tower - Case2 and Case3

The vortex shedding behind Case2 exhibits a power density in the same order as Case1. Because the cylinders are in a staggered arrangement, it is not expected that the frequencies are similar [16]. A more thorough study is necessary to understand the details of wake interactions, but this will not be the focus of this report. However, a frequency peak at about 3.5Hz seems to be a trend for Case2, with a power density of about 10^{-2}. This corresponds to a Strouhal number of 0.26 using the main cylinder diameter ($d_2 = 0.9$m).

The shedding downstream from truss tower Case3 has the largest frequency peaks, in the order of 10^0, which is higher than for Case1 and Case2. The measuring lines are now closer to the last cylinder and the frequency peak will therefore have higher power density. The frequencies seems reasonable from their expected values calculated from the Strouhal number.
The power spectra for the Case2 and Case3 can be found in appendix D.

3.6 Discussion

Results from numerical simulations of flow past two dimensional cross sections of a monopile and a truss tower in the supercritical Reynolds number regime, have been studied. Numerical analysis of flows with Reynolds numbers are very time consuming. It is therefore very important that the numerical model and the parameters are carefully chosen.
The objective were to compare the mean velocity flow field downstream of the structures and to study the wake development. Mean velocity profiles have been measured at multiples of the monopile diameter D, where distances up to 5D downstream is considered the near-wake region. The far-wake region are considered from 5D and further downstream.
Since the blades on a downwind wind turbine is rotating at a distance equal to 3D from the tower, the turbulence intensity and power spectra is considered most thoroughly here. However, additional results from 6D, and 9D for the turbulence intensity, are included to see the effects in the far-wake region. Wake development in both near- and far-wake regions are different behind the two towers. One von Karman street is observed behind the monopile and several von Karman streets behind the truss tower.

As the cylinder representing the monopile is larger than the cylinders representing the truss tower, the resulting eddies are also larger. This does not necessarily mean that the resulting velocity deficit is larger. Although the truss tower is more transparent, the tandem configurations results in a larger velocity deficit downstream. At a distance 3D downstream of the towers, the velocity deficit is almost the same for the monopile and Case1-A. For Case1-B,C,D the velocity deficit is even larger. This clearly indicates that more cylinders involved in a tandem configuration gives larger deficits. This is, however, not representative in reality. When wind flows past a truss tower structure, it is not expected that the wind hits the tower at the exact 0 degree angle. This means that the tandem configurations is more likely to be staggered arrangements, like Case2. Results show that the staggered arrangements give a velocity deficit which is much lower than the deficit behind the monopile (Fig. 3.10), Case1 (Fig. 3.11, 3.12, 3.13 and 3.14) and Case3 (Fig. B). Another interesting observation is that truss tower Case2 gives a similar wake for all the different cases (Case2 A,B,C and D), which is very promising considering that

a steady tower shadow model could predict this more easily (Figures in appendix B). This is studied in chapter 4: *Tower shadow models*.

It is not only tandem effects that effects the wake development. Earlier literature, like Kiya et al. [16], Gao et al. [12] and Alam and Zhou [2], study flow past varied arrangements of cylinders and their results show that distances between the cylinders affect the wake. These features are not studied in detail here, but the numerical results clearly show that wake interactions change the downstream environment.

The total turbulence intensity is larger for the monopile by about 20%, compared to truss tower Case1. Interestingly, the results from Case2 and Case3 show a larger turbulence intensity in some of the cases, but still not larger than the monopile. This means that wake interactions does not necessarily contribute to the turbulence intensity. Unfortunately, the lack of earlier literature that studies the turbulence intensity behind such structures makes it difficult for comparisons.

Both the truss tower spectrum and the monopile spectrum is dominated by the vortex shedding frequencies. The velocity profiles indicated that wake interactions changed the shedding process. As discussed in Kiya et al. [16] the staggered arrangements, seen in Case2, will change the shedding process and thereby the frequencies. Most of the frequencies resulted in a Strouhal number in the expected region. The monopile spectrum showed an order of magnitude more fluctuations than the truss towers.

One major conclusion is that the Reynolds-Averaged-Navier-Stokes (RANS) approach is able to give accurate results for transient simulations of flow around two dimensional structures. This is very important considering that RANS is mostly used for steady-state simulations. This takes us to another important aspect that was discovered.

The turbulence intensity have two contributions. One from the sub-grid parameterization and another from the unsteady fluid motion. The section 3.5.2: *Turbulence intensity* made it clear that calculating this properly is a key issue that should be considered when using RANS for transient simulations. Since steady-state tower shadow models (see chapter 4: *Tower shadow models*) are commonly used for a complete wind turbine simulations, this important aspect could also result in large underpredictions of the loads on the blades.

Chapter 4

Tower shadow models

A numerical approach to predict the mean velocity profile behind a wind turbine tower have been tested in the previous chapter. Another approach is to use a steady-state tower shadow model. These tower shadow models could be a beneficial tool if they are able to predict the mean velocity profiles accurately. A complete wind turbine setup simulation already demands lots of computational capacity and a tower shadow model can avoid time consuming preparations with unsteady numerical simulations. The positive aspect of such an idea could help develop faster simulation tools in the wind turbine design field.

There are several tower shadow models that could be used for wind turbine tower structures. One of the popular choices has been Powles' model [23]. This model was originally designed for wind turbine monopiles and predict the velocity deficit downwind of a circular cylinder, which represent the two dimensional cross section of a monopile. The wake development is predicted using two parameters, respectively the wake width (w) and the velocity deficit (Δ) (see section 4.1). Blevins [4] proposed another two-parameter model. His interesting proposal of introducing an upstream virtual origin (x_0) of the wake allows for more flexibility, and could therefore give better predictions of the velocity deficit in the region of interest (see section 4.5). A third possibility is Schlichting's model [29]. This model was originally made for predicting the wake behind a thin plate, but it is also applicable for circular cylinders. Schlichting's model have three parameters that needs to be chosen (see section 4.5).

Unfortunately, no tower shadow models are available for a multimember structure, such as the truss tower. Powles' model has, however, recently been extended to predict the velocity deficit for such setups. Two dimensional cross sections of the truss tower are represented as several cylinders with different size and in different arrangements. The extended model simply superpose each of these cylinders linearly.
The multimember extension of Powles' model is an idea that simplifies the wake development in such an arrangement. Depending on the distances between the cylinders, and if they are placed in a tandem or side-by-side configuration, it is expected that some wake interactions would occur [16, 12, 2, 18]. How much these differences affect the velocity profile is studied by comparing the steady tower shadow models and the statistics from the numerical results.

4.1 Powles' model

Powles' tower shadow model [23] is implemented in GH Bladed (Version 3.82, Garrad Hassan and Partners Ltd.) [5]. This software is used to analyze a complete wind turbine setup and is the only one available with a multimember tower shadow model.

The tower shadow model has three different solutions in GH Bladed: a potential solution, a semi-empirical solution (Powles' model) and a combination of the two. The potential solution model the velocity magnitude upwind of the monopile an is expressed with a x- and y-component:

$$U_x = \left(1 - \frac{x^2 - y^2}{(x^2 + y^2)^2}\left(\frac{D}{2}\right)^2\right)U_0, \tag{4.1}$$

$$U_y = \frac{-2xy}{(x^2 + y^2)^2}\left(\frac{D}{2}\right)^2 U_0 \ . \tag{4.2}$$

Both components are calculated in the terms of the cylinder diameter D and a free-stream reference velocity U_0. The velocity magnitude is then $U = \sqrt{U_x^2 + U_y^2}$. Note that a potential solution for flow past multiple cylinders is not possible to approach with a linear superposition method [9] and will not be discussed for the multimember structures of this report.

Downstream of the cylinder, Powles' model is a semi-empirical model that predicts the velocity profiles, given by

$$U = U_0\left(1 - \Delta cos^2\left(\frac{y}{wd}\pi\right)\right), \tag{4.3}$$

where Δ is the velocity deficit at the middle of the wake. The dimensionless parameter $w = W/2d$ is represented by the physical wake width W and the cylinder diameter d. This empirical model assumes a cosine bell-shape within a ± 60 degree angle, which means it is cut of when the deficit falls to $\Delta/4$. Outside this region the potential solution model the velocity.

4.2 Multimember extension of Powles' model

The simplest idea to make Powles' model applicable for a multimember structure, such as the truss tower, is to use a linear superposition approach. This means that Powles' model is applied for each individually member, which is represented as circular cylinders.

Superposing the velocities for each member gives

$$U = \sum_{i=1}^{n} U_i + (1 + n)U_0 \ , \tag{4.4}$$

for the final velocity. Here, $U_i (i = 1, 2, ..., n)$ is the individual velocity for the i-th member of the structure. The maximum potentially velocity is $2U_0$ and limited to zero.

Two parameters, respectively the wake-width w and velocity deficit Δ, needs to be fitted to the individual member. Following Bossanyi et al. [5], the wake characteristics were assumed to use a square-root model:

$$\Delta(x) = \Delta_r / \sqrt{\frac{x}{x_r d}}, \tag{4.5}$$

$$w(x) = w_r \sqrt{\frac{x}{x_r d}}\,. \tag{4.6}$$

The parameters are made non-dimensional in order to be comparable with each other, using the reference length $x_r = 2.825D$, which is a typical distance for a downwind rotor. In addition, the reference values of w_r and Δ_r are used to scale w and Δ with the member diameter d.

Several aspects of the wake characteristics are not accounted for. Off course, there are three dimensional effects, which are not possible to model in two dimensions [38]; and depending on the placements of the cylinders, their wake can interact exhibiting new wake features and render the model less valid (Fig. 4.1). Wake interaction is captured by the numerical simulations. Studied later in this chapter is comparison on how much the steady-state tower shadow models and numerical results differ.

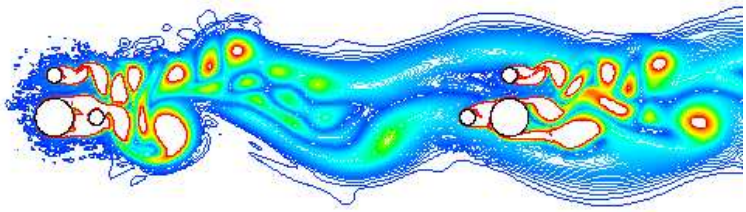


Figure 4.1: Contour plot of wake interactions, a part of cross section Case1-D. Figure clearly show wake interactions from cylinders placed at certain distance with each other. Color indicate vorticity magnitude. Red is 10[1/s]. Blue is 0[1/s]

4.3 Parameter estimation

As the multimember extension of Powles' model is applied, the two parameters w and Δ needs to be known for all the members in the truss tower. As already mentioned, Eq. 4.6 gives the behavior of the parameters, and the reference values w_r and Δ_r are used to scale w and Δ. In order to find good choices of these parameters, all cases were fitted by minimizing the root-mean-square (RMS) error between simulation results and model predictions for each individual mean velocity profile, with the Nelder-Mead search algorithm [20]. This was done by choosing a random value between 0.0 and 1.0 for Δ_r and 0.5 and 5.0 for w_r and then optimized ten times with five hundred iterations each.

In order to see if there could be optimized parameters that apply for a number of mean velocity profiles (instead of a single individual fit), global parameters including the four cases A,B,C,D have been estimated for Case1, Case2 and Case3. Four different optimization methods have been applied here:

1. By minimizing maximum error from the cross section with the largest such error
2. By minimizing the sum of maximum error from all cases
3. By minimizing RMS-error from the cross section with the largest RMS-error
4. By minimizing the sum of RMS-error from all cases

This has been done at distances 3D and 6D downstream of the tower.

4.4 Results and discussion

4.4.1 Monopile

The potential solution was able to follow the pattern of the numerical results upstream of the tower. However, the velocity was overpredicted at 1D upstream of the tower (Fig. 4.2a). Interestingly, the speed-up at the sides were also predicted. This feature, which also is exhibited downstream, were not captured there. At 3D upstream, a potential tower shadow is visible and the potential model slightly overpredicts this (Fig. 4.2b). Powles' tower shadow model was originally designed to model the time-averaged velocity deficit downstream of a monopile. Figure 4.3 shows that Powles' model predicts the velocity deficit quite well, both at a distance 3D and 6D downstream of the tower. The root-mean-square error at 3D is 0.092m/s and about the same at 6D. The wake width parameter is 1.8D and the velocity deficit about 22% at a 3D distance downstream (Fig. 4.3a). Compared with empirical measurement in earlier literature[1] [23] the parameters is about w=1D and a velocity deficit (Δ) of about 40%.Powles' model accurately predict that the wake widens downstream and at 6D it is found to be 2.292D with a deficit of 11%. Notice here that the speed up is still visible for the CFD-results, but not for Powles' model.

The dip between $y = \pm$6m is where the semi-empirical solution of the model describes the

[1]These parameters are not from flow around a circular cylinder, but a hexagonal cylinder. Also the wind speed is different, but should still be comparable.

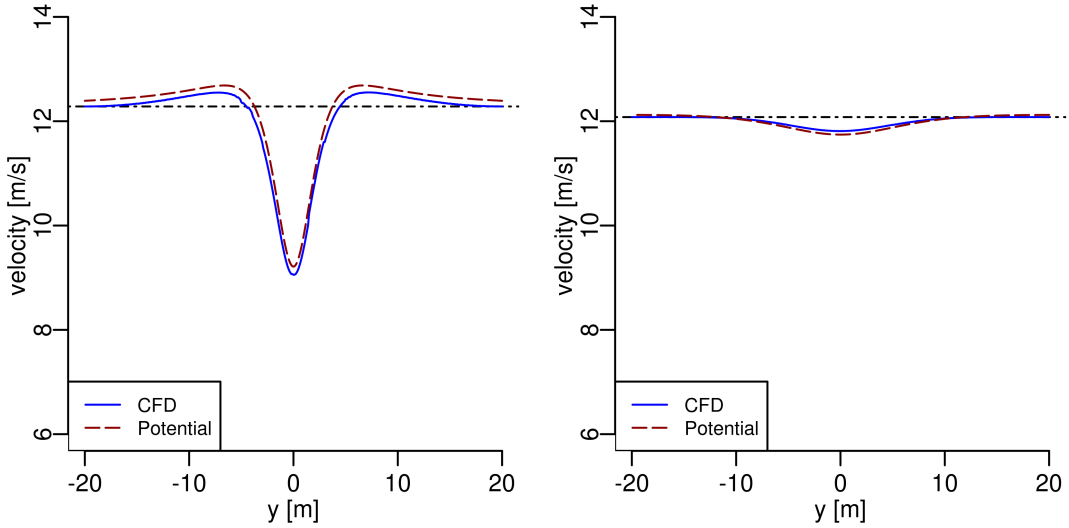
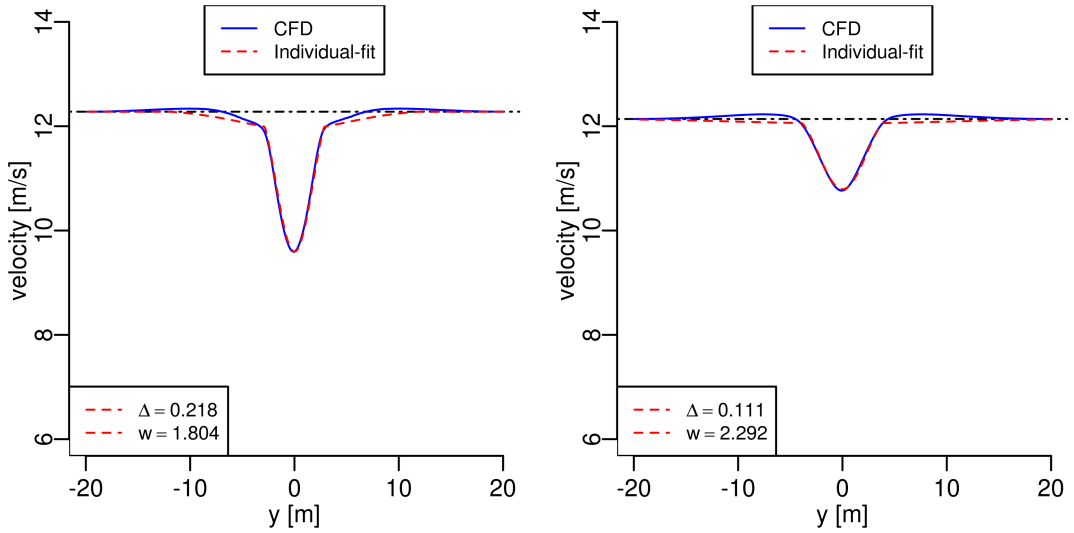


Figure 4.2: Monopile velocity profiles from CFD-simulations and Powles' model. a) 1D upstream. b) 3D upstream

wake. Outside this region, where the velocity has fallen to $\Delta/4$, the potential model was unable to predict the expected speed up at the sides of the cylinder. The free-stream velocity is larger than the inflow of 12m/s at the domain sides. This is an artifact from the periodic boundary conditions and the limited domain size. This will, however, not affect the tower shadow model predictions, if the reference velocity is set to be equal to the free-stream velocity at the sides of the domain.



(a) (b)

Figure 4.3: Monopile velocity profiles from CFD-simulations and Powles' model. a) 3D downstream. b) 6D downstream

4.4.2 Truss towers

The truss tower cases were divided into Case1, Case2 and Case3. Here, Case1 is four different two dimensional cross sections angled 0 degrees transverse to incoming wind direction. Case2 and Case3 are the same two dimensional cross sections angled 22.5 and 45 degrees, respectively. The four different cross sections (A,B,C and D) represents the X-brace at four different heights (See details in section 3.1: *Geometry and mesh*).

As explained above, the parameters Δ_r and w_r are estimated in order to give the best velocity profile prediction, both with an individual fit and a global fit.

Individual estimates versus CFD results

As for the monopile, the tower shadow model was unable to predict the speed-up at the sides of the cylinders. The velocity deficit was, on the other hand, quite accurately predicted for Case1-A, with an underestimation of about 2% behind the smaller cylinders (Fig. 4.4a). Behind the main cylinder no discrepancies are observed. It seems that Powles' model have problems with the wake development behind the smaller cylinders, especially when the small cylinders are placed in a close side-by-side arrangement as in Case1-B. For Case1-B, the deficit underestimation is almost 20% (Fig. 4.4b). The deficit behind the main cylinders was only overpredicted by about 2%.
Applying Powles' model for Case1-C, it also slightly overpredicts the deficit, about 1%, behind

the main cylinders (Fig. 4.4c). One could argue that the extra cylinder in these tandem configurations, compared to Case1-A, is the reason for this. The velocity deficit behind the small cylinders were again underestimated, now about 5%. When looking at Case1-A,B and C, one can observe that the deficit parameter Δ_r is quite similar, about 20% (Table 4.1). Note that when the small cylinders are placed close to each other (Case1-B), Powles' model was unable to predict the large deficit. The wake width parameter w_r is very similar, about 2.2D for the three cases.
Surprisingly, the velocity deficit behind Case1-D was quite well predicted, with only a small displacement of the dip (Fig.4.4d). Now, however, the velocity deficit parameter is much larger than for Case1-A,B,C, about $\Delta_r = 24\%$. The wake width (w_r) is somewhat lower, about 2.0D. In reality, the smaller wakes are absorbed by the larger ones, and the tower shadow model does not capture this. Fortunately, the wake width behind the main cylinders overlap the area where the smaller wake is exhibited. Therefore the model fits quite well, even though there are wake interactions. It seems like the individual fits in Powles' model give accurate prediction when the velocity deficit are of the same scale, as they were in Case1-D. The velocity deficit behind different sized cylinders, especially when placed in a side-by-side configuration, is difficult to predict.

So, are wake interactions the problem? There are wake interactions when the circular members are closer than five diameters to each other [2, 18], but Powles' model was still able to quite accurately predict the deficit behind the larger members, when in tandem configuration, even though their arrangements should give wake interactions. However, the close side-by-side members in Case1-B, gave a much larger deficit than Powles' model was able to reproduce. Case1-D also have members close to each other, but here the problem was avoided since the wake behind the main cylinders absorbed the smaller wakes. If applying Powles' model closer to the cylinder, one would expect the tower shadow model to fail. Another problem could be the different sized cylinders, or at least the large differences in the wake. The wake behind the small cylinders of Case1-A,B,C, is much different from the wake behind the main cylinders, and Powles' model performance is probably weakened by this.

If wake interactions are a problem, how well does Powles' model perform for Case2, where the tower is angled 22.5 degrees and is less transparent to the incoming wind? Powles' model performs quite well. The discrepancies are only displacements of the dips, but the major features are reproduced (Fig. 4.5). This is good news as Case2 is the most representative case.

Interestingly, the wake parameters for Case2 behaves quite similarly compared with Case1. The deficit parameters (Δ_r) for Case2-A,B,C are about 20-21%, with a wake width (w_r) of about 2.0D. Again Case-D is the odd one out, with a deficit parameter of 25% and wake width of 1.8D.

It is also worth mentioning that many of the same problems seen in Case1 occur for Case3 (Figures in appendix E). Accurate predictions for Case3-A, but less accurate for Case3-B and C. Case3-D have larger deficit discrepancies also the displacements. The most interesting observation regarding this case, however, is that the parameters vary more ($\Delta_r = 19 - 29\%$ and $w_r = 1.4 - 2.4D$) than for Case1 and Case2 (see table 4.1).

Wake interactions are an important feature in the wake development behind the truss tower.

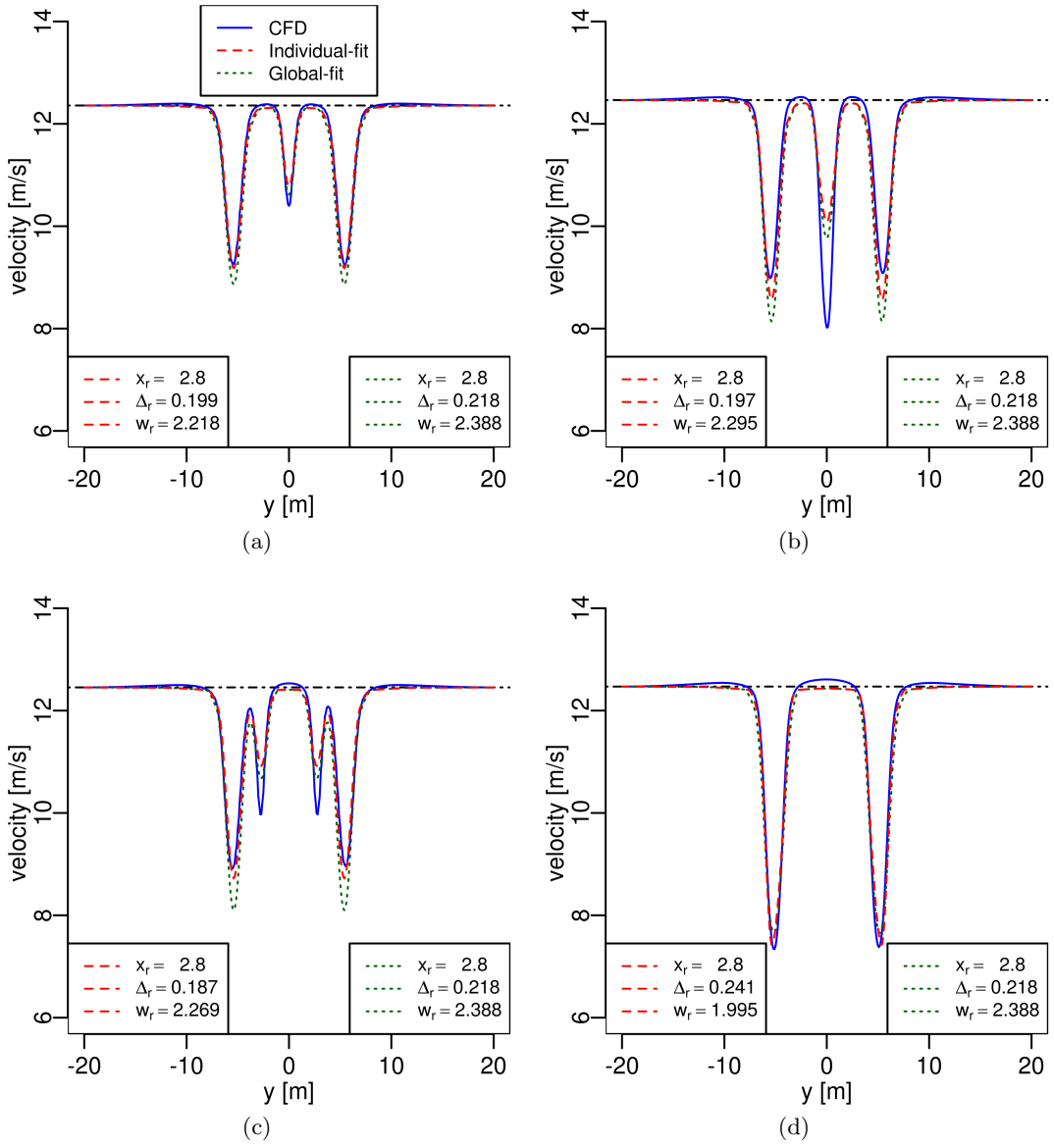


Figure 4.4: Mean velocity profiles for Case1 represented by the CFD-simulations and Powles' model, including both the individual(red line) and global fit(green line). a) Case1-A b) Case1-B c) Case1-C d) Case1-D

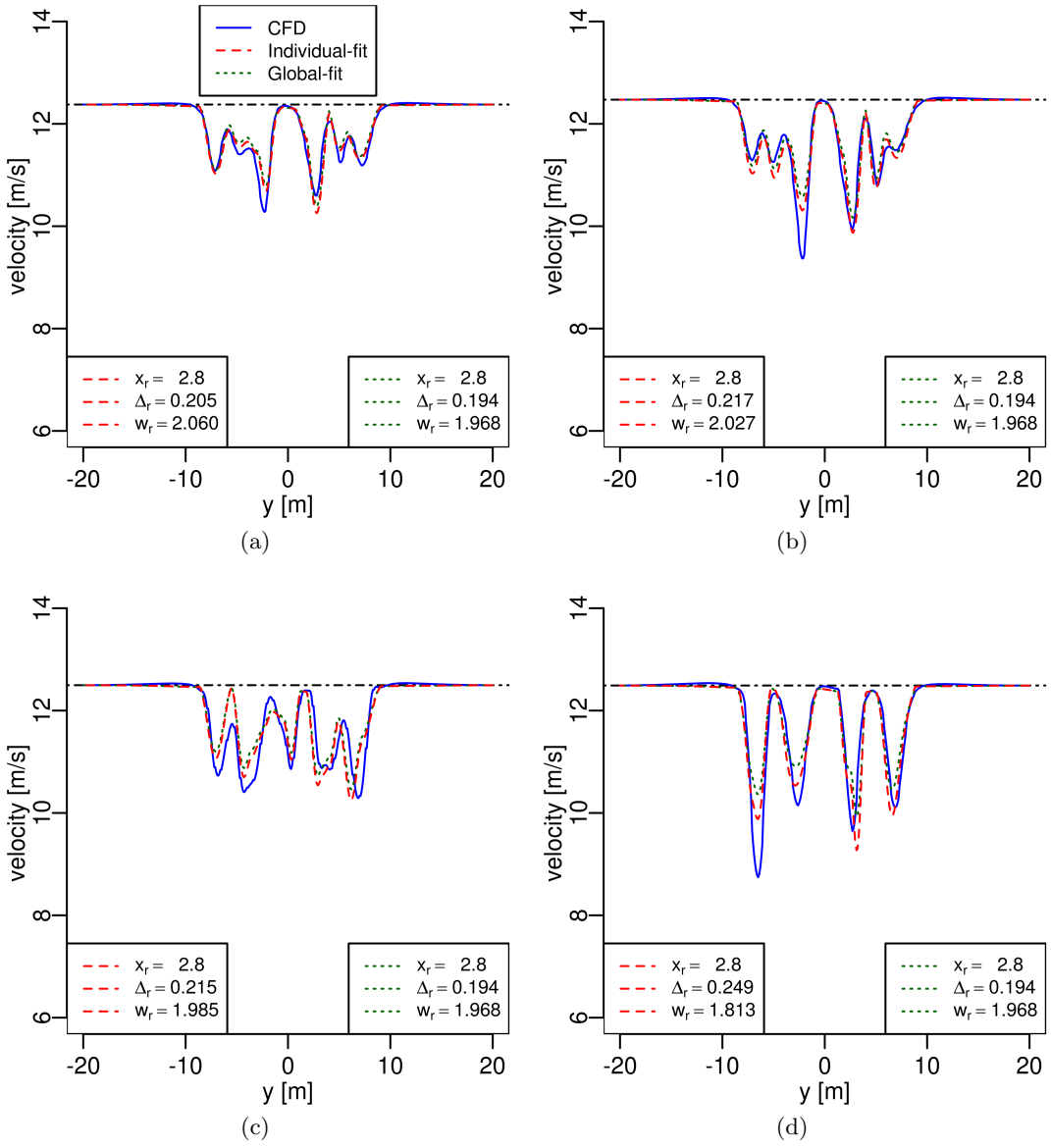


Figure 4.5: Mean velocity profiles for Case2 represented by the CFD-simulations and Powles' model, including both the individual(red line) and global fit(green line). a) Case2-A b) Case2-B c) Case2-C d) Case2-D

Table 4.1: Individually estimated parameters for Powles' model

Tower setup	3D		6D	
	w_r	Δ_r	w_r	Δ_r
Monopile	1.804	0.219	2.292	0.111
Case1-A	2.218	0.199	2.195	0.203
Case1-B	2.295	0.197	2.212	0.200
Case1-C	2.269	0.187	2.342	0.195
Case1-D	1.995	0.241	2.114	0.215
Case2-A	2.060	0.205	1.736	0.291
Case2-B	2.027	0.217	1.691	0.289
Case2-C	1.985	0.215	1.836	0.285
Case2-D	1.813	0.249	1.966	0.258
Case3-A	2.028	0.211	2.083	0.243
Case3-B	1.913	0.235	2.078	0.236
Case3-C	2.378	0.187	1.900	0.262
Case3-D	1.434	0.294	2.062	0.211

Although Powles' model does not account for such, it was possible to get more or less accurate predictions with individual estimations, at least for some of the cases. Fortunately, the largest discrepancies are found for Case1 and Case3. Aligned members, such as this are not common in reality and can be seen as more extreme cases, but still, they were quite well predicted.

Further downstream, at 6D, some of the smaller wakes should have dissipated and perhaps Powles' model should give more accurate results. In a way they do, but here the same problems predicting the wake behind the small cylinders occur, only now the discrepancies are smaller. This is discussed in detail below.

Global fit parameter estimates

Four different optimization methods were used to find global parameters for Powles' model (see section 4.3: *Parameter estimation*). Global parameters were found for Case1, Case2 and Case3. The parameters were estimated in order to fit the four cross sections in each case.

The results are very different. When looking at velocity profiles for Case2, there is no doubt that minimizing the maximum error give the best overall parameters (Fig. 4.6a). If estimated from minimizing the maximum error from a single cross section or the sum of all errors, does not change the profiles considerably (Fig. 4.6b). As expected, both root-mean-square (RMS) error estimates over- and underpredicts the deficit to a larger scale for all cross sections (Fig. 4.6c and Fig. 4.6d). For Case1, the differences between the cross sections are larger. None of the estimate methods is better than the other, since each of them performs well on at least one cross section. The same thing can be said for Case3, but here the RMS-approach performs more inaccurate to a larger extent.

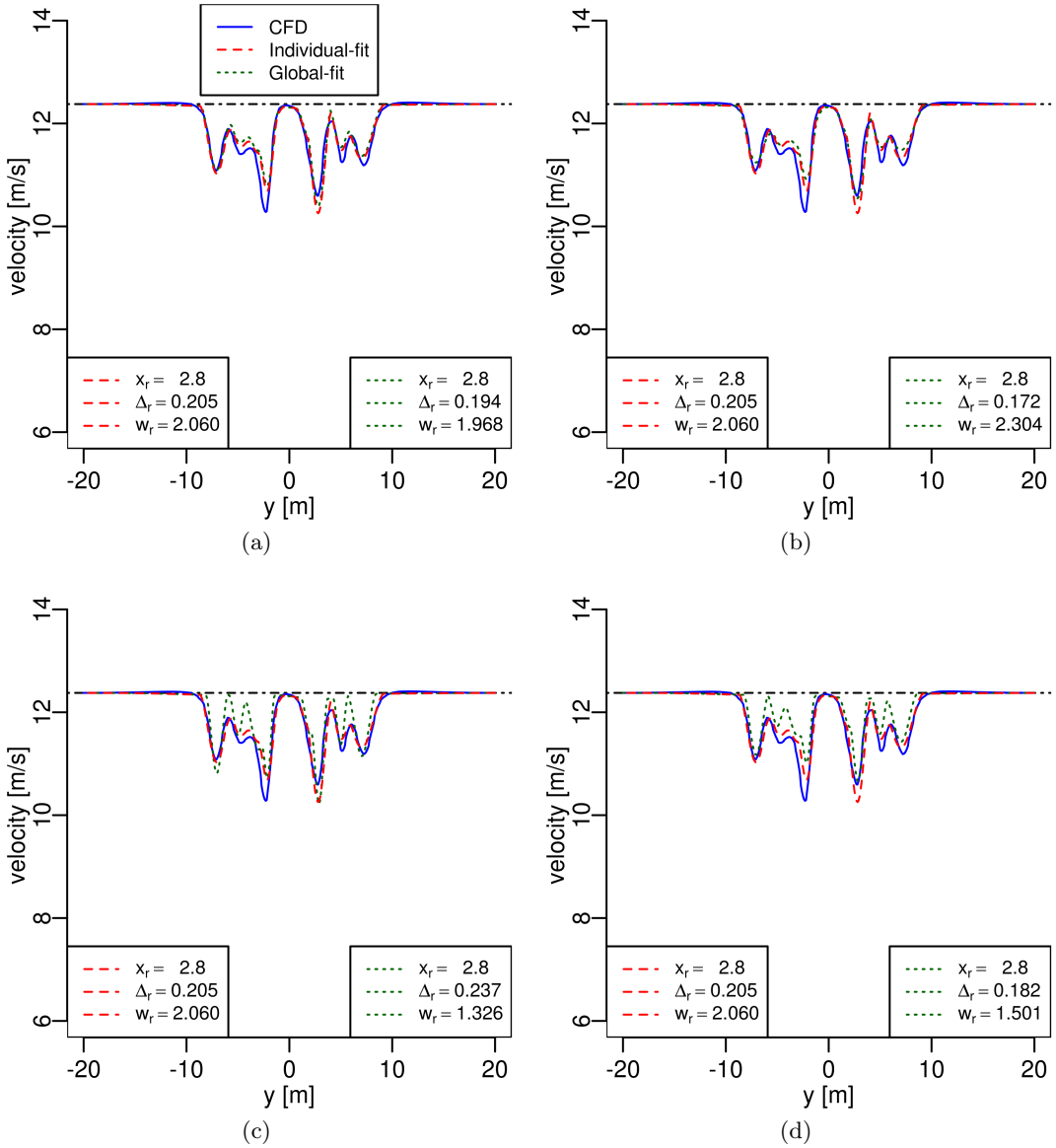


Figure 4.6: Velocity profiles 3D downstream for Case2-A including four different global fit estimate methods. a) Minimizing maximum error from a single profile with the largest single such error. b) Minimizing the sum of the maximum error from all cases included in the fitting. c) Minimizing the RMS error from a single profile. d) Minimizing the sum of the RMS error from all cases included in the fitting.

Table 4.2: Powles' model. Individual errors for four different global fit estimate minimized Root-Mean-Square (RMS) error and minimized maximum error at a 3D distance downwind of the truss towers. The global fit just includes each case e.g., Case1-error includes only the Case1 profiles.

Tower setup	Error estimates 3D			
Estimate method	Single max error	Tot. max error	Single RMS error	Tot. RMS error
Case1-A	0.86	0.48	0.35	0.35
Case1-B	1.40	1.77	0.56	0.56
Case1-C	1.40	0.96	0.46	0.46
Case1-D	1.16	0.71	0.55	0.54
Case2-A	0.88	0.77	0.39	0.36
Case2-B	1.31	1.21	0.50	0.48
Case2-C	1.31	1.21	0.51	0.49
Case2-D	1.31	1.47	0.54	0.55
Case3-A	0.77	0.58	0.37	0.32
Case3-B	0.96	0.77	0.48	0.47
Case3-C	1.26	1.07	0.51	0.44
Case3-D	1.26	1.48	0.51	0.55

Error estimates are found in table 4.2, and by looking at the numbers, one would expect that RMS-estimates give best results since, the errors are smaller and quite consistent. However, by looking at the velocity profiles, it is clear that minimizing the maximum error gives the best velocity profile predictions, and it therefore seems that using the parameters from this global fit gives most reasonable results, especially for Case2. The global parameters found by minimizing maximum error from a single cross section are found in table 4.3. Note that the parameters for Case2 could be used as a global set of parameters and would give reasonable results for all cross sections.

As the best parameters are found by minimizing maximum error from a single profile, this is the only method pursued in this study.

Table 4.3: Globally estimated parameters by minimizing maximum error from a single cross section for Powles' model 3D downstream of the truss tower.

Tower setup	3D		Min. Max.
Parameter	w_r	Δ_r	Error
Case1	2.388	0.218	1.40
Case2	1.968	0.194	1.31
Case3	1.718	0.231	1.26

At a distance 6D downstream of the truss tower it was expected that Powles' model would

perform better (see Fig. 4.8 below). The global estimates, however, does not work well here. All estimate methods gives similar results, with large discrepancies for Case2 and Case3. Case1, on the other hand, is very well reproduced by again minimizing the maximum error from one cross section, and it would therefore make sense to use the same estimation method 6D downstream.

Individual versus global fitting

The global estimates do not, as expected, give as accurate results as the individual fit. For Case1 and Case3 this is seen as over- and underpredictions, typically for the largest deficit dip. However, this is usually seen in the same places where the individual estimates also show discrepancies. This means that the global estimate give slightly different results overall. Note that an overprediction is especially large for Case1-C and Case3-C using the global fittings, mainly because the deficit parameter Δ_r (22% and 23%), deviates with the individual parameters, which is 19% for both cases. The opposite problem, that is an underprediction, occurs for Case2-D, where $\Delta_r = 25\%$ for the individual estimate and 19% for the global estimate.

For Case1 and Case2, the individual wake width parameter is similar to the global wake width parameter, but the global deficit parameter is larger for Case1 (except Case1-D) and overall smaller for Case2 (Fig. 4.4 and 4.5).
For Case3, where the wake width parameters vary more, the wake width is found lower except for Case3-D where the wake width is larger. The individual velocity deficit parameters are about the same as the global deficit parameter (except Case3-C as explained above).

The results show that using global estimates for the aligned member cases (Case1 and Case3) would give more inaccurate results for some of the cross sections. For the more representative scenario, Case2, the individual parameters vary less and the global parameters do not deviates that much from the individual ones. This means that it is possible to use only one set of parameters for all cases. This is good, because it is time consuming to estimate parameters for many different cases.

3D versus 6D downstream

It was expected that the tower shadow model would be able to predict the wake development better at a distance 6D downstream, since some of the wakes should have decayed here (see Fig. 4.8 below). The individual estimates are somewhat better, but the global estimates are actually worse; mainly underpredicting the wake width. What is also interesting, is that the individual estimates suggest a higher deficit parameter for most cases at 6D compared to 3D. Further downstream, where the wake grows larger, there is probably more interaction effects which results in a different behavior than reproduced by Powles' model. The wake widens, which probably means that the smaller wakes are absorbed by the larger ones, and this is not foreseen by Powles' model. This could perhaps be resolved by changing the reference distance when predicting the velocity profile in the far-wake regime or implement an 'absorption limit'[2], but this is not pursued further here.

[2]Term invented by the author. A limit depending on the spacing between the cylinders and their size, and when the smaller wake is absorbed

4.5 Alternative tower shadow models

4.5.1 Blevins' model

Blevins' model is designed to predict the mean velocity profile behind a circular cylinder. The main difference from this model and Powles' model is the parameter describing the upstream virtual origin of the wake x_0. This parameter makes the model more flexible in a way that it is not limited to just consider wake development downstream of the cylinder. This could mean that the model could give better predictions of the mean velocity profiles. The second parameter is the drag coefficient C_d, and depends analytically both on the wake width and the velocity deficit.

Blevins' model is represented by three equations. In the first equation b is the half-width,

$$b = 0.23[C_d d(x + x_0)]^{\frac{1}{2}} \ , \tag{4.7}$$

and represents the length from the centerline to the position where one-half of the centerline velocity deficit c is reached,

$$c = 1.02U_0\left(\frac{C_d d}{(x + x_0)}\right)^{\frac{1}{2}} \ . \tag{4.8}$$

Note that a misprint was detected in Blevins [4] and conected in equation 4.8, following Fredheim [11]. The total velocity deficit profile is given by

$$U(x, y) = U_0(1 - ce^{\frac{-0.69y^2}{b^2}}) \ . \tag{4.9}$$

To extend this model for multimember structures, the same extension approach used for Powles' model applies here. This approach superpose each wake from the individual member linearly, to find the resulting velocity (See section 4.2: *Multimember extension on Powles' model*). Blevins' velocity wake model has a square-root-model as the basis (Eq. 4.7 and Eq. 4.8) and the reference parameter values have been estimated by the same method explained in section 4.3: *Parameter estimates.*

4.5.2 Schlichting's model

Schlichting's model is not originally made for circular cylinders, but it models the wake behind a thin flat plate [29]. This model has three parameters that need to be chosen. The fluid viscosity is represented by ν and l represents the length of the thin flat plate. The latter parameter can here be interpreted as the virtual origin of the wake, similar to x_0 in Blevins' model. The third parameter is the drag coefficient and this has been chosen to be 0.37, which is the drag coefficient calculated from the validation run (section 3.4: *Validating the model*).

The velocity deficit in Schlichting's model is given by

$$U(x, y) = U_0\frac{C_d}{4\sqrt{\pi}}\sqrt{\frac{U_0 l}{\nu}}\left(\frac{x}{l}\right)^{-\frac{1}{2}} exp\left(-\frac{y^2 U_0}{4x\nu}\right) \ , \tag{4.10}$$

and the same multimember extension approach used for Powles' and Blevins' model is applied.

Since the drag coefficient parameter is chosen to be 0.37, it is only the fluid viscosity ν and the length l that needs to be adjusted by minimizing the root-mean-square (RMS) error. However, note that Eq. 4.10 is Reynolds number dependent which means that l needs to be large for large Reynolds numbers. This could make the model unstable, which implies an adjustment could be made for the model.

4.5.3 Comparison

Applying Blevins' and Schlichting's model for the monopile gives similarly accurate results as Powles' model, but none of them are able to capture the speed-up at the sides of the cylinder. Figures are found in appendix E.

Both Blevins' and Schlichting's model have been tested using the same linear superposition approach as Powles' model. Blevins' model shows similar accuracy as Powles' model for Case1, where Schlichting's model shows larger discrepancies, usually seen as over- or/and underpredictions of the velocity deficit for both the global and individual estimates.
The individual fits from Powles' model follows the deficit pattern for Case2 better than Blevins' model, but just slightly. Schlichting's model has again largest discrepancies, but here are the global estimates more consistent. This also applies for Case3. Examples of the differences are found in figure 4.7. Rest of the velocity profiles are found in appendix E.
There is not much difference between the three steady-state tower shadow models. All of them are able to predict the main features of the velocity profile. The lack of applicability when wake interactions occur is a trend, and it is probably necessary to include additional features in the equations to treat this complexity. However, it is expected that all the models, with the applied superposition approach, are able to give reasonable predictions of the mean velocity profiles. Again, to give even more accurate results, some modifications could be made for all models.

Blevins' and Schlichting's parameters are found in table 4.6 and table 4.7, and their best estimated global parameters are found in table 4.4 and table 4.5. In the far wake region all models are equally inaccurate. None of the profiles are able to capture the wake interactions that occur here. An example of this is found in figure 4.8.

Table 4.4: Globaly estimated parameters by minimizing maximum error from a single cross section for Blevins' model 3D downstream of the truss tower.

Tower setup	3D		Min. Max.
Parameter	C_d	x_0	Error
Case1	0.538	52.371	1.26
Case2	0.381	72.018	1.38
Case3	0.368	8.871	1.46

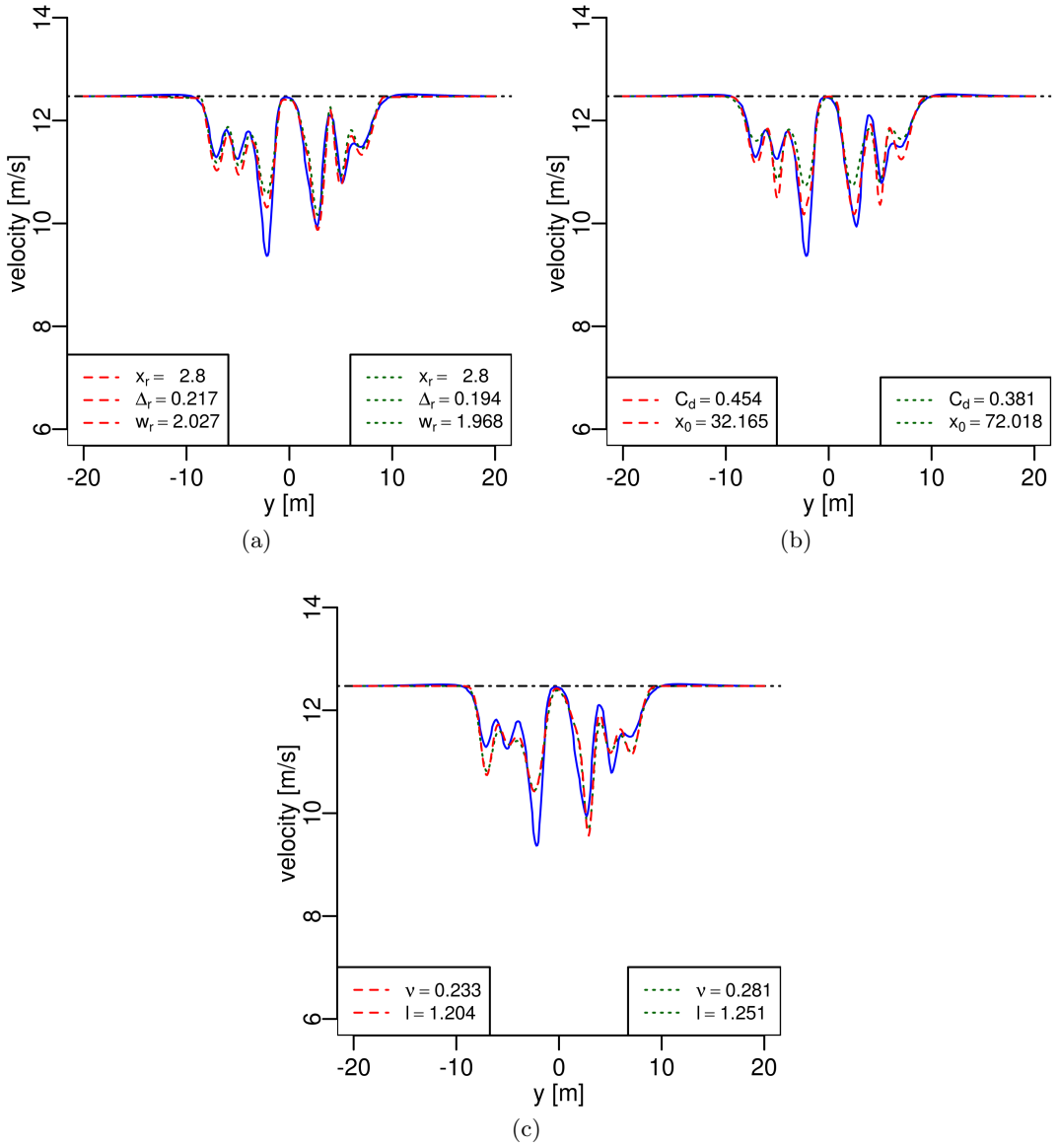


Figure 4.7: Mean velocity profiles 3D downstream represented by Powles', Blevins' and Schlichting's model together with CFD-simulations for both global and individual estimates. a) Powles' model b) Blevins' model c) Schlichting's model

Table 4.5: Globaly estimated parameters by minimizing maximum error from a single cross section for Schlichting's model 3D downstream of the truss tower.

Tower setup	3D		Min. Max.
Parameter	ν	l	Error
Case1-A	0.334	1.575	1.72
Case2-B	0.281	1.251	1.40
Case3-D	0.316	1.256	1.31

Table 4.6: Individually estimated parameters for Blevins' model

Tower setup	3D		6D	
	C_d	x_0	C_d	x_0
Monopile	0.415	5.806	-	-
Case1-A	0.453	41.507	0.467	90.944
Case1-B	0.479	54.030	0.477	118.567
Case1-C	0.449	56.931	0.477	122.457
Case1-D	0.489	28.391	0.464	80.789
Case2-A	0.451	34.379	0.511	25.427
Case2-B	0.454	32.165	0.488	23.423
Case2-C	0.449	32.718	0.519	21.429
Case2-D	0.465	21.158	0.510	49.127
Case3-A	0.443	32.498	0.511	59.625
Case3-B	0.466	29.176	0.483	69.838
Case3-C	0.452	52.526	0.494	40.678
Case3-D	0.414	0.971	0.449	87.083

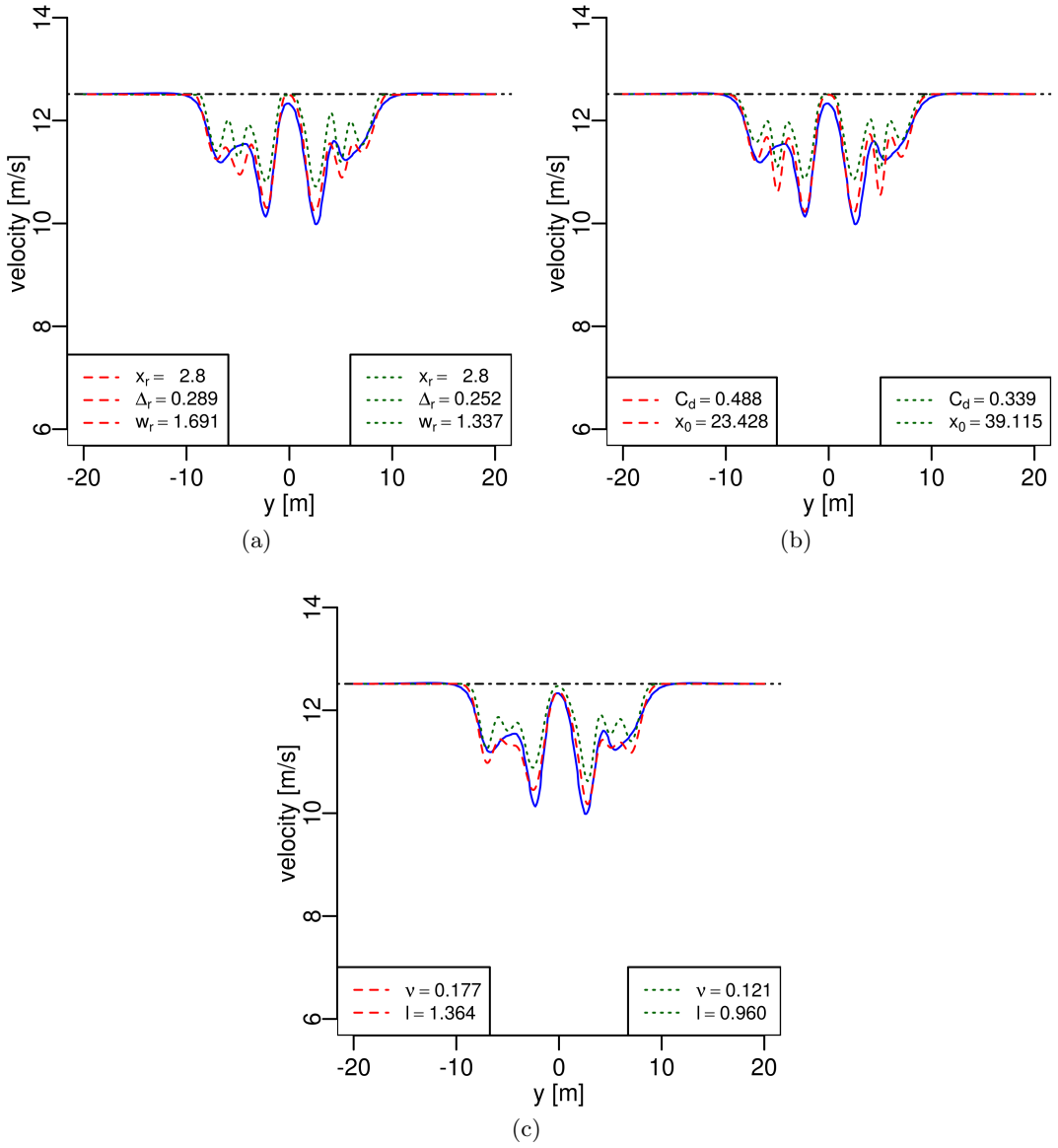


Figure 4.8: Mean velocity profiles at 6D downstream represented by Powles', Blevins' and Schlichting's model together with CFD-simulations for both global and individual estimates. a) Powles' model b) Blevins' model c) Schlichtings model

Table 4.7: Individually estimated parameters for Schlichting's model

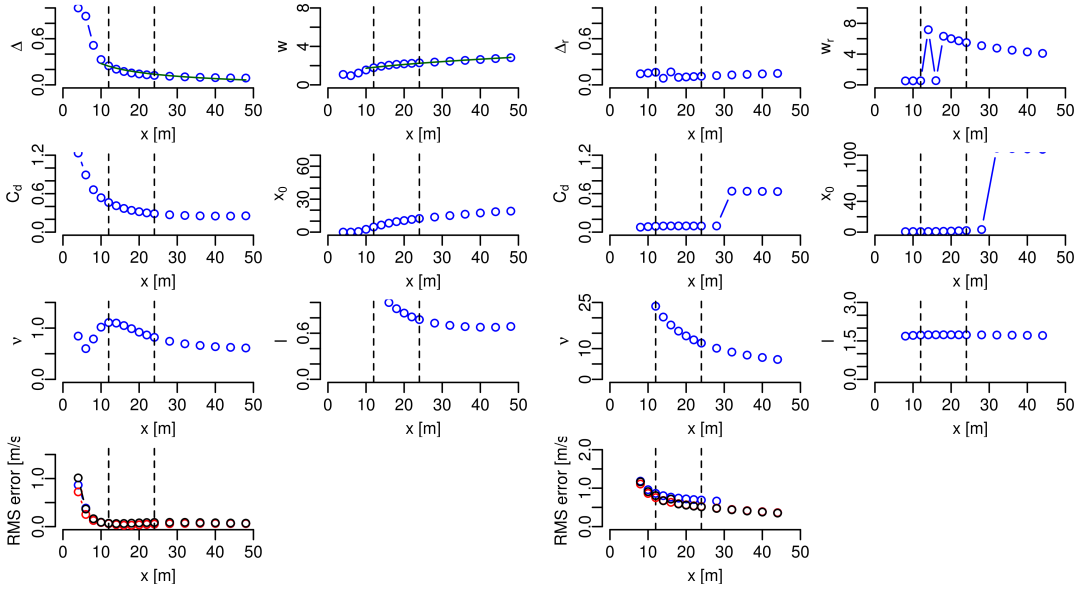
Tower setup	3D		6D	
	ν	l	ν	l
Monopile	1.146	1.122	-	-
Case1-A	0.311	1.213	0.302	1.227
Case1-B	0.277	1.212	0.252	1.179
Case1-C	0.278	1.144	0.340	1.250
Case1-D	0.249	1.330	0.251	1.260
Case2-A	0.297	1.166	0.227	1.416
Case2-B	0.233	1.204	0.177	1.364
Case2-C	0.259	1.216	0.195	1.445
Case2-D	0.197	1.271	0.201	1.386
Case3-A	0.273	1.191	0.254	1.359
Case3-B	0.205	1.229	0.217	1.350
Case3-C	0.339	1.191	0.277	1.406
Case3-D	0.143	1.224	0.231	1.208

4.6 Individual parameter estimates

The estimated parameters for the tower shadow models at several distances downstream of the monopile are found in figure 4.9a. As the parameters are estimated at the actual distance downstream of the monopile, they should exhibit a square-root behavior from Eq. 4.6, for Powles' model. From a distance 3D, or 12m, downstream this is more or less the case for both parameters. There are more discrepancies closer to the monopile, where the wake is not fully developed, compared to a square-root-law. This means that Powles' model is more adequate for the far-wake region, but at a 3D distance, where the rotor is designed to be, the Powles' model is good. The error is also found to be quite low, about 0.09m/s, from 3D and further downstream. For the other tower shadow models (Blevins' and Schlichting's), which are estimated with a reference distance, the parameters should be constant downstream. This is only the case in the far-wake region (5D and further downstream).

The estimated parameters for the truss tower have been scaled to a reference distance $x_r = 2.825D$, which is the distance from the tower center to the rotor. This means that the parameters should be constant with respect to the distance, for all the truss tower cases. From 3D and further downstream, this is more or less the case for the velocity deficit parameter for Powles' model, and the drag coefficient estimated for Blevins' model. The wake width parameter and the virtual wake origin vary more, but are roughly constant in the far-wake region. Schlichting's viscosity parameter actually seems to be distance dependent and gives the worst results here. This reflects back to what was seen in the mean velocity profiles. An example of the parameter estimates is found in figure 4.9b(Case1-A) and the rest is found in appendix F.

The unexpected transitions (differences in near- and far-wake region), seen from the tower



(a) (b)

Figure 4.9: Individual parameter estimates for Powles'(top row), Blevins'(second row) and Schlichtings(third row) model. RMS-error found in bottom row. a) Monopile b) Case1-A

shadows point of view, makes it difficult to estimate the parameters. This identifiability issue is seen as large 'jumps' for w_r, C_d and x_0 parameters (Fig. 4.9b). The residuals are still quite low for the truss tower, about 0.5m/s, and this means that when individually estimating the parameters, the tower shadow models are able to accurately reproduce the mean velocity profile.

4.7 Summary

All tower shadow models applied for the monopile give accurate results. However, all models are unable to predict the speed-up at the sides of the cylinder. This important feature could be important to capture since it could exhibit more fluctuating blades on a downwind wind turbine.

Extending the tower shadow models for a multimember structure is very useful. When parameters are found for each individual case, the predicted mean velocity profiles are quite accurately reproduced. However, the different sized members troubles the tower shadow models and some underestimation of the deficit is seen for the smaller members. This underestimation is highly increased for the cross sections which have the small members in a side-by-side configuration, where the deficit is much larger. Interestingly, the deficit is accurately predicted when the small members are placed close to the main members. The wakes behind the small members are absorbed by the larger wakes and because of this, the tower shadow models preform better.

Another important aspect is choosing the correct parameters. There is no earlier literature

that suggests parameters for multimember structures, but parameters for the monopile are usually suggested lower to what was found here. As the individual estimates reproduce the mean velocity profiles quite well, are the parameters somewhat different for the different for each cross section. For the more extreme Case1 and Case3, vary the parameters more.

So, are there any global parameters that could work for all cross sections at a 3D distance? Yes, there are. The more extreme cases (Case1 and Case3, with aligned members) showed some discrepancies with the global parameters, but they were not much larger than for the individual estimates. The tower shadow models work better for the more representative case (Case2), but the deficit dips are displaced. This is probably not a severe error when the steady-state tower shadow models are introduced in a fatigue analysis of wind turbine blades. When minimizing the maximum error, it is found to be about 1.4m/s (roughly 0.5m/s for RMS-error) for all cases. Comparing that with the velocity inflow (12m/s), it is a maximum error of 10%. That is not much when considered in an industrial application.

All major features of the mean velocity profiles are predicted by the tower shadow models when using the global estimates, and it would be expected that the largest induced vibrations are captured in order to produce accurate fatigue analysis.

For a far-wake study, the tower shadow models with global parameters show large discrepancies for numerous reasons. The main reason is the growing wakes downstream. That results in more wake interactions, which is not predicted by the tower shadow models. Here, the individual fit is better.

Chapter 5

Simulations with turbulent inflow

The turbulence used in the simulations in chapter 3: *Numerical simulations and wake development* is not structured, but imposed with turbulence intensity calculated using the parametrization from the $k-\omega$ SST viscosity model. Ideally, the flow should be implemented with structured turbulence. This is a better representation of a realistic flow. Unfortunately, this turbulent inflow has very low turbulence intensity, which could make it difficult to "trigger" the vortex shedding. This could again result in a long initialization time. Therefore, the turbulent inflow is added on top of the parameterized turbulence, which is set to be 10%, to have sufficient turbulence intensity. The turbulent inflow is implemented with a user-defined function in Fluent (see script in appendix H).

Turbulent inflow can be implemented with a spectrum. One option is the Kaimal spectrum, which fits better to the empirical observations of atmospheric turbulence, and the von Karman spectrum, which has proven to give good descriptions of the turbulence in wind tunnels. The latter represent the atmospheric turbulence above 150m quite well. As this is a study of flow past a wind turbine tower, the von Karman spectrum have been chosen here.

With turbulent inflow it is expected that the wake will fluctuate more and thereby exhibit different frequencies. This could change the loads, and thereby the fatigue damage on a wind turbine rotor. The following explains how the von Karman spectrum is implemented in Fluent, and some results of the numerical simulations with turbulent inflow.

5.1 Implementation

The von Karman spectrum follows the form given in Burton et al.(pg. 23,Eq. 2.24 and Eq. 2.26) [7], where the x-component is given by

$$\frac{nS_u(n)}{\sigma_u^2} = \frac{4nL_{2u}/U_0}{(1+70.8(nL_{2u}/\overline{U})^2)^{\frac{5}{6}}} \tag{5.1}$$

and the y-component is given by

$$\frac{nS_v(n)}{\sigma_v^2} = \frac{4(nL_{2v}/U_0)(1+755.2(nL_{2v}/\overline{U})^2)}{(1+283.2(nL_{2v}/U_0)^2)^{\frac{11}{6}}}\ , \tag{5.2}$$

where $L_{2u} = 73.5$m is the longitudinal length scale and $L_{2v} = 37.25$m is the lateral length scale. The terms $S_u(n)$ and $S_v(n)$ are the autospectral density functions, U_0 is the mean velocity and n is the frequency. Standard deviations for the components were taken to be $\sigma_u = 1.6$m/s and $\sigma_v = 1.2$m/s.
Taylor's frozen turbulence hypothesis and von Karman spectrum are used to analytically derive the correlations between the velocity with respect to distance. Refer Burton et al. [7] for details about the coherence functions. The diagonal mixing matrix, introduced in Veers [34], was simplified to be wrapped around 20 neighboring grid cells to correlate the lateral velocities with respect to distance. A frequency band between 0-10Hz was discretized using 100 frequency components. This was done for efficiency and to maintain stability. The user-defined function implemented in C-code can be found in appendix H.

5.2 Postprocessing

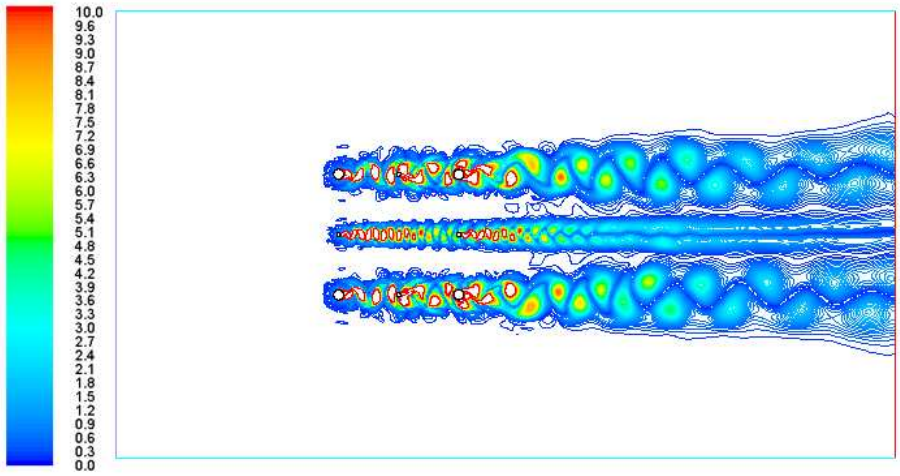
In order to get a fully developed turbulence through the whole domain, which is 150m along the flow direction, it was necessary to compute the flow as long as possible. Unfortunately, the computational time increased, about 10 times, drastically when the flow was implemented with the user-defined turbulence. The most time-consuming part of the simulations is outputting files, since the speed of the harddrive is limited.

The simulation procedures followed the same recipe as seen in section 3.2: *Numerical method* with the same time-step size, 0.005s (see section 3.3: *Postprocessing* for details). It was important to make the turbulent inflow fully developed through out the domain. Here, 30000 time-steps was simulated, which results in a real flow time of 150s. An additional 5000 time-steps initialized the flow. A longer initialization would have been preferred, but due to the time consuming process, this was all that was managed. The same outputs as before were saved. This resulted in approximately six weeks of simulation time, together for Case1 A,B,C and D, using a high performance workstation. The total number of output files was about $3.5 \cdot 10^6$, which demanded a large harddrive and time consuming postprocessing, using the statistical computing software R [24]. All 30000 time-steps was used for postprocessing in order to capture as much data from the inflow as possible.

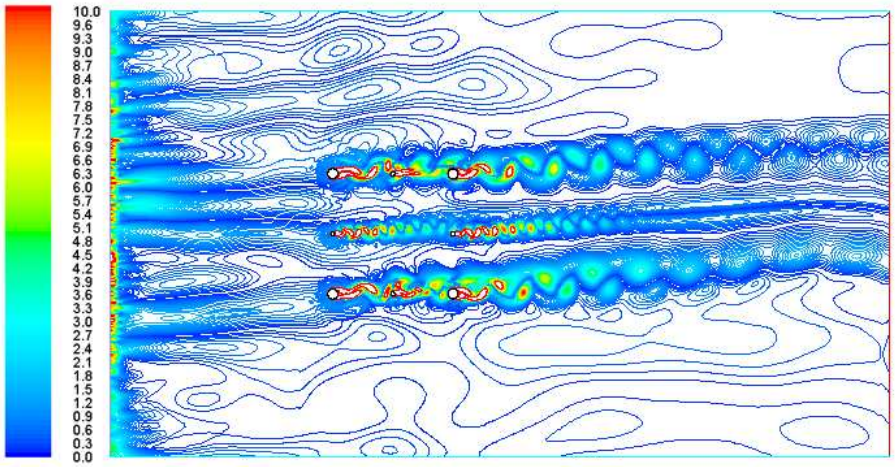
5.3 Results and discussion

The inflow now has unsteady motion (or fluctuations) in addition to the turbulence intensity parameterized by the $k - \omega$ SST viscosity model. Fig. 5.1 shows the differences in a vorticity magnitude contour plot. The vortex shedding is no longer aligned directly behind the cylinders, and vortices can be shed with angles from the cylinders.

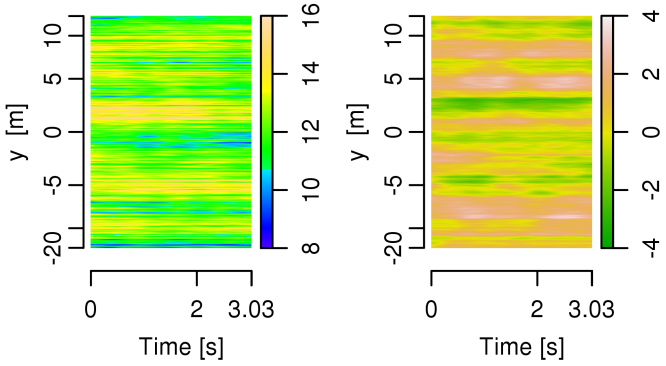
The velocity profiles upstream of the truss tower shows that the flow fluctuates around the mean velocity inflow of 12m/s, or actually about 12.3m/s due to blockage effect (Fig. 5.2a). Notice that the potential tower shadow model give the upstream velocity profile without turbulent inflow. With sufficient simulation time, the mean velocity profile upstream should be constant. However, the turbulence intensity should be higher, as discussed below.



(a)



(b)



(c)

Figure 5.1: Vorticity magnitude[1/s] for Case1-A a) Without von Karman turbulent inflow b) With von Karman turbulent inflow c) An example of turbulent inflow(3s) is shown. Left: x-velocity component. Right: y-velocity component. Colors represent velocity magnitude [m/s]

The mean velocity profiles downstream shows the same pattern as before, but now with less symmetry (Fig. 5.2b). This is probably due to insufficient calculation time of the turbulent inflow. Nevertheless, Powles' model are able to predict most of the features, which means the results are very good.

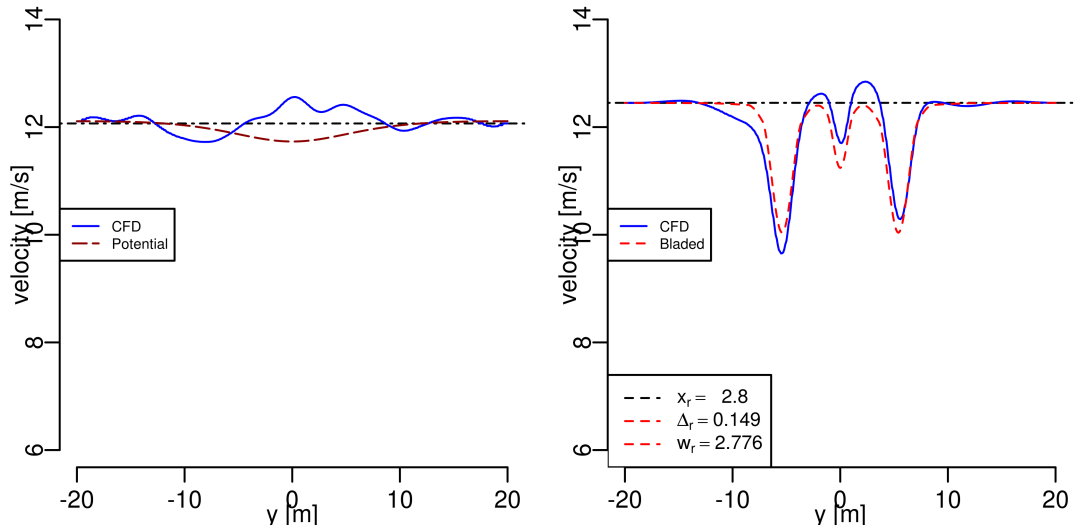


Figure 5.2: Mean velocity profiles for Case1-A with von Karman turbulent inflow a) 3D Upstream b) 3D Downstream

In the previous chapter 3: *Numerical simulations and wake development*, the turbulent inflow gave no contribution from unsteady motions to the turbulence intensity. Now, with the von Karman spectrum, Fig. 5.3a the inflow fluctuations now has a turbulence intensity of about 5%, and even higher for some of the cases.

Downstream of Case1-A with turbulent inflow, the maximum intensity is about 15%(Fig. 5.3d), compared with 12% without the turbulent inflow (Fig. 5.3c). Here, the turbulence inflow is added on top of the parameterized inflow and this would not be the case in reality. Ideally, it would be better to just have the structured turbulent inflow, which is a better representation of the wind and the turbulence intensity. At the sides of the domain the turbulence intensity is 8% with only parameterized turbulence and 10% with the turbulent inflow on top of the parameterized turbulence. This is not much difference which means it is probably not necessary to simulate complete wind turbine setups with a von Karman spectrum.

The main frequencies exhibited from the vortex shedding is still very similar, only now the magnitude is about one order higher than before (Fig. 5.4). For Case1-A the high frequency peak at about 13Hz is lower and the peak at 6.9Hz is the dominant one. This could mean that the shedding from the smaller cylinders, which give the frequency of 13Hz, is more affected by the turbulent fluctuations from the inflow, compared with the vortex shedding from the main

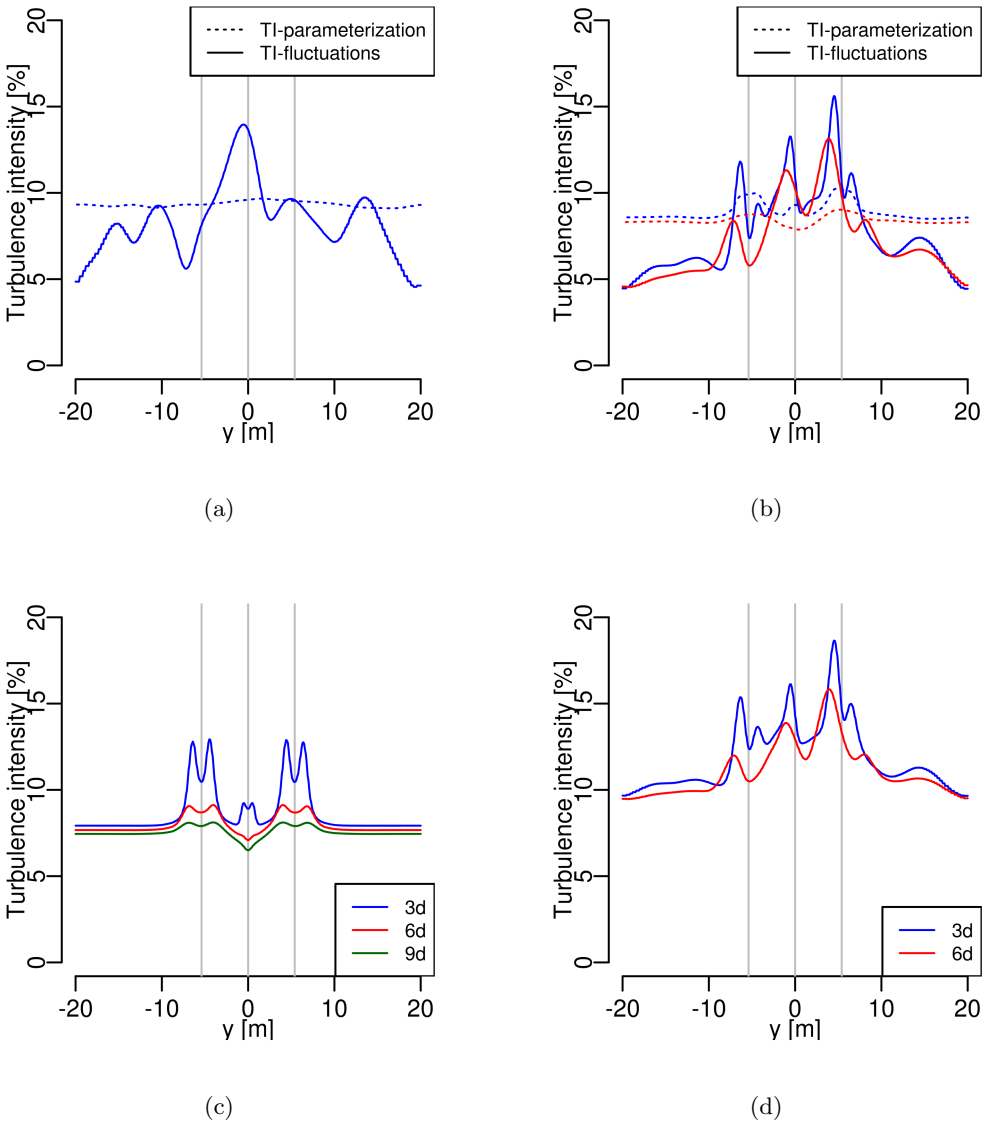
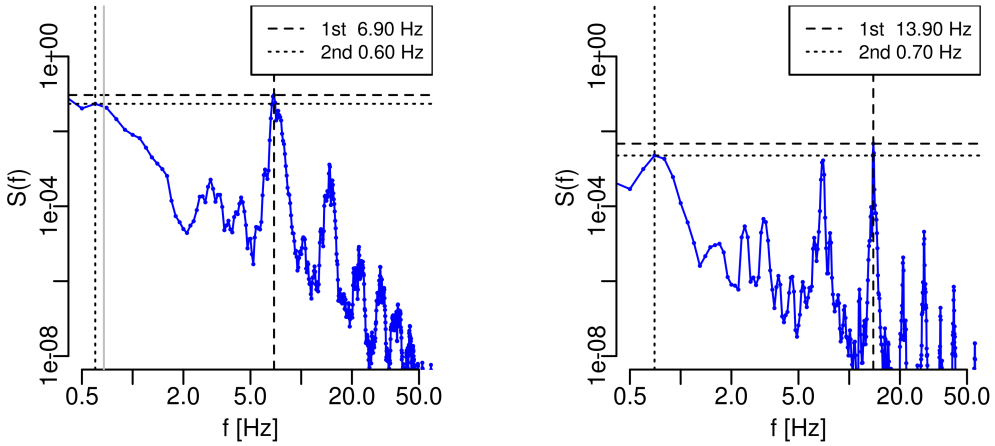


Figure 5.3: Turbulence intensity at 3D and 6D for Case1-A with von Karman turbulent inflow a) Upstream: Turbulence intensity form the two components b) Downstream: Turbulence intensity form the two components c)Downstream: Total turbulence intensity with only parameterized turbulence d) Downstream: Total turbulence intensity

cylinders. The latter has a corresponding frequency of 6.9Hz. Also notice that the low frequency from the inflow at 0.60Hz, which is also captured with the von Karman spectrum.



(a) (b)

Figure 5.4: Frequency spectrum for Case1-A. a) With von Karman turbulent inflow. b) Without von Karman turbulent inflow

When dealing with fatigue analysis of a wind turbine a realistic scenario is preferable. A turbulent inflow provides this, but increase the computational time drastically (about 10 times for flow simulations). Results above showed that the frequencies were the same for both the turbulence inflow simulations and the parameterized turbulence simulations. There was only a larger power density for the turbulent inflow case. However, considering that the turbulent inflow was added on top of the parameterized turbulence, the power density is probably reduced when the parameterized turbulence is reduced. Differences were also seen for the turbulence intensity. The fluctuation component gave a contribution from the turbulent inflow, but also this will be reduced when only turbulent inflow is considered. The mean velocity profile was quite accurately predicted, but small discrepancies due to insufficient simulation time, were observed.

This means that it is probably not necessary to use structured turbulence inflow in order to get realistic predictions of the flow profiles behind a truss tower.

Chapter 6

Aftermath

The numerical model used in this study have been pushed to its limit. A Reynolds-Averaged-Navier-Stokes (RANS) approach has been used to run transient simulations of flow past cross sections of a monopile and a truss tower. The computational demand has proven to be a challenge even though several simplifications have been made. However, results show that using a numerical model like the $k-\omega$ Shear-Stress Transport (SST) viscosity model give very accurate results in decent amount of time. The high Reynolds number ($\sim 10^6$) made the simulations a challenge, but by adapting the computational grid and triggering the flow, realistic results were captured.

One of the key issues when using RANS model for transient simulations, is the turbulent intensity. RANS only calculates the sub-grid parameterization turbulence, but an additional contribution from the unsteady motions also needs to be considered. It is very important that this is done properly, as section 3.5.2: *Turbulence intensity* showed.

In addition to the study above, the same numerical approach was tried on a model scale (about 1:130) of the same truss tower and monopile. Unfortunately, it was not possible to get good validation results with the same numerical approach. Results showed that the wake was not symmetric as it should have been, which is probably due to numerical instabilities.

The time-series resulted in mean velocity profiles which were compared with steady-state tower shadow models. These models were modified to be applicable for two dimensional cross sections of a truss tower, and their parameters were estimated to each of the individual truss tower cross section. The tower shadow models was unable to predict features like wake interactions, since the wake are calculated for each individual member and then linearly superposed. However, the tower shadow models proved to be very efficient. Some discrepancies were found for the more extreme cases (Case1 and Case2), but much better for the more representative Case2.

Different estimation methods were performed when trying to find the best set of global parameters. The best optimization method was to minimize the largest maximum error for a single profile. The global set of parameters for each of the tower shadow models are found in table 4.3,4.4 and 4.5. With these parameters, the tower shadow models were able to reproduce the mean velocity deficit very well. Note, however, that when using a steady-state tower

shadow model for a complete wind turbine setup, one should also consider the contribution to the turbulence intensity from the fluctuations. Avoiding this could result in underpredictions of damaging loads. One question remains. The periodic motions of the vortex shedding are not captured when they are modeled with an increased turbulence intensity. This could mean that the loads are not correctly reproduced. This interesting idea could be studied by comparing results from this study with a "dynamic tower shadow model". However, this is left to future work.

Simulations with the turbulent inflow is not necessary to predict a realistic tower shadow behind truss tower cross sections. The power spectra was about one order higher compared with the structured turbulent inflow, but in these simulations the turbulent inflow was added on top of the parameterized turbulence. Together, they probably gave a larger turbulence intensity than they should have.

There are several aspects of these studies that could be modified in order to get more realistic results. One important is that two dimensional simulations is a simplified model and important three dimensional effects [38] is lost.

For now it is possible to implement the time-series from the numerical studies or the tower shadow models into GH Bladed. This study has been started, but unfortunately, it was not finished to be included in this report. However, some early results show that a complete wind turbine setup simulation with Powles' model underpredict the damage equivalent loadings (DEL) on the blades. The same simulations have been done implementing the time-series, where DEL is calculated to be higher. Considering that Powles' model is a steady-state model, the important unsteady motions are disregarded, which results in too low turbulence intensity.

List of publications

- Hagen, T.R., Reiso, M. and Muskulus, M., *"Numerical Analysis of Turbulent Flow Past a Truss Tower for Offshore Downwind Turbines"*, Proceedings of the ISOPE-2011, The 21st International Offshore (Ocean) and Polar Engineering Conference, June 19-24-2011, Maui, Hawaii, USA
- Hagen, T.R., Reiso, M. and Muskulus, M., *"Numerical Tower Shadow Modeling for a Downwind Wind Turbine Truss Tower"*, Proceedings of the ASME 2011 30th International Conference on Ocean, Offshore and Arctic Engineering, OMAE-2011, June 19-24-2011 Rotterdam, The Netherlands

Bibliography

[1] Achenbach, E. "Distribution of local pressure and skin friction around a circular cylinder in cross-flow up to $Re = 5 \cdot 10^6$", J. Fluid Mech., 34(4) pp.625-639 (1968)

[2] Alam, M.M. and Zhou, Y. "Strouhal numbers, forces and flow structures around two tandem cylinders of different diameters", Journal of Fluids and Structures 24 pg. 505-526, (2008)

[3] ANSYS FLUENT 12.0 Theory Manual, ANSYS Inc., Cantonburg USA - (2009)

[4] Blevins, R.D., *Flow-Induced Vibrations*, Van Nostrand Reinhold, 450 pp. (1990)

[5] Bossanyi, E.A., *GH Bladed theory Manual*, Technical Report 282/BR/009, Garrad Hassan and Partners Limited, Bristol, 111 pp. (2009)

[6] Breton, S.P. and Moe, G., "Status plans and technologies for offshore wind turbines in Europe and North America", *Renewable Energy*, Vol. 34, pp 646-654 - (2009)

[7] Burton, T., Sharpe, D., Jenkins, N. and Bossanyi, E., *Wind Energy Handbook* ISBN 13: 978-0-471-48997-9(H/B), John Wiley & Sons Ltd. The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England (2001)

[8] Cook, R.D., Malkus, D.S. and Plesha, M.E., "Concepts and Applications of Finite Element Analysis", Third edition, University of Wisconsin-Madison, John Wiley & Sons, Inc. (1989)

[9] Crowdy, D.G., *Analytical solutions for uniform potential flow past multiple cylinders*, European Journal of Mechanics B/Fluids 25 pg. 459-470, (2006)

[10] Csanady, G.T., "Air-Sea Interaction", Cambridge University Press Cambridge, New York (2001)

[11] Fredheim, A. *Current forces on net structures*, PhD thesis, Department of Marine Technology, NTNU Trondheim

[12] Gao, Y., Yu, D., Tan, S., Wang, X. and Hao, Z. "Experimental study on the near wake behind two side-by-side cylinders on unequal diameters", Fluid Dyn. Res. 42, 055509, The Japan Society of Fluid Mechanics and IOP Publishing Ltd, Printed in UK (2010)

[13] Hagen, T.R., *Numerical Analysis of Flow Past Cross-Sections of a Wind Turbine Tower*, Project work TFY4510, NTNU Trondheim (2010)

[14] Hedde, T. and Durand, P., "Turbulence intensities and bulk coefficients in the surface layer above the sea", Boundary-Layer Meteorology 71, 415-432 (1994)

[15] Jones, W.P. and Launders, B.E. "The prediction of laminarization with a two-equation model of turbulence", *Dept. of Mechanical Engineering*, Int. J. Heat Mass Transfer, Vol. 15, pp. 301-314, Pergamont Press (1972)

[16] Kiya, M., Arie, M., Tamura, H. and Mori, H. "Vortex Shedding From Two Circular Cylinders in Staggered Arrangements", Transaction of the ASME Vol.102, *Department of Mechanical Engineering*, Hokkaido University, Sapporo 060, Japan, (June 1980)

[17] Long, H. and Moe, G., "Truss type towers in offshore wind turbines", *Proceedings of European Offshore Wind Conference* (EOW2007) - (2007)

[18] Meneghini, J.R., Saltara, F., Siqueira, C.L.R., and Ferrari Jr, J.A., "Numerical Simulations of Flow Interference Between Two Circular Cylinders in Tandem and Side-by-Side Arrangements", *Journal of Fluids and Strucutres 15, pg. 327-350* Academic press (2001)

[19] Menter, F.R., *Two-equation Eddy-Viscosity Turbulence Models for Engineering Applications*, AIAA, 32:1598 - 1605 (1994)

[20] Nelder, J.A., and Mead, R., "A simplex method for function minimizing", Comput J., Vol 7, pp 308-313, (1965)

[21] Ong, M.C., Utnes, T., Holmedal, L.E., Myrhaug, D. and Pettersen, B., *Numerical simulation of flow around a smooth circular cylinder at very high Reynolds numbers*, Marine Structures 22, pg. 142-153 (2009)

[22] Pope, S.B., *Turbulent Flows*, Cambridge University Press - (2000)

[23] Powles, S.R.J., *The effects of tower shadow on the dynamics of a HAWT*, Wind Engineering Vol.7, No.1 (1983)

[24] R Develpment Core Team , *"R: A Language and Environment for Statistical Computing"*, R Foundation for Statistical Computing, Vienna. Software available ar http://www.r-project.org/. (2009)

[25] Reiso, M., Muskulus, M. and Moe, G. "Tower shadow-experiment comparing wake behind tubular and truss towers" isope-2011-tcp-855. In *Proceedings of the 21st International Offshore (Ocean) and Polar Engineering Conference*, June (2011)

[26] Reynolds, O., "An Experimental Investigation of the Circumstances which determine whether the Motion of Water shall be Direct or Sinuous, and of the Law of Resistance in Parallel Channels", *Philosophical Transactions of the Royal Society of London*, pp. 935-982, May (1883)

[27] Roshko, A., "Experiments on the flow past a circular cylinder at very high Reynolds number", Guggenheim Aeronautical Laboratory, California Institute of Technology, Pasadena, California, (1960)

[28] Salim, S.M. and Cheah, S.C., "Wall y^+ Strategy for Dealing with Wall-bounded Turbulent Flows", *Proceeding of the Int. MultiConference of Eng. and Comp. Sci. 2009 Vol. 2* IMECS (2009)

[29] Schlichting, H. and Gersten, K. *Boundary-Layer Theory*, Springer, 799 pp. (2000)

[30] Shinozuka, M. "Digital Simulation of Rnadom Process and its Application", *J Sound Vibration*, Vol. 25, pp. 111-128 (1972)

[31] http://www.flickr.com/photos/mitopencourseware/4150128499/, downloaded 14.06.2011

[32] Tennekes, H. and Lumley, J.L., "A First Course in Turbulence", The MIT Press, Cambridge, Massachusetts, and London, England (1972)

[33] Trk, M. and Emeis, S., "The dependance of offshore turbulence intensity on wind speed", J. Wind Eng. Ind. Aerodyn. 98 466-471 (2010)

[34] Veers, PS., *Three-Dimensional Wind Simulations*, Technical Report SAND88-0152, Sandia National Laboratories, Albuquerque 36pp. (1988)

[35] Venables, W.N. and Ripley, B.D., *Modern applied statistics*, Springer, 501 pp. (2002)

[36] Warchauer, K.A. and Leene, J.A., "Experiments on mean and fluctuating pressures of circular cylinders at cross flow at very high Reynolds numbers", *Proc. Int. Conf. Wind Effects Buildings Structures*, Tokyo, pp. 305-315 (2009)

[37] White, F.M., *Viscous Fluid Flow*, Third edition, McGraw-Hill international edition, ISBN 007-124493-X, (2006)

[38] Williamson, C.H.K., "Vortex dynamics in the cylinder wake", *Annu Rev Fluid Mech. 28*, pp.477-539, (1996)

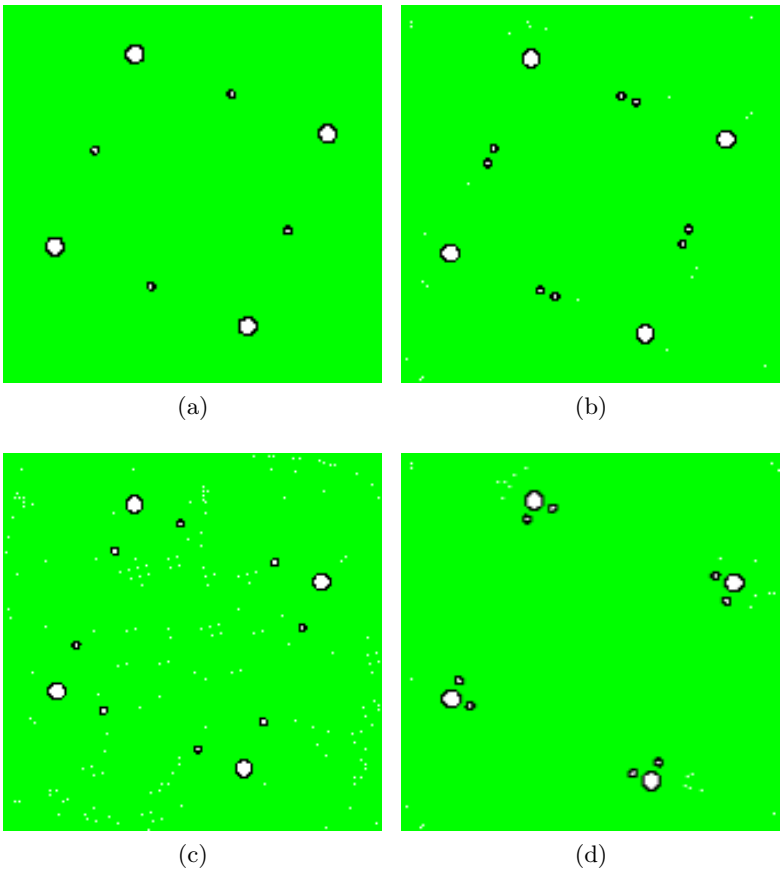
[39] Zdravkovich, M.M., *Flow Around Circular Cylinders*, Vol.1: Fundamentals, Oxford University Press, Oxford (1997)

[40] Zdravkovich, M.M., *Flow Around Circular Cylinders*, Vol.2: Application, Oxford University Press, Oxford (2005)

[41] Zdravkovich, M.M., "REVIEW-Review of Flow Interference Between Two Circular Cylinders in Various Arrangements", Journal of Fluid Engineering, Transaction of the ASME, pg.618-633 (1977)

Appendix A

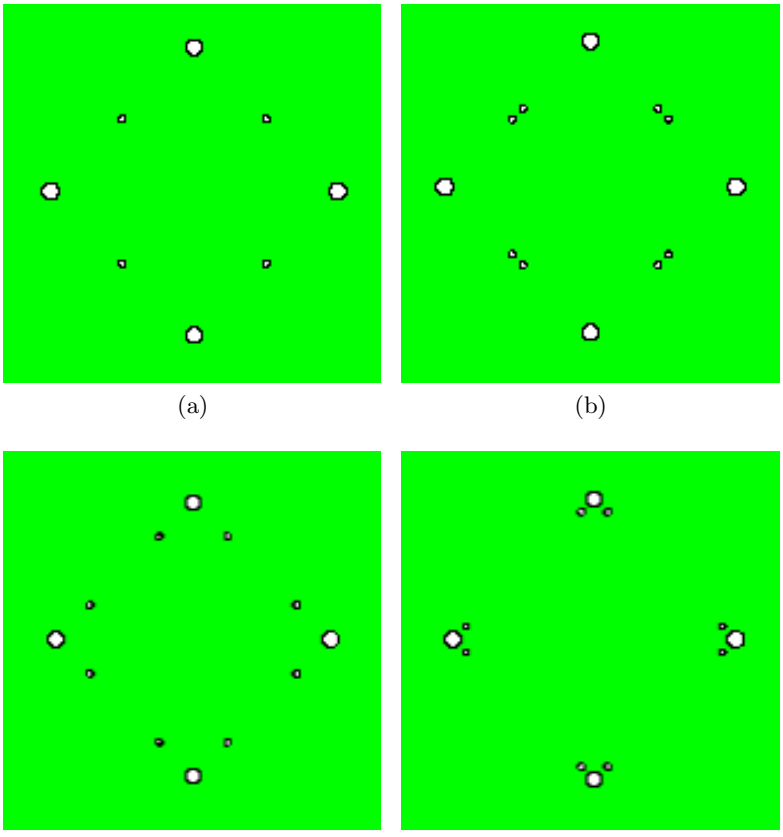
Additional figures: Geometry



(a) (b)

(c) (d)

Figure A.1: Case2: 22.5 degree truss tower cases embedded in mesh. a) Case2-A b) Case2-B c) Case2-C d) Case2-D



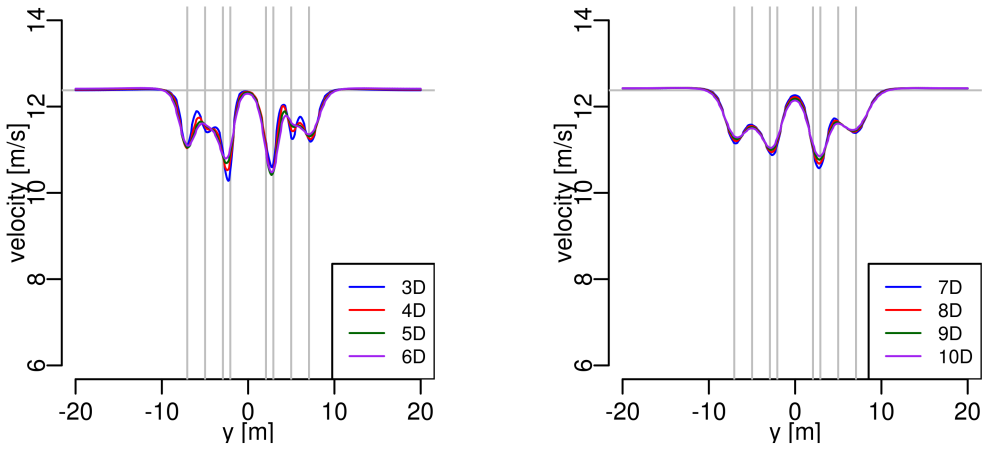
(a) (b)

(c) (d)

Figure A.2: Case3: 45 degree truss tower cases embedded in mesh. a) Case3-A b) Case3-B c) Case3-C d) Case3-D

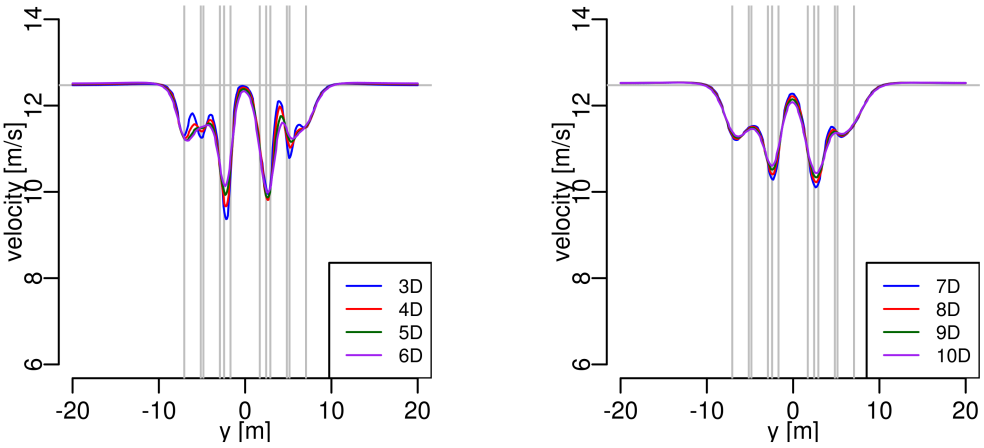
Appendix B

Additional figures: Mean velocity profiles



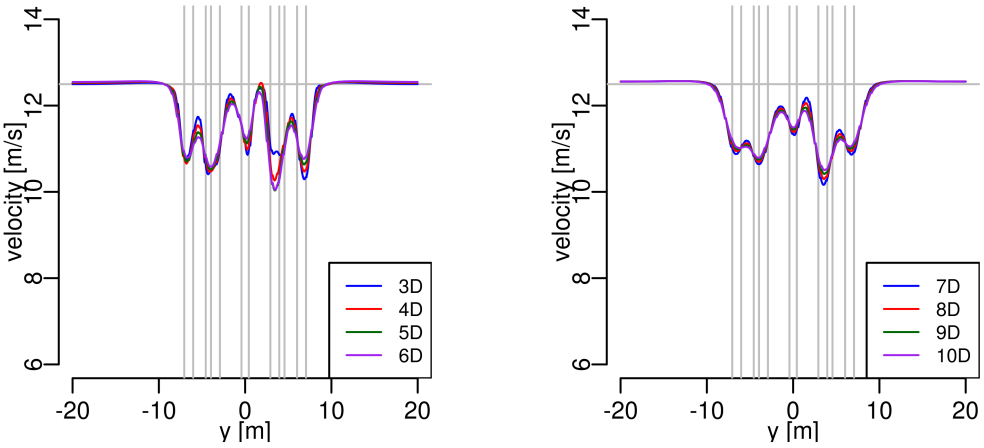
(a) (b)

Figure B.1: Mean velocity profiles for Case2-A. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



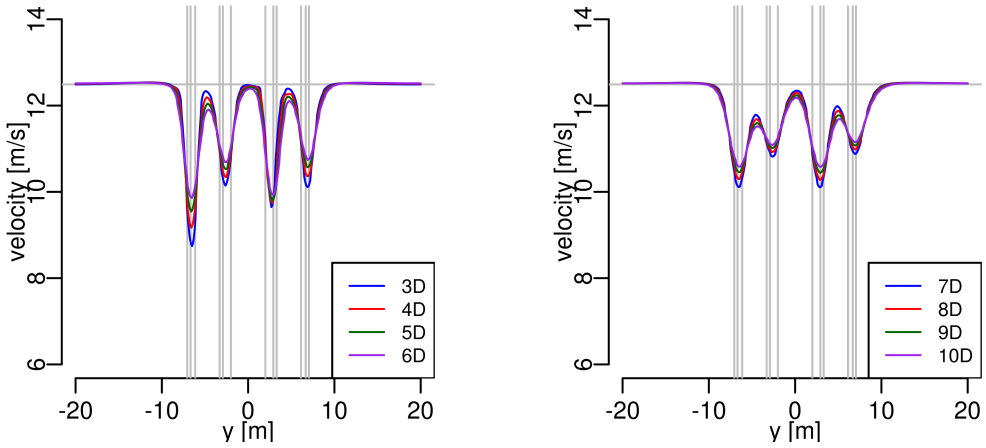
(a) (b)

Figure B.2: Mean velocity profiles for Case2-B. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



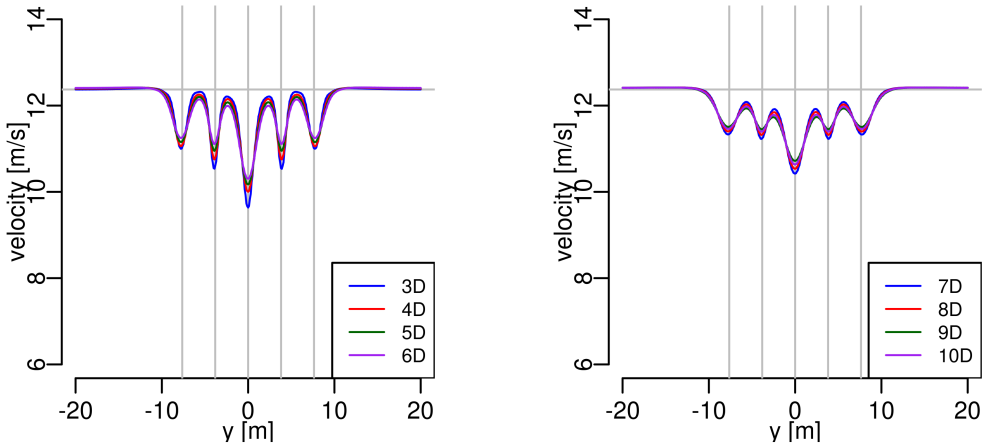
(a) (b)

Figure B.3: Mean velocity profiles for Case2-C. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



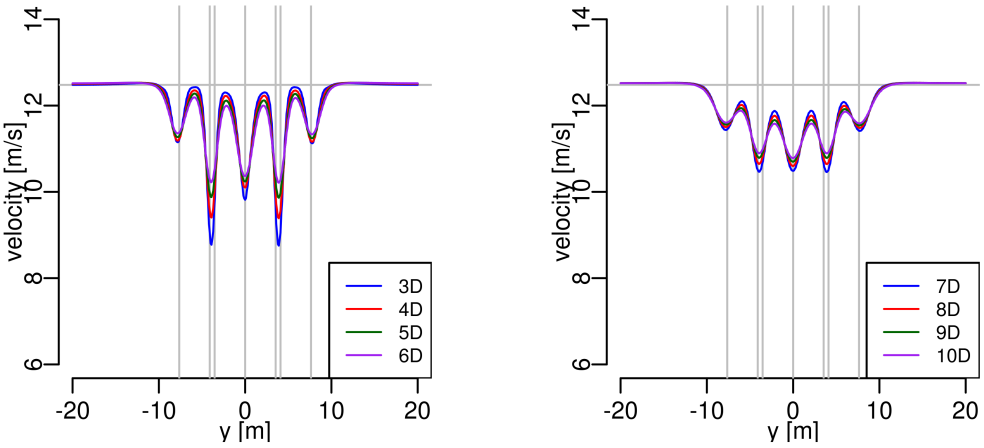
(a) (b)

Figure B.4: Mean velocity profiles for Case2-D. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



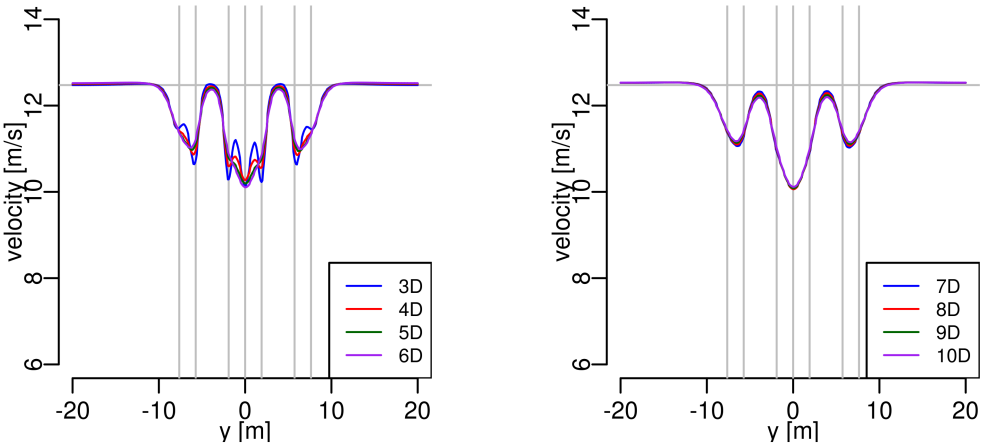
(a) (b)

Figure B.5: Mean velocity profiles for Case3-A. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



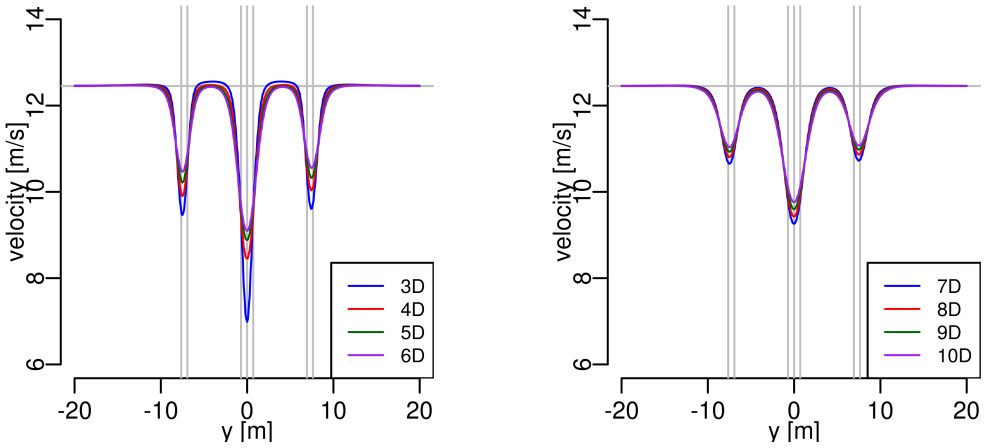
(a) (b)

Figure B.6: Mean velocity profiles for Case3-B. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.



(a) (b)

Figure B.7: Mean velocity profiles for Case3-C. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.

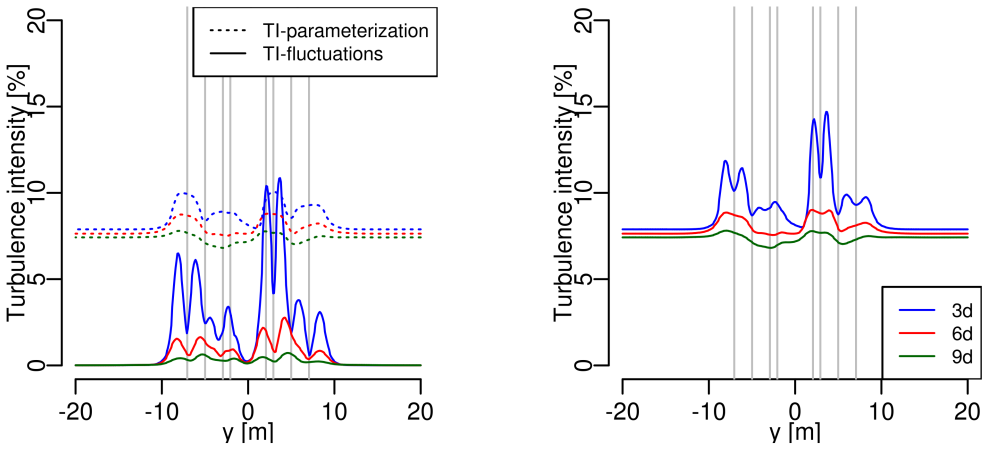


(a) (b)

Figure B.8: Mean velocity profiles for Case3-D. The vertical lines indicates the center of each cylinder. a) Near-wake region, 3D-6D downstream of the cylinders. b) Far-wake region, 7D-10D downstream of the cylinders.

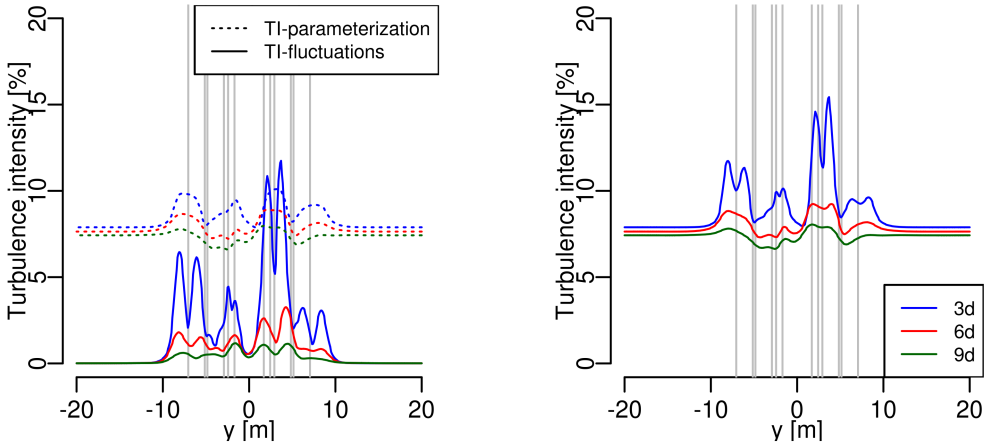
Appendix C

Additional figures: Turbulence intensity profiles



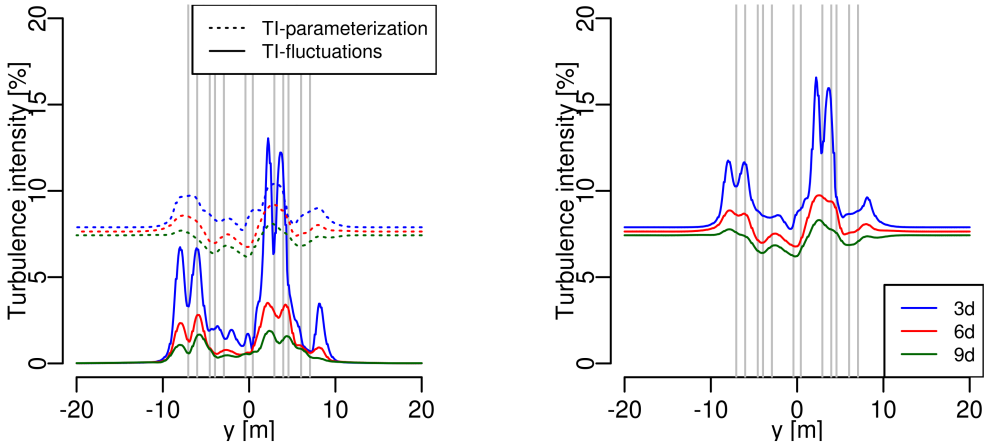
(a) (b)

Figure C.1: Turbulence intensity for the Case2-A. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



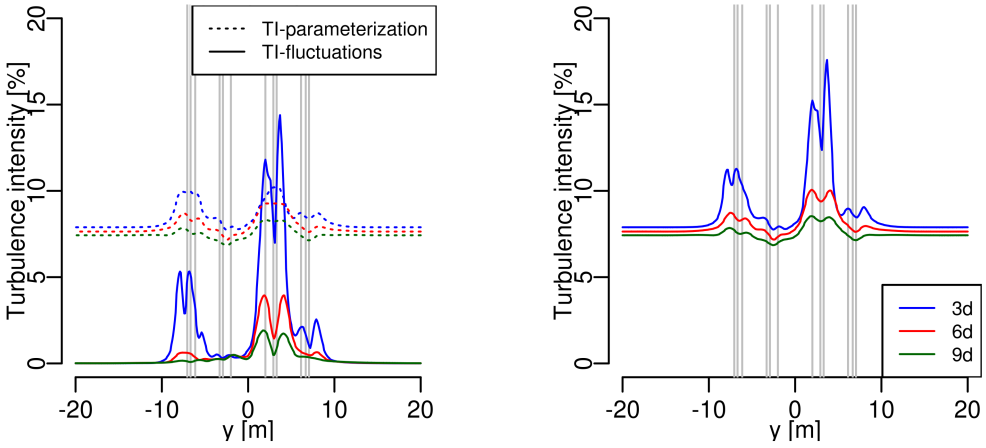
(a) (b)

Figure C.2: Turbulence intensity for the Case2-B. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



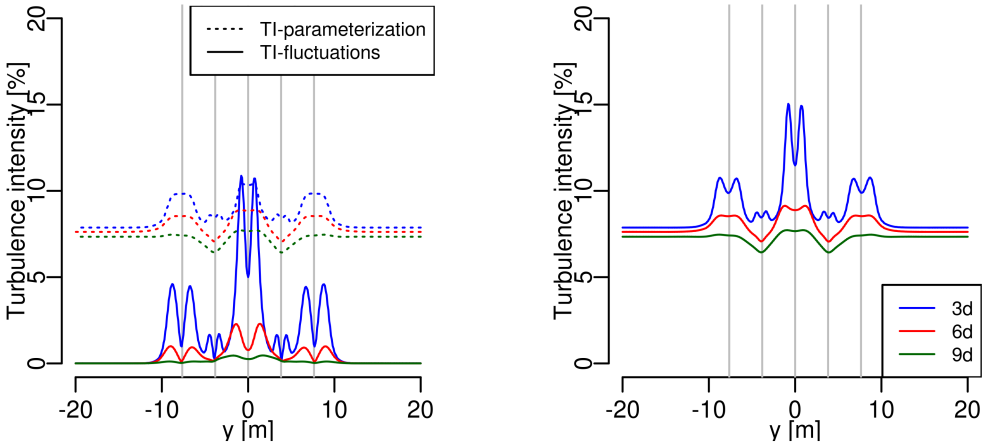
(a) (b)

Figure C.3: Turbulence intensity for the Case2-C. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



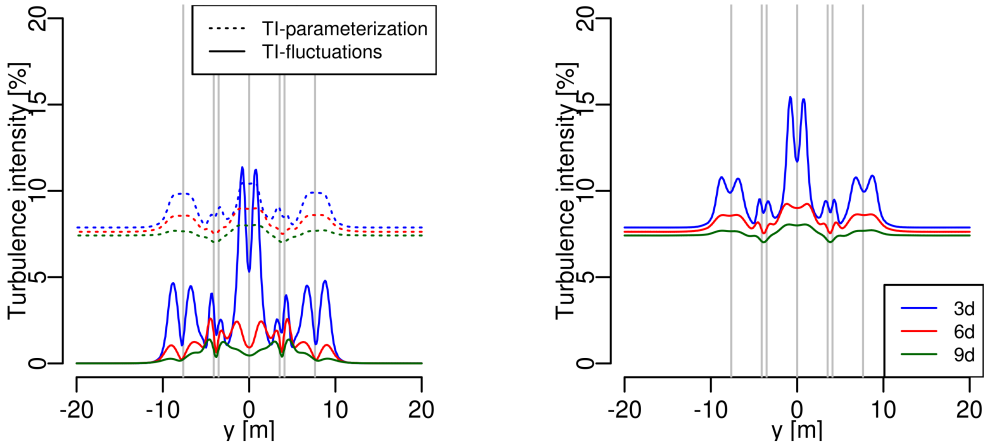
(a) (b)

Figure C.4: Turbulence intensity for the Case2-D. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



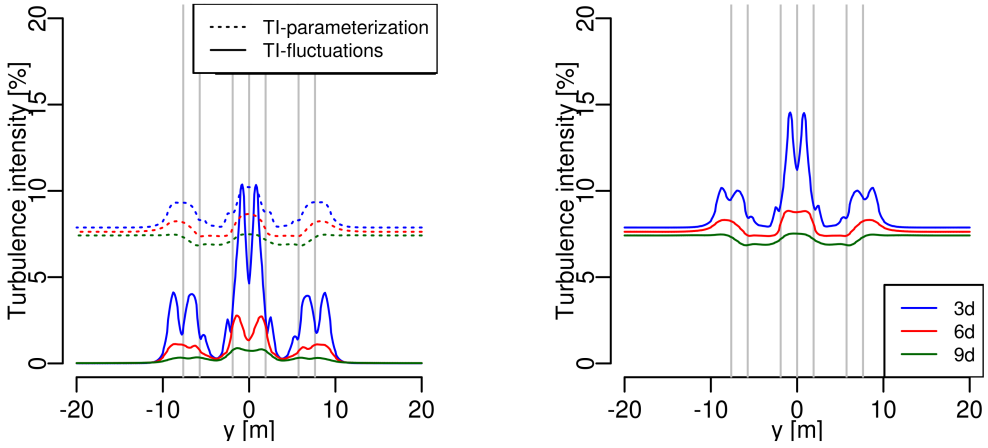
(a) (b)

Figure C.5: Turbulence intensity for the Case3-A. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



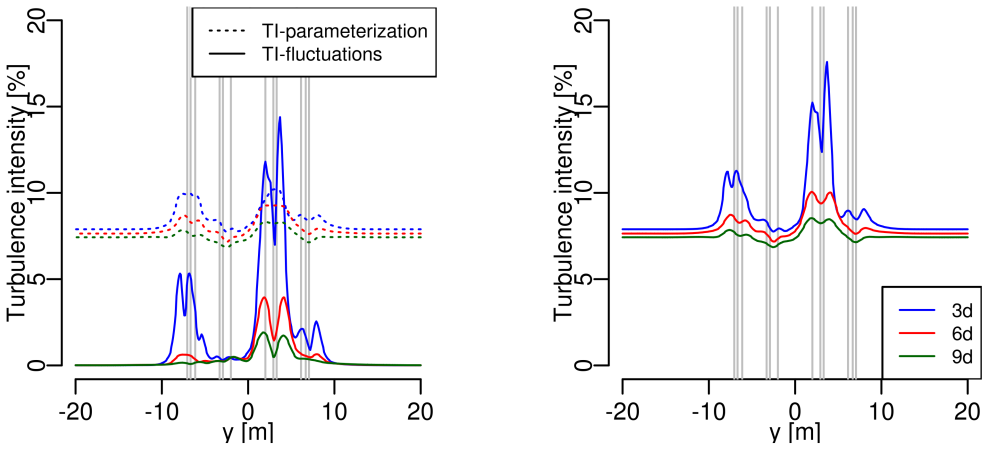
(a) (b)

Figure C.6: Turbulence intensity for the Case3-B. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity



(a) (b)

Figure C.7: Turbulence intensity for the Case3-C. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity

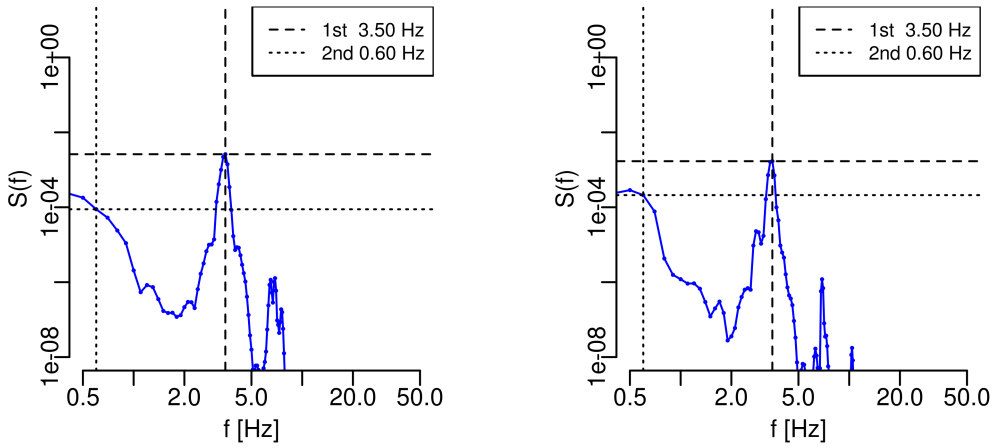


(a) (b)

Figure C.8: Turbulence intensity for the Case3-D. The vertical lines indicates the center of each cylinder. a) The two different components. b) Total turbulence intensity

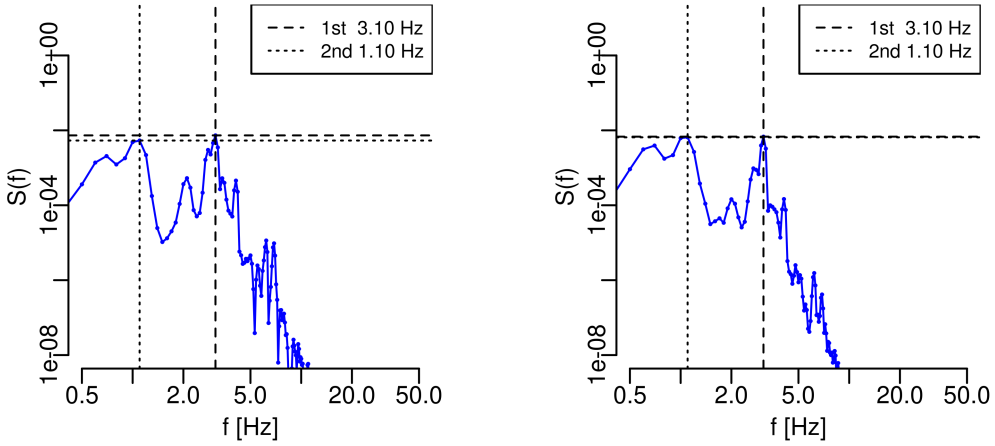
Appendix D

Additional figures: Power spectra plots



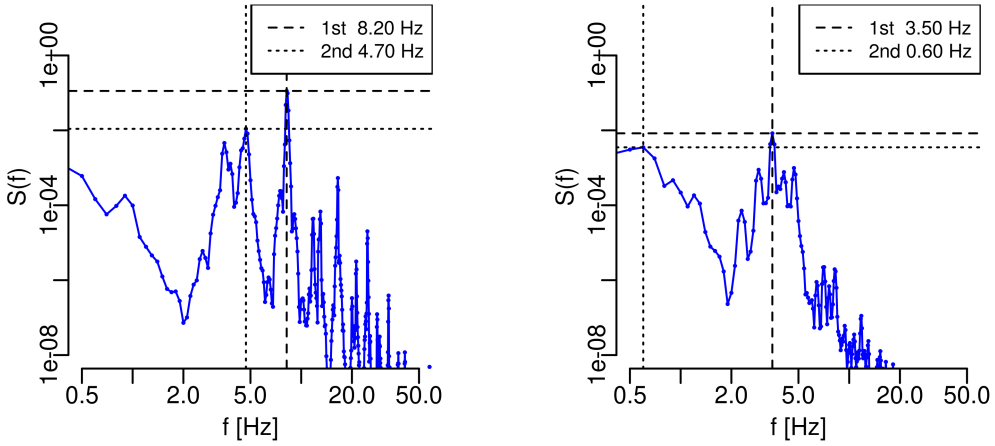
(a) (b)

Figure D.1: Power spectral density downstream of the truss tower, Case2-A. a) 3D downstream. b) 6D downstream



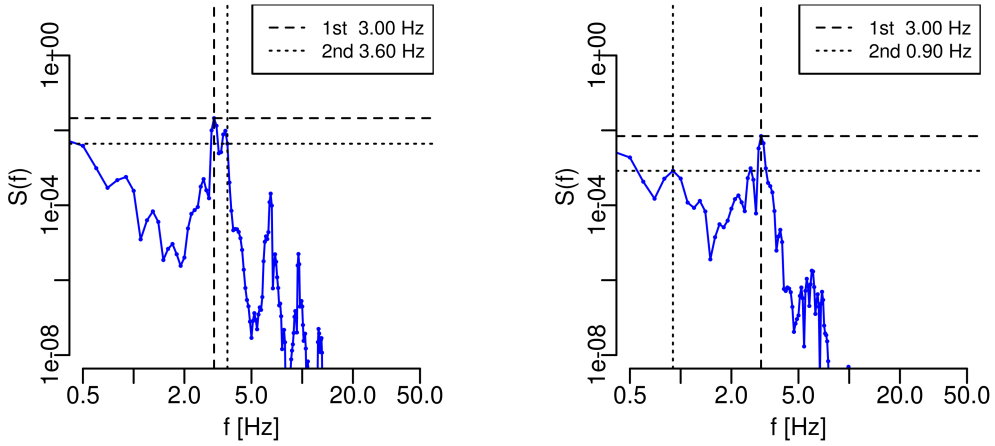
(a) (b)

Figure D.2: Power spectral density downstream of the truss tower, Case2-B. a) 3D downstream. b) 6D downstream



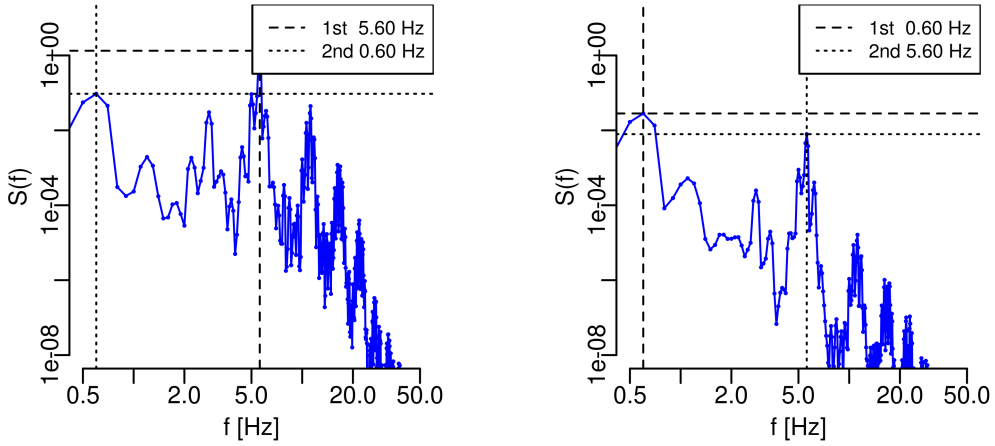
(a) (b)

Figure D.3: Power spectral density downstream of the truss tower, Case2-C. a) 3D downstream. b) 6D downstream



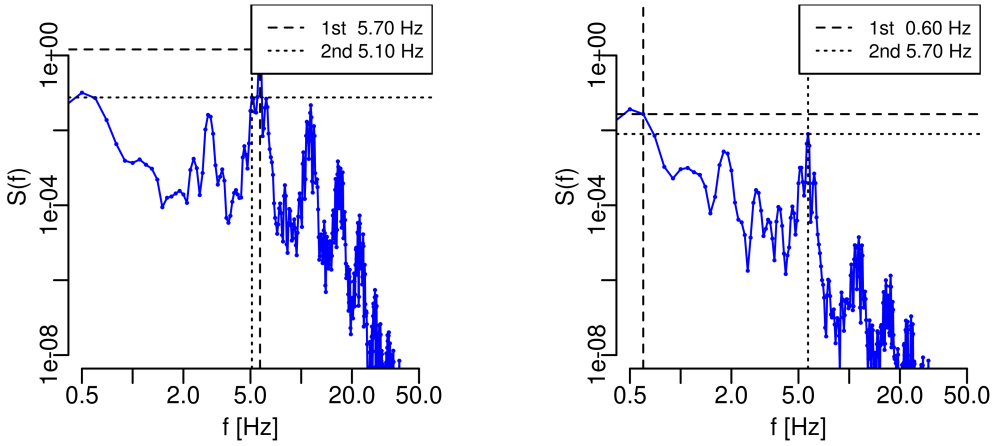
(a) (b)

Figure D.4: Power spectral density downstream of the truss tower, Case2-D. a) 3D downstream. b) 6D downstream



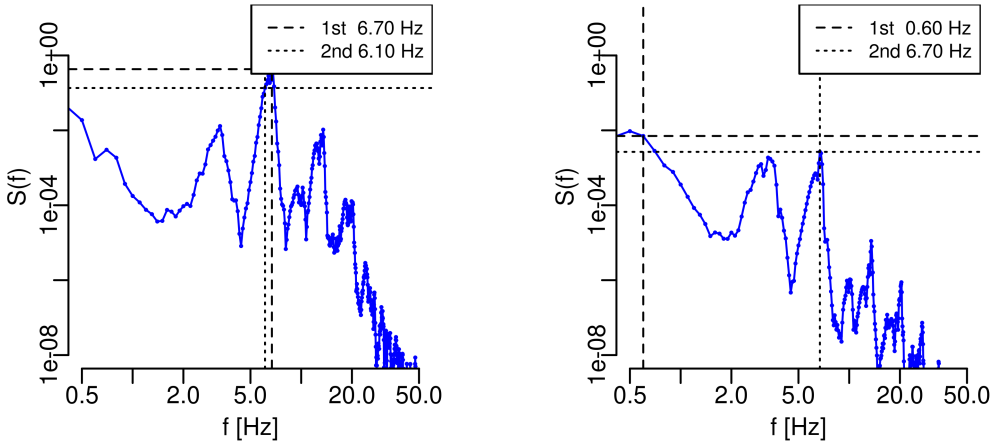
(a) (b)

Figure D.5: Power spectral density downstream of the truss tower, Case3-A. a) 3D downstream. b) 6D downstream



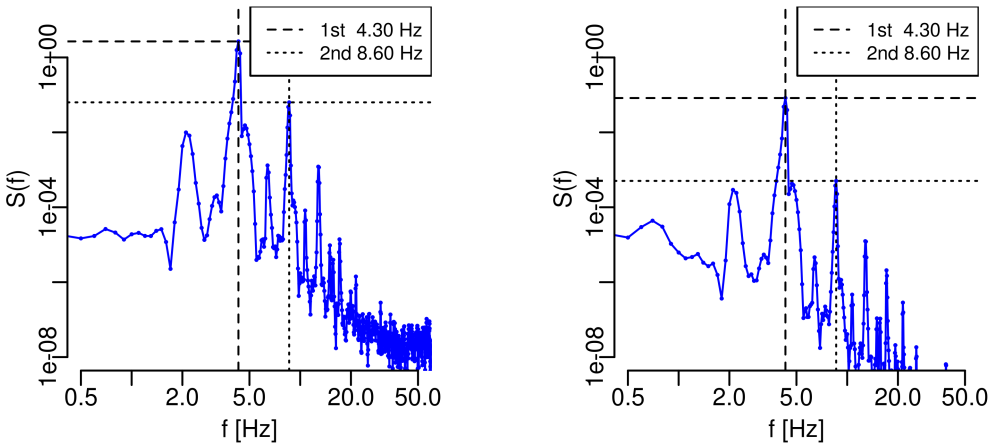
(a) (b)

Figure D.6: Power spectral density downstream of the truss tower, Case3-B. a) 3D downstream. b) 6D downstream



(a) (b)

Figure D.7: Power spectral density downstream of the truss tower, Case3-C. a) 3D downstream. b) 6D downstream

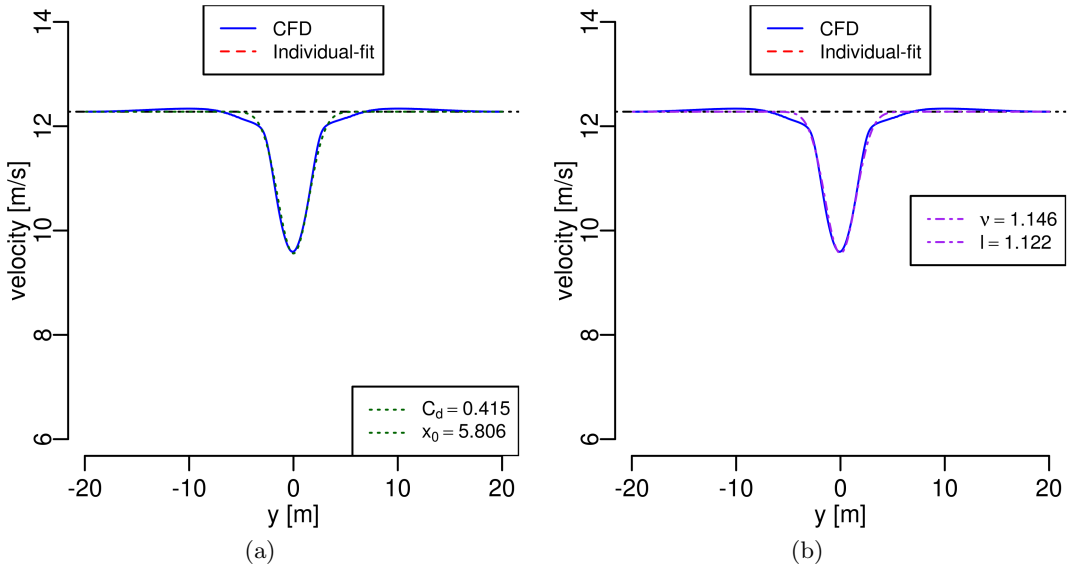


(a) (b)

Figure D.8: Power spectral density downstream of the truss tower, Case3-D. a) 3D downstream. b) 6D downstream

Appendix E

Additional figures: Tower shadow profiles



(a) (b)

Figure E.1: Mean velocity profile for the monopile represented by the CFD-simulations, Blevins' and Schlichting's model 3D downstream. a) Blevins' model b) Schlichtings model

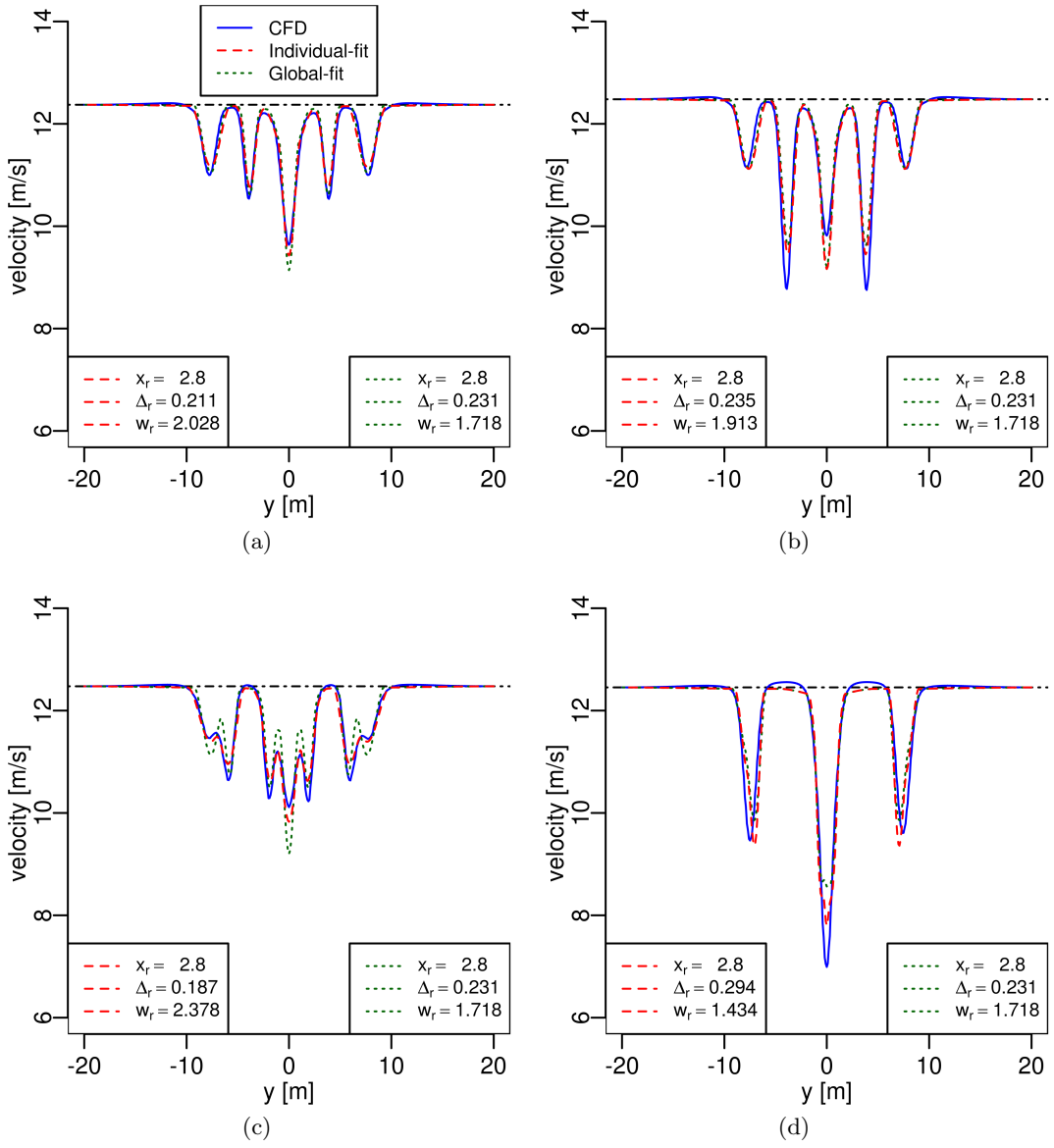


Figure E.2: Mean velocity profiles for Case3 represented by the CFD-simulations and Powles' model 3D downstream, including both the individual(red line) and global fit(green line). a) Case3-A b) Case3-B c) Case3-C d) Case3-D

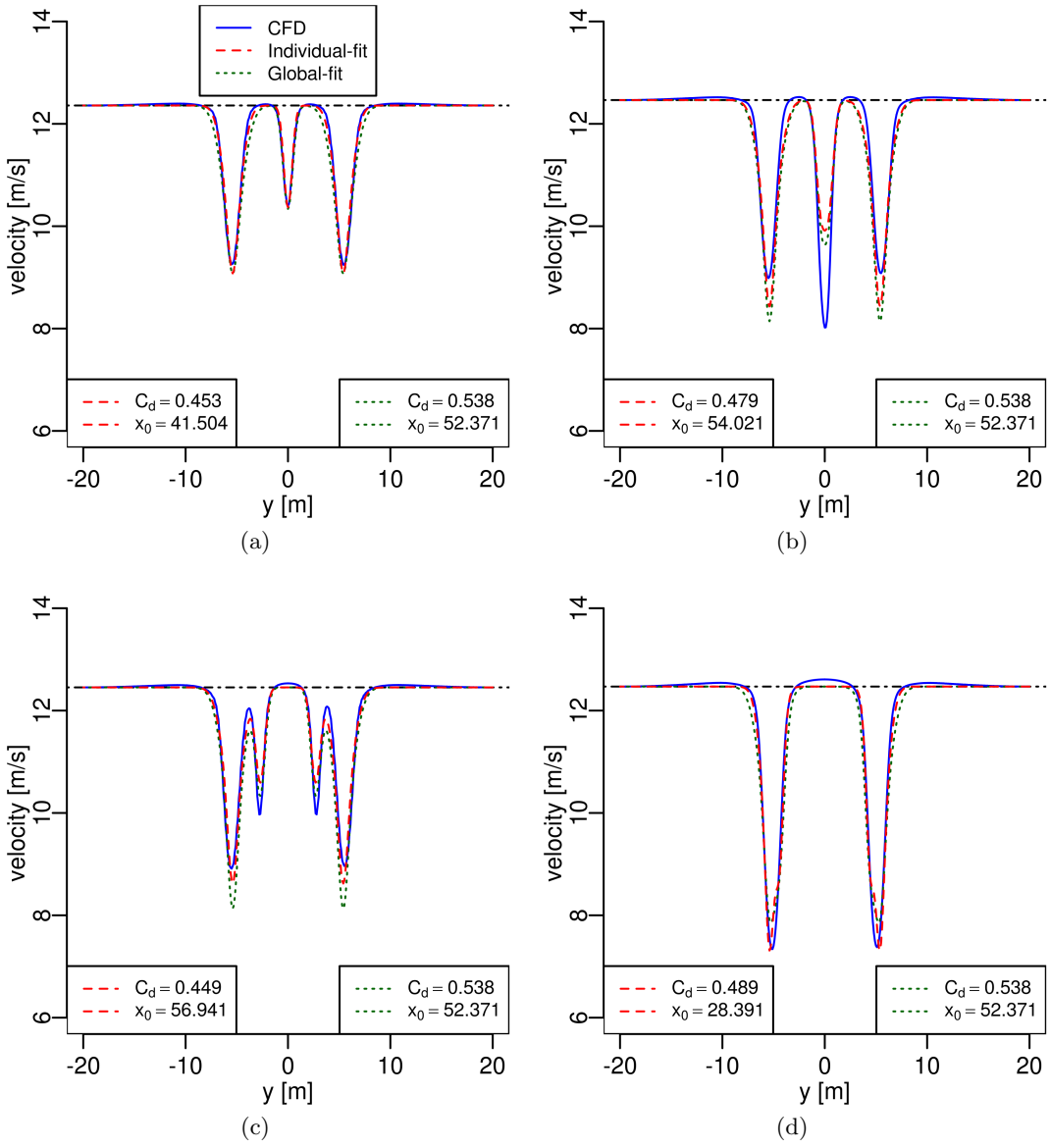


Figure E.3: Mean velocity profiles for Case1 represented by the CFD-simulations and Blevins' model 3D downstream, including both the individual(red line) and global fit(green line). a) Case1-A b) Case1-B c) Case1-C d) Case1-D

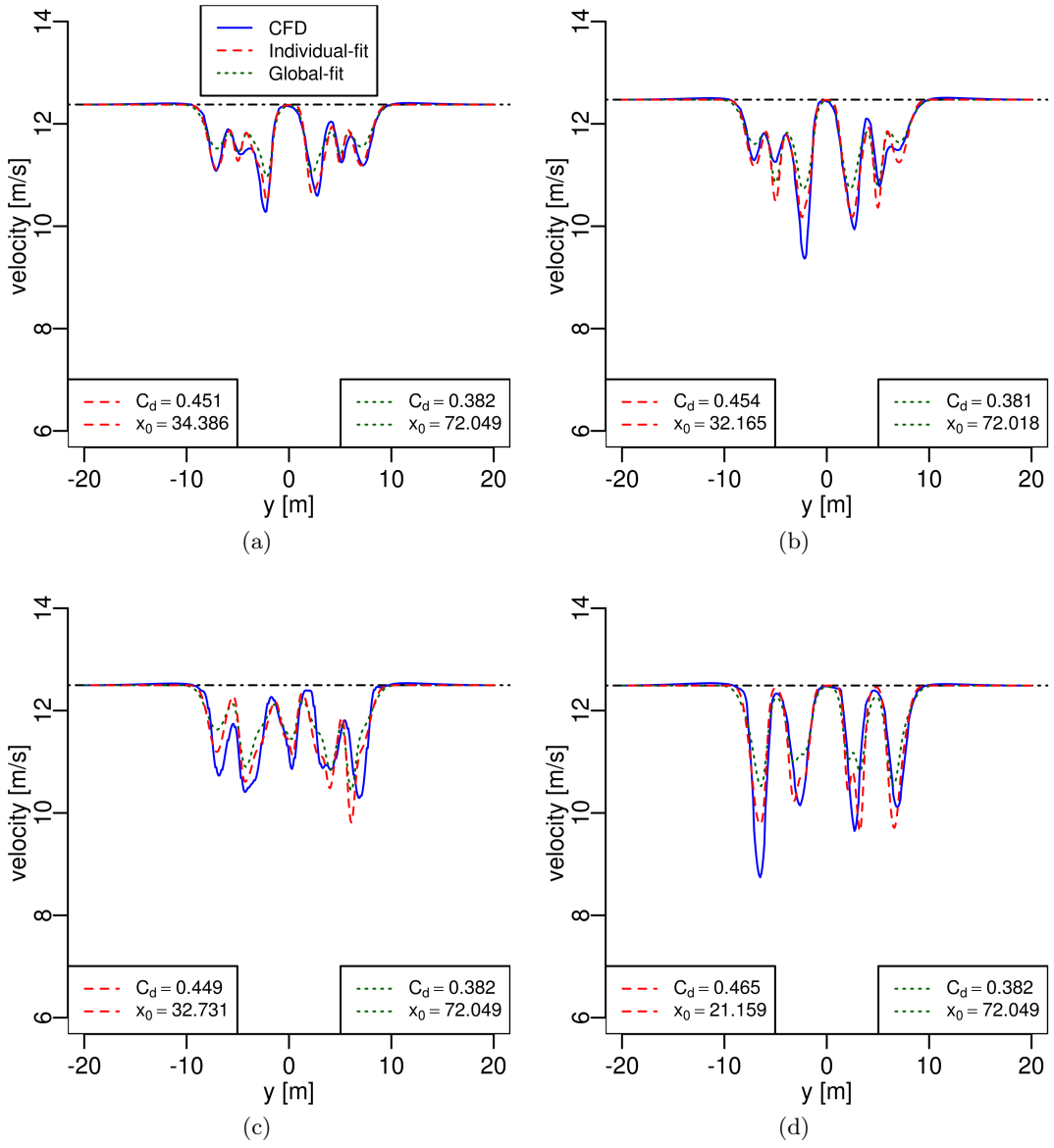


Figure E.4: Mean velocity profiles for Case2 represented by the CFD-simulations and Blevins' model 3D downstream, including both the individual(red line) and global fit(green line). a) Case2-A b) Case2-B c) Case2-C d) Case2-D

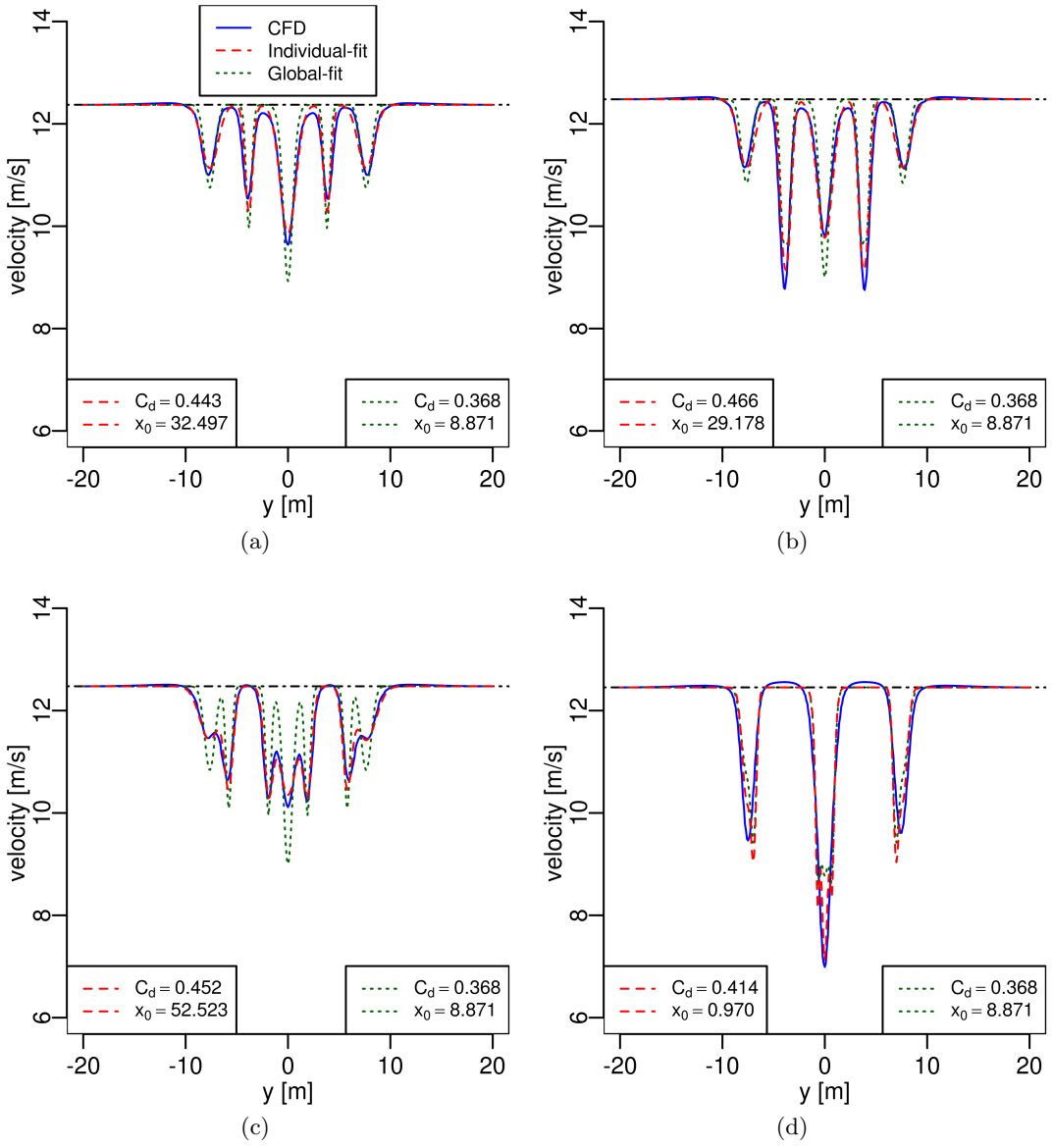


Figure E.5: Mean velocity profiles for Case3 represented by the CFD-simulations and Blevins' model 3D downstream, including both the individual(red line) and global fit(green line). a) Case3-A b) Case3-B c) Case3-C d) Case3-D

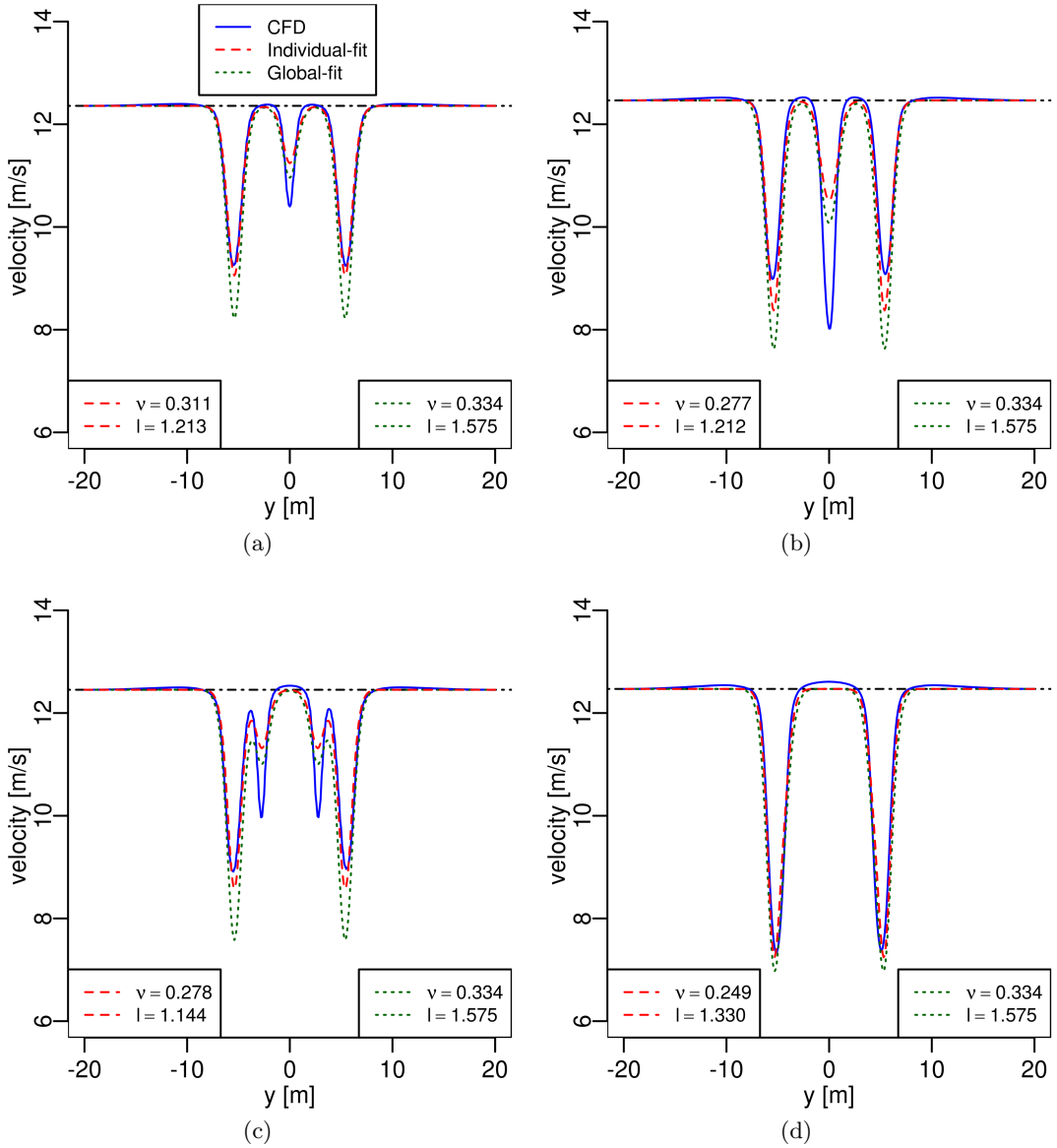


Figure E.6: Mean velocity profiles for Case1 represented by Schlichting's model 3D downstream, including both the individual(red line) and global fit(green line). a) Case1-A b) Case1-B c) Case1-C d) Case1-D

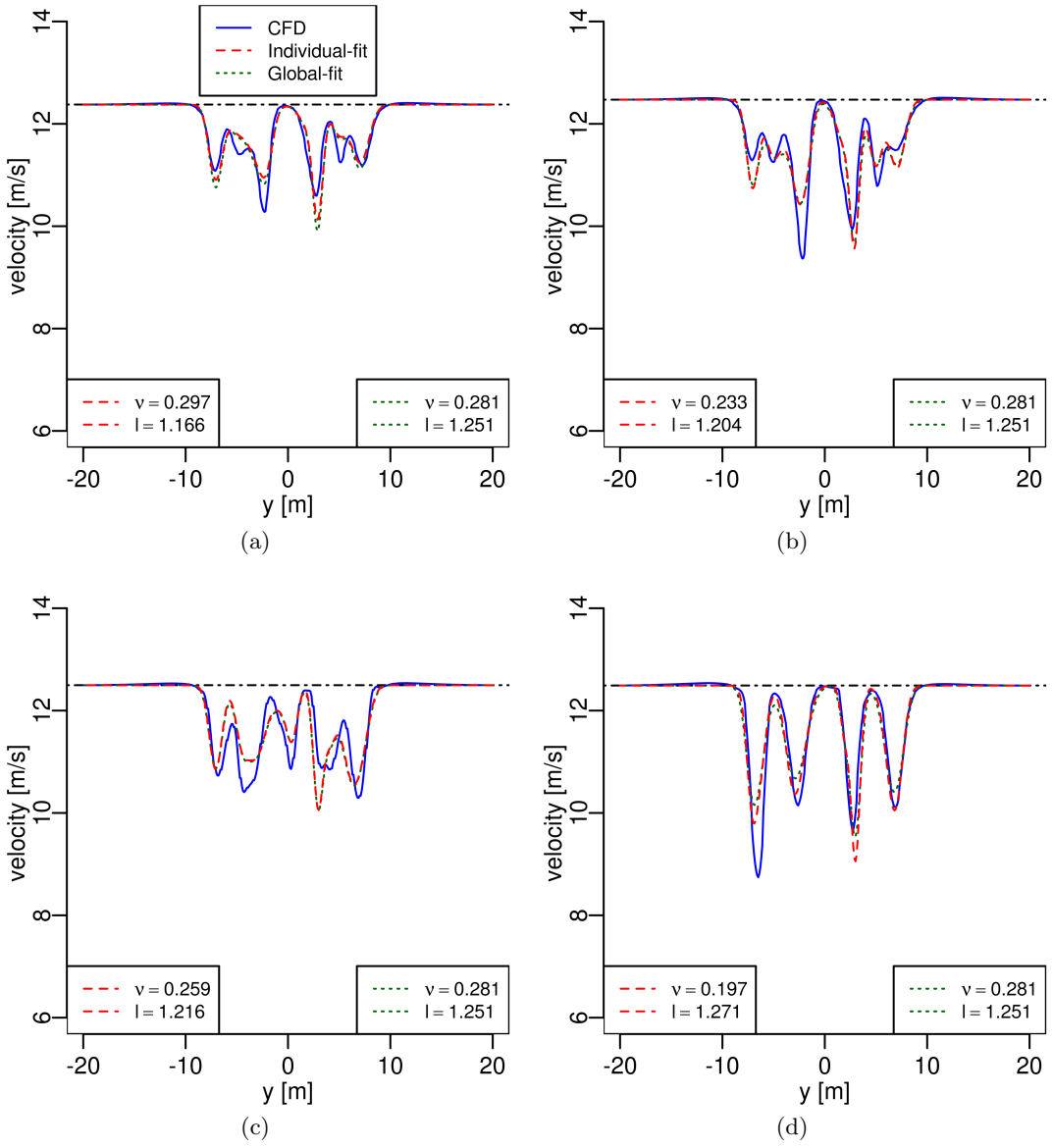


Figure E.7: Mean velocity profiles for Case2 represented by Schlichting's model 3D downstream, including both the individual(red line) and global fit(green line). a) Case2-A b) Case2-B c) Case2-C d) Case2-D

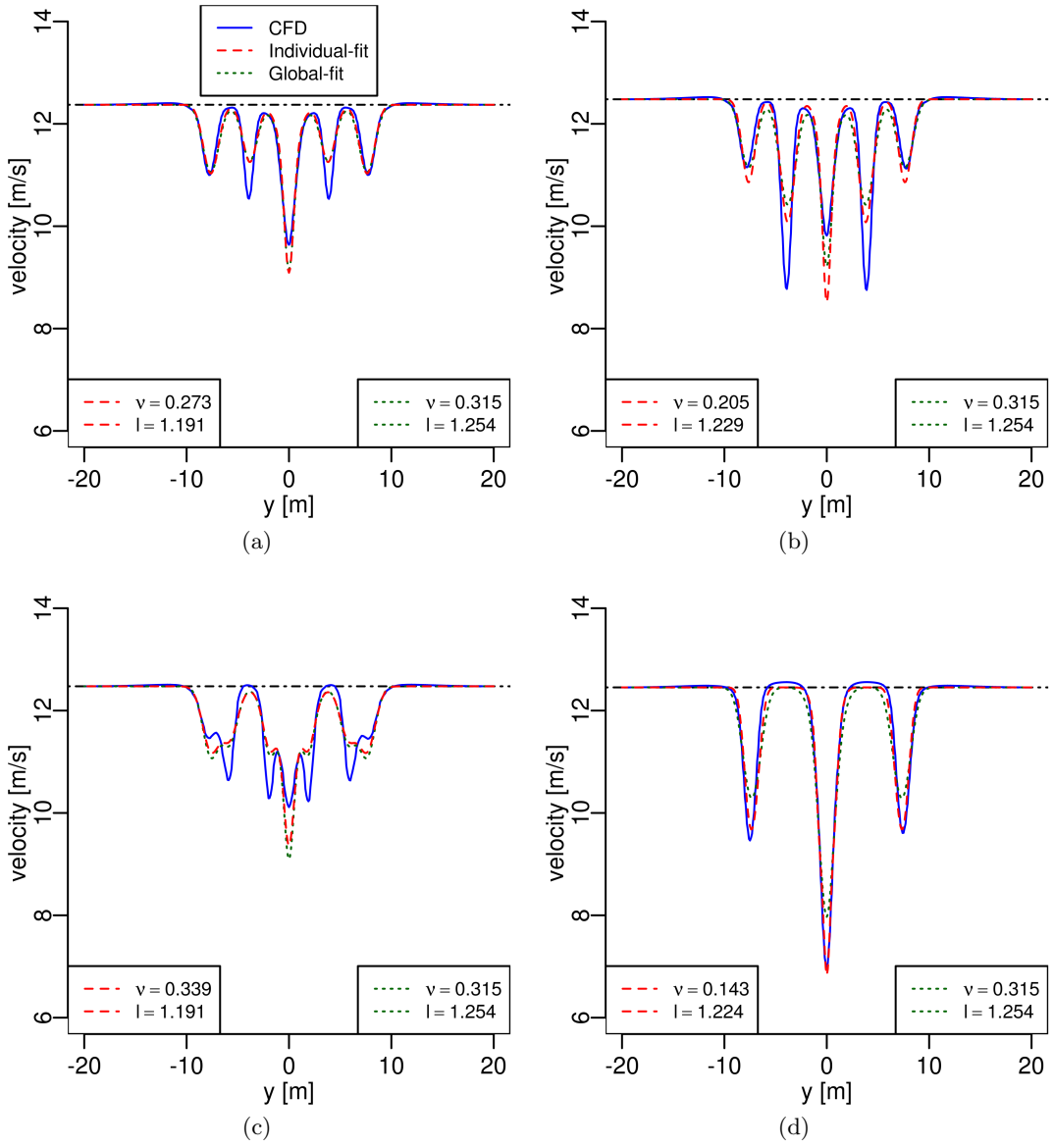


Figure E.8: Mean velocity profiles for Case3 represented by Schlichting's model 3D downstream, including both the individual(red line) and global fit(green line). a) Case3-A b) Case3-B c) Case3-C d) Case3-D

Appendix F

Additional figures: Individual parameter estimates

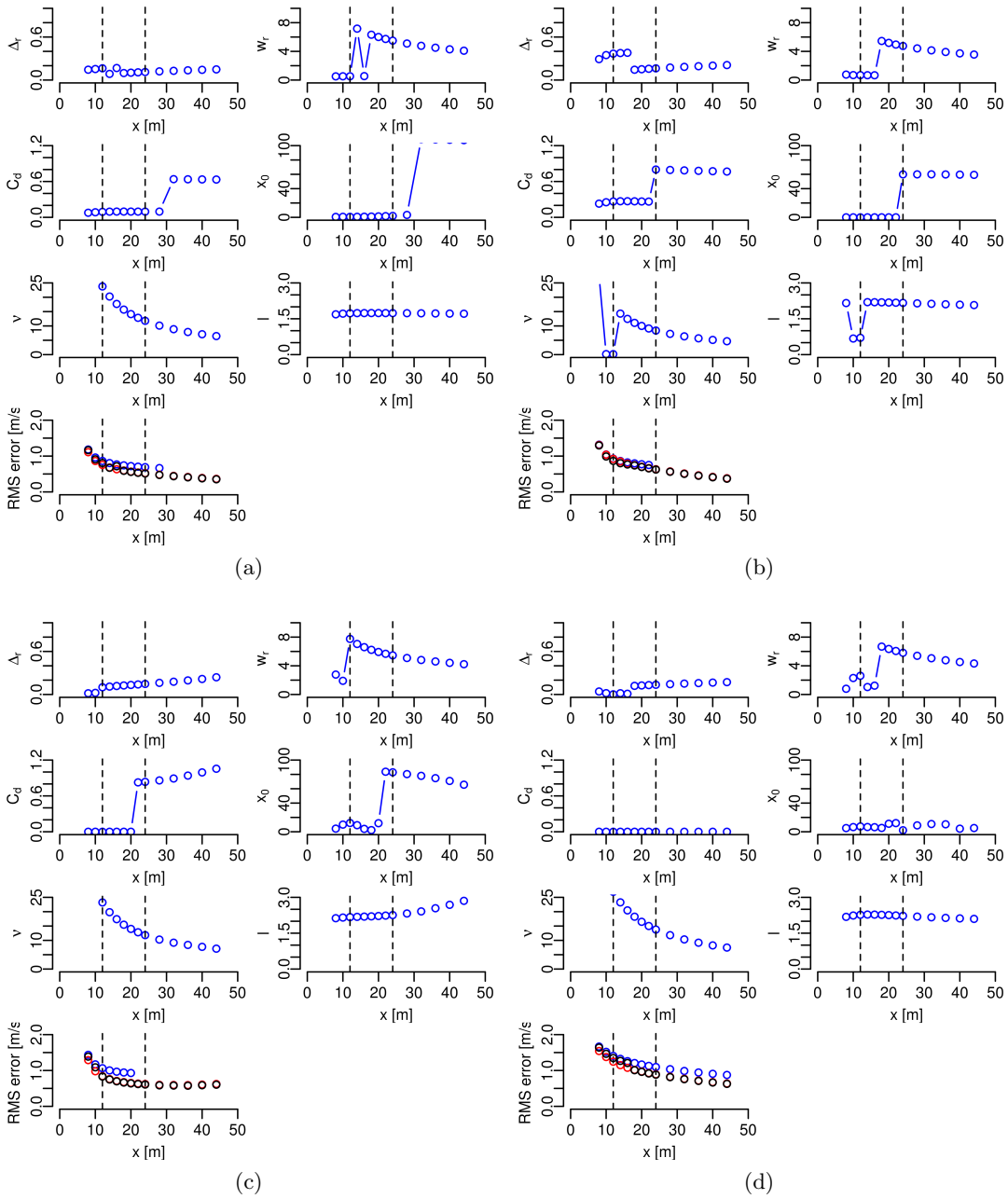


Figure F.1: Individual parameter estimates for Case1 for Powles'(top row), Blevins'(second row) and Schlichting's(third row) model. RMS-error found in bottom row. a) Case1-A b) Case1-B c) Case1-C d) Case1-D

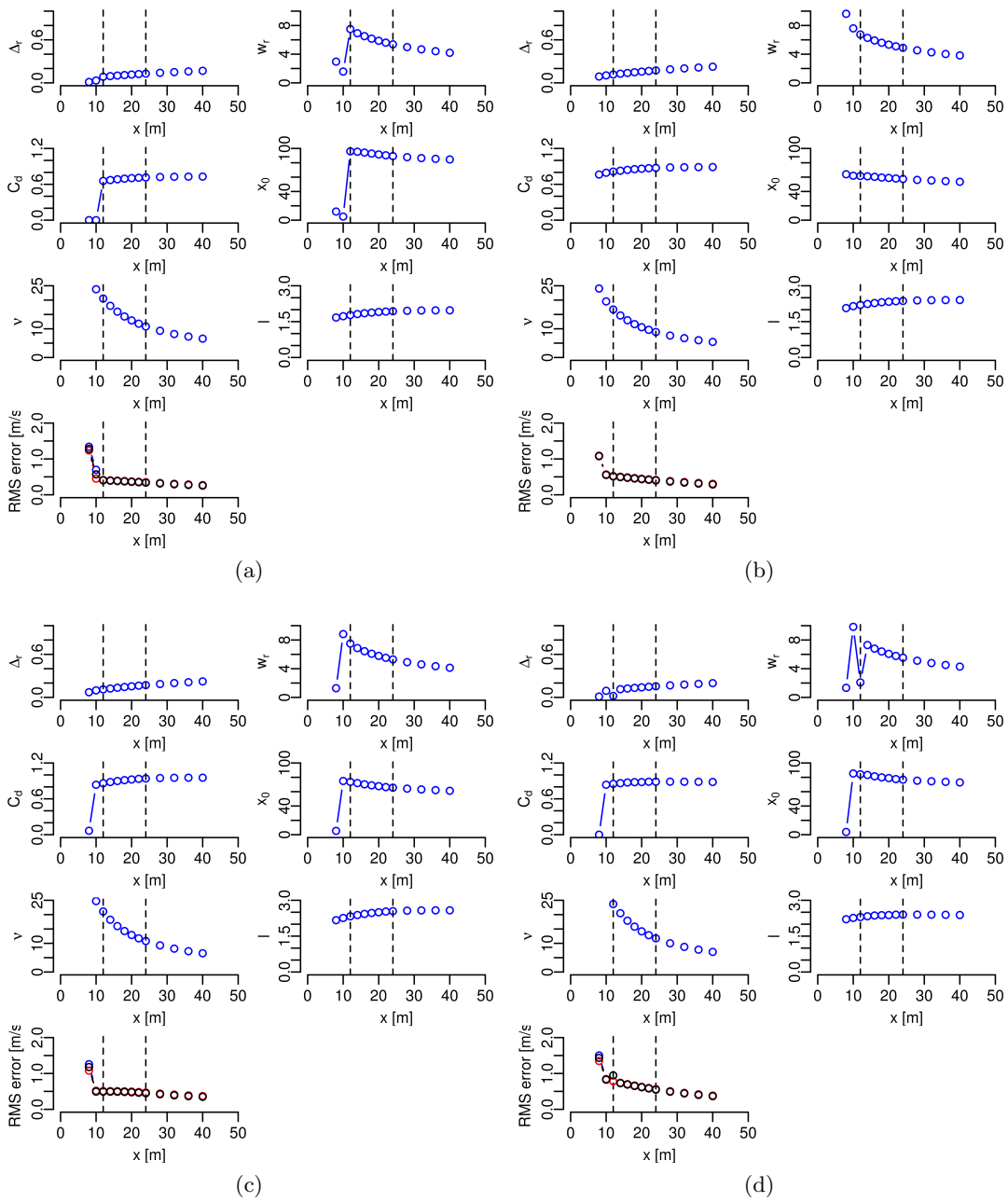


Figure F.2: Individual parameter estimates for Case2 for Powles'(top row), Blevins'(second row) and Schlichting's(third row) model. RMS-error found in bottom row. a) Case2-A b) Case2-B c) Case2-C d) Case2-D

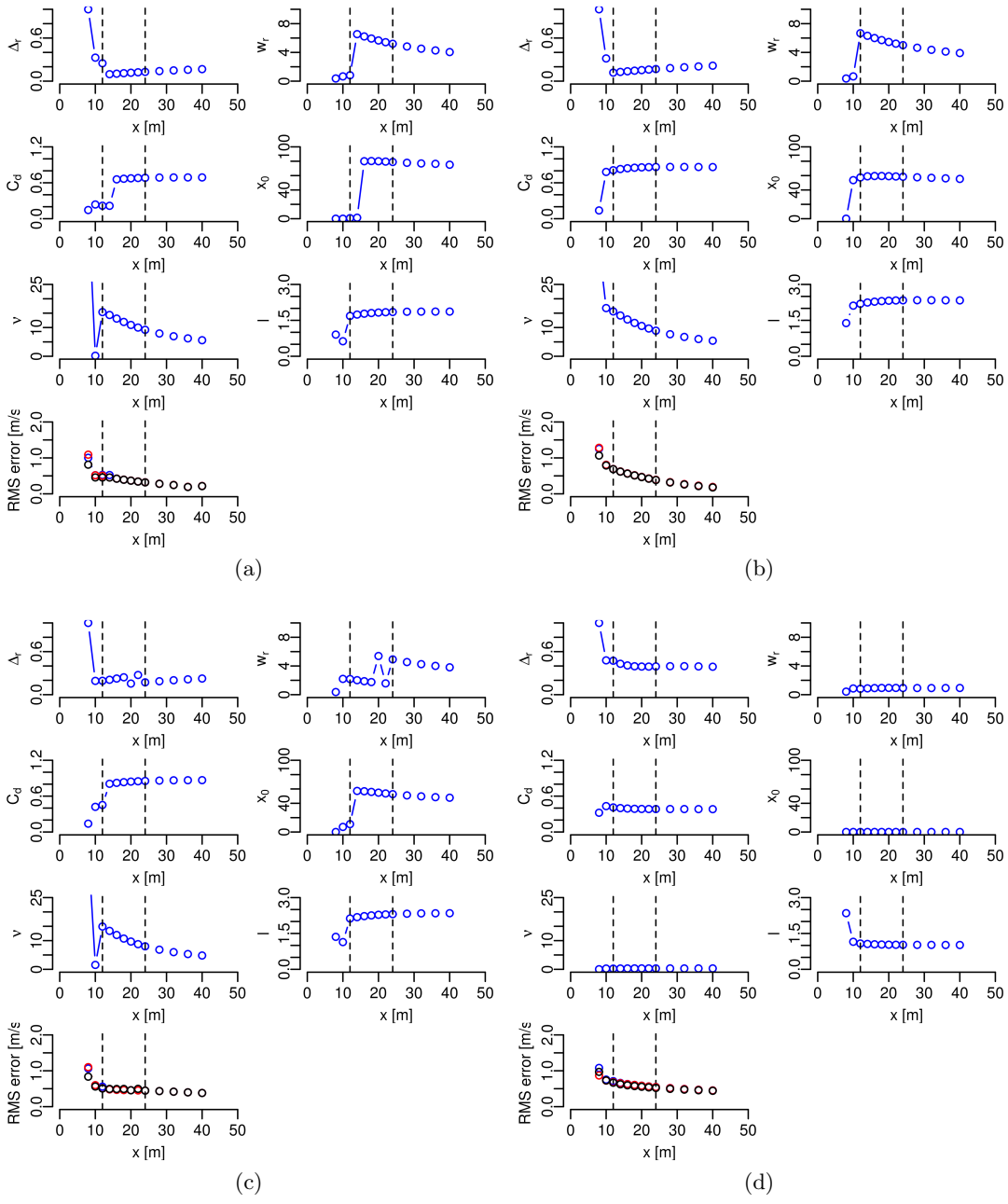


Figure F.3: Individual parameter estimates for Case3 for Powles'(top row), Blevins'(second row) and Schlichting's(third row) model. RMS-error found in bottom row. a) Case3-A b) Case3-B c) Case3-C d) Case3-D

Appendix G

Derivation of total turbulence intensity

The turbulence intensity TI is defined as a ratio between velocity fluctuations (u') and the mean free-stream velocity (U_0):

$$TI = \frac{u'}{U_0} \, . \tag{G.1}$$

There are two contributions to the total velocity fluctuations $u' = u'_{tot}$. Together with the contribution from the sub-grid parameterization there is another contribution from the unsteady motions (in time).

Since the numerical study is a stochastic process the velocity in each time step, the random variable U_i, includes a predication of turbulent motion. This gives the decomposition

$$U_i = \overline{U_i} + {U_t}^{(i)}, i = 1, ..., N, \tag{G.2}$$

where i is the index number of N time-steps. The turbulent contribution from each time-step is denoted as ${U_t}^{(i)}$ and $\overline{U_i} = \overline{U} + {U_u}^{(i)}$ is the ensemble mean written in terms of the overall mean velocity $\overline{U}$ and the unsteady fluctuations ${U_u}^{(i)}$ for each time-step i. The ensemble mean is usually written in brackets $< U_i >$,

$$\overline{U}_i =< U_i >= \lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} U_i^{(j)}, \tag{G.3}$$

where j is the ensemble index. Note that $< {U_t}^{(i)} >= 0$ and $\frac{1}{N} \sum_{i=1}^{N} {U_u}^{(i)} = 0$.

The total mean velocity is $\overline{U}$ or the total sum of all ensemble mean velocities divided by the total number of N time-steps,

$$\overline{U} = \frac{1}{N} \sum_{i=1}^{N} \overline{U_i} = \frac{1}{N} \sum_{i=1}^{N} < U_i > = \frac{1}{N} \sum_{i=1}^{N} < U >_i \tag{G.4}$$

Further, the variance of the total velocity describes the total fluctuations due to turbulence:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} < (U_i - \overline{U})^2 > \tag{G.5}$$

$$= \frac{1}{N}\sum_{i=1}^{N} < ({U_i}^2 - 2U_i\overline{U} + \overline{U}^2) > \tag{G.6}$$

$$= \overline{U}^2 - 2\overline{U}^2 + \frac{1}{n}\sum_{i=1}^{N} < {U_i}^2 > \tag{G.7}$$

$$= \frac{1}{N}\sum_{i=1}^{N} < {U_i}^2 > - \overline{U}^2 \ . \tag{G.8}$$

The term $< {U_i}^2 >$ is unknown, but since the ensemble variance is known from the sub-grid parameterization,

$$\sigma^2_{\text{ensamble},i} = < (U_i - \overline{U_i})^2 > \tag{G.9}$$

$$= < U_i^2 > - \overline{U_i}^2, \tag{G.10}$$

and the variance for the unsteady turbulent motions are known

$$\sigma^2_{\text{unsteady}} = \frac{1}{N}\sum_{i=1}^{N} (\overline{U_i} - \overline{U})^2 \tag{G.11}$$

$$= \frac{1}{N}\sum_{i=1}^{n} \overline{U_i}^2 - \overline{U}^2 \tag{G.12}$$

then

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (\sigma^2_{\text{ensamble},i} + \overline{U_i}^2) - \overline{U}^2 \tag{G.13}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \sigma^2_{\text{ensamble},i} + \sigma^2_{\text{unsteady}} \tag{G.14}$$

The total turbulent intensity is then,

$$TI_{(tot)} = \frac{\sqrt{\sigma_{\text{unsteady}} + (\frac{1}{N}\sum_{i=1}^{N} \sigma_{\text{ensemble},i})}}{U_0} \tag{G.15}$$

$$= \frac{\sqrt{\sigma^2}}{U_0} \ . \tag{G.16}$$

Appendix H

Scripts

H.1 Tower shadow functions, R-code

Tower shadow functions. This code was provided by Michael Muskulus and modified by Torbjørn Ruud Hagen.

```
# shadow.R: compute, estimate and plot tower shadow models
# Copyright (C) 2011 Michael Muskulus (michael.muskulus@ntnu.no)
# Version 0.9c -- ISOPE-2011 paper bugfix

# CHANGELOG
#
# Version 0.9e -- Master Thesis (TorbjÃ¸rn)
#   - Ready for plotting turbulence intensity
#
#
# Version 0.9d -- Master Thesis adds (TorbjÃ¸rn)
#   - Made ready for implementing 22.5 degree cases
#   - Addjusted plots for truss and monopile, ekstra points for black and white
    plots
#   - Updated: Possiblities for choosing which model you want to plot
#
# Version 0.9c -- ISOPE-2011 paper bugfix
#   - Corrected calculation of tower shadow models for truss towers: all members
    now used
#   - Fixed a sign in case A1
#   - Fixed plotting: showed wrong profiles (relative to begin of structure, not
    center)
#
# Version 0.9b -- ISOPE-2011 paper
#   - Version used for the paper submitted to ISOPE-2011
#


D    <- 4.0        # monopile diameter
Dl   <- 0.9        # truss large diameter
Ds   <- 0.36       # truss small diameter
ln   <- 401        # no. points on rake line
ly   <- seq(-20.00,+20.00,0.1) # y-coordinates along rake line
```

```
V0    <- 12.0       # free stream velocity
ref   <- 2.825      # reference distance in D for Powles

# options

adapt.v <- 1        # use actual mean velocity, not free-stream V0?
                    # values: 0 (use V0), 1 (use V at border), 2 (use profile mean
                       V)
powles.verbose <- F # show details of Powles / Bladed model?

#fdir <- "Rdata/"   # where to find / store the data
#fdir <- "/media/Iomega_HDD/Work/Rdata/" #If the data is stored on external
    harddrive
fdir <- "/media/Iomega_HDD/Work/Rdata/TurbInflow/" #If the data is stored on
    external harddrive plus turbulent inflow
# CAVEAT
# lx, mx are "real positions" (from beginning of structure)
# x       is "relative position" (from center of structure)
# conversion: x + sx is "real position"
#             mx- sx is "relative position"

# monopile data
# rake lines
fn.m <- c("-4d","-3d","-2d","-1d","0d","1d","2d","3d","4d","5d","6d","7d","8d","9
    d","10d","11d","12d",
"-4p5d","-3p5d","-2p5d","-1p5d","-0p5d","0p5d","1p5d","2p5d","3p5d","4p5d","5p5d")
ds.m <- c(seq(-4,12),seq(-4.5,5.5))
stopifnot(length(ds.m) == length(fn.m))
# center of structure
sx.m <- 0.0
# sort rake lines
ix <- sort(ds.m,index.return=TRUE)$ix
fn.m <- fn.m[ix]
ds.m <- ds.m[ix]
# position of rake lines
lx.m <- ds.m * D + sx.m

# truss tower data
# case A:  0 degrees
# case B: 45 degrees
# case C: 22p5 degrees
# rake lines
fn.A <- c("-4d","-3d","-2d","-1d","0d","1d","2d","3d","4d","5d","6d","7d","8d","9
    d","10d","11d","12d",
"-4p5d","-3p5d","-2p5d","-1p5d","-0p5d","0p5d","1p5d","2p5d","3p5d","4p5d","5p5d")
ds.A <- c(seq(-4,12),seq(-4.5,5.5))
stopifnot(length(ds.A) == length(fn.A))
fn.B <- fn.A
ds.B <- ds.A
stopifnot(length(ds.B) == length(fn.B))
fn.C <-fn.A
ds.C <-ds.A
stopifnot(length(ds.C) == length(fn.C))
# center of structure
sx.A <- 5.4
```

```
sx.B  <- 7.637
sx.C  <- 7.056
# sort rake lines
ix <- sort(ds.A,index.return=TRUE)$ix
fn.A <- fn.A[ix]
ds.A <- ds.A[ix]
ix <- sort(ds.B,index.return=TRUE)$ix
fn.B <- fn.B[ix]
ds.B <- ds.B[ix]
ix <- sort(ds.C,index.return=TRUE)$ix
fn.C <- fn.C[ix]
ds.C <- ds.C[ix]
# position of rake lines (CAVEAT: in terms of monopile diameter D)
lx.A  <- ds.A * D + sx.A
lx.B  <- ds.B * D + sx.B
lx.C  <- ds.C * D + sx.C

# truss tower cases
cases.A <- c(1,4,7,10)
cases.B <- c(2,5,8,11)
cases.C <- c(3,6,9,12)

# truss tower geometry
# large members
# case A: 0 degrees; cases 1,4,7,10
mx.A <- c(0,      0,  10.8,   10.8)
my.A <- c(5.4, -5.4,   5.4,   -5.4)
# case B: 45 degrees: cases 2,5,8,11
mx.B <- c(0, 15.274, 7.637,  7.637)
my.B <- c(0,      0, 7.637, -7.637)
# case C: 22p5 degrees: cases 3,6,9,12
mx.C <- c(0,      14.111, 9.978,  4.133)
my.C <- c(-2.923, 2.923, -7.056,  7.056)
# smaller members
# depend on case
mx.a1  <- c(  0, 10.8,    5.4,    5.4)
my.a1  <- c(  0,    0,    5.4,   -5.4)
mx.a4  <- c(  0,    0,   10.8,   10.8,    2.7,    2.7,    8.1,  8.1)
my.a4  <- c(2.7, -2.7,    2.7,   -2.7,    5.4,   -5.4,    5.4, -5.4)
mx.a7  <- c(  0,    0,   10.8,   10.8,      1,      1,    9.8,  9.8)
my.a7  <- c(4.4, -4.4,    4.4,   -4.4,    5.4,   -5.4,    5.4, -5.4)
mx.a10 <- c(  0,    0,   10.8,   10.8,      5,      5,    5.8,  5.8)
my.a10 <- c(0.4, -0.4,    0.4,   -0.4,    5.4,   -5.4,    5.4, -5.4)
mx.b2  <- c(3.818,  3.818, 11.455, 11.455)
my.b2  <- c(3.818, -3.818,  3.818, -3.818)
mx.b5  <- c(1.909,  1.909,  5.782,  5.782,  9.546,  9.546, 13.364, 13.364)
my.b5  <- c(1.909, -1.909,  5.782, -5.782,  5.782, -5.782,  1.909, -1.909)
mx.b8  <- c(0.707,  0.707, 14.567, 15.567,   6.93,   6.93,  8.344,  8.344)
my.b8  <- c(0.707, -0.707,  0.707, -0.707,   6.93,  -6.93,   6.93,  -6.93)
mx.b11 <- c(3.536,  3.536,  4.101,  4.101, 11.738, 11.738, 11.173, 11.173)
my.b11 <- c(3.536, -3.536,  4.101, -4.101,  3.536, -3.536,  4.101, -4.101)
mx.c3  <- c(  4.989,  12.044,   9.122,  2.067)
my.c3  <- c( -4.989,  -2.067,   4.989,  2.067)
mx.c6  <- c(  2.494,   7.483,  11.617,   6.628,  11.011, 13.078,  3.100,  1.033)
my.c6  <- c( -3.954,  6.0227,   3.956,  6.0227,  -4.562,  0.427,  4.562, -0.427)
```

```
mx.c9  <- c( 0.9239,  0.9054, 10.3607, 13.7283, 13.1871,  5.057, 3.7503, 0.3827)
my.c9  <- c(-3.3057, -6.6733, -6.1321,  1.998,  3.3057, 6.6733, 6.1321, -1.998)
mx.c12 <- c( 4.6194,  5.3585, 11.8914, 12.1976,  9.4916, 8.7525, 2.2196, 1.9134)
my.c12 <- c(-4.8364, -5.1426, -2.4366, -1.6975,  4.8364, 5.1426, 2.4366, 1.6975)
# tower shadow models
# potential solution (Wikipedia)
# U(r,theta) = U0*(1 + (R/r)^2)*sqrt(1-2*cos^2 theta)

"potential" <- function(x,y,V0,D=1) { # from Bladed theory manual
  Vx <- V0 + (y^2 - x^2)/(y^2 + x^2)^2 * (D/2)^2 * V0
  Vy <- (-2*x*y)   /(y^2 + x^2)^2 * (D/2)^2 * V0       # CAVEAT: V0 is vector in
      the manual (!)
  R  <- D/2
  r  <- sqrt(x^2 + y^2)
  list(Vx=Vx,Vy=Vy,V=sqrt(Vx^2 + Vy^2))
}

"potential2" <- function(x,y,V0,d) {  # alternative (Wikipedia)
  R  <- d/2
  r  <- sqrt(x^2 + y^2)
  #Vx <- (x^2 - y^2)/(y^2 + x^2) * V0 - V0 * (R/r)^2 # ??
  #Vy <- (-2*x*y)   /(y^2 + x^2)^2 * (D/2)^2 * V0    # ??
  #V  <- sqrt(Vx^2 + Vy^2)
  Va <- V0 * sqrt(1 + (R/r)^4 + 2*(R/r)^2 * (y^2 - x^2) / r^2)
  #list(Vx=Vx,Vy=Vy,V=sqrt(Vx^2 + Vy^2),Va=Va)
  list(Va=Va)
}

"powles" <- function(x,y,V0,d,delta,w,ref=NULL,debug=FALSE,bladed=TRUE,cut=TRUE,
    deg=60,mixed=TRUE
,smooth=TRUE,verbose=powles.verbose) {
  if (delta < 0) return(rep(Inf,length(y))) # for optimization
  if (delta > 1) return(rep(Inf,length(y))) # for optimization
  #if (w > 4) return(rep(Inf,length(y))) # for optimization
  if (!is.null(ref)) {  # just assume that we are in the wake region
    if (length(x) != 1) stop("Cannot process more than one x value yet...")
    if (verbose) cat(paste("Scaling given w=",round(w,3),"delta=",round(delta,3),"
        to "))
    f.ref <- sqrt(abs(x)/abs(ref*d))  # NEW: ref is in terms of diameter (!!)
    w     <- w * f.ref # increase with sqrt of distance
    delta <- delta / f.ref # decrease with sqrt of distance
    if (verbose) cat(paste("w=",round(w,3),"delta=",round(delta,3),"at x=",
x," instead of xref=",round(ref,3),", d=",round(d,3)," "))
  }

  x1 <- x
  if (length(x1) == 1) x1 <- rep(x,length(y))
  ym <- w*d/2
  if (debug) {
    cat(paste("Powles:\n"))
    cat(paste("x = ",x,"\n"))
    cat(paste("W = ",w,"\n"))
    cat(paste("D = ",d,"\n"))
    cat(paste("delta = ",delta,"\n"))
    print(delta)
```

```
  }
  A <- 1 - delta*(cos(pi*y/(w*d)))^2
  # Bladed: potential flow inside 60 degree cone
  if (bladed) {
    A2 <- potential2(x1,y,V0=V0,d)$Va / V0
    if (mixed) {
      # restrict Powles' model to one half-period
      ix1    <- which(y > (w*d) / 180 * 90  | x <= 0)
      ix2    <- which(y < -(w*d) / 180 * 90 | x <= 0)
      A[ix1] <- 1  # NEW
      A[ix2] <- 1  # NEW
      A1     <- A
      # use potential flow if it exhibits the larger deficit
      # gives continuous results
      ix     <- which(A2 < A)
      #print(A2[ix])
      A[ix] <- A2[ix]
      if (smooth) {
        # where do we have speedup and Powles exhibits slowdown?
        ix3 <- which(A2 > 1 & A < 1)
        # average both solutions if they exhibit different signs
        A[ix3] <- (A1[ix3] + A2[ix3])/2
      }
    } else { # restrict to cone around centerline
      # default is 60 degrees in argument
      ix1 <- which(y > (w*d) / 180 * 60  | x <= 0)
      ix2 <- which(y < -(w*d) / 180 * 60 | x <= 0)
      A[ix1] <- A2[ix1]
      A[ix2] <- A2[ix2]
    }
  } else {
    if (cut) {
      A[y > ym]  <- 1
      A[y < -ym] <- 1
    }
  }
  ix    <- which(sqrt(x^2 + y^2) <= d/2)
  A[ix] <- 0 # invalid contribution (inside cylinder)
  V     <- A*V0
  if (verbose) cat(paste("min.V=",round(min(V),3),"\n"))
  V
}

"blevins" <- function(x,y,V0,d,cd,x0,verbose=FALSE) {
  if (x < 0) return(rep(V0,length(y)))
  if (x0 < 0) return(rep(V0,length(y))) # NEW: virtual wake before cylinder (!)
  x0 <- x0 * d    # NEW: x0 in terms of diameter!
  #if (x0+x < 0 | cd < 0) cat(paste("*** Blevins x = ",x,", cd = ",cd," x0 = ",x0
      ,"\n"))
  if (x0+x < 0) return(rep(Inf,length(y)))  # for optimization algorithms
  if (cd < 0)   return(rep(Inf,length(y)))  # for optimization algorithms
  b  <- 0.23 * sqrt(cd*d*(x+x0))            # half-width of wake
  ud <- 1.02 * V0 * sqrt(cd*d / (x+x0))     # velocity deficit at centerline
  if (verbose) cat(sprintf("Blevins: b = %8.5f, ud = %8.5f \n",b,ud))
  v  <- V0 - ud * exp(-0.69*(y^2) / b^2)
```

```
  v
}

# mean drag: 1.485 +- 0.005 #This is wrong
#Correct mean drag 0.37 +-0.005

"schlichting" <- function(x,y,V0,d,nu,l) {
  if (x < 0) return(rep(V0,length(y)))
  #nu=1.460e-5
  l <- l * d  # NEW: assume l in terms of D
  if (l < 0) return(rep(Inf,length(y)))  # for optimization algorithms
  if (nu < 0) return(rep(Inf,length(y)))  # for optimization algorithms
  cd <- 0.37
  v <- V0 * cd / (4 * sqrt(pi)) * sqrt(V0*l/nu) / sqrt(x/(l)) * exp(-y^2 * V0 / (4
      *x*nu))
  # nu: kinematic viscosity, in m2/s
  v <- V0 - v
  v
}

# tower shadow models
# truss tower case

"bladed.a" <- function(x,y,V0,delta,w,case) {
  foo  <- 0.0
  # larger members
  if (case %in% cases.A) { mx <- mx.A - sx.A; my <- my.A }
  if (case %in% cases.B) { mx <- mx.B - sx.B; my <- my.B }
  if (case %in% cases.C) { mx <- mx.C - sx.C; my <- my.C }
  for (i in seq(along=mx)) { # larger members
    foo1 <- powles(x-mx[i], y-my[i], d= Dl, delta= delta, w= w, ref= ref,
bladed= TRUE, V0=V0)  # need to use reference
    foo  <- foo + (foo1 - V0)
  }
  # smaller members
  if (case == 1 ) { mx <- mx.a1;  my <- my.a1  }
  if (case == 2 ) { mx <- mx.b2;  my <- my.b2  }
  if (case == 3 ) { mx <- mx.c3;  my <- my.c3  }
  if (case == 4 ) { mx <- mx.a4;  my <- my.a4  }
  if (case == 5 ) { mx <- mx.b5;  my <- my.b5  }
  if (case == 6 ) { mx <- mx.c6;  my <- my.c6  }
  if (case == 7 ) { mx <- mx.a7;  my <- my.a7  }
  if (case == 8 ) { mx <- mx.b8;  my <- my.b8  }
  if (case == 9 ) { mx <- mx.c9;  my <- my.c9  }
  if (case == 10) { mx <- mx.a10; my <- my.a10 }
  if (case == 11) { mx <- mx.b11; my <- my.b11 }
  if (case == 12) { mx <- mx.c12; my <- my.c12 }
  if (case %in% cases.A) mx <- mx - sx.A
  if (case %in% cases.B) mx <- mx - sx.B
  if (case %in% cases.C) mx <- mx - sx.C
  for (i in seq(along=mx)) {
    foo1 <- powles(x-mx[i], y-my[i], d= Ds, delta= delta, w= w, ref= ref, bladed=
        TRUE,V0=V0)
    foo  <- foo + (foo1 - V0)
  }
```

```
  foo  <- sign(foo) * unlist(lapply(abs(foo), function(x) { min(x,V0) }))
  v    <- foo + V0
  v
}

"blevins.a" <- function(x,y,V0,cd,x0,case) {
  foo  <- 0.0
  # larger members
  if (case %in% cases.A) { mx <- mx.A - sx.A; my <- my.A }
  if (case %in% cases.B) { mx <- mx.B - sx.B; my <- my.B }
  if (case %in% cases.C) { mx <- mx.C - sx.C; my <- my.C }
  for (i in seq(along=mx)) {
    foo1 <- blevins(x-mx[i], y-my[i], Dl, cd, x0,V0=V0)
    foo  <- foo + (foo1 - V0)
  }
  # smaller members
  if (case == 1 ) { mx <- mx.a1;  my <- my.a1  }
  if (case == 2 ) { mx <- mx.b2;  my <- my.b2  }
  if (case == 3 ) { mx <- mx.c3;  my <- my.c3  }
  if (case == 4 ) { mx <- mx.a4;  my <- my.a4  }
  if (case == 5 ) { mx <- mx.b5;  my <- my.b5  }
  if (case == 6 ) { mx <- mx.c6;  my <- my.c6  }
  if (case == 7 ) { mx <- mx.a7;  my <- my.a7  }
  if (case == 8 ) { mx <- mx.b8;  my <- my.b8  }
  if (case == 9 ) { mx <- mx.c9;  my <- my.c9  }
  if (case == 10) { mx <- mx.a10; my <- my.a10 }
  if (case == 11) { mx <- mx.b11; my <- my.b11 }
  if (case == 12) { mx <- mx.c12; my <- my.c12 }
  if (case %in% cases.A) mx <- mx - sx.A
  if (case %in% cases.B) mx <- mx - sx.B
  if (case %in% cases.C) mx <- mx - sx.C
  for (i in seq(along=mx)) {
    foo1 <- blevins(x-mx[i], y-my[i], Ds, cd, x0,V0=V0)
    foo  <- foo + (foo1 - V0)
  }
  foo  <- sign(foo) * unlist(lapply(abs(foo), function(x) { min(x,V0) }))
  v    <- foo + V0
  v
}

"schlichting.a" <- function(x,y,V0,nu,l,case) {
  foo  <- 0.0
  # larger members
  if (case %in% cases.A) { mx <- mx.A - sx.A; my <- my.A }
  if (case %in% cases.B) { mx <- mx.B - sx.B; my <- my.B }
  if (case %in% cases.C) { mx <- mx.C - sx.C; my <- my.C }
  for (i in seq(along=mx)) {
    foo1 <- schlichting(x-mx[i], y-my[i], Dl, nu, l,V0=V0)
    foo  <- foo + (foo1 - V0)
  }
  # smaller members
  if (case == 1 ) { mx <- mx.a1;  my <- my.a1  }
  if (case == 2 ) { mx <- mx.b2;  my <- my.b2  }
  if (case == 3 ) { mx <- mx.c3;  my <- my.c3  }
  if (case == 4 ) { mx <- mx.a4;  my <- my.a4  }
```

```
  if (case == 5 ) { mx <- mx.b5;  my <- my.b5  }
  if (case == 6 ) { mx <- mx.c6;  my <- my.c6  }
  if (case == 7 ) { mx <- mx.a7;  my <- my.a7  }
  if (case == 8 ) { mx <- mx.b8;  my <- my.b8  }
  if (case == 9 ) { mx <- mx.c9;  my <- my.c9  }
  if (case == 10) { mx <- mx.a10; my <- my.a10 }
  if (case == 11) { mx <- mx.b11; my <- my.b11 }
  if (case == 12) { mx <- mx.c12; my <- my.c12 }
  if (case %in% cases.A) mx <- mx - sx.A
  if (case %in% cases.B) mx <- mx - sx.B
  if (case %in% cases.C) mx <- mx - sx.C
  for (i in seq(along=mx)) {
    foo1 <- schlichting(x-mx[i], y-my[i], Ds, nu, l,V0=V0)
    foo  <- foo + (foo1 - V0)
  }
  foo  <- sign(foo) * unlist(lapply(abs(foo), function(x) { min(x,V0) }))
  v    <- foo + V0
  v
}

# fitting the models

"fit.bladed" <- function(x,y,u,d,V0,quiet=TRUE,ntry=10,ref=NULL) {
  fn <- function(par) {
    delta <- par[1]
    w     <- par[2]
    v     <- powles(x, y, V0, d, delta, w, bladed= TRUE, ref= ref)
    rms   <- sqrt(mean((v-u)^2))
    rms
  }
  foo        <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.0,max=1.0),runif(n=1,min=0.5,max=3.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) print(foo)
  delta <- foo$par[1]
  w     <- foo$par[2]
  rms   <- foo$value
  list(delta=delta,w=w,rms=rms)
}

"fit.blevins" <- function(x,y,u,d,V0,quiet=TRUE,ntry=10) {
  fn <- function(par) {
    cd   <- par[1]
    x0   <- par[2]
    if (cd < 0) return(+Inf)
    #if (x1 < 0) return(+Inf)
    v    <- blevins(x,y,V0,d,cd,x0)
    rms  <- sqrt(mean((v-u)^2))
```

```
    rms
  }
  foo         <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) print(foo)
  cd   <- foo$par[1]
  x0   <- foo$par[2]
  rms  <- foo$value
  list(cd=cd,x0=x0,rms=rms)
}


"fit.schlichting" <- function(x,y,u,d,V0,quiet=TRUE,ntry=10) {
  fn <- function(par) {
    nu   <- par[1]
    l    <- par[2]
    if (nu < 0) return(+Inf)
    #if (x1 < 0) return(+Inf)
    v    <- schlichting(x,y,V0,d,nu,l)
    rms  <- sqrt(mean((v-u)^2))
    rms
  }
  foo         <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) print(foo)
  nu   <- foo$par[1]
  l    <- foo$par[2]
  rms  <- foo$value
  list(nu=nu,l=l,rms=rms)
}

# fitting the models
# truss tower cases

"fit.bladed.a" <- function(x,y,u,case,V0,quiet=FALSE,ntry=10) {
  fn <- function(par) {
    delta <- par[1]
    w     <- par[2]
    v     <- bladed.a(x,y,V0,delta,w,case)
```

```
    rms   <- sqrt(mean((v-u)^2))
    rms
  }
  foo        <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.0,max=1.0),runif(n=1,min=0.5,max=5.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) { cat(paste("BLADED FIT:\n")); print(foo) }
  delta <- foo$par[1]
  w     <- foo$par[2]
  rms   <- foo$value
  list(delta=delta,w=w,rms=rms)
}

"fit.blevins.a" <- function(x,y,u,case,V0,quiet=FALSE,ntry=10) {
  fn <- function(par) {
    cd   <- par[1]
    x0   <- par[2]
    v    <- blevins.a(x,y,V0,cd,x0,case)
    rms  <- sqrt(mean((v-u)^2))
    rms
  }
  foo        <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) { cat(paste("BLEVINS FIT:\n")); print(foo) }
  cd   <- foo$par[1]
  x0   <- foo$par[2]
  rms  <- foo$value
  list(cd=cd,x0=x0,rms=rms)
}


"fit.schlichting.a" <- function(x,y,u,case,V0,quiet=FALSE,ntry=10) {
  fn <- function(par) {
    nu   <- par[1]
    l    <- par[2]
    v    <- schlichting.a(x,y,V0,nu,l,case)
    rms  <- sqrt(mean((v-u)^2))
    rms
  }
  foo        <- NULL
```

```
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars, fn=fn, method="Nelder-Mead",control=list(maxit=500))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  if (!quiet) { cat(paste("SCHLICHTING FIT:\n")); print(foo) }
  nu   <- foo$par[1]
  l    <- foo$par[2]
  rms  <- foo$value
  list(nu=nu, l=l, rms=rms)
}

# plotting mean velocity profiles for black and white figs
largestep <- 6
#Potential
#p1 <- seq(0,400,20) #Extra points for potential
#Powles
p2 <- c(seq(0,150,40),seq(250,400,40)) #Extra points for Powles
mp2 <- seq(178,223,5) #Extra points monopile in the middle monopile
Ptm0deg <- c(seq(137,161,largestep),seq(245,263,largestep)) # Extra points for
    truss main cylinder 0deg
Ptm22p5deg <- c(seq(121,139,largestep),seq(261,279,largestep),seq(161,179,
    largestep),seq(221,239,largestep)) # Extra points for truss main cylinder 22
    p5deg
Ptm45deg <- c(seq(191,209,largestep),seq(115,133,largestep),seq(267,285,largestep)
    ) # Extra points for truss main cylinder 45deg
#Blevins
p3 <- seq(5,400,40) #Extra points for Blevins
mp3 <- seq(181,221,5) #Extra points monopile in the middle
#Btm0deg <- c(seq(137,161,largestep),seq(245,263,largestep)) # Extra points for
    truss main cylinder 0deg

#Schlichting
p4 <- seq(10,400,40) #Extra points for Schlichting
mp4 <- seq(179,219,5) #Extra points in the middle monopile
#Stm0deg <- c(seq(137,161,largestep),seq(245,263,largestep)) # Extra points for
    truss main cylinder 0deg

#CFD
p5 <- seq(3,400,40) #Extra points for CFD
mp5 <- seq(180,220,5) #Extra points in the middle monopile
#Ctm0deg <- c(seq(137,161,largestep),seq(245,263,largestep)) # Extra points for
    truss main cylinder 0deg

# Truss small members need special treatment

smallstep <- 6
smallstep22p5 <-20

#Case1
sm1 <- seq(196,204,smallstep)
```

```
#Case2
sm2 <- c(seq(158,166,smallstep),seq(234,242,smallstep))
#Case3
sm3 <- c(seq(146,154,smallstep22p5),seq(246,254,smallstep22p5),seq(216,224,
    smallstep22p5),seq(176,184,smallstep22p5))
#Case4
sm4 <- c(seq(169,177,smallstep),seq(223,231,smallstep))
#Case5
sm5 <- c(seq(169,177,smallstep),seq(223,231,smallstep),seq(142,150,smallstep),seq
    (250,258,smallstep))
#Case6
sm6 <- c(seq(121,279,smallstep22p5)) #This adds points to the whole section of
    cylinders
#Case7
#Case8
#Case9
#Case10
sm10 <- c(seq(185,215,smallstep))
#Case11
sm11 <- c(seq(147,177,smallstep),seq(223,253,smallstep))
#case12
sm12 <- c(seq(121,279,smallstep)) # Points for whole section

# monopile

exp.y           <- c(0.075,0.05,0.025,0,-0.025,-0.05,-0.075)
exp.y           <- exp.y/0.025
expmononorm     <- c(0.93,1.04,0.85,0.41,0.85,1.04,0.93)

"plot.monopile" <- function(id,delta=0.4,w=1.0,cdb=1.0,nu=1.0,x0=6,l=6,yl=c(2,17),
    V0=V0,verbose=TRUE,best.fit=FALSE,lg=TRUE,turb=FALSE,use.vx=FALSE,Global=FALSE
    ,CFD=TRUE,bladed=TRUE,blevins=TRUE,schlichting=TRUE,potential=TRUE,BW=FALSE,
    norm=FALSE,exp=FALSE) {
  # options
  #   - best.fit  -- find best parameters
  #   - lg        -- plot legend (names of different methods)
  #   - turb      -- use turbulent cases (different data file)
  #   - use.vx    -- use x-velocity (instead of velocity magnitude)
  # distances
  x <- lx.m[id] - sx.m # relative position
  y <- ly
  cat(paste("Monopile at",x,"m distance from center\n"))
  # setting up the coordinate system
  if(norm)plot(y/D,y/D,type="n",ylim=c(0.5,1.2),xlim=c(-5,5),xlab=expression(y/D),
      ylab=expression(U/U[0]),bty="n")
  else plot(y,y,type="n",ylim=yl,xlab="y [m]",ylab="velocity [m/s]",bty="n")
  if (verbose) title(main=paste("Monopile @",x,"m (",fn.A[id],")",sep=""))
  # read data
  fn <- paste(fdir,"monopile-",fn.m[id],".Rdata",sep="")
  if (turb) fn <- paste(fdir,"turb-monopile-",fn.m[id],".Rdata",sep="")
  cat(paste("Looking for file",fn,"\n"))
  if (!file.exists(fn)) stop("Not found -- check your files")
  load(fn)
  y <- fl$y
  u <- fl$mean.v
```

```
  if (use.vx)        u  <- fl$mean.vx
  if (adapt.v == 1) V0 <- u[1]
  if (adapt.v == 2) V0 <- mean(u)
  if (verbose) {
    du <- fl$mean.v - fl$mean.vx
    cat(sprintf("Max. difference between velocity magnitude and x-component = %8.5
        g\n",max(du)))
  }
  if(norm) abline(h=V0/12,lty=4)
  else abline(h=V0,lty=4)
  cat(paste("V0 = ",round(V0,4),"\n"))

  # CFD profile
  if(CFD){
  if(norm)lines(y/D,u/12,col="blue",lwd=1)
  if(exp)points(exp.y,expmononorm,type="o",lty=2,col="red")
  if(exp)legend("bottomright",col=c("blue","red"),pch=c(46,21),lty=c(1,2),legend=c
      ("CFD","Physical experiment"),bg="white")
  else lines(y,u,col="blue",lwd=1)
  if(BW){
  lines(y[p5],u[p5],type="o",pch=4,col="blue",lty=0) #extra points
  lines(y[mp5],u[mp5],type="o",pch=4,col="blue",lty=0) #extra points
  }
  }
  # Potential profile
  if(potential){
  p  <- potential2(x,y,V0=V0,D)$Va
  lines(y,p,col="darkred",lty=5)
  legend("bottomleft",col=c("blue","darkred"),lty=c(1,5),legend=c("CFD","Potential
      "),bg="white",cex=0.8)
  }
  if(bladed){
  # Bladed  profile
  if (best.fit) {
    foo <- fit.bladed(x=x,y=y,u=u,d=D,V0=V0) # CAVEAT: no reference values here
    delta <- foo$delta
    w     <- foo$w
    legend1 <- substitute(Delta == v,list(v=sprintf("%5.3f",delta)))
    legend2 <- substitute(w == v,list(v=sprintf("%5.3f",w)))
    # legend3 <- substitute(x[r] == v,list(v=sprintf("%5.1f",ref)))
    # legend("bottomleft",col=c("black","red","red"),lty=2,legend=do.call("
        expression",c(legend3,legend1,legend2)),bg="white",cex=0.8)
    if(BW)legend("bottomleft",col=c("red","red"),pch=22,lty=2,legend=do.call("
        expression",
c(legend1,legend2)),bg="white",cex=0.8)
    else legend("bottomleft",col=c("red","red"),lty=2,legend=do.call("expression",
c(legend1,legend2)),bg="white",cex=0.8)
  }
  foo  <- powles(x=x,y=y,d=D,delta=delta,w=w,V0=V0,bladed=TRUE)
  if(norm)lines(y/D,foo/12,col="red",lwd=1,lty=2)
  else lines(y,foo,col="red",lwd=1,lty=2)
  if(BW){
  lines(y[p2],foo[p2],type="o",pch=22,col="red",lty=0) #extra points
  lines(y[mp2],foo[mp2],type="o",pch=22,col="red",lty=0) #extra points
  }
```

```
  }
  # The following profiles are not defined in front of the monopile
  if (x > D) {
    if(blevins){
    # Blevins profile
    if (best.fit) {
      bar <- fit.blevins(x=x,y=y,u=u,d=D,V0=V0)
      cdb <- bar$cd
      x0  <- bar$x0
      legend1 <- substitute(C[d] == v,list(v=sprintf("%5.3f",cdb)))
      legend2 <- substitute(x[0] == v,list(v=sprintf("%5.3f",x0)))
     if(BW) legend("bottomright",col=c("darkgreen","darkgreen"),pch=24,lty=3,
         legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
     else legend("bottomright",col=c("darkgreen","darkgreen"),lty=3,legend=do.call
         ("expression",c(legend1,legend2)),bg="white",cex=0.8)
    }
    bar  <- blevins(x=x,y=y,d=D,cd=cdb,x0=x0,V0=V0)
    lines(y,bar,col="darkgreen",lwd=1,lty=3)
    if(BW){
    lines(y[p3],bar[p3],type="o",pch=24,col="darkgreen",lty=0) #extra points
    lines(y[mp3],bar[mp3],type="o",pch=24,col="darkgreen",lty=0) #extra points
    }
    }
    if(schlichting){
    # Schlichting profile
    if (best.fit) {
      bar <- fit.schlichting(x=x,y=y,u=u,d=D,V0=V0)
      nu <- bar$nu
      l  <- bar$l
      legend1 <- substitute(nu == v,list(v=sprintf("%5.3f",nu)))
      legend2 <- substitute(l == v,list(v=sprintf("%5.3f",l)))
    }

    bar  <- schlichting(x=x,y=y,d=D,nu=nu,l=l,V0=V0)
    if(CFD==FALSE) lty1 <- 1
    else lty1 <- 4
    lines(y,bar,col="purple",lwd=1,lty=lty1)
    if(BW){
    lines(y[p4],bar[p4],type="o",pch=21,col="purple",lty=0) #extra points
    lines(y[mp4],bar[mp4],type="o",pch=21,col="purple",lty=0) #extra points
    }
    if (best.fit & BW) legend("bottomleft",col=c("purple","purple"),pch=21,lty=
        lty1,legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
    if (best.fit & BW==FALSE) legend("right",col=c("purple","purple"),lty=lty1,
        legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
    }
  }
  # Legend
  if (lg & BW==FALSE & Global==FALSE) legend("left",col=c("blue","red","darkgreen"
      ,"purple","black"),
lty=c(1,2,3,4,5),legend=c("CFD","Bladed","Blevins","Schlichting","Potential"),cex
    =0.8,bg="white")
  if (lg & BW & !CFD) legend("right",pch=c(24,21),col=c("darkgreen","purple"),lty=
      c(3,1),legend=c("Blevins","Schlichting"),cex=0.5,bg="white")
```

```
  if (lg & BW & CFD) legend("left",col=c("blue","black","red"),pch=c(4,46,22),lty=
      c(1,5,2),legend=c("CFD","Potential","Bladed"),cex=0.5,bg="white")
  if (Global) legend("top",col=c("blue","red"),lty=c(1,2),legend=c("CFD","
      Individual-fit"),cex=0.8,bg="white")
  res <- NULL
  # return optimal parameters if fitting
  if (best.fit) res <- list(delta=delta,w=w,cdb=cdb,nu=nu,x0=x0,l=l)
  res
}

# plotting mean velocity profiles
# truss tower cases

"plot.truss.a" <- function(id,case,delta=0.4,w=1.0,cdb=1.0,nu=1.0,x0=6,l=6,yl=c
    (2,17),V0=V0,verbose=TRUE,best.fit=FALSE,use.vx=FALSE,lg=TRUE,turb=FALSE,
    Global=FALSE,CFD=TRUE,schlichting=TRUE
,blevins=TRUE,bladed=TRUE,potential=TRUE,BW=FALSE,norm=FALSE) {
  # options
  #   - best.fit  -- find best parameters
  #   - lg        -- plot legend (names of different methods)
  #   - turb      -- use turbulent cases (different data file)
  #   - adapat.v  -- use actual mean velocity (not free-stream V0)
  #   -           -- values: 0 (use V0), 1 (use V at border), 2 (use profile mean
      V)
  #   - use.vx    -- use x-velocity (instead of velocity magnitude)
  # distances
  if (case %in% cases.A) x <- lx.A[id] - sx.A  # relative position
  if (case %in% cases.B) x <- lx.B[id] - sx.B
  if (case %in% cases.C) x <- lx.C[id] - sx.C
  y <- ly
  cat(paste("Truss case",case,"at ",x,"m distance from center\n"))
  # setting up the coordinate system
  if(norm)plot(y/D,y/D,type="n",ylim=c(0.5,1.2),xlim=c(-5,5),xlab=expression(y/D),
      ylab=expression(U/U[0]),bty="n")
  else plot(y,y,type="n",ylim=yl,xlab="y [m]",ylab="velocity [m/s]",bty="n")
  if (verbose & case %in% cases.A) title(main=paste("Case ",case," @ ",x," m (",fn
      .A[id],")",sep=""))
  if (verbose & case %in% cases.B) title(main=paste("Case ",case," @ ",x," m (",fn
      .B[id],")",sep=""))
  if (verbose & case %in% cases.C) title(main=paste("Case ",case," @ ",x," m (",fn
      .C[id],")",sep=""))
  # read data
  if(!turb){
  if (case %in% cases.A) fn <- paste(fdir,"truss",case,"-",fn.A[id],".Rdata",sep="
      ")
  if (case %in% cases.B) fn <- paste(fdir,"truss",case,"-",fn.B[id],".Rdata",sep="
      ")
  if (case %in% cases.C) fn <- paste(fdir,"truss",case,"-",fn.C[id],".Rdata",sep="
      ")
  }
  if (turb) {
    if (case %in% cases.A) fn <- paste(fdir,"turb-truss",case,"-",fn.A[id],".Rdata
        ",sep="")
    if (case %in% cases.B) fn <- paste(fdir,"turb-truss",case,"-",fn.B[id],".Rdata
        ",sep="")
```

```
  if (case %in% cases.C) fn <- paste(fdir,"turb-truss",case,"-",fn.C[id],".Rdata
      ",sep="")
}
cat(paste("Looking for file ",fn,"\n"))
if (!file.exists(fn)) stop("Not found -- check your files")
load(fn)
y  <- fl$y
u  <- fl$mean.v
if (use.vx)       u  <- fl$mean.vx
if (adapt.v == 1) V0 <- u[1]
if (adapt.v == 2) V0 <- mean(u)
if (verbose) {
  du <- fl$mean.v - fl$mean.vx
  cat(sprintf("Max. difference between velocity magnitude and x-component = %8.5
      g\n",max(du)))
}
if(norm) abline(h=V0/12,lty=4)
else abline(h=V0,lty=4)
cat(paste("V0 = ",round(V0,4),"\n"))


if(CFD){
# CFD profile
if(norm)lines(y/D,u/12,col="blue",lwd=1)
#if(norm)legend("bottomright",col=c("blue","red"),lty=c(1,2),legend=c("CFD","
    Physical experiment"),bg="white")
else lines(y,u,col="blue",lwd=1)
if(BW){
lines(y[p5],u[p5],type="o",pch=4,col="blue",lty=0) #extra points
      if(case==1){ lines(y[sm1],u[sm1],type="o",pch=4,col="blue",lty=0) #extra
          points
                   lines(y[Ptm0deg],u[Ptm0deg],type="o",pch=4,col="blue",lty
                       =0) #extra points
      }
      if(case==2){ lines(y[sm2],u[sm2],type="o",pch=4,col="blue",lty=0) #extra
          points
                   lines(y[Ptm45deg],u[Ptm45deg],type="o",pch=4,col="blue",
                       lty=0) #extra points
      }
      if(case==3){ lines(y[sm3],u[sm3],type="o",pch=4,col="blue",lty=0) #extra
          points
                   lines(y[Ptm22p5deg],u[Ptm22p5deg],type="o",pch=4,col="blue
                       ",lty=0) #extra points
      }
      if(case==4){ lines(y[sm4],u[sm4],type="o",pch=4,col="blue",lty=0) #extra
          points
                   lines(y[Ptm0deg],u[Ptm0deg],type="o",pch=4,col="blue",lty
                       =0) #extra points
      }
      if(case==5){ lines(y[sm5],u[sm5],type="o",pch=4,col="blue",lty=0) #extra
          points
                   lines(y[Ptm45deg],u[Ptm45deg],type="o",pch=4,col="blue",
                       lty=0) #extra points
      }
```

```
        if(case==6){ lines(y[sm6],u[sm6],type="o",pch=4,col="blue",lty=0) #extra
            points
                     lines(y[Ptm22p5deg],u[Ptm22p5deg],type="o",pch=4,col="blue
                         ",lty=0) #extra points
        }
        if(case==7){ #lines(y[sm1],u[sm1],type="o",pch=4,col="blue",lty=0) #extra
            points
                     lines(y[Ptm0deg],u[Ptm0deg],type="o",pch=4,col="blue",lty
                         =0) #extra points
        }
        if(case==8){ #lines(y[sm2],u[sm2],type="o",pch=4,col="blue",lty=0) #extra
            points
                     lines(y[Ptm45deg],u[Ptm45deg],type="o",pch=4,col="blue",
                         lty=0) #extra points
        }
        if(case==9){ #lines(y[sm3],u[sm3],type="o",pch=4,col="blue",lty=0) #extra
            points
                     lines(y[Ptm22p5deg],u[Ptm22p5deg],type="o",pch=4,col="blue
                         ",lty=0) #extra points
        }
        if(case==10){ lines(y[sm10],u[sm10],type="o",pch=4,col="blue",lty=0) #
            extra points
                     lines(y[Ptm0deg],u[Ptm0deg],type="o",pch=4,col="blue",lty
                         =0) #extra points
        }
        if(case==11){ lines(y[sm11],u[sm11],type="o",pch=4,col="blue",lty=0) #
            extra points
                     lines(y[Ptm45deg],u[Ptm45deg],type="o",pch=4,col="blue",
                         lty=0) #extra points
        }
        if(case==12){ lines(y[sm12],u[sm12],type="o",pch=4,col="blue",lty=0) #
            extra points
                     lines(y[Ptm22p5deg],u[Ptm22p5deg],type="o",pch=4,col="blue
                         ",lty=0) #extra points
        }
}
lcol <- "blue"
ln   <- "CFD"
lt   <- 1
points <- 4
}


# Potential profile
if(potential){
p  <- potential2(x,y,V0=V0,D)$Va
lines(y,p,col="darkred",lty=5)
if(CFD){
       lcol <- c(lcol,"darkred")
       ln   <- c(ln,"Potential")
       lt   <- c(lt,5)
       points <- c(points,0)
}
else {
```

```
      lcol <- "black"
      ln   <- "Potential"
      lt   <- 5
      points <- 0
}
}
if (bladed) {
  # Bladed profile
  # Global fitting

 if(Global){
    legend1 <- substitute(Delta[r] == v, list(v=sprintf("%5.3f",delta)))
    legend2 <- substitute(w[r] == v, list(v=sprintf("%5.3f",w)))
    legend3 <- substitute(x[r] == v, list(v=sprintf("%5.1f",ref)))
    legend("bottomright",col=c("darkgreen","darkgreen","darkgreen"),lty=3,legend
        =do.call("expression",c    (legend3,legend1,legend2)),bg="white",cex=0.8)


      foo  <- bladed.a(x=x,y=y,delta=delta,w=w,case=case,V0=V0)
      if(norm)lines(y/D,foo/12,col="darkgreen",lwd=1,lty=3)
      else lines(y,foo,col="darkgreen",lwd=1,lty=3)
 }
  # Individual fitting
  if (best.fit) {
    foo <- fit.bladed.a(x=x,y=y,u=u,case,V0=V0) # CAVEAT: global reference value
        used
    delta <- foo$delta
    w     <- foo$w
    legend1 <- substitute(Delta[r] == v, list(v=sprintf("%5.3f",delta)))
    legend2 <- substitute(w[r] == v, list(v=sprintf("%5.3f",w)))
    legend3 <- substitute(x[r] == v, list(v=sprintf("%5.1f",ref)))
    if(BW) legend("bottomleft",col=c("black","red","red"),pch=c(46,22,22),lty=2,
        legend=do.call("expression",c(legend3,legend1,legend2)),bg="white",cex
        =0.8)
    if(Global) legend("bottomleft",col=c("red","red","red"),lty=2,legend=do.call
        ("expression",c(legend3,legend1,legend2)),bg="white",cex=0.8)
    else legend("bottomleft",col=c("black","red","red"),lty=2,legend=do.call("
        expression",c(legend3,legend1,legend2)),bg="white",cex=0.8)

      foo  <- bladed.a(x=x,y=y,delta=delta,w=w,case=case,V0=V0)
      if(norm)lines(y/D,foo/12,col="red",lwd=1,lty=2)
      else lines(y,foo,col="red",lwd=1,lty=2)
  }

      if(Global==FALSE & best.fit==FALSE){
      foo  <- bladed.a(x=x,y=y,delta=delta,w=w,case=case,V0=V0)
      if(norm)lines(y/D,foo/12,col="red",lwd=1,lty=2)
      else lines(y,foo,col="red",lwd=1,lty=2)
      }

  if(BW){
  lines(y[p2],foo[p2],type="o",pch=22,col="red",lty=0) #extra points
      if(case==1){ lines(y[sm1],foo[sm1],type="o",pch=22,col="red",lty=0) #extra
          points
```

```
                lines(y[Ptm0deg],foo[Ptm0deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==2){ lines(y[sm2],foo[sm2],type="o",pch=22,col="red",lty=0) #extra
    points
                lines(y[Ptm45deg],foo[Ptm45deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==3){ lines(y[sm3],foo[sm3],type="o",pch=22,col="red",lty=0) #extra
    points
                lines(y[Ptm22p5deg],foo[Ptm22p5deg],type="o",pch=22,col="
                    red",lty=0) #extra points
}
if(case==4){ lines(y[sm4],foo[sm4],type="o",pch=22,col="red",lty=0) #extra
    points
                lines(y[Ptm0deg],foo[Ptm0deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==5){ lines(y[sm5],foo[sm5],type="o",pch=22,col="red",lty=0) #extra
    points
                lines(y[Ptm45deg],foo[Ptm45deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==6){ lines(y[sm6],foo[sm6],type="o",pch=22,col="red",lty=0) #extra
    points
                lines(y[Ptm22p5deg],foo[Ptm22p5deg],type="o",pch=22,col="
                    red",lty=0) #extra points
}
if(case==7){ #lines(y[sm1],foo[sm1],type="o",pch=22,col="red",lty=0) #
    extra points
                lines(y[Ptm0deg],foo[Ptm0deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==8){ #lines(y[sm2],foo[sm2],type="o",pch=22,col="red",lty=0) #
    extra points
                lines(y[Ptm45deg],foo[Ptm45deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==9){ #lines(y[sm3],foo[sm3],type="o",pch=22,col="red",lty=0) #
    extra points
                lines(y[Ptm22p5deg],foo[Ptm22p5deg],type="o",pch=22,col="
                    red",lty=0) #extra points
}
if(case==10){ lines(y[sm10],foo[sm10],type="o",pch=22,col="red",lty=0) #
    extra points
                lines(y[Ptm0deg],foo[Ptm0deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==11){ lines(y[sm11],foo[sm11],type="o",pch=22,col="red",lty=0) #
    extra points
                lines(y[Ptm45deg],foo[Ptm45deg],type="o",pch=22,col="red",
                    lty=0) #extra points
}
if(case==12){ lines(y[sm12],foo[sm12],type="o",pch=22,col="red",lty=0) #
    extra points
```

```
                    lines(y[Ptm22p5deg],foo[Ptm22p5deg],type="o",pch=22,col="
                        red",lty=0) #extra points
        }
    }
    if(CFD==TRUE || potential==TRUE){
    lcol <- c(lcol,"red")
    ln   <- c(ln,"Bladed")
    lt   <- c(lt,2)
    points <- c(points,22)
    }
    else{
        lcol <- "red"
        ln <- "Bladed"
        lt <- 2
        points <- 22
    }
}
# The following profiles are not defined in front of the monopile
if (x > D) {
    if (blevins) {

     if(Global){
        legend1 <- substitute(C[d] == v,list(v=sprintf("%5.3f",cdb)))
        legend2 <- substitute(x[0] == v,list(v=sprintf("%5.3f",x0)))
        if(BW) legend("bottomright",col=c("darkgreen","darkgreen"),pch=24,lty=3,
            legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
        else legend("bottomright",col=c("darkgreen","darkgreen"),lty=3,legend=do.
            call("expression",c(legend1,legend2)),bg="white",cex=0.8)

      bar  <- blevins.a(x=x,y=y,cd=cdb,x0=x0,case=case,V0=V0)
      lines(y,bar,col="darkgreen",lwd=1,lty=3)
      }


      # Blevins profile
      if (best.fit) {
        bar <- fit.blevins.a(x=x,y=y,u=u,case,V0=V0)
        cdb <- bar$cd
        x0  <- bar$x0
        legend1 <- substitute(C[d] == v,list(v=sprintf("%5.3f",cdb)))
        legend2 <- substitute(x[0] == v,list(v=sprintf("%5.3f",x0)))
        if(BW) legend("bottomleft",col=c("darkgreen","darkgreen"),pch=24,lty=3,
            legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
        else legend("bottomright",col=c("darkgreen","darkgreen"),lty=3,legend=do.
            call("expression",c(legend1,legend2)),bg="white",cex=0.8)

        #if(BW) legend("bottomleft",col=c("red","red"),pch=24,lty=2,legend=do.call
            ("expression",c(legend1,legend2)),bg="white",cex=0.8)
        #else legend("bottomleft",col=c("red","red"),lty=2,legend=do.call("
            expression",c(legend1,legend2)),bg="white",cex=0.8)

      bar  <- blevins.a(x=x,y=y,cd=cdb,x0=x0,case=case,V0=V0)
      lines(y,bar,col="darkgreen",lwd=1,lty=3)
      #lines(y,bar,col="red",lwd=1,lty=2)
      }
```

```
if(Global==FALSE & best.fit==FALSE){
bar  <- blevins.a(x=x,y=y,cd=cdb,x0=x0,case=case,V0=V0)
lines(y,bar,col="darkgreen",lwd=1,lty=3)
}


if(BW){
lines(y[p3],bar[p3],type="o",pch=24,col="darkgreen",lty=0) #extra points
  if(case==1){ lines(y[sm1],u[sm1],type="o",pch=24,col="darkgreen",lty=0) #
      extra points
                  lines(y[Ptm0deg],u[Ptm0deg],type="o",pch=24,col="darkgreen
                      ",lty=0) #extra points
  }
  if(case==2){ lines(y[sm2],bar[sm2],type="o",pch=24,col="darkgreen",lty=0)
      #extra points
                  lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==3){ lines(y[sm3],bar[sm3],type="o",pch=24,col="darkgreen",lty=0)
      #extra points
                  lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==4){ lines(y[sm4],bar[sm4],type="o",pch=24,col="darkgreen",lty=0)
      #extra points
                  lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==5){ lines(y[sm5],bar[sm5],type="o",pch=24,col="darkgreen",lty=0)
      #extra points
                  lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==6){ lines(y[sm6],bar[sm6],type="o",pch=24,col="darkgreen",lty=0)
      #extra points
                  lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==7){ #lines(y[sm1],bar[sm1],type="o",pch=24,col="darkgreen",lty=0)
       #extra points
                  lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==8){ #lines(y[sm2],bar[sm2],type="o",pch=24,col="darkgreen",lty=0)
       #extra points
                  lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
  if(case==9){ #lines(y[sm3],bar[sm3],type="o",pch=24,col="darkgreen",lty=0)
       #extra points
                  lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
  }
```

```
    if(case==10){ lines(y[sm10],bar[sm10],type="o",pch=24,col="darkgreen",lty
        =0) #extra points
                  lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
    }
    if(case==11){ lines(y[sm11],bar[sm11],type="o",pch=24,col="darkgreen",lty
        =0) #extra points
                  lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
    }
    if(case==12){ lines(y[sm12],bar[sm12],type="o",pch=24,col="darkgreen",lty
        =0) #extra points
                  lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=24,col="
                      darkgreen",lty=0) #extra points
    }
  }
  if(CFD==TRUE || potential==TRUE || bladed==TRUE){
  lcol <- c(lcol,"darkgreen")
  ln   <- c(ln,"Blevins")
  lt   <- c(lt,3)
  points <- c(points,24)
  }
  else{
    lcol <- "darkgreen"
    ln   <- "Blevins"
     lt   <- 3
     points <- 24
  }
}
if (schlichting) {
 if(Global){

    legend1 <- substitute(nu == v,list(v=sprintf("%5.3f",nu)))
    legend2 <- substitute(l == v,list(v=sprintf("%5.3f",l)))
          if (best.fit & BW) legend("bottomright",col=c("darkgreen","darkgreen
              "),pch=21,lty=3,legend=do.call("expression",c(legend1,legend2)),
              bg="white",cex=0.8)
  if (best.fit & BW==FALSE) legend("bottomright",col=c("darkgreen","darkgreen"
      ),lty=3,legend=do.call("expression",c(legend1,legend2)),bg="white",cex
      =0.8)

    bar  <- schlichting.a(x=x,y=y,nu=nu,l=l,case,V0=V0)
    lines(y,bar,col="darkgreen",lwd=1,lty=3)
 }

  # Schlichting profile
  if (best.fit) {
    bar <- fit.schlichting.a(x=x,y=y,u=u,case,V0=V0)
    nu <- bar$nu
    l   <- bar$l
    legend1 <- substitute(nu == v,list(v=sprintf("%5.3f",nu)))
    legend2 <- substitute(l == v,list(v=sprintf("%5.3f",l)))

    if (best.fit & BW) legend("right",col=c("purple","purple"),pch=21,lty=1,
        legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)
```

```
  if (best.fit & BW==FALSE) legend("right",col=c("purple","purple"),lty=4,
      legend=do.call("expression",c(legend1,legend2)),bg="white",cex=0.8)


  #legend("bottomleft",col=c("red","red"),lty=2,legend=do.call("expression",
      c(legend1,legend2)),bg="white",cex=0.8)

  bar <- schlichting.a(x=x,y=y,nu=nu,l=l,case,V0=V0)
  lines(y,bar,col="purple",lwd=1,lty=4)
  #lines(y,bar,col="red",lwd=1,lty=2)
}
if(CFD==FALSE) lty1 <- 1
else lty1 <- 4
if(Global==FALSE & best.fit==FALSE){
bar <- schlichting.a(x=x,y=y,nu=nu,l=l,case,V0=V0)
lines(y,bar,col="purple",lwd=1,lty=4)
}

if(BW){
lines(y[p4],bar[p4],type="o",pch=21,col="purple",lty=0) #extra points
  if(case==1){ lines(y[sm1],bar[sm1],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=21,col="purple"
                    ,lty=0) #extra points
  }
  if(case==2){ lines(y[sm2],bar[sm2],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=21,col="
                    purple",lty=0) #extra points
  }
  if(case==3){ lines(y[sm3],bar[sm3],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=21,col="
                    purple",lty=0) #extra points
  }
  if(case==4){ lines(y[sm4],bar[sm4],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=21,col="purple"
                    ,lty=0) #extra points
  }
  if(case==5){ lines(y[sm5],bar[sm5],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=21,col="
                    purple",lty=0) #extra points
  }
  if(case==6){ lines(y[sm6],bar[sm6],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=21,col="
                    purple",lty=0) #extra points
  }
  if(case==7){ #lines(y[sm1],bar[sm1],type="o",pch=21,col="purple",lty=0) #
      extra points
                lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=21,col="purple"
                    ,lty=0) #extra points
  }
```

```
        if(case==8){ #lines(y[sm2],bar[sm2],type="o",pch=21,col="purple",lty=0) #
            extra points
                    lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=21,col="
                        purple",lty=0) #extra points
        }
        if(case==9){ #lines(y[sm3],bar[sm3],type="o",pch=21,col="purple",lty=0) #
            extra points
                    lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=21,col="
                        purple",lty=0) #extra points
        }
        if(case==10){ lines(y[sm10],bar[sm10],type="o",pch=21,col="purple",lty=0)
            #extra points
                    lines(y[Ptm0deg],bar[Ptm0deg],type="o",pch=21,col="purple"
                        ,lty=0) #extra points
        }
        if(case==11){ lines(y[sm11],bar[sm11],type="o",pch=21,col="purple",lty=0)
            #extra points
                    lines(y[Ptm45deg],bar[Ptm45deg],type="o",pch=21,col="
                        purple",lty=0) #extra points
        }
        if(case==12){ lines(y[sm12],bar[sm12],type="o",pch=21,col="purple",lty=0)
            #extra points
                    lines(y[Ptm22p5deg],bar[Ptm22p5deg],type="o",pch=21,col="
                        purple",lty=0) #extra points
        }
      }
      lcol <- c(lcol,"purple")
      ln   <- c(ln,"Schlichting")
      lt   <- c(lt,4)
      points <- c(points,21)
     }
  }
  # Legend
  if (lg & BW==FALSE & Global == FALSE) legend("left",col=lcol,lty=lt,legend=ln,
      cex=0.6,bg="white")
  if (lg & BW & CFD==FALSE) legend("right",pch=points,col=lcol,lty=lt,legend=ln,
      cex=0.6,bg="white")
  if (lg & BW & CFD) legend("left",col=lcol,pch=c(4,46,22),lty=c(1,5,2),legend=c("
      CFD","Potential","Bladed"),cex=0.8,bg="white")
 if (Global & best.fit & lg) legend("top",col=c("blue","red","darkgreen"),lty=c
     (1,2,3),legend=c("CFD","Individual-fit","Global-fit"),cex=0.8,bg="white")
 #if (Global & best.fit & lg & blevins) legend("top",col=c("blue","darkgreen","
     darkred"),lty=c(1,3,4),legend=c("CFD","Individual-fit","Global-fit"),cex=0.8,
     bg="white")
 #if (Global & best.fit & lg & schlichting) legend("top",col=c("blue","purple","
     darkorange"),lty=c(1,4,5),legend=c("CFD","Individual-fit","Global-fit"),cex
     =0.8,bg="white")
  res <- NULL
  # return optimal parameters if fitting
  if (best.fit) res <- list(delta=delta,w=w,cdb=cdb,nu=nu,x0=x0,l=l)
  res
}

#plot turbulent intensity
#monopile
```

```
"plot.monopile.ti" <- function(id,delta=0.4,w=1.0,cdb=1.0,nu=1.0,x0=6,l=6,yl=c
    (7,20),xl=c(-20,20),labx="y [m]",laby="Turbulence Intesity [%]",V0=V0,verbose=
    TRUE,best.fit=FALSE,lg=TRUE,turb=FALSE,ti.tot=TRUE,ti.fluent=FALSE,ti.fluc=
    FALSE,MS=FALSE){
  x <- lx.m[id] - sx.m  # relative position
  y <- yl
  cat(paste("Monopile at",x,"m distance from center\n"))
  # setting up the coordinate system
  plot(y,y,type="n",ylim=yl,xlim=xl,xlab=labx,ylab=laby,bty="n")
  if (verbose) title(main=paste("Monopile @",x,"m (",fn.A[id],")",sep=""))
  # read data
  fn <- paste(fdir,"ts-monopile-",fn.m[id],".Rdata",sep="")
  if (turb) fn <- paste(fdir,"ts-turb-monopile-",fn.m[id],".Rdata",sep="")
  cat(paste("Looking for file ",fn,"\n"))
  if (!file.exists(fn)) stop("Not found -- check your files")
  load(fn)
  if(MS) y <- fl$y/D
  else y <- fl$y
  ti1 <- apply(fl$ti,2,mean)
  ti2 <- apply(fl$v,2,var)
  titot <- sqrt(12*ti1+ti2)/12*100
  if(ti.tot)lines (y,sqrt(12*ti1+ti2)/12*100,ylim=yl,col="blue")
  if(ti.fluent) lines(y,sqrt(12*ti1)/12*100,ylim=yl,col="red",lty=2)
  if(ti.fluc) lines(y,sqrt(ti2)/12*100,ylim=yl,col="darkgreen",lty=1)
  if(ti.tot)legend("topright",col="blue",legend=c("Total Turb int"),lty=1,cex=0.6,
      bg="white")
  if(ti.fluc & ti.fluent) legend("topright",col=c("red","darkgreen"),legend=c("TI-
      kinetic energy","TI-fluctuations"),lty=c(2,1),cex=0.6,bg="white")
}
#truss tower
"plot.truss.ti" <- function(id,case,delta=0.4,w=1.0,cdb=1.0,nu=1.0,x0=6,l=6,yl=c
    (7,20),xl=c(-20,20),labx="y [m]",laby="Turbulence Intesity [%]",V0=V0,verbose=
    TRUE,turb=FALSE,ti.tot=TRUE,ti.fluent=FALSE,ti.fluc=FALSE,MS=FALSE){
if (case %in% cases.A) x <- lx.A[id] - sx.A  # relative position
  if (case %in% cases.B) x <- lx.B[id] - sx.B
  if (case %in% cases.C) x <- lx.C[id] - sx.C
  y <- ly
  cat(paste("Truss case",case,"at",x,"m distance from center\n"))
  # setting up the coordinate system
  plot(y,y,type="n",ylim=yl,xlim=xl,xlab=labx,ylab=laby,bty="n")
  if (verbose & case %in% cases.A) title(main=paste("Case",case,"@",x,"m (",fn
      .A[id],")",sep=""))
  if (verbose & case %in% cases.B) title(main=paste("Case",case,"@",x,"m (",fn
      .B[id],")",sep=""))
  if (verbose & case %in% cases.C) title(main=paste("Case",case,"@",x,"m (",fn
      .C[id],")",sep=""))
  # read data
  if (case %in% cases.A) fn <- paste(fdir,"ts-truss",case,"-",fn.A[id],".Rdata",
      sep="")
  if (case %in% cases.B) fn <- paste(fdir,"ts-truss",case,"-",fn.B[id],".Rdata",
      sep="")
  if (case %in% cases.C) fn <- paste(fdir,"ts-truss",case,"-",fn.C[id],".Rdata",
      sep="")
  if (turb) {
```

```
    if (case %in% cases.A) fn2 <- paste(fdir,"ts-turb-truss",case,"-",fn.A[id],".
        Rdata",sep="")
    if (case %in% cases.B) fn2 <- paste(fdir,"ts-turb-truss",case,"-",fn.B[id],".
        Rdata",sep="")
    if (case %in% cases.C) fn2 <- paste(fdir,"ts-turb-truss",case,"-",fn.C[id],".
        Rdata",sep="")
  }
  cat(paste("Looking for file ",fn,"\n"))
  if (!file.exists(fn)) stop("Not found -- check your files")
  load(fn)
  if(MS) y <- fl$y/D
  else y  <- fl$y
  ti1  <- apply(fl$ti,2,mean)
  ti2  <- apply(fl$v,2,var)
  if(ti.tot)lines (y,sqrt(12*ti1+ti2)/12*100,ylim=yl,col="blue")
  if(ti.fluent) lines(y,sqrt(12*ti1)/12*100,ylim=yl,col="red",lty=2)
  if(ti.fluc) lines(y,sqrt(ti2)/12*100,ylim=yl,col="darkgreen",lty=1)
  if(ti.tot)legend("topright",col="blue",legend=c("Total Turb int"),lty=1,cex=0.6,
      bg="white")
  if(ti.fluc & ti.fluent) legend("topright",col=c("red","darkgreen"),legend=c("TI-
      kinetic energy","TI-fluctuations"),lty=c(2,1),cex=0.6,bg="white")
}

# fit at all rake lines
# monopile

"fit.monopile" <- function(V0=V0) {
  n <- length(ds.m)
  deltas <- NULL
  ws     <- NULL
  rmss   <- NULL
  ds     <- NULL
  xs     <- NULL
  cdbs   <- NULL
  x0s    <- NULL
  rms2s  <- NULL
  nus    <- NULL
  ls     <- NULL
  rms3s  <- NULL
  for (i in seq(n)) {

    if (lx.m[i] <= 0) next  # selection: fit only behind center of structure

    x <- lx.m[i] - sx.m  # relative position"
    cat(paste("Fit at ",sprintf("%3d",x),"m  => ",sep=""))
    fn <- paste(fdir,"monopile-",fn.m[i],".Rdata",sep="")
    cat(paste("Looking for file ",fn,"\n"))
    if (!file.exists(fn)) {
      warning("Not found -- check your files");
      cat("Not found -- check your files\n")
      next
    }
    load(fn)
    u <- fl$mean.vx
    if (adapt.v == 1) V0 <- u[1]
```

```
    if (adapt.v == 2) V0 <- mean(u)
    cat(paste("V0 = ",round(V0,3),"\n"))
    y <- fl$y

    foo <- fit.bladed(x=x,y=y,u=u,d=D,V0=V0,quiet=TRUE)
    delta <- foo$delta
    w     <- foo$w
    rms   <- foo$rms
    deltas <- c(deltas,delta)
    ws     <- c(ws,w)
    rmss   <- c(rmss,rms)
    cat(sprintf(" delta = %5.3f  W = %5.3f  rms = %5.3f\n",delta,w,rms))

    bar  <- fit.blevins(x=x,y=y,u=u,d=D,V0=V0)
    cdb  <- bar$cd
    x0   <- bar$x0
    rms2 <- bar$rms
    cdbs <- c(cdbs,cdb)
    x0s  <- c(x0s,x0)
    rms2s<- c(rms2s,rms2)
    cat(sprintf(" cd = %5.3f  x0 = %5.3f  rms = %5.3f\n",cdb,x0,rms2))

    baz  <- fit.schlichting(x=x,y=y,u=u,d=D,V0=V0)
    nu   <- baz$nu
    l    <- baz$l
    rms3 <- baz$rms
    nus  <- c(nus,nu)
    ls   <- c(ls,l)
    rms3s<- c(rms3s,rms3)
    cat(sprintf(" nu = %5.3f  l = %5.3f  rms = %5.3f\n",nu,l,rms3))

    ds <- c(ds, ds.m[i])
    xs <- c(xs, x)
  }
  save(file=paste(fdir,"fit_monopile.Rdata", sep=""),ascii=TRUE,deltas,ws,rmss,ds,
       xs,cdbs,x0s,rms2s,nus,ls,rms3s)
  list(delta=deltas,w=ws,rms=rmss,ds=ds,xs=xs,cdb=cdbs,x0=x0s,rmsb=rms2s,nu=nus,l=
       ls,rmss=rms3s)
}

# fit at all rake lines
# truss tower cases

"fit.truss" <- function(case,V0=V0) {
  if (case %in% cases.A) ds1 <- ds.A
  if (case %in% cases.B) ds1 <- ds.B
  if (case %in% cases.C) ds1 <- ds.C
  n <- length(ds1)
  deltas <- NULL
  ws     <- NULL
  rmss   <- NULL
  ds     <- NULL
  xs     <- NULL
  cdbs   <- NULL
  x0s    <- NULL
```

```
rms2s <- NULL
nus   <- NULL
ls    <- NULL
rms3s <- NULL
for (i in seq(n)) {

  if (case %in% cases.A) x <- lx.A[i] - sx.A  # relative position
  if (case %in% cases.B) x <- lx.B[i] - sx.B
  if (case %in% cases.C) x <- lx.C[i] - sx.C

  if (x <= 7.7) next  # selection: at least 7.7m behind center

  cat(paste("Fit at ",sprintf("%4.3f",x),"m  => ",sep=""))
  fn <- paste(fdir,"truss",case,"-",fn.A[i],".Rdata",sep="")
  cat(paste("Looking for file ",fn,"\n"))

  if (!file.exists(fn)) {
    warning("Not found -- check your files");
    cat("Not found -- check your files\n")
    next
  }
  load(fn)
  u <- fl$mean.vx
  if (adapt.v == 1) V0 <- u[1]
  if (adapt.v == 2) V0 <- mean(u)
  cat(paste("V0 = ",round(V0,3),"\n"))
  y <- fl$y

  foo <- fit.bladed(x=x,y=y,u=u,d=D,ref=ref, quiet=TRUE,V0=V0)
  delta <- foo$delta
  w     <- foo$w
  rms   <- foo$rms
  deltas <- c(deltas,delta)
  ws     <- c(ws,w)
  rmss   <- c(rmss,rms)
  cat(sprintf(" delta= %5.3f  W= %5.3f  rms= %5.3f\n",delta,w,rms))

  ds <- c(ds, ds1[i])
  xs <- c(xs, x)

  bar  <- fit.blevins(x=x,y=y,u=u,d=D,V0=V0)
  cdb  <- bar$cd
  x0   <- bar$x0
  rms2 <- bar$rms
  cdbs <- c(cdbs,cdb)
  x0s  <- c(x0s,x0)
  rms2s<- c(rms2s,rms2)
  cat(sprintf(" cd = %5.3f  x0 = %5.3f  rms= %5.3f\n",cdb,x0,rms2))

  baz  <- fit.schlichting(x=x,y=y,u=u,d=D,V0=V0)
  nu   <- baz$nu
  l    <- baz$l
  rms3 <- baz$rms
  nus  <- c(nus,nu)
  ls   <- c(ls,l)
```

```
    rms3s<- c(rms3s,rms3)
    cat(sprintf("  nu = %5.3f   l = %5.3f   rms= %5.3f\n",nu,l,rms3))

  }
  save(file=paste(fdir,"fit_truss_",case,".Rdata",sep=""),ascii=TRUE,deltas,ws,
      rmss,ds,xs,cdbs,x0s,rms2s,nus,ls,rms3s)
  list(delta=deltas,w=ws,rms=rmss,ds=ds,xs=xs,cdb=cdbs,x0=x0s,rmsb=rms2s,nu=nus,l=
      ls,rmss=rms3s)
}

# plot rake line fits
# monopile

"plot.fit.monopile" <- function(foo,bladed=TRUE,blevins=TRUE,schlichting=TRUE,rms=
    TRUE,lg=FALSE) {
  # foo contains results from fit.monopile
  if(!rms)par(mfrow=c(1,2))
  if(blevins & schlichting & bladed & rms) par(mfrow=c(4,2))
  if(rms & !bladed) par(mfrow=c(2,2))
  par(mgp=c(1.7,0.5,0),mar=c(2.75,2.75,0.5,0.25))

  q <- which(ds.m %in% c(3,6)) # rake lines to show in plot: 3D, 6D

  x <- foo$xs
  ix  <- which(x > 2)          # selection: only show x > 2

  foo$delta  <- foo$delta[ix]
  foo$w      <- foo$w[ix]
  foo$rms    <- foo$rms[ix]
  foo$cdb    <- foo$cdb[ix]
  foo$x0     <- foo$x0[ix]
  foo$rmsb   <- foo$rmsb[ix]
  foo$nu     <- foo$nu[ix]
  foo$l      <- foo$l[ix]
  foo$rmss   <- foo$rmss[ix]
  x          <- foo$xs[ix]
  foo$xs     <- foo$xs[ix]
  dx         <- 1/sqrt(x)
  dx1        <- sqrt(x)

  library(MASS)
  xl  <- c(0,50)                 # x-limits for plot
  ix2 <- which(foo$xs > 8)       # selection: only base fits on these

  f <- data.frame(delta=foo$delta[ix2],w=foo$w[ix2],x=foo$xs[ix2],dx=1/sqrt(foo$xs
      [ix2]),dx1=sqrt(foo$xs[ix2]))
  dfit <- lm(delta ~ dx ,data=f)   # inverse square-root law
  print(f)
  print(dfit)
  dbar <- as.numeric(predict(dfit))
  print(dbar)
  wfit <- lm(w ~ dx1 ,data=f)      # square-root law
  # predict w_r (at reference point, among others)
  wbar <- as.numeric(predict(wfit))
```

```
# parameters for panel labels
ll <- -0.5
ad <- -0.12
# plot type and symbols
pc <- 1
ty <- "b"
if(bladed){
# plot Bladed model
if (TRUE) {
plot(x,foo$delta,ylim=c(0,1),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(Delta),col="blue")
#mtext("A",line=ll,adj=ad,cex=1.2)
lines(f$x,dbar,col="darkgreen")
abline(v=lx.m[q] - sx.m,lty=2) # relative position
}
if (TRUE) {
plot(x,foo$w,ylim=c(0,8),pch=1,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(w),col="blue")
#mtext("B",line=ll,adj=ad,cex=1.2)
lines(f$x,wbar,col="darkgreen")
abline(v=lx.m[q] - sx.m,lty=2)
}
}
if(blevins){
# plot Blevins model
if (TRUE) {
plot(x,foo$cdb,ylim=c(0,1.2),pch=1,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(C[d]),col="blue")
#mtext("C",line=ll,adj=ad,cex=1.2)
abline(v=lx.m[q] - sx.m,lty=2)
}
if (TRUE) {
plot(x,foo$x0,ylim=c(0,70),pch=1,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(x[0]),col="blue")
#mtext("D",line=ll,adj=ad,cex=1.2)
abline(v=lx.m[q] - sx.m,lty=2)
}
}
if(schlichting){
# plot Schlichting model
if (TRUE) {
plot(x,foo$nu,ylim=c(0,1.5),pch=1,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(nu),col="blue")
#mtext("E",line=ll,adj=ad,cex=1.2)
abline(v=lx.m[q] - sx.m,lty=2)
}
if (TRUE) {
plot(x,foo$l,ylim=c(0,1),pch=1,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(l),col="blue")
#mtext("F",line=ll,adj=ad,cex=1.2)
abline(v=lx.m[q] - sx.m,lty=2)
}
}
if(rms){
# RMS plot
```

```
  if (TRUE) {
  plot(x,foo$rms,xlim=xl,ylim=c(0,1.6),pch=1,bty="n",type="n",col="blue",xlab="x [
      m]",ylab="RMS error [m/s]")
  if(blevins)       lines(x,foo$rmsb,lty=2,col="blue",type=ty)
  if(schlichting)   lines(x,foo$rmss,lty=3,col="red",type=ty)
  if(bladed)        lines(x,foo$rms,lty=1,col="black",type=ty)
  if(lg & blevins & schlichting & bladed)legend("topright",col=c("black","blue","
      red"),lty=c(1,2,3),legend=c("Bladed","Blevins","Schlichting"))
  abline(v=lx.m[q] - sx.m,lty=2)
  abline(v=lx.m[q] - sx.m,lty=2)
  #mtext("G",line=ll,adj=ad,cex=1.2)
  }
  }
}

# plot rake line fits
# truss tower cases

"plot.fit.truss" <- function(foo,case,bladed=TRUE,blevins=TRUE,schlichting=TRUE,
    rms=TRUE, lg=FALSE) {
  if(!rms)par(mfrow=c(1,2))
  if(blevins & schlichting & bladed & rms) par(mfrow=c(4,2))
  if(rms & !bladed) par(mfrow=c(2,2))
  par(mgp=c(1.7,0.5,0),mar=c(2.75,2.75,0.5,0.25))

  if (case %in% cases.A) vdx <- lx.A - sx.A  # "behind the structure", relative
      position
  if (case %in% cases.B) vdx <- lx.B - sx.B
  if (case %in% cases.C) vdx <- lx.C - sx.C
  x <- foo$xs

  q <- which(ds.A %in% c(3,6)) # selection which rake lines to show on plot: 3D, 6
      D
  if (case %in% cases.B) q <- which(ds.B %in% c(3,6)) # 3D, 6D
  if (case %in% cases.C) q <- which(ds.C %in% c(3,6))

  ix  <- which(x > 7.8)         # selection: only show x > 7.8
  foo$delta <- foo$delta[ix]
  foo$w     <- foo$w[ix]
  foo$rms   <- foo$rms[ix]
  foo$cdb <- foo$cdb[ix]
  foo$x0    <- foo$x0[ix]
  foo$rmsb   <- foo$rmsb[ix]
  foo$nu <- foo$nu[ix]
  foo$l     <- foo$l[ix]
  foo$rmss   <- foo$rmss[ix]
  x   <- x[ix]
  dx  <- 1/sqrt(x)
  dx1 <- sqrt(x)

  print(foo)

  library(MASS)
  xl  <- c(0,50)                # x-limits for plot
  ix2 <- which(foo$xs > 7.8)    # selection: only base fits on these
```

```
f <- data.frame(delta=foo$delta[ix2],w=foo$w[ix2],x=x[ix2],dx=1/sqrt(x[ix2]),dx1
    =sqrt(x[ix2]))

# parameters for panel labels
ll <- -0.5
ad <- -0.12
# plot type and symbols
pc <- 1
ty <- "b"
if(bladed){

# plot Bladed model
if (TRUE) {
plot(x,foo$delta,ylim=c(0,1),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(Delta[r]),col="blue")
#mtext("A",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
if (TRUE) {
plot(x,foo$w,ylim=c(0,10),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(w[r]),col="blue")
#mtext("B",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
}
if(blevins){
# plot Blevins model
if (TRUE) {
plot(x,foo$cdb,ylim=c(0,1.2),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(C[d]),col="blue")
#mtext("C",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
if (TRUE) {
plot(x,foo$x0,ylim=c(0,100),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(x[0]),col="blue")
#mtext("D",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
}
if(schlichting){
# plot Schlichting model
if (TRUE) {
plot(x,foo$nu,ylim=c(0,25),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(nu),col="blue")
#mtext("E",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
if (TRUE) {
plot(x,foo$l,ylim=c(0,3),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
    expression(l),col="blue")
#mtext("F",line=ll,adj=ad,cex=1.2)
abline(v=vdx[q],lty=2)
}
```

```
  }
  if(rms){
  # RMS plot
  if (TRUE) {
  plot(x,foo$rms,xlim=xl,ylim=c(0,2),type="n",pch=pc,bty="n",col="blue",xlab="x [m
      ]",ylab="RMS error [m/s]")
  abline(v=vdx[q],lty=2)
  #mtext("G",line=ll,adj=ad,cex=1.2)
  if(blevins)lines(x,foo$rmsb,lty=2,col="blue",type=ty)
  if(schlichting)lines(x,foo$rmss,lty=3,col="red",type=ty)
  if(bladed)lines(x,foo$rms,lty=1,col="black",type=ty)
  if(lg & blevins & schlichting & bladed)legend("topright",col=c("black","blue","
      red"),lty=c(1,2,3),legend=c("Bladed","Blevins","Schlichting"))
  }
  if (FALSE) {
  plot(x,foo$x0,ylim=c(0,10),pch=pc,xlim=xl,bty="n",type=ty,xlab="x [m]",ylab=
      expression(x[0]),col="blue")
  mtext("H",line=ll,adj=ad,cex=1.2)
  abline(v=vdx[q],lty=2)
  legend("topright",bg="white",pch=c(pc),col=c("blue"),legend=c("Detail"),cex=0.8)
  }
  }
}
```

H.2 Global estimations functions, R-code

Code to find and plot global parameters for the tower shadow models. This code was provided by Michael Muskulus and modified by Torbjørn Ruud Hagen.

```
# global-fit.R: global fitting of tower shadow models
# Copyright (C) 2011 Michael Muskulus (michael.muskulus@ntnu.no)
# Version 0.2 -- Initial idea

# CHANGELOG
#
#

source("shadow.R")
#graphics.off()
# optimization criteria
# note: single profile comparisons lead to either RMS (2-norm) or maximal error (
    max-norm)
#       all profile comparisons lead to either sum of errors or maximal error
# this is a hierarchical process, with two levels (single profiles and between
    profiles)
# don't confuse this!

max.rms     <- TRUE  # minimize maximal error for single profile; alternative: sum
    of errors #MAX ERROR = TRUE, SNITT ERROR = FALSE
max.error   <- TRUE # minimize maximal error for all profiles; alternative: sum of
    errors #TOT ERROR FOR ALL CASES
change      <- TRUE #If a True/False bool needs to be changed during a loop
```

```
# cases to use in the optimization

id=22 #16 = 3d 22=6d
id_name="6d"
id_x=24
#filename <- paste('fig-truss',case[1],'-global',id_name,'.png',sep="")

case      <- c(1, 4, 7, 10)   # number
#case      <- c(2, 5, 8, 11)   # number
#case      <- c(3, 6, 9, 12)   # number
#case      <- c(7, 9, 10, 12) #Marits article
#case      <- c(1,2,3,4,5,6,7,8,9,10,11,12)   # number
case_x    <- c(id_x,id_x,id_x,id_x,id_x,id_x,id_x,id_x,id_x,id_x,id_x,id_x)   # x
    -coordinate; relative position, as in: lx.A[id] - sx.A
case_id   <- c(id,id,id,id,id,id,id,id,id,id,id,id)   # section id (number);
case_files <- c(paste('truss',case[1],'-',id_name,'.Rdata',sep=""),paste('truss',
    case[2],'-',id_name,'.Rdata',sep=""),paste('truss',case[3],'-',id_name,'.Rdata
    ',sep=""),paste('truss',case[4],'-',id_name,'.Rdata',sep=""))
#case_files <- c(paste('truss',case[1],'-',id_name,'.Rdata',sep=""),paste('truss',
    case[2],'-',id_name,'.Rdata',sep=""),paste('truss',case[3],'-',id_name,'.Rdata
    ',sep=""),paste('truss',case[4],'-',id_name,'.Rdata',sep=""),paste('truss',
    case[5],'-',id_name,'.Rdata',sep=""),paste('truss',case[6],'-',id_name,'.Rdata
    ',sep=""),paste('truss',case[7],'-',id_name,'.Rdata',sep=""),paste('truss',
    case[8],'-',id_name,'.Rdata',sep=""),paste('truss',case[9],'-',id_name,'.Rdata
    ',sep=""),paste('truss',case[10],'-',id_name,'.Rdata',sep=""),paste('truss',
    case[11],'-',id_name,'.Rdata',sep=""),paste('truss',case[12],'-',id_name,'.
    Rdata',sep=""))
#case_files <- c('truss1-3d.Rdata','truss4-3d.Rdata','truss7-3d.Rdata','truss10-3d
    .Rdata')
#case_files <- c('truss2-3d.Rdata','truss5-3d.Rdata','truss8-3d.Rdata','truss11-3d
    .Rdata')
#case_files <- c('truss3-3d.Rdata','truss6-3d.Rdata','truss9-3d.Rdata','truss12-3d
    .Rdata')
#case_files <- c('truss10-3d.Rdata','truss12-3d.Rdata','truss7-3d.Rdata','truss9-3
    d.Rdata') #Marits cases
#filename <- paste('fig-truss',case[1],'-global',id_name,'.png',sep="")

######### FIT FOR MORE THAN ONE MEASURING LENGTH ##########

#case      <- c(1, 4, 7, 10)   # number
#case      <- c(2, 5, 8, 11)   # number
#case      <- c(case, 3, 6, 9, 11)   # number
#case_x    <- c(case_x,24,24,24,24,24,24)   # x-coordinate; relative position, as
     in: lx.A[id] - sx.A
#case_id   <- c(case_id,22,22,22,22,22,22)   # section id (number); 22 = 6d
#case_files <- c(case_files,'truss1-6d.Rdata','truss4-6d.Rdata','truss7-6d.Rdata
    ','truss10-6d.Rdata')
#case_files <- c(case_files,'truss2-6d.Rdata','truss5-6d.Rdata','truss8-6d.Rdata
    ','truss11-6d.Rdata')
#case_files <- c(case_files,'truss3-6d.Rdata','truss6-6d.Rdata','truss9-6d.Rdata
    ','truss12-6d.Rdata')
#case_files <- c(case_files,'truss10-6d.Rdata','truss12-6d.Rdata','truss7-6d.Rdata
    ','truss9-6d.Rdata') #Marits cases
```

```
##################

#
# BELOW HERE SHOULD NOT REQUIRE USER CHANGES
#

no_cases   <- length(case_files)
no_points  <- length(ly)
# load all relevant files and put the profiles into a matrix

cfd <- matrix(numeric(no_cases*no_points),ncol=no_points)
for (fi in seq(along=case_files)) {
  f <- case_files[fi]
  f <- paste("/media/Iomega_HDD/Work/Rdata/",f,sep="")
  cat(paste("Loading profile from file ",f,"\n",sep=""))
  load(f)
  if (length(fl$mean.v) != no_points) stop('Profile does not have the required
      number of points?!')
  cfd[fi,] <- fl$mean.v

}
rm('fl') # might be quite large

# fitting the models
# truss tower cases

"fit.bladed.all" <- function(V0,quiet=FALSE,ntry=5) {
  fn <- function(par) {
    delta <- par[1]
    w     <- par[2]
    rms   <- 0.0
    for (fi in seq(no_cases)) {
      v     <- bladed.a(case_x[fi],ly,V0,delta,w,case[fi])

      if (max.rms) {
        rms1  <- max(abs(v-cfd[fi,]))
      } else {
        rms1  <- sqrt(mean((v-cfd[fi,])^2))
      }
      if (max.error) {
        rms <- max(rms,rms1)   #IF TRUE MINIMIZING MAXIMUM ERROR FOR HIGHEST ERROR
      } else {
        rms <- rms + rms1      #IF TRUE MINIMIZING MAX ERROR FOR SUM OF ALL
            PROFILES
      }
    }
    rms
  }
  foo        <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.0,max=1.0),runif(n=1,min=0.5,max=5.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    cat(paste("  rms=",round(foo1$value,6),"\n",sep=""))
```

```
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  cat("\n")
  # show details
  rms <- 0.0
  for (fi in seq(no_cases)) {

    v      <- bladed.a(case_x[fi],ly,V0,foo$par[1],foo$par[2],case[fi])

    if (max.rms) {
      rms1  <- max(abs(v-cfd[fi,]))
    } else {
      rms1  <- sqrt(mean((v-cfd[fi,])^2))
    }
    if (max.error) {
      rms <- max(rms,rms1)
    } else {
      rms <- rms + rms1
    }
    cat(paste("Error for case #",case[fi]," => ",round(rms1,6),"\n",sep=""))
  }
  cat(paste("TOTAL error => ",round(rms),"\n",sep=""))

  #if (!quiet) { cat(paste("BLADED FIT:\n")); print(foo) }
  delta <- foo$par[1]
  w     <- foo$par[2]
  rms   <- foo$value
  list(delta=delta,w=w,rms=rms)
}

"fit.blevins.all" <- function(V0,quiet=FALSE,ntry=5) {
  fn <- function(par) {
    cd   <- par[1]
    x0   <- par[2]
    rms  <- 0.0
    for (fi in seq(no_cases)) {
      v    <- blevins.a(case_x[fi],ly,V0,cd,x0,case[fi])
      if (max.rms) {
        rms1  <- max(abs(v-cfd[fi,]))
      } else {
        rms1  <- sqrt(mean((v-cfd[fi,])^2))
      }
      if (max.error) {
        rms <- max(rms,rms1)
      } else {
        rms <- rms + rms1
      }
    }
    rms
  }
  foo        <- NULL
  best.value <- Inf
```

```
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars,fn=fn,method="Nelder-Mead",control=list(maxit=500))
    cat(paste("  rms = ",round(foo1$value,6),"\n",sep=""))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  cat("\n")
  # show details
  rms <- 0.0
  for (fi in seq(no_cases)) {
    v     <- blevins.a(case_x[fi],ly,V0,foo$par[1],foo$par[2],case[fi])
    if (max.rms) {
      rms1  <- max(abs(v-cfd[fi,]))
    } else {
      rms1  <- sqrt(mean((v-cfd[fi,])^2))
    }
    if (max.error) {
      rms <- max(rms,rms1)
    } else {
      rms <- rms + rms1
    }
    cat(paste("Error for case #",case[fi]," => ",round(rms1,6),"\n",sep=""))
  }

  if (!quiet) { cat(paste("BLEVINS FIT:\n")); print(foo) }
  cd   <- foo$par[1]
  x0   <- foo$par[2]
  rms  <- foo$value
  list(cd=cd,x0=x0,rms=rms)
}


"fit.schlichting.all" <- function(V0,quiet=FALSE,ntry=5) {
  fn <- function(par) {
    nu   <- par[1]
    l    <- par[2]
    rms  <- 0.0
    for (fi in seq(no_cases)) {
      v    <- schlichting.a(case_x[fi],ly,V0,nu,l,case[fi])
      if (max.rms) {
        rms1  <- max(abs(v-cfd[fi,]))
      } else {
        rms1  <- sqrt(mean((v-cfd[fi,])^2))
      }
      if (max.error) {
        rms <- max(rms,rms1)
      } else {
        rms <- rms + rms1
      }
    }
    rms
  }
```

```
  foo         <- NULL
  best.value <- Inf
  for (i in seq(ntry)) {
    pars <- c(runif(1,min=0.25,max=3.0),runif(n=1,min=0.5,max=9.0))
    foo1 <- optim(par=pars, fn=fn, method="Nelder-Mead",control=list(maxit=500))
    cat(paste("  rms = ",round(foo1$value,6),"\n",sep=""))
    if (foo1$value < best.value) {
      foo <- foo1
      best.value <- foo1$value
    }
  }
  cat("\n")
  # show details
  rms <- 0.0
  for (fi in seq(no_cases)) {
    v     <- schlichting.a(case_x[fi],ly,V0,foo$par[1],foo$par[2],case[fi])
    if (max.rms) {
      rms1  <- max(abs(v-cfd[fi,]))
    } else {
      rms1  <- sqrt(mean((v-cfd[fi,])^2))
    }
    if (max.error) {
      rms <- max(rms,rms1)
    } else {
      rms <- rms + rms1
    }
    cat(paste("Error for case #",case[fi]," => ",round(rms1,6),"\n",sep=""))
  }


  if (!quiet) { cat(paste("SCHLICHTING FIT:\n")); print(foo) }
  nu   <- foo$par[1]
  l    <- foo$par[2]
  rms  <- foo$value
  list(nu=nu,l=l,rms=rms)
}

# for plotting: use plot.truss.a and give it the estimated parameters
#
# example:
#   > res <- fit.bladed.all(V0=12.0)
#   > plot.truss.a(id=case_id[1],case[1],delta=res$delta,w=res$w)  # id gives the
    x-section

#graphics.off()

run.example <- function() {
  #res.bladed <- fit.bladed.all(V0=12)
  #res.blevins <- fit.blevins.all(V0=12)
  res.schlichting <- fit.schlichting.all(V0=12)
#for(counter in 1:4){

#png(paste('fig-truss-bladed',case[counter],'-global4-',id_name,'.png',sep=""),
    pointsize=9,width=7,height=7,res=600,units="cm")
#par(mfrow=c(1,1))
```

```
#par(mgp=c(1.4,0.5,0),mar=c(2.5,2.5,0.25,0.25))
#plot.truss.a(id,case[counter], yl=c(6,14), delta=res.bladed$delta,verbose=F,w=res
    .bladed$w,schlichting=F,blevins=F,potential=F,Global=T,norm=F,best.fit=T,lg=
    change)
#if(counter == 1) change <- FALSE
#}
#for(counter in 1:4){
#png(paste('fig-truss-blevins',case[counter],'-global1-',id_name,'.png',sep=""),
    pointsize=9,width=7,height=7,res=600,units="cm")
#par(mfrow=c(1,1))
#par(mgp=c(1.4,0.5,0),mar=c(2.5,2.5,0.25,0.25))
#plot.truss.a(id,case[counter], yl=c(6,14), x0=res.blevins$x0,verbose=F,cd=res.
    blevins$cd,schlichting=F,bladed=F,blevins=T,potential=F,Global=T,norm=F,best.
    fit=T,lg=change)
#if(counter == 1) change <- FALSE
#}
for(counter in 1:4){
png(paste('fig-truss-schlichting',case[counter],'-global1-',id_name,'.png',sep="")
    ,pointsize=9,width=7,height=7,res=600,units="cm")
par(mfrow=c(1,1))
par(mgp=c(1.4,0.5,0),mar=c(2.5,2.5,0.25,0.25))
plot.truss.a(id,case[counter], yl=c(6,14), l=res.schlichting$l,verbose=F,nu=res.
    schlichting$nu,schlichting=T,bladed=F,blevins=F,potential=F,Global=T,norm=F,
    best.fit=T,lg=change)
if(counter == 1) change <- FALSE
}
graphics.off()

}



run.example.blevins <- function() {
  res <- fit.blevins.all(V0=12)
  par(mfrow=c(3,2))
  plot.truss.a(16,1, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)
  plot.truss.a(16,2, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)
  plot.truss.a(16,7, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)
  plot.truss.a(16,8, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)
  plot.truss.a(16,10, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)
  plot.truss.a(16,11, bladed=F,schlichting=F,blevins=T,cd=res$cd,x0=res$x0)

}


run.example.schlichting <- function() {
  res <- fit.schlichting.all(V0=12)
  par(mfrow=c(3,2))
  plot.truss.a(16,1, bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
  plot.truss.a(16,2, bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
  plot.truss.a(16,7, bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
  plot.truss.a(16,8, bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
  plot.truss.a(16,10,bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
  plot.truss.a(16,11,bladed=F,schlichting=T,blevins=F,nu=res$nu,l=res$l)
```

```
}
```

H.3 Spectral estimations functions, R-code

Code to get frequency peaks from the numerical time-series. This code was provided by Michael Muskulus.

```
# get-peaks.R: find local maxima in discrete data
# Copyright (C) 2011 Michael Muskulus (michael.muskulus@ntnu.no)
# Version 0.9 -- ISOPE-2011 paper

"get.peaks" <- function(x,y,n=NULL) {
  # get a list of peaks (y-profile changing direction) ordered by magnitude
  yd  <- diff(y)
  ydl <- yd[-length(yd)]
  ydr <- yd[-1]
  ix  <- which(ydl > 0 & ydr < 0) + 1 # local maxima
  xm  <- x[ix]
  ym  <- y[ix]
  foo <- sort(ym, index.return=TRUE, decreasing=TRUE)
  if (!is.null(n)) { foo$ix <- foo$ix[seq(n)] }
  xm  <- xm[foo$ix]
  ym  <- ym[foo$ix]
  list(xm=xm,ym=ym)
}
```

Code to plot frequency spectrum. This code was provided by Michael Muskulus and modified by Torbjørn Ruud Hagen.

```
# spectra.R: estimate and plot spectra
# Copyright (C) 2011 Michael Muskulus (michael.muskulus@ntnu.no)
# Version 0.9 -- ISOPE-2011 paper

source("get-peaks.R")


"plot.spec" <- function(spec,mt,n) {
  yl   <- c(5,14)
  xl1  <- c(0,8)
  yl1  <- c(1e-6,8)
  yl2  <- c(1e-8,1e2)
  fsp  <- 5

  spx     <- spec$freq
  spy     <- 2 * spec$spec # correction for one-sided spectrum
  plot(spx,spy,log="xy",lwd=1,ylim=yl2,type="o",bty="n",col="blue",xlab="f [Hz]",
       ylab="S(f)",cex=0.2,pch=1,cex.lab=1.5,cex.axis=1.5,xlim=c(0.5,50))
  mtext(mt,line=-0.7,adj=-0.2,cex=1.2)
  foo <- get.peaks(spx,spy,n=10)
```

```
  print(foo)

  # find peak
  spy <- spy[spx > 0.5]
  spx <- spx[spx > 0.5]
  peak1 <- spx[which.max(spy)]
  oof   <- sort(spy,index.return=TRUE,decreasing=TRUE)
  ix2 <- 2
  while (abs(oof$ix[ix2] - oof$ix[1]) <= fsp) ix2 <- ix2 + 1
  peak1b <- spx[oof$ix[ix2]]
  loc1b  <- spy[oof$ix[ix2]]
  loc1   <- max(spy)
  abline(h=loc1,lty=2,lwd=1)
  abline(v=peak1,lty=2,lwd=1)
  abline(v=peak1b,lty=3,lwd=1)
  abline(h=loc1b,lty=3,lwd=1)
  cat(sprintf("First  peak at %5.3f Hz with power %6.4f\n",peak1,loc1))
  cat(sprintf("Second peak at %5.3f Hz with power %6.4f\n",peak1b,loc1b))
  legend("topright",lty=c(2,3),legend=c(sprintf("1st  %3.2f Hz",peak1),sprintf("2
      nd %3.2f Hz",peak1b)),cex=0.8)
  #legend("topright",lty=c(1),legend=c(sprintf("%d points",n)),cex=0.8)
}

#source("shadow.R")

graphics.off()
png("fig-spectra.png",width=9,height=12,pointsize=9,res=600,units="cm")

  par(mfrow=c(3,2))
  par(mgp=c(1.6,0.5,0),mar=c(4,2.7,0.5,0.5),oma=c(1.5,2,1,1))

  anew <- FALSE

# source on external harddrive "/media/Elements/Work/Rdata/"

  # load data
  if (!exists("ym") || anew) {
    load("/media/Elements/Work/Rdata/ts-monopile-3d.Rdata")
    xm <- fl$v
    ym <- fl$y
    rm(fl) # save space
  }
  if (!exists("ym.t") || anew) {
    load("/media/Elements/Work/Rdata/ts-truss1-2d.Rdata")
    xm.t <- fl$v
    ym.t <- fl$y
    rm(fl) # save space
  }
  if (!exists("y10") || anew) {
    load("/media/Elements/Work/Rdata/ts-truss4-2d.Rdata")
    x10 <- fl$v
    y10 <- fl$y
    rm(fl) # save space
  }
  if (!exists("x10.t") || anew) {
```

```
  load("/media/Elements/Work/Rdata/ts-truss7-2d.Rdata")
  x10.t <- fl$v
  y10.t <- fl$y
  rm(fl) # save space
}
if (!exists("x10front") || anew) {
  load("/media/Elements/Work/Rdata/ts-truss10-2d.Rdata")
  x10front <- fl$v
  y10front <- fl$y
  rm(fl) # save space
}
if (!exists("x10front.t") || anew) {
  load("/media/Elements/Work/Rdata/ts-truss1-1d.Rdata")
  x10front.t <- fl$v
  y10front.t <- fl$y
  rm(fl) # save space
}

dt <- 0.005

span <- NULL
#span <- c(2)

i  <- 201 # centerline for truss
im <- 201 # centerline for monopile

cat(paste("Truss 1: ",nrow(xm),"time steps available\n",sep=""))
cat(paste("Truss 2: ",nrow(xm.t),"time steps available\n",sep=""))
cat(paste("Truss 3: ",nrow(x10),"time steps available\n",sep=""))
cat(paste("Truss 4: ",nrow(x10.t),"time steps available\n",sep=""))
cat(paste("Truss 5: ",nrow(x10front),"time steps available\n",sep=""))
cat(paste("Truss 6: ",nrow(x10front.t),"time steps available\n",sep=""))

#trans <- seq(200) # remove first 200 data points
last <- 400
last <- 2000   # 10 s
prelim <- FALSE # no truss turbulent


zm    <- xm[nrow(xm)-rev(seq(last)),]
zm.t  <- xm.t[nrow(xm.t)-rev(seq(last)),]
z10   <- x10[nrow(x10)-rev(seq(last)),]
z10front <- x10front[nrow(x10front)-rev(seq(last)),]
# z10.t <- x10.t[,]
if (prelim) {
  z10.t <- x10.t[nrow(x10.t)-rev(seq(600)),]  # need at least 3 seconds (to
      travel 30 m downstream), i.e., 600 time steps
} else {
  z10.t <- x10.t[nrow(x10.t)-rev(seq(last)),]
}
if (prelim) {
  z10front.t <- x10front.t[nrow(x10front.t)-rev(seq(600)),]  # need at least 3
      seconds (to travel 30 m downstream), i.e., 600 time steps
} else {
  z10front.t <- x10front.t[nrow(x10front.t)-rev(seq(last)),]
```

```
  }

  cat(paste("i = ",i,"; y[i] = ",ym[i]," ",ym.t[im]," ",y10[i]," ",y10.t[i]," ",
      y10front[i]," ",y10front.t[i],"\n",sep=""))
  #ts.m   <- ts(xm[-trans,i],    deltat=dt)
  #ts.mt  <- ts(xm.t[-trans,im], deltat=dt)
  #ts.10  <- ts(x10[-trans,i],   deltat=dt)
  #ts.10t <- ts(x10.t[-trans,i], deltat=dt)
  ts.m   <- ts(zm[,i],    deltat=dt)
  ts.mt  <- ts(zm.t[,im], deltat=dt)
  ts.10  <- ts(z10[,i],   deltat=dt)
  ts.10t <- ts(z10.t[,i], deltat=dt)
  tsfront.10  <- ts(z10front[,i], deltat=dt)
  tsfront.10t <- ts(z10front.t[,i], deltat=dt)


  spec.m    <- spectrum(ts.m,    span=span, plot=FALSE)
  spec.mt   <- spectrum(ts.mt,   span=span, plot=FALSE)
  spec.10   <- spectrum(ts.10,   span=span, plot=FALSE)
  spec.10t  <- spectrum(ts.10t,  span=span, plot=FALSE)
  specfront.10 <- spectrum(tsfront.10, span=span, plot=FALSE)
  specfront.10t <- spectrum(tsfront.10t, span=span, plot=FALSE)


  spec    <- spec.m
  plot.spec(spec,"A", n=length(ts.m))
  spec    <- spec.mt
  plot.spec(spec,"B", n=length(ts.mt))
  spec    <- spec.10
  plot.spec(spec,"C", n=length(ts.10))
  spec    <- spec.10t
  plot.spec(spec,"D", n=length(ts.10t))
  spec    <- specfront.10
  plot.spec(spec,"E",n=length(tsfront.10))
  spec    <- specfront.10t
  plot.spec(spec,"F", n=length(tsfront.10t))

  cat(paste("Truss 1: ",nrow(xm)," time steps available\n",sep=""))
  cat(paste("Truss 2: ",nrow(xm.t)," time steps available\n",sep=""))
  cat(paste("Truss 3: ",nrow(x10)," time steps available\n",sep=""))
  cat(paste("Truss 4: ",nrow(x10.t)," time steps available\n",sep=""))
  cat(paste("Truss 5: ",nrow(x10front)," time steps available\n",sep=""))
  cat(paste("Truss 6: ",nrow(x10front.t)," time steps available\n",sep=""))

  cat(paste("i = ",i," im = ",im,"; y[i] = ",ym[i]," ",ym.t[im]," ",y10[i]," ",
      y10.t[i]," ",y10front[i]," ",y10front.t[i],"\n",sep=""))

graphics.off()
```

H.4 Turbulent inflow implementations, C-Code

This code is implemented in ANSYS Fluent as a user-defined function in order to run simulations with von Karman turbulent inflow. Provided by Michael Muskulus. Code to plot frequency spectrum. This code was provided by Michael Muskulus and modified by Torbjørn Ruud Hagen.

```
/* inflow.c -- Von Karman turbulent inflow for Fluent simulations */
/* ---------------------------------------------------------------- */
/* Copyright 2011 by Michael Muskulus (michael.muskulus@ntnu.no)  */

#include "udf.h"
#include "/usr/local/include/gsl/gsl_math.h"
#include "/usr/local/include/gsl/gsl_sf.h"
#include "/usr/local/include/gsl/gsl_ieee_utils.h"

const real TPI = 2.0 * M_PI;

/* model constants */
const double Lu   = 73.5;
const double Um   = 12.0;
const double stdU = 1.6;
const double stdV = 1.2;
const double c    = 1.0;   /* definition of eta */

/* method constants */
int nbins;  /* dynamically determined */
const int nfreqs    = 100;
const double flower = 0.0;
const double fupper = 10.0;
const double deltaf = 0.01;
const double df     = 0.1; /* (fupper - flower) / nfreq; */
const double cutoff = 1e-6;
const int bw        = 20;   /* adjacent processes */

/* derived constants */
const double Lv       = 36.75;     /* Lu/2.0; */
const double Lv2      = 73.5;      /* Lv*2.0; */

/* precomputed for speed */
const double stdU2    = 2.56;         /* stdU*stdU; */
const double stdV2    = 1.44;         /* stdV*stdV; */
const double psd1uC   = 62.72;        /* stdU2*4*Lu/Um; */
const double psd1vC   = 17.64;        /* stdV2*4*Lv/Um; */
const double psd2uC1  = 2.58230e-05;  /* (0.747/Lu2)*(0.747/Lu2); */ /* first term
    in eta */
const double psd2uC2  = 0.5235988;    /* c * TPI / Um; */
const double psd2vC1  = 0.000103292;  /* (0.747/Lv2)*(0.747/Lv2); */
const double psd2vC2  = 0.5235988;    /* c * TPI / Um; */
const double LudUm    = 6.125;        /* Lu/Um; */
const double LvdUm    = 3.0625;       /* Lv/Um; */
const double c56      = 0.8333333;    /* 5.0/6.0;  */
const double c116     = 1.8333333; /* 11.0/6.0; */
const double fuc      = 0.4472136; /* sqrt(2.0 * df); */

/* does this work? */
```

```
int first_run = 1;
int tdone_x, tdone_y;
FILE* ret;

/* main coefficient data */
double* mH1;
double* mH2;
double* phi1;
double* phi2;

/* helpers */
double* freqs;
double* ys;     /* we are paranoid and check y values at each time step */
double  ymin, ymax;
const double YDIM = 30.0;

double fmin(double x1, double x2) {
  if (x1 < x2) return x1; else return x2;
};

double fmax(double x1, double x2) {
  if (x1 > x2) return x1; else return x2;
};

double psd1u(double f) {
  double x1 = gsl_pow_2(LudUm*f);
  double s = psd1uC / pow(1.0 + 70.8 * x1, c56);
  return s;
};

double psd1v(double f) {
  double x1 = gsl_pow_2(LvdUm*f);
  double s = psd1vC * (1 + 755.2 * x1) / pow(1 + 283.2 * x1, c116);
  return s;
};

double psd2u(double f, double r) {
  double eta = r * sqrt(psd2uC1 + gsl_pow_2(psd2uC2 * f));
  double coh = 0.994 * (pow(eta,c56) * gsl_sf_bessel_Knu(c56, eta) - 0.5 * pow(eta
      ,c116) * gsl_sf_bessel_Knu(c116, eta));
  coh = fmin(coh, 1.0);  /* restrict to realistic values */
  coh = fmax(coh,-1.0);
  double s = coh * psd1u(f);
  return s;
};

double psd2v(double f, double r) {
  double eta = r * sqrt(psd2vC1 + gsl_pow_2(psd2vC2 * f));
  double g   = eta * Lv2 / r;
  double g2  = gsl_pow_2(g);
  double coh = 0.597 * (4.781 * g2 * pow(eta,c56) * gsl_sf_bessel_Knu(c56,eta) -
                        pow(eta,c116) * gsl_sf_bessel_Knu(c116,eta)) / (2.869 * g2
                            - 1.0);
  coh = fmin(coh, 1.0);  /* restrict to realistic values */
  coh = fmax(coh,-1.0);
```

```
  double s = coh * psd1v(f);
  return s;
};

int ind_y(int j) { /* wrapping around the y values */
  while (j < 0)      j += nbins;
  while (j >= nbins) j -= nbins;
  return j;
};

int ind_yf(int j, int l) {  /* channel, frequency */
  /* allow for wrap in j */
  int i = ind_y(j) + l*nbins;  /* CAVEAT: here was a terrible memory access
      mistake, nfreq instead of nbins */
  return i;
};

int ind_bwf(int j, int l) {  /* channel, frequency */
  if (j < 0) j = -j;
  int i = j + l*bw;
  return i;
};

double get_r(int i, int j) {
  /* ys[i] - ys[ind_y(j)]; */
  double d1 = fabs(ys[ind_y(i)]-ys[ind_y(j)]);
  double d2 = fabs(ys[ind_y(i)]-ys[ind_y(j)]+(YDIM));
  double d3 = fabs(ys[ind_y(i)]-ys[ind_y(j)]-(YDIM));
  d1 = fmin(d1,d2);
  d1 = fmin(d1,d3);
  return d1;
};

void initialize() {
  int i,j,k;
  double r;
  char s[255];
  /* mask IEEE underflow ... in Bessel computation */
  gsl_ieee_env_setup();

  /* initialize frequency bins */
  freqs = (double*) malloc(nfreqs * sizeof(double));
  for (i=0; i<nfreqs; ++i) {
    freqs[i] = ((double) i + 0.5 + flower) * df + (double) rand() * deltaf / (
        double) RAND_MAX;;
    Message("freq. %i = %8.5f\n",i,freqs[i]);
  };

  /* initialize coefficient space */
  mH1  = (double*) malloc(nfreqs*bw*sizeof(double));
  mH2  = (double*) malloc(nfreqs*bw*sizeof(double));
  phi1 = (double*) malloc(nfreqs*nbins*sizeof(double));
  phi2 = (double*) malloc(nfreqs*nbins*sizeof(double));

  for (k=0; k<bw;  ++k)
```

```
    for (i=0; i<nfreqs; ++i) {
      mH1[ind_bwf(k,i)] = 0.0;
      mH2[ind_bwf(k,i)] = 0.0;
    };

  Message("Building H matrices\n");
  /* average spectral contributions */
  for (j=0; j<nbins; ++j) {
    for (k=0; k<bw; ++k) {
      /*Message("j = %3d, k = %3d => r = %8.5g\n",j,k,r);*/
      for (i=0; i<nfreqs; ++i) {
        if (k == 0) {
          mH1[ind_bwf(0,i)] += (psd1u(freqs[i]) / (double)(nbins));
          mH2[ind_bwf(0,i)] += (psd1v(freqs[i]) / (double)(nbins));

          sprintf(s, "f = %8.5g : y = %8.5g, k = %3d, H1 += %12.8g, H2 += %12.8g\n
              ",freqs[i],ys[j],k,psd1u(freqs[i]),psd1v(freqs[i]));
          fputs(s,ret);
        } else {
          r = get_r(j,j+k);
          mH1[ind_bwf(k,i)] += (psd2u(freqs[i],r) / (2.0 * (double)(nbins)));
          mH2[ind_bwf(k,i)] += (psd2v(freqs[i],r) / (2.0 * (double)(nbins)));

          sprintf(s, "f = %8.5g : y1 = %8.5g, y2 = %8.5g, k = %3d, r = %8.5g, H1
              += %12.8g, H2 += %12.8g\n",freqs[i],ys[j],ys[ind_y(j+k)],k,r,psd2u(
              freqs[i],r),psd2v(freqs[i],r));
          fputs(s,ret);

          r = get_r(j,j-k);
          mH1[ind_bwf(k,i)] += (psd2u(freqs[i],r) / (2.0 * (double)(nbins)));
          mH2[ind_bwf(k,i)] += (psd2v(freqs[i],r) / (2.0 * (double)(nbins)));

          sprintf(s, "f = %8.5g : y1 = %8.5g, y2 = %8.5g, k = %3d, r = %8.5g, H1
              += %12.8g, H2 += %12.8g\n",freqs[i],ys[j],ys[ind_y(j-k)],k,r,psd2u(
              freqs[i],r),psd2v(freqs[i],r));
          fputs(s,ret);
        };
        /* Message("j = %d, k = %d, i = %d, H1= %8.5g  H2= %8.5g\n",j,k,i,mH1[
            indcf(k,i)],mH2[indcf(k,i)]); */
      };
    };
  };
  /* normalize */
  double sq1, sq2;
  for (i=0; i<nfreqs; ++i) {
    sq1 = sq2 = 0.0;
    for (k=0; k<bw; ++k) {
      sq1 += mH1[ind_bwf(k,i)] * mH1[ind_bwf(k,i)];
      sq2 += mH2[ind_bwf(k,i)] * mH2[ind_bwf(k,i)];
    };
    sq1 = sqrt(sq1) / sqrt(mH1[ind_bwf(0,i)]);
    sq2 = sqrt(sq2) / sqrt(mH2[ind_bwf(0,i)]);
    for (k=0; k<bw; ++k) {
      mH1[ind_bwf(k,i)] /= sq1;
      mH2[ind_bwf(k,i)] /= sq2;
```

```
    };
    Message("#1 f= %8.5g : ",freqs[i]);
    for (k=0; k<bw; ++k) Message("%12.8g ",mH1[ind_bwf(k,i)]);
    Message("\n#2 f= %8.5g : ",freqs[i]);
    for (k=0; k<bw; ++k) Message("%12.8g ",mH2[ind_bwf(k,i)]);
    Message("\n");
    sprintf(s, "H1 / H2 f= %8.5g\n",freqs[i]);
    fputs(s,ret);
    for (k=0; k<bw; ++k) {
      sprintf(s, "%12.8g ",mH1[ind_bwf(k,i)]);  fputs(s,ret);
    };
    sprintf(s, "\n");
    fputs(s,ret);
    for (k=0; k<bw; ++k) {
      sprintf(s, "%12.8g ",mH2[ind_bwf(k,i)]);  fputs(s,ret);
    };
    sprintf(s, "\n");
    fputs(s,ret);
  };

  /* random phases */
  Message("Random phases\n");
  for (j=0; j<nbins; j++)
    for (i=0; i<nfreqs; ++i) {
      phi1[ind_yf(j,i)] = (double) rand() * TPI / (double) RAND_MAX;
      phi2[ind_yf(j,i)] = (double) rand() * TPI / (double) RAND_MAX;
    };
};



double evaluate_u(double t, int j) {
  int m,l;
  double fu = 0.0;
  double f;
  for (m=-bw; m<bw; ++m)
    for (l=0; l<nfreqs; ++l) {
      f = fabs(mH1[ind_bwf(m,l)]);
      f *= fuc;
      f *= cos(freqs[l]*t + phi1[ind_yf(m+j,l)]);
      fu += f;
    };
  return fu;
};

double evaluate_v(double t, int j) {
  int m,l;
  double fv = 0.0;
  double f;
  for (m=-bw; m<bw; ++m)
    for (l=0; l<nfreqs; ++l) {
      f = fabs(mH2[ind_bwf(m,l)]);
      f *= fuc;
      f *= cos(freqs[l]*t + phi2[ind_yf(m+j,l)]);
```

```
      fv += f;
    };
  return fv;
};



DEFINE_PROFILE(inlet_x_velocity, t, i)
{
  real x[ND_ND]; /* position vector; ND_ND = 2 for 2D model */
  real y;
  face_t f;
  real ts = CURRENT_TIME; /* in seconds */
  char s[255];
  int  nt = N_TIME;      /* no. time step */

  if (first_run) {
    /* find out the number of cells */
    nbins = 0;
    begin_f_loop(f,t)
    {
      F_CENTROID(x,f,t);
      y = x[1];
      nbins++;
    }
    end_f_loop(f,t)
    int bin = 0;
    /* write out some information */
    char s[20];
    first_run = 0;
    ret = fopen("inflow.out","w");
    fputs("INITIALIZATION IN U\n", ret);
    sprintf(s, "TIME STEP = %i\n", (int)N_TIME);
    sprintf(s, "NO. FACES = %i\n", nbins);
    fputs(s,ret);
    ys = (double*) malloc(nbins*sizeof(double));
    ymin = ymax = 0.0;
    begin_f_loop(f,t)
    {
      F_CENTROID(x,f,t);
      y = x[1];
      ys[bin] = y;
      bin++;
      if (y < ymin) ymin = y;
      if (y > ymax) ymax = y;
      sprintf(s, "face #%i: y = %8.4f\n", bin, y);
      Message("face #%i: y = %8.4f\n", bin, y);
      fputs(s,ret);
    }
    end_f_loop(f,t)
    sprintf(s, "ymin = %8.4f\nymax = %8.4f\n",ymin,ymax);
    fputs(s,ret);
    /* initialize */
    Message("INITIALIZING H\n");
```

```
    initialize();
    sprintf(s,"\nRunning simulation\nSampling velocity at  y[40] = %6.4f,  y[80] =
        %6.4f, y[120] = %6.4f\n\n",ys[39],ys[79],ys[119]);
    fputs(s,ret);
    fclose(ret);
  };

  int bin = 0;
  double v1,v2,v3;
  double v;
  begin_f_loop(f,t)
  {
    F_CENTROID(x,f,t);
    y = x[1];
    if (abs(y - ys[bin]) > 1e-4) exit(1);       /* being paranoid */
    v = Um + evaluate_u(ts,bin);
    if (bin ==  39) v1 = v;
    if (bin ==  79) v2 = v;
    if (bin == 119) v3 = v;
    F_PROFILE(f,t,i) = v;
    bin++;
  }
  end_f_loop(f,t)
  if (tdone_x != nt) {
    ret = fopen("inflow.out","a");
    sprintf(s, "Evaluating x-velocity at time step = %i\n", (int)N_TIME);
    fputs(s,ret);
    sprintf(s, "  vx[40] = %8.6g,  vx[80] = %8.6g, vx[120] = %8.6g\n",v1,v2,v3);
    fputs(s,ret);
    fclose(ret);
    tdone_x = nt;
  };
}

DEFINE_PROFILE(inlet_y_velocity, t, i)
{
  real x[ND_ND]; /* position vector; ND_ND = 2 for 2D model */
  real y;
  face_t f;
  char s[255];
  real ts = CURRENT_TIME; /* in seconds */
  int  nt = N_TIME;       /* no. time step */

  if (first_run) {
    /* find out the number of cells */
    nbins = 0;
    begin_f_loop(f,t)
    {
      F_CENTROID(x,f,t);
      y = x[1];
      nbins++;
    }
    end_f_loop(f,t)
    int bin = 0;
    /* write out some information */
```

```
    char s[20];
    first_run = 0;
    ret = fopen("inflow.out","w");
    fputs("INITIALIZATION IN V\n", ret);
    sprintf(s, "TIME STEP = %i\n", (int)N_TIME);
    sprintf(s, "NO. FACES = %i\n", nbins);
    fputs(s,ret);
    ys = (double*) malloc(nbins*sizeof(double));
    ymin = ymax = 0.0;
    begin_f_loop(f,t)
    {
      F_CENTROID(x,f,t);
      y = x[1];
      ys[bin] = y;
      bin++;
      if (y < ymin) ymin = y;
      if (y > ymax) ymax = y;
      sprintf(s, "face #%i: y = %8.4f\n", bin, y);
      fputs(s,ret);
    }
    end_f_loop(f,t)
    sprintf(s, "ymin = %8.4f\nymax = %8.4f\n",ymin,ymax);
    fputs(s,ret);
    /* initialize */
    Message("INITIALIZING H\n");
    initialize();
    fclose(ret);
  };

  int bin = 0;
  double v1, v2, v3;
  double v;
  begin_f_loop(f,t)
  {
    F_CENTROID(x,f,t);
    y = x[1];
    if (abs(y - ys[bin]) > 1e-4) exit(1);       /* being paranoid */
    v = evaluate_v(ts,bin);
    if (bin ==  39) v1 = v;
    if (bin ==  79) v2 = v;
    if (bin == 119) v3 = v;
    F_PROFILE(f,t,i) = v;
    bin++;
  }
  end_f_loop(f,t)
  if (tdone_y != nt) {
    ret = fopen("inflow.out","a");
    sprintf(s, "Evaluating y-velocity at time step = %i\n", (int)N_TIME);
    fputs(s,ret);
    sprintf(s, "  vy[40] = %8.6g,  vy[80] = %8.6g, vy[120] = %8.6g\n",v1,v2,v3);
    fputs(s,ret);
    fclose(ret);
    tdone_y = nt;
  };
}
```