Upgrade of Fuga

Title (of deliverable): Upgrade of Fuga
Deliverable no.: D1.7

Delivery date: 12.09.2019
Lead beneficiary: DTU
Dissemination level: PU

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 727680

Author(s) information (alphabetical):		
Name	Organisation	Email
Søren Ott Paul van der Laan Gunner Larsen	DTU DTU DTU	sqot@dtu.dk

Acknowledgements/Contributions:		
Name	Name	Name

Document information

Version	Date	Description			
			Prepared by	Reviewed by	Approved by
1	12.09.2019	Name	Søren Ott		Gunner Chr. Larsen

Definitions

DWM	Dynamic Wake Meandering
WT	Wind turbine
WF	Wind farm
ABL	Atmospheric boundary layer
CFD	Computational fluid mechanics
RANS	Reynolds averaged Navier Stokes
LES	Large eddy simulation

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This report describes results from task 1.7 of the TotalControl project: ‘Upgrade of Fuga for yawed rotors and strongly stable stratification, and report with testing and validation results. The deliverable is referring to task 1.3.1. Measure of success: Accuracy in new regimes with same level as Fuga accuracy in standard regimes.’

Fuga is a linearized model and the generalization to yawed cases is straight forward. It merely consists in solving the first order equations for the case of transverse forcing. However, the results are useless because there is almost no deflection of the resulting yawed wake, and the deflection was going in the wrong direction. An investigation using non-linear CFD was then conducted in order to find out what was going on. The setup is a single actuator disk in flat, homogeneous terrain. Everything was kept fixed except for the thrust coefficient C_T. The results show that the velocity deficit roughly scales with C_T, thus explaining the relative success of linearized models. However, the left-right anti-symmetric part of the deficit profile, which is responsible for the wake deflection, did not scale with C_T at all in the central part of the wake (it did in fact in the outskirts of the wake). We propose a solution to this dilemma which combines the linearization with a suitable coordinate transformation. It turns out that the first order equations expressed in the transformed coordinates have exactly the same form as before. The solution is therefore the same as before except that it refers to a transformed coordinate system, which can be chosen afterwards. The best choice is determined in such a way that the term $v\,\partial u_i/\partial y$ appears when the first order equations are transformed back to the original coordinates.

Due these unexpected difficulties, the main focus has been on reformulating the model so that it can deal with yaw effects. We demonstrate that the new strategy gives meaningful results for neutral stratification when compared with CFD model results. A quantitative comparison could, however, not be made because two different closures were used by for two models. Lately, Paul van der Laan has made it possible to use the Fuga closure with his CFD model, thus enabling a more quantitative analysis. There are still problems with strongly stable cases and work on improving the solver will continue. Finally, we are waiting for the experimental results to be generated within the TotalControl project for validation.

INTRODUCTION

The Fuga model [1, 2] is a linearized CFD flow model that calculates wake effects in wind farms. The present version works in homogeneous, flat terrain including a sea surface. Fuga is many orders of magnitude more cost efficient than the simple, non-linear closure model on which it is based (or any other non-linear CFD model for that matter). Even if the Annual Energy Production (AEP) is the main output, sensitivities with respect to layout design parameters (e.g. turbine positions) can also be made, using a method which is much faster than simple finite differencing. This feature facilitates the use of fast, gradient based algorithms for windfarm layout optimization. In the TotalControl project it is an objective to control yaw misalignment and power curtailment in order to maximize wind farm power production for given meteorological conditions. The main aim of the work presented here is to upgrade Fuga to incorporate yaw control.

The main aim of the work presented here is to upgrade Fuga for yaw control in order to make possible to estimate the feasibility of yaw control within the TotalControl project. Atmospheric stability is likely to be very important because it affects both small and large scales of the atmospheric turbulence. The enhanced small scale turbulence in unstable conditions speeds up the wake decay and broadens the wakes which should make yaw control less efficient in terms of optimization of power output. At the same time the enhanced large scale turbulence causes wake meandering and increased uncertainty of predictions of 'the' wind speed and direction, which needs to be made at a distance from the measurement point (e.g. a met mast) and ahead in time. It therefore seems likely that yaw control makes more harm than good if the atmospheric conditions are too much on the unstable side. In stable conditions, on the other hand, the wind field and hence the wakes tend to be more regular and predictable and the prospects of yaw control seem to be much better. Fuga does take stability into account, but there has been numerical problems for the solver in stable conditions. Making an attempt to overcome these difficulties is the secondary aim of this work. Including (strongly) stable conditions is necessary for the study of yaw control in the situations where is looks most promising, and the fact that the present solver fails for stable conditions is a major obstacle for progress.

A SHORT DESCRIPTION OF FUGA

2.1 GOVERNING EQUATIONS

The starting point for Fuga is a full, non-linear CFD model. Any CFD model, which is based on the Boussinesq approximation and an eddy viscosity closure, can be used. In such a setting the Reynolds averaged Navier-Stokes equation may look like this:

$$U_j \frac{\partial U_i}{\partial x_j} = \frac{\partial}{\partial x_j} K \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{\partial P}{\partial x_i} + f_i \tag{1}$$

where U_i is the mean velocity, P is the pressure divided by the (constant) density, K is the eddy viscosity (molecular viscosity is neglected), and f_i is the external forcing from one or more actuator discs representing the drag forces exerted by turbines. For the sake of simplicity the buoyancy term $-g\,\theta\,\delta_{3i}/T$ and the corresponding transport equation for potential temperature θ have been neglected. This means that standing gravity waves are not supported, but turbulent mixing can still be affected by stability through the eddy viscosity.

Incompressibility yields the continuity

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{2}$$

Boundary conditions are just as important as governing equations. Assuming a 'lid driven' flow with an imposed velocity U_i^{lid} at the top of the boundary layer and rough lower boundary, the boundary conditions are

$$\begin{aligned} U_i(z_0) &= 0 \\ U_i(z_i) &= U_i^{\mathrm{lid}} \end{aligned} \tag{3}$$

where z_0 is the roughness length and z_i is the boundary layer height (the i in z_i stands for capping $i$nversion, not component index). Specification of the momentum flux u_*^2 at the lid is an alternative, but in practice the choice is not so important.

Finally, the eddy viscosity K has to be determined by means of a closure scheme. In report1 three different closures were studied for neutral conditions: the standard k-ϵ model, the mixing length closure and the 'simple' closure, where $K = \kappa\, u_*\, z$. Somewhat surprisingly, the simple closure performed better than the other two and has been used subsequently. The simple closure totally neglects any feed-back mechanism that modifies turbulent mixing (of momentum) in the presence of wakes. The two other models are supposed to do that, but not with much success, apparently. At the time of this investigation it was realized that the (non-linear) standard k-ϵ model yields pure results for wakes, and a modified k-ϵ model (i.e. Paul's model [3]) is necessary in order to beat the performance of the simple model. It is therefore possible that a better choice than the simple closure exists. Investigating this is, however, outside the scope of the present work. In Fuga the simple closure is extended to non-neutral conditions by means of Monin-Obukhov theory setting

$$K = \frac{\kappa\, u_*\, z}{\varphi_m(z/L)} \tag{4}$$

where L is the Monin-Obukhov length and φ_m is the profile function,

2.2 Linearization

The linearization is a result of a perturbation expansion using the drag force as the perturbation. There is no unique way of doing this, so we shall first concentrate on how this has been done in Fuga. In the present version of Fuga the drag force f_i is formally replaced by $s\, f_i$ where s is a 'small' parameter. The solution then depends on s, and we may express this dependence by writing U_i and P as Taylor series:

$$\begin{aligned} U_i &= U_i^0 + s\, u_i^1 + s^2\, u_i^2 + s^3\, u_i^3 + \cdots \\ P &= P^0 + s\, p^1 + s^2\, p^2 + s^3\, p^3 + \ldots \end{aligned} \tag{3}$$

Inserting this into (1) and (2), the n^{th} order equations are obtained by applying $\frac{\partial^n}{\partial s^n}$ on both sides and setting $s = 0$. In other words, the n^{th} order equation is obtained by balancing all terms proportional to s^n. Thus the 0^{th} order equations are

$$\begin{gathered} U_j^0 \frac{\partial U_i^0}{\partial x_j} = \frac{\partial}{\partial x_j}\, K \left(\frac{\partial U_i^0}{\partial x_j} + \frac{\partial U_j^0}{\partial x_i} \right) - \frac{\partial P^0}{\partial x_i} \\ \frac{\partial U_i^0}{\partial x_i} = 0 \\ U_i^0(z_0) = 0 \\ U_i^0(z_i) = U_i^{\text{lid}} \end{gathered} \tag{4}$$

This is just the same as (1), (2) and (3) without any external forcing. Assuming that the mean pressure gradient vanishes we can set P^0=0, and the solution is the usual Monin-Obukhov velocity profile (the familiar logarithmic profile for neutral conditions).

The first order equations are

$$
\begin{gathered}
U^0 \frac{\partial u_i^1}{\partial x} + w^1 \frac{\partial U^0}{\partial z} \delta_{i1} = \frac{\partial}{\partial x_j} K \left(\frac{\partial u_i^1}{\partial x_j} + \frac{\partial u_j^1}{\partial x_i} \right) - \frac{\partial p^1}{\partial x_i} + f_i \\
\frac{\partial u_i^1}{\partial x_i} = 0 \\
u_i^1(z_0) = 0 \\
u_i^1(z_i) = 0
\end{gathered} \tag{5}
$$

where we have assumed the approaching flow is along the x-axis so that we have set $U_i^0 = \delta_{i1} U^0(z)$.

The first order equations are most conveniently solved in a mixed-spectral setting. This means that the equations are Fourier transformed in the horizontal x and y coordinates, so that solutions depend on z and a 2D wave vector $\boldsymbol{k} = (k \cos\beta, k \sin\beta)$. The mixed spectral equations are ordinary differential equations in the z variable.

Note that there is one separate set of equations for each $\boldsymbol{k}$ and no coupling between equations for different $\boldsymbol{k}$s. This decoupling is the big advantage of the mixed-spectral formulation. Instead of having coupled equations for millions of variables, four for each grid point, the mixed-spectral setting only has four equations and four variables.

In order to make the notation tidy we will write u, v, w and p rather than u_1^1, u_2^1, u_3^1 and p^1 in the following.

Finally we use (6d) to eliminate derivatives of w in (6c) and introduce two new variables u' and v' that enable us to write the equations as 6 first order equations, viz

$$\frac{\partial u}{\partial z} = u' \tag{7a}$$

$$
\begin{aligned}
\frac{\partial u'}{\partial z} = (k^2 + \frac{U^0 i\, k \cos\beta}{K}) u - \frac{1}{K} \frac{\partial K}{\partial z} u' + \left(\frac{1}{K} \frac{\partial U^0}{\partial z} - \frac{\partial K}{\partial z} \frac{i\, k \cos\beta}{K} \right) w \\
+ \frac{i\, k \cos\beta}{K} p - \frac{f_1}{K}
\end{aligned} \tag{7b}
$$

$$\frac{\partial v}{\partial z} = v' \tag{7c}$$

(7d)

$$\frac{\partial v'}{\partial z} = (k^2 + \frac{U^0 i\, k \cos\beta}{K}) v - \frac{1}{K}\frac{\partial K}{\partial z} v' - \frac{\partial K}{\partial z}\frac{i\, k \sin\beta}{K} w + \frac{i\, k \sin\beta}{K} p - \frac{f_2}{K}$$

$$\frac{\partial w}{\partial z} = -i\, k \cos\beta\, u - i\, k \sin\beta\, v \tag{7e}$$

$$\frac{\partial p}{\partial z} = -2\frac{\partial K}{\partial z}\, ik \cos\beta\, u - Kik \cos\beta\, u' - 2\frac{\partial K}{\partial z} ik \sin\beta\, v - Kik \sin\beta\, v' - (U^0 ik \cos\beta + k^2 K) w + f_3 \tag{7f}$$

$$u(z_0) = 0, v(z_0) = 0, w(z_0) = 0, u(z_i) = 0, v(z_i) = 0, w(z_i) = 0 \tag{7g}$$

Setting $X_i = (u, u', v, v', w, p)$ and $F_i = (0, f_1, 0, f_2, f_3, 0)$, the equations can be written in the form

$$\frac{\partial X_i}{\partial z} = A_{ij} X_j + F_i \tag{8}$$

where A_{ij} and F_i are functions of z. Equations (7a-g) and (8) should be written in non-dimensional form. All velocities and K_i should scale with u_*, pressure should scale with u_*^2, and kz should be used instead of z as independent variable. The non-dimensional solutions then become functions of kz and three parameters: z_0, the angle β between the wave vector and the wind direction, and a stability parameter $\zeta_0 \equiv z_0/L$, say. Note that there is no additional, explicit dependence on k or u_* or z_0, so we solve the equations without actually knowing k, u_* and z_0. We will not rewrite the equations in non-dimensional form and invent new notation etc. Instead we keep the dimensional notation and note that the non-dimensional form is obtained very easily by replacing k with kz and setting k =1 and u_*=1.

2.3 The solver

It is a set of linear equations with boundary conditions both at $z = z_0$ and $z = z_i$. It may sound as an easy problem, but it is not. The NDSolve routine in Mathematica attacks the problem with the chasing method. This method first transforms boundary conditions at one boundary to an equivalent boundary conditions at the other boundary. With all boundary conditions at the same boundary we have an initial value problem which can be integrated using e.g. Runge-Kutta. The transformation works like this: we have a boundary condition at z_1

$$y_i^1\, X_i(z_1) = b^1 \tag{9}$$

which should be equivalent to

$$y(z)\, X_i(z) = b(z) \tag{10}$$

at some other point . The trick is first to solve the following initial value problem

$$\frac{\partial y_j}{\partial z} = -y_i A_{ij} \tag{11}$$

$$y_i(z_1) = y_i^1$$

Then

$$\frac{\partial y_i X_i}{\partial z} = -y_i A_{ij} X_j + y_i A_{ij} X_j + y_i F_i = y_i F_i \tag{12}$$

and therefore

$$b(z) = b^1 + \int_{z_1}^{z} y_i(z') F_i(z') dz' \tag{13}$$

Note that we don't have to know X_i to find y_i and b. In this way the three upper boundary conditions can be transformed into lower boundary conditions, say, and the problem can be integrated up from z_0 to z_i as an ordinary initial value problem.

Unfortunately the method often fails because the determination of the initial value turns out to be numerically ill-conditioned, and the integration cannot start. Even if the integration sometimes does start, there is no guarantee that numerical solution ends up obeying the original boundary conditions at the other end as it is supposed to. The problem is numerical precision. As the transportation from one end to the other progresses, the three boundary conditions tend to look more and more alike and extremely high numerical precision is needed to tell them apart. Investigations using Mathematica indicate that as much as 60 significant digits are sometimes needed. This is about twice as many as are available using quadruple precision floating-point reals.

A modified version of the chasing method was therefore used. The idea is to reformulate the transported conditions every now and then in order to separate them before running out of numerical precision. The integration start from below at $z = z_0$, where we have the three boundary conditions are $X_1 = X_3 = X_5 = 0$ (i.e. $u = v = w = 0$), each with a corresponding y_i and b (that happen to be =0). It is practical to use the ys as the first (upper) three rows in a 6x6 matrix $Y_{ij}(z_0)$. The three lower rows represent three additional conditions which are unknown at this point. We choose to define them in such a way as to make $Y_{ij}(z_0)$ unitary, e.g.

$$Y_{ij}(z_0) = \left\{\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}\right\} \tag{14}$$

The bs for the three lower rows are unknown, but according (13) to we can use b=0 as starting value and add the actual starting value later.

We now rewrite (11) in matrix form (which does not mix the rows of Y_{ij})

$$\frac{\partial Y_{ij}}{\partial z} = -Y_{ik} A_{kj} \tag{15}$$

$$\frac{\partial b_i}{\partial z} = Y_{ij}\, F_j$$
$$b_j(z_0) = 0$$

The forcing F_j can be divided into three components: Longitudinal acting against the approacing wind direction, transverse forcing (relevant for yawed rotors) and vertical forcing (not actually used here). It is an advantage to make three separate solutions which later can be scaled and superimposed to give the full forcing. There will therefore be a separate solution for each of the three choices

$$\begin{aligned} F_j(z) &= f_1(z)\,\delta_{j2} \\ F_j(z) &= f_2(z)\,\delta_{j4} \\ F_j(z) &= f_3(z)\,\delta_{j5} \end{aligned} \tag{16}$$

Thus there will be three sets of b_i and Y_{ij} for the three types. In principle $f_j(z)$ can be any function of z, but we will restrick $f_j(z)$ to be piecewize linear. To this end we define a number of levels

$$z_j = z_0\, e^{j\, ds} \tag{17}$$

between which $f_j(z)$ is approximated by a linear function. The value $ds = 0.05$ is small enough to give a good approximations. Integrating (13) from one level to the next we note that

$$\begin{aligned} b_i(z_j) &= b_i(z_{j-1}) + \Delta b_i \\ \Delta b_i &= \int_{z_j}^{z_{j+1}} Y_{ik}\,(z) F_k(z)\, dz \end{aligned} \tag{18}$$

where the forcing is linear in z. We therefore only need to calculate the integral for the special cases $f_j(z)$=1and $f_j(z)$=z and store the six results in a look-up table. The solution for more any linear $f_j(z)$ can then be obtained as a linear combination of the spatial cases. $Y_{ij}\,(z_j)$, which does not depend on the forcing, is also tabulated.

These tables can be used to find X_i for a particular, piecewise forcing in the following way. First the forcing $f_j(z)$ is worked out for the Fourier component in question. Using the Δb table we then find $b_i(z_j) - b_i(z_0)$ for all levels up to z_i. Here we know that $b_1(z_0) = b_2(z_0) = b_3(z_0)$ =0, hence $b_1(z_j)$, $b_2(z_j)$ and $b_3(z_j)$ are all known, but $b_4(z_0)$, $b_5(z_0)$ and $b_6(z_0)$ are unknow. They can, however, be determined because at the upper boundary we know six conditions: $Y_{ij}X_j = b_i$ for $i = 1$, 2 and 3 plus the three upper boundary conditions. With six conditions we have enough information to determine the X_j and the remaining $b_i = Y_{ij}X_j$ and hence also $b_i(z_0)$ for $i = 4$, 5 and 6. Now having six conditions at every level, all $X_j(z_j)$ can be calculated.

In the original chasing algorithm $X_j(z)$ is found by solving the initial value problem starting from $X_j(z_i)$ and integrating downwards. This is not a good idea here because particular solutions increase or decrease roughly exponentially and a tiny bit of numerical noise make the solutions go berserk so there is chance that we end up with the original lower boundary conditions $u(z_0) = v(z_0) = w(z_0) = 0$. The proposed solutions method is better because the lower boundary conditions will automatically be fulfilled.

That should work *in principle*, but, unfortunately, it does not work at all in practice. What happens is that the matrix Y_{ij} becomes ill-conditioned (having a mixture of very small and very large eigenvalues) and cannot be inverted without major loss of numerical precision. However, we can avoid this if the condition $Y_{ij}X_j = b_i$ at each level is modified. Another, equivalent condition $\tilde{Y}_{ij}X_j = \tilde{b}_i$ where $\tilde{Y}_{ik} = M_{ij}Y_{jk}$ and $\tilde{b}_i = M_{ij}b_j$. The matrix M_{ij} should be regular, it should not mix the three upper rows with the lower three (i.e. $M_{ij} = 0$ for $i \leq 3$ and $j > 3$), and $\tilde{Y}_{ij}$ should be well-conditioned. This is achieved by requiring that M_{ij} to be lower triangular ($M_{ij} = 0$ for $i \leq j$) and $\tilde{Y}_{ij}$ to be unitary. Unitary matrices are as well-conditioned as they can be, and they are also extremely easy to invert (the same goes for triangular matrices).The inverse of a lower triangular matrix is also lower triangular, so the procedure is to write Y_{ij} as a product of a lower triangular matrix (the inverse of M_{ij}) and a unitary matrix $\tilde{Y}_{ij}$. The so-called QR decomposition, which essentially is a Gram-Schmidt orthonormalization, can be used. The integration of $Y_{ij}(z)$ then uses the well-conditioned $\tilde{Y}_{ij}$ as start value andchances are that $Y_{ij}(z)$ s remains reasonably well-conditioned up to the next level, where the next decomposition takes place. Otherwise the step is too large and must be divided into smaller steps with QR decompositions in between. The look-up tables contain $\tilde{b}_i$, $\tilde{Y}_{ij}$ and M_{ij} for each level. We need to store M_{ij} because of (18) is replaced with

$$\begin{aligned} \tilde{b}_i(z_j) &= M_{ik}b_k(z_{j-1}) + \Delta\tilde{b}_i \\ \Delta\tilde{b}_i &= M_{ip}\int_{z_j}^{z_{j+1}} Y_{pk}(z)F_k(z)\,dz \end{aligned} \qquad (19)$$

The modified chasing algorithm outlined above seems to work, even if there are some issues. It gets rid of ill-conditioned matrices, and produces wakes that look ok. It is definitely doing better than the state-of-the-art solver NDSolve in Mathematica, which simply fails, but it is difficult to say how accurate the solutions are because we lack an even better solver to compare with. Spurious behaviour has been observed, however, as can be seen in fig. 1. The figure shows the park efficiency for a given wind climate assuming different values of the stability parameter $\zeta_0 = z_0/L$ corresponding to stabilities ranging from very unstable to very stable conditions. The efficiency decreases with increasing ζ_0 as expected, but only up to $\zeta_0 \approx 3 \times 10^{-7}$ where it begins to increase. There is no reason to believe that the rise at strong stability is a correct behavior.

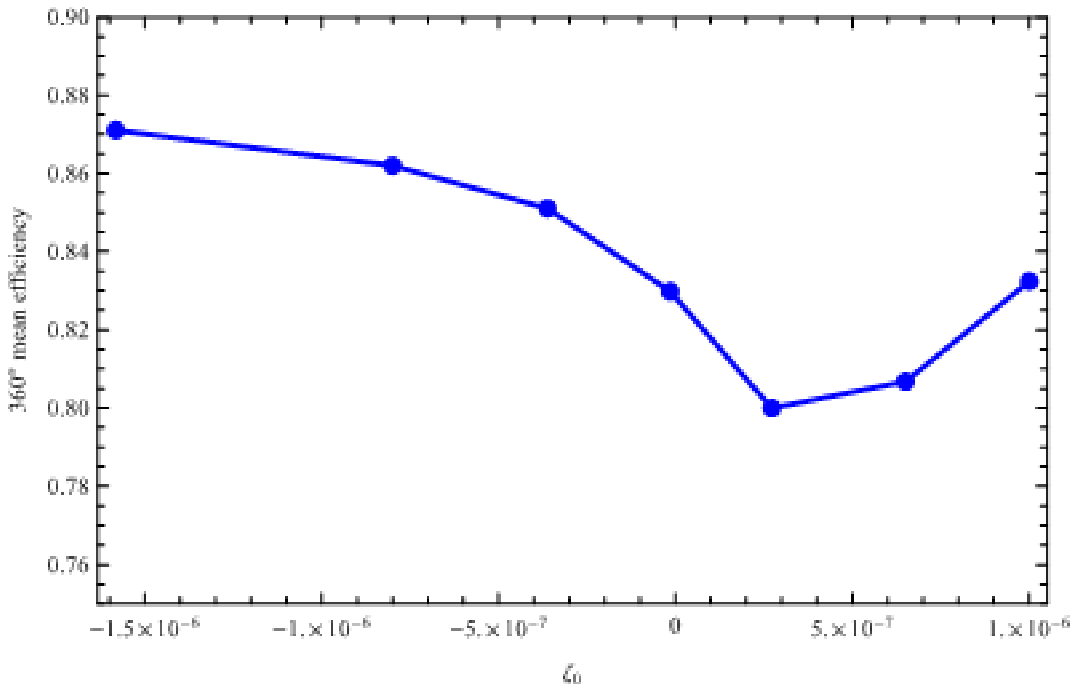

Fig 1. Top: Park efficiency as a function of $\zeta_0 = z_0/L$ using the present solver. The rise in the stable side is spurious.

It is difficult to pinpoint exactly what goes wrong. Diffusion tends to wipe out numerical noise and inaccuracy, so it is perhaps not surprising that things go wrong in the stable end where K is smallest. When the eddy viscosity is small, the spurious forces resulting from numerical inaccuracies can more easily perturb the solution.

Various attempts have been made to improve the solver further, most of which led to catastrophic results. The best bit is the following.

Major simplifications occur K in equations (7a-f) if we assume that U^0 and K are constants. Discarding all terms involving derivatives of U^0 and K we end up with a constant matrix A_{ij} of the form

$$A_{ij} = \left\{ \begin{array}{cccccc} 0 & 1 & 0 & 0 & 0 & 0 \\ q^2 & 0 & 0 & 0 & 0 & i\,\cos\beta/K \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & q^2 & 0 & 0 & i\,\sin\beta/K \\ -i\,\cos\beta & 0 & -i\,\sin\beta & 0 & 0 & 0 \\ 0 & -i\,K\,\cos\beta & 0 & i\,K\,\sin\beta & 0-K\,q^2 & 0 \end{array} \right\} \tag{20}$$

where

$$q^2 = k^2 + i\,k\frac{U^0\cos\beta}{K} \tag{21}$$

The interesting thing is that analytical solutions to (8) and (13) exist for this constant matrix. The solution to (13) is $Y(z) = Y(z_0)\, e^{-A\,(z-z_0)}$ where the matrix exponential $e^{-A\,(z-z_0)}$ can be written (by Mathematica) as a relatively compact, analytical expression. It is still several pages long so there is no need write it here.

This can be used to make approximate solutions to the integration from one level to the next using values of U^0 and K representative for the given atmospheric stability. K is therefore not really constant, but rather piecewise constant. In order for the flux $K\,\partial U^0/\partial z$ to be continuous, $\partial U^0/\partial z$ must also be piecewise constant and hence U^0 must be continuous and piecewise linear, not piecewise constant as we assume. However, the difference should be small when the levels are closely spaced,

When crossing a level z_j, where K and U^0 jump, the fluxes Ku' and Kv' should be continuous so that u', v' need to make appropriate jumps while u, v and w are simply continuous. The pressure also jumps due to the δ function generated by $\partial K/\partial z = \Delta K\ \delta(z - z_j)$ in equation (7f). These adjustments can be expressed as a matrix operating on $X_i \to X_i + B_{ij}X_j$ each time the integration of level (8) passes a level. We are actually integrating (15) where the corresponding operation should be $Y_{ij} \to Y_{ij} - Y_{ik}B_{kj}$.

The hope has been that the use of analytical expressions would help minimizing the generation of numerical noise and make the solver more robust. Unfortunately there has been some problems with the implementation of the method in Mathematica which are not yet solved. The eigenvalues of A_{ij} are $\pm k$ and $\pm q$. For neutral and unstable conditions $U^0/K \to 0$ for $z \to \infty$, but for stable conditions we have $U^0/K \to \infty$ for $z \to \infty$. The problems seem to be related to this.

YAWED WAKES AND LINEARIZATION

Linearized models work surprisingly well for wakes from rotors that are not in yaw. The transverse component of the drag force from a yawed rotor can be treated in a similar way as the longitudinal component. The linear responses to each of the two force components can be calculated separately and the total linear response (in terms of wake deficit) is just their sum. This is all very convenient, but the response is only truly linear for an infinitesimal drag force, and the question is how far away we are from a linear response regime. In order to figure this out we made a numerical experiment using Paul van der Laan's model [3]. The set-up is a single actuator disk placed in flat, rough terrain (z_0=3.52cm) and neutral conditions. The rotor is yawed 20 degrees with respect to the incoming wind, and the thrust is uniformly distributed over the actuator disk and normal to it. The total thrust is fixed to

$$T = \frac{1}{2} C_T\, \rho\, U_0^2\, \pi\, R^2 \tag{22}$$

where C_T is the thrust coefficient, U_0 is the approaching wind speed at hub neight (90m), ρ is the air density and R=63m is the rotor radius. The exercise consists in keeping everything fixed except the thrust coefficient C_T. Runs were made using the values C_{T}=0.8, 0.4, 0.2, 0.1 and 0, and the velocity deficit was calculated by subtracting results for C_{T}=0 from the other results. For C_{T} =0 there is of course no wake and the obtained velocity profile is logarithmic with tiny deviations, probably resulting from the zooming nature of the grid. These deviations are likely to cancel out when the deficit is calculated as the difference between two numerical solutions.

The detailed thrust distribution is probably not very important except near the rotor. Further away, where the wake has spread out, the wake will only 'remember' the total force. This could justify a simplified forcing where only the force vector is turned, while the actuator disk remains perpendicular to the incoming wind direction. For numerical reasons the force field is also smeared out so that it is not concentrated on an infinitely thin disk. With this setup only two cases, one for longitudinal and one for transverse forcing, are needed to cover all yaw angles. Note that the wake deficit ΔU is a left-right (spanwise) symmetric for longitudinal forcing and antisymmetric for transverse forcing.

Results are presented in figs 2-5 which show cross-wind profiles of $\Delta U/(U_o\ C_T)$ at hub height for various downwind distances and C_T values. For a linear model $\Delta U/C_T$ should be independent of independent of C_T and all curves should collapse. Judged from the top plots there is a decent collapse except for the $C_T = 0.8$ curve which lies a bit higher than the rest except for the $x = 12D$ case where it is lower. There is also clear tendency for the wakes to shift more to the right as C_T increases. For the symmetric part the collapse is also fair, but for the anti-symmetric part there is no collapse at all except for $x = 0D$ and in the tails. Fig. 6 shows a nice collapse in the tails indicating that the response is indeed linear in the region outside of the main wake. Inside the wake, on the other hand, there is no collapse at all. The dashed yellow curve is an extrapolation to C_T=0 based on a polynomial fit to the data, and is thus an estimate of the linear response. It looks completely different with opposite sign compared to the rest of the curves. It is also very small and actually shifts the wake in the wrong direction.

The lesson to learn from the numerical experiment is that the symmetric part of the wake responds fairly linearly while the anti-symmetric part is more complicated and even for C_T = 0.1 the higher order terms dominate. This is bad news for Fuga which calculates the antisymmetric part of the wake from the linear response to transverse forcing.

The observation that the shifting of the wake is a second order effect should not come as a surprise (even if it actually did). Linearization basically consists in dropping the non-linear term $\Delta\boldsymbol{u}\cdot\nabla\,\Delta\boldsymbol{u}$. This means that the perturbation is not allowed to advect itself, and without this self-interaction the wake is unable to drift to the side as it moves downwind. We can also observe that the shape of the deficit profiles is approximately self-similar at downwind distances of more than a few D, but shifted to the side so that

$$\Delta u \sim A\,F\left(\frac{y-\lambda}{\sigma}\right) \tag{23}$$

where A, λ and σ are functions of x, z and C_T. F is of approximately Gaussian shape and we may use the maximum deficit Δu_{max} as an estimate of A. Fig. 7 shows that Δu_{max} is proportional to C_T up to at least C_T=0.4.

Both A and λ vanish for C_T=0 while σ is positive and their Taylor expansions therefore look like this

$$\begin{aligned} A &= \sum_{n=1}^{\infty} A_n\, C_T^n \\ \lambda &= \sum_{n=1}^{\infty} \lambda_n\, C_T^n \\ \sigma &= \sum_{n=0}^{\infty} \sigma_n\, C_T^n \end{aligned} \tag{24}$$

hence

$$\Delta u \sim A_1 F\left(\frac{y}{\sigma_0}\right) C_T + \left\{A_2 F\left(\frac{y}{\sigma_0}\right) + A_1 F'\left(\frac{y}{\sigma_0}\right)\left(\frac{y\,\sigma_1}{\sigma_0^2} - \frac{y\lambda_1}{\sigma_0}\right)\right\} C_T^2 + \cdots \tag{25}$$

Here we observe that there is no λ in the first term and hence no shift to first order.

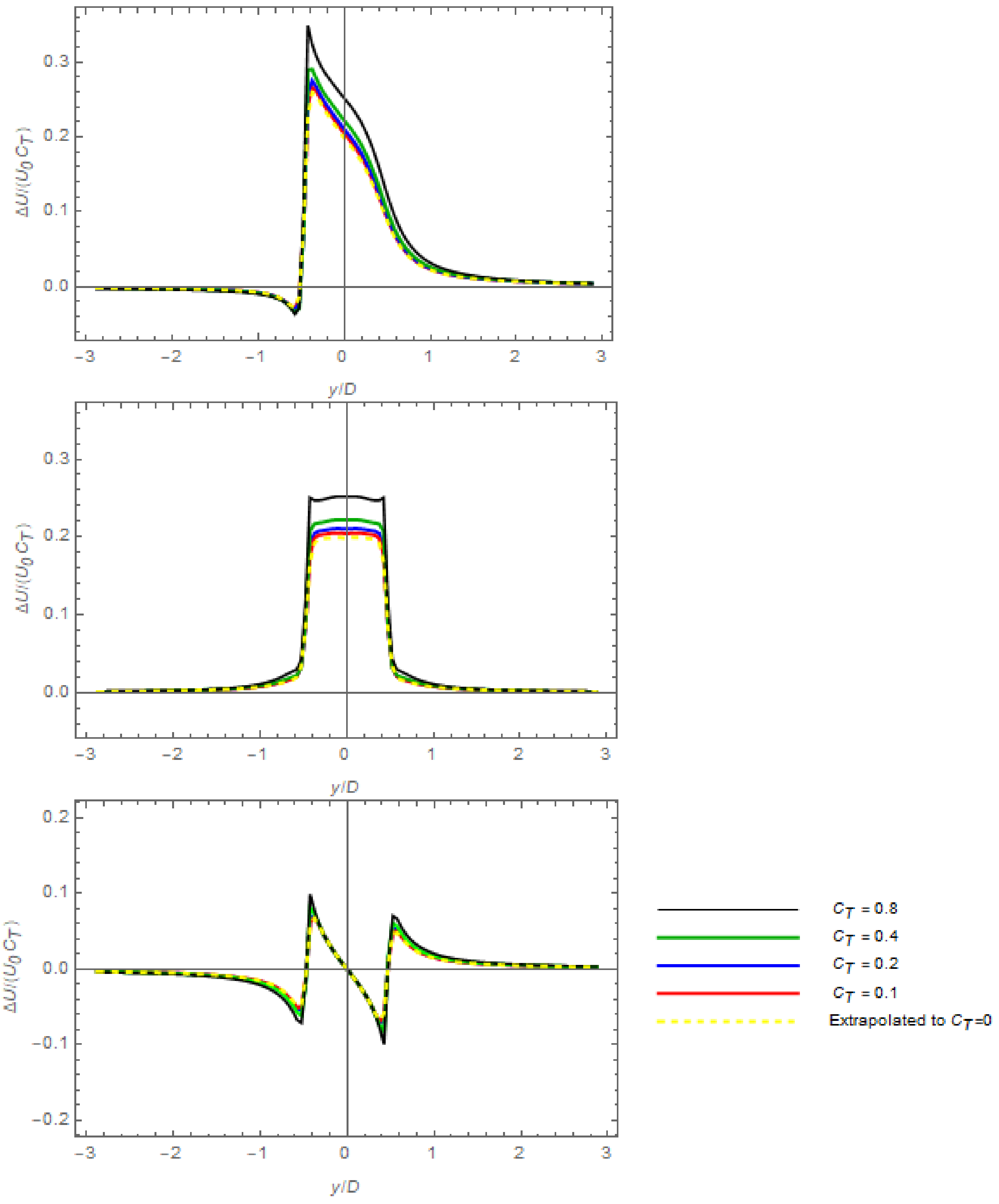

Fig 2. Top: $\Delta u/(U_o\ C_T)$at hub height 0D downwind of the rotor. Middle: the left-right symmetric part of $\Delta u/(U_o\ C_T)$. Bottom: antisymmetric part of $\Delta u/(U_o\ C_T)$.

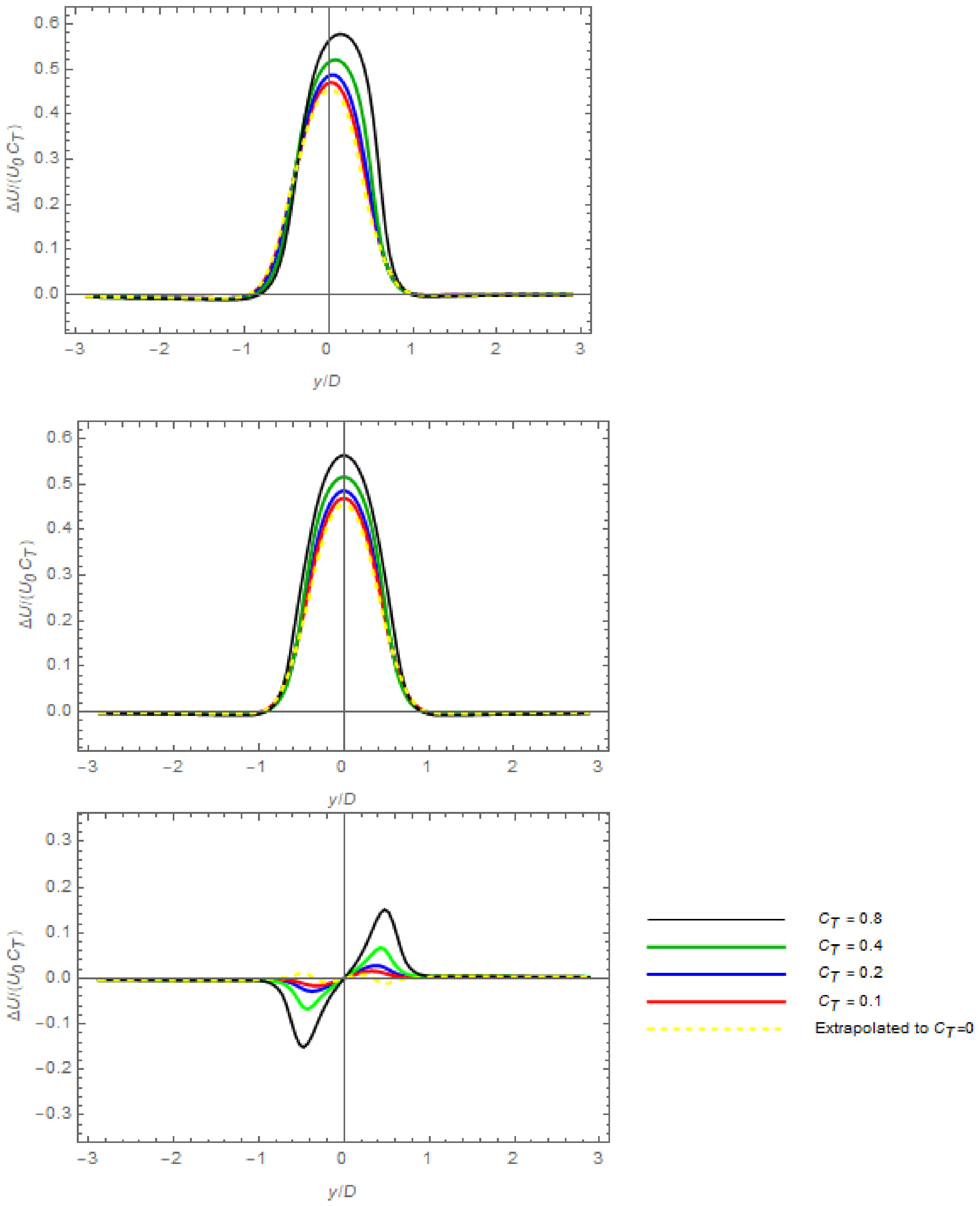

Fig 3. Top: $\Delta u/(U_o\ C_T)$ at hub height 2D downwind of the rotor. Middle: the left-right symmetric part of $\Delta u/(U_o\ C_T)$. Bottom: antisymmetric part of $\Delta u/(U_o\ C_T)$.

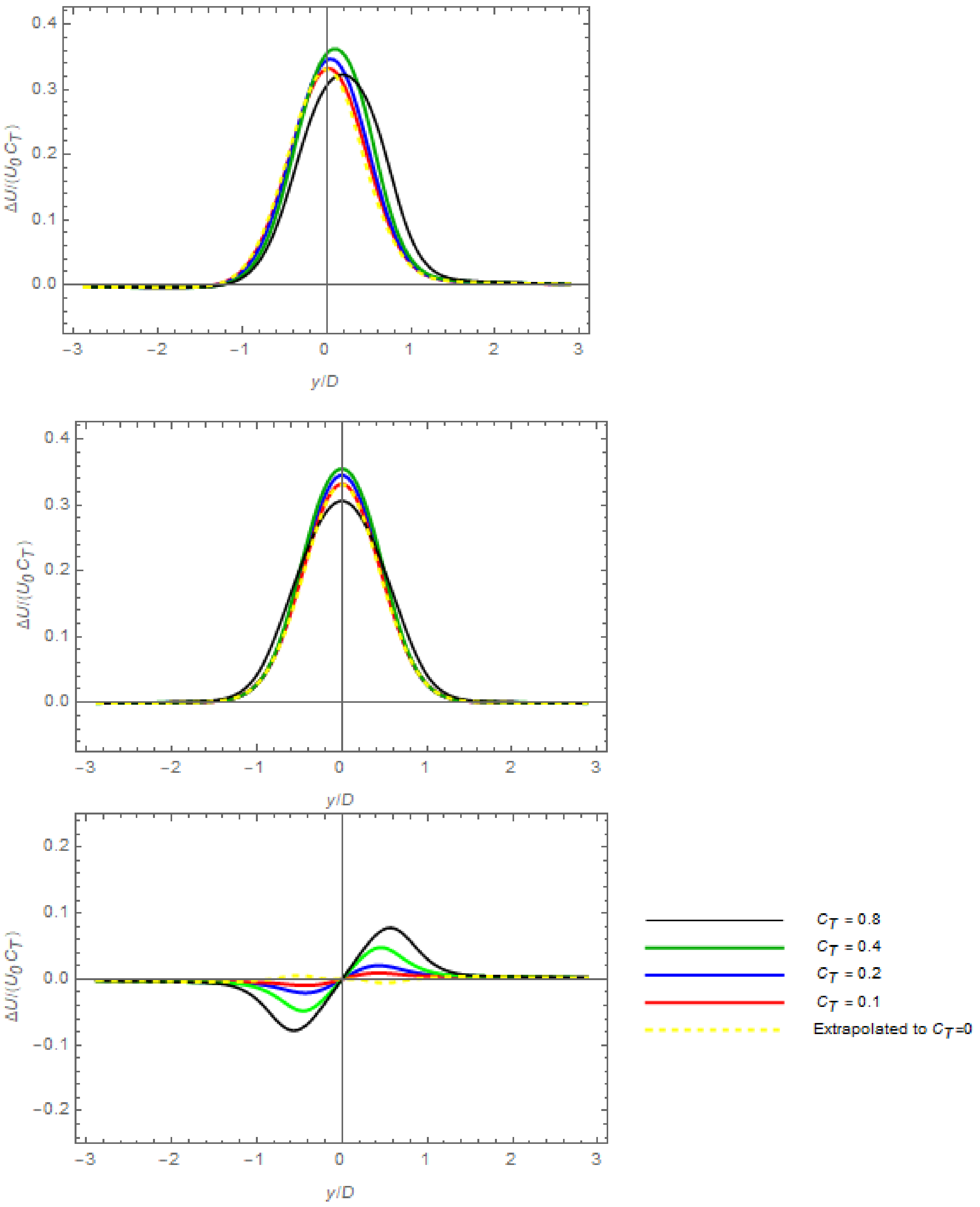

Fig 4. Top: $\Delta u/(U_o\ C_T)$ at hub height 5D downwind of the rotor. Middle: the left-right symmetric part of $\Delta u/(U_o\ C_T)$. Bottom: antisymmetric part of $\Delta u/(U_o\ C_T)$.

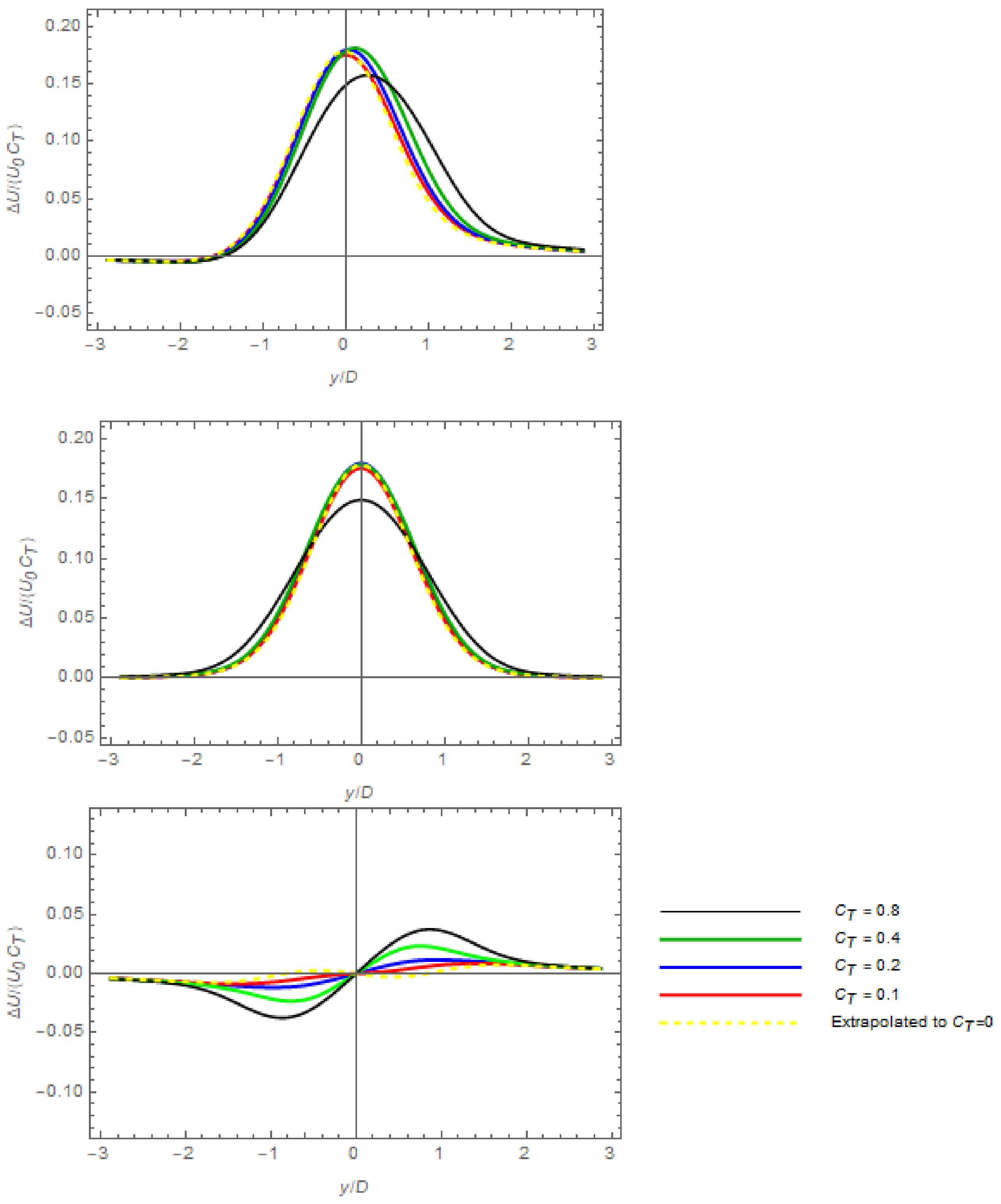

Fig 5. Top: $\Delta u/(U_o\ C_T)$ at hub height 12D downwind of the rotor. Middle: the left-right symmetric part of $\Delta u/(U_o\ C_T)$. Bottom: antisymmetric part of $\Delta u/(U_o\ C_T)$.

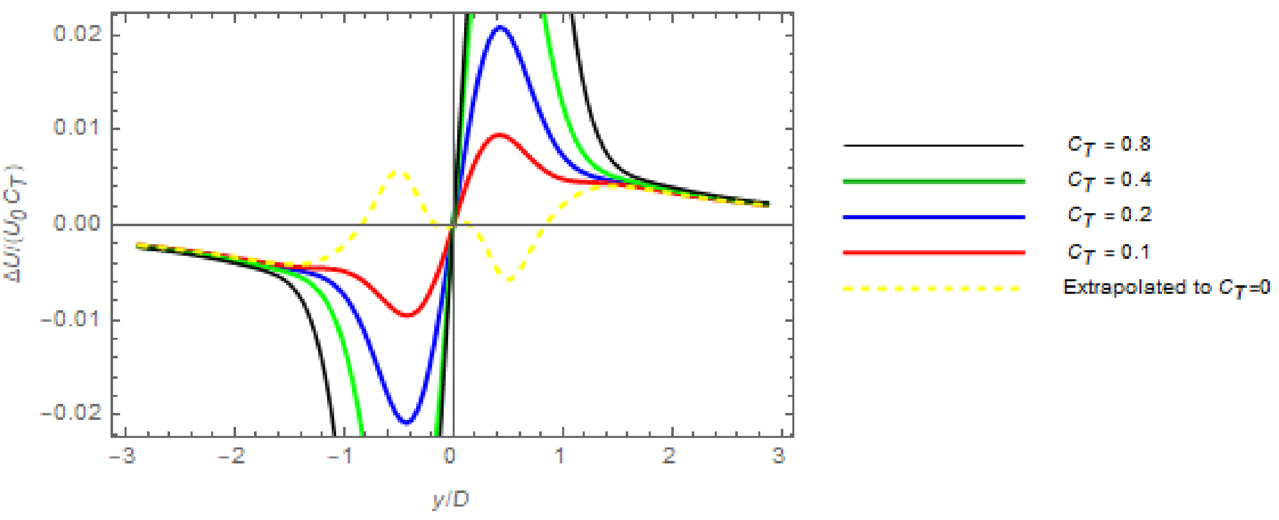

Fig 6. The same as fig 4 (middle) except that the ordinate axis has been stretched.

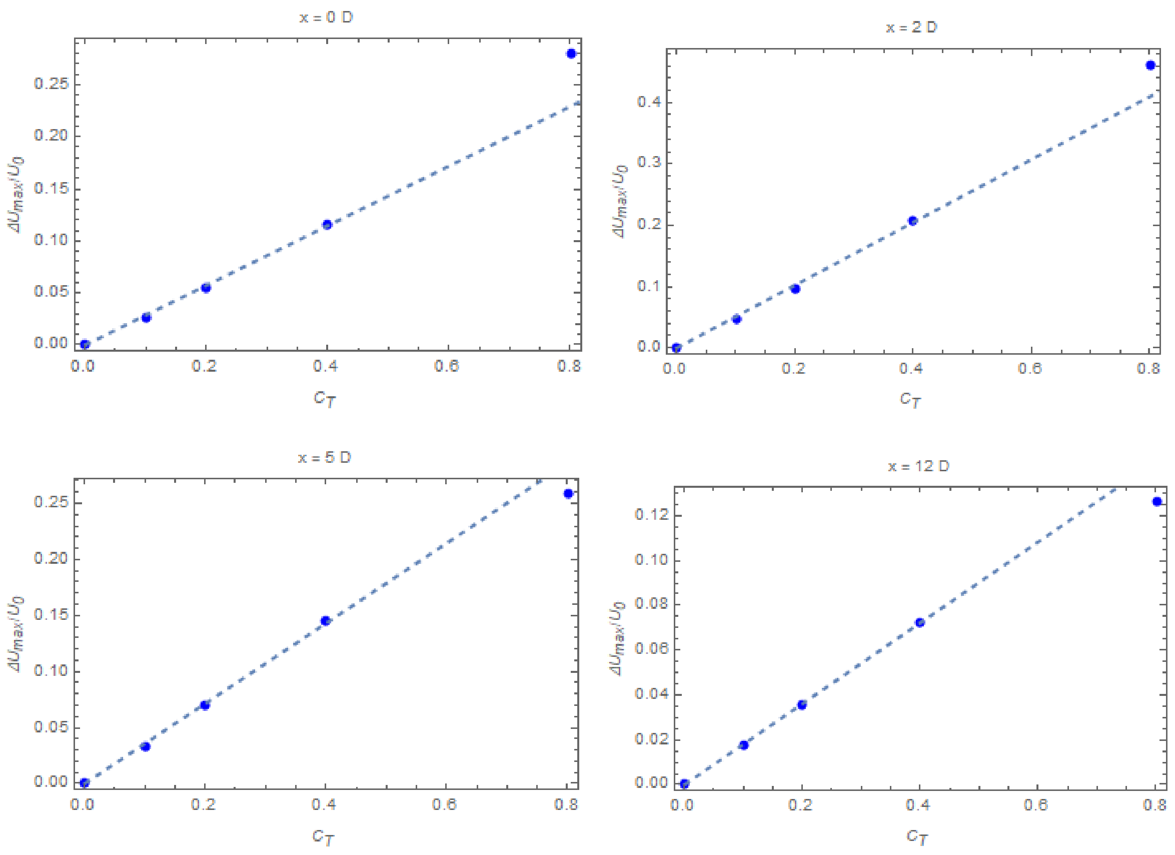

Fig 7. Plots of $\Delta u_{max}/U_o$ against C_T for x/D = 0, 2, 5 and 12.

A NEW LINEARIZATION TECHNIQUE

The last section revealed that wake deflection is a second order effect. This is a serious problems for linearized models if they are to cope with yawed rotors. In this section we will propose a way out of the dilemma.

The idea is to use new coordinates $\acute{x}_\iota$ that are shifted in the transverse direction (along the y-axis)

$$\begin{gathered} \acute{x} = x \\ \acute{y} = y - \lambda \\ \acute{z} = z \end{gathered} \tag{26}$$

In other words, the perturbation expansion is for the problem where f_i is replaced with sf_i and the shift λ is a functions of s, x, y and z. We can choose smooth function as long as $\lambda \to 0$ for $s \to 0$ and the coordinate transformation $x_i \to \acute{x}_\iota$ is one-to-one. We will return to the choice of λ.

The governing equations (1) and (2) should be written in new coordinates, which amounts to replacing derivatives using the chain rule

$$\frac{\partial}{\partial x_i} = \frac{\partial \acute{x}_j}{\partial x_i}\frac{\partial}{\partial \acute{x}_j} = \frac{\partial}{\partial \acute{x}_i} - \frac{\partial \lambda}{\partial x_i}\frac{\partial}{\partial \acute{y}} = \frac{\partial}{\partial \acute{x}_i} - \frac{\frac{\partial \lambda}{\partial \acute{x}_i}}{1+\frac{\partial \lambda}{\partial \acute{y}}}\frac{\partial}{\partial \acute{y}} \sim \frac{\partial}{\partial \acute{x}_i} - \frac{\partial \lambda}{\partial \acute{x}_i}\frac{\partial}{\partial \acute{y}} \tag{27}$$

The first order equations will then contain the same terms as before with $\partial/\partial\acute{x}_i$ replacing $\partial/\partial x_i$ and extra terms where $\partial/\partial x_i$ is replaced with - $\partial\lambda/\partial\acute{x}_i\,\partial/\partial\acute{y}$. However, since λ is first order, $\partial\lambda/\partial\acute{x}_i$ must operate on a zero order term in order to yield a first order term, but they only depend on $\acute{z}$, not on $\acute{y}$. Therefore there are no extra terms and the linearized equations therefore look exactly as before except for the accents on $\acute{x}_i$. It is natural to require λ=0 on the actuator disk so that it stays circular in the new coordinate system. The problem is therefore exactly the same as before we made the coordinate transformation. The solution is therefore the same as before except that we have to decide which coordinate system it refers to.

We shall choose λ in such a way that $\acute{y}$ does not change along a streamline. This eliminates wake deflection in the new coordinates system (it will of course reappear when transforming back to old coordinates). The requirement means that

$$0 = U_i\frac{\partial \acute{y}}{\partial x_i} = U_i\left\{\frac{\partial \acute{y}}{\partial \acute{x}_i} - \frac{\frac{\partial \lambda}{\partial \acute{x}_i}}{1+\frac{\partial \lambda}{\partial \acute{y}}}\frac{\partial \acute{y}}{\partial \acute{y}}\right\} = V - \frac{U_i\frac{\partial \lambda}{\partial \acute{x}_i}}{1+\frac{\partial \lambda}{\partial \acute{y}}} \tag{28}$$

Since λ is first order (or higher) so the first order approximation to (24) is simply

$$U^0 \frac{\partial \lambda}{\partial \acute{x}} = v \tag{29}$$

so that

$$\lambda(\acute{x}, \acute{y}, \acute{z}) = \frac{1}{U^0(\acute{z})} \int_{\acute{x}_o}^{\acute{x}} v(\tilde{x}, \acute{y}, \acute{z})\, d\tilde{x} \tag{30}$$

where $(\acute{x}_o, \acute{y}, \acute{z})$ is a point in the rotor plane. The integral is easy to do numerically. It yields λ as a function of $\acute{y}$, but this can easily be converted to a table of $\lambda(x, y, z)$ values which it is more convenient since.

Fig, 7 shows the new coordinate system for a case similar to the one studied in the last section. The exaggerated value C_T=2 was used for clarity. It is comforting to see that lines of constant $\acute{y}$ do not cross indicating that the transformation is indeed one-to-one. The highest possible value is C_T=2.18 where $\partial\lambda/\partial\acute{y} = -1$ at a point and the denominator in (23) vanishes.

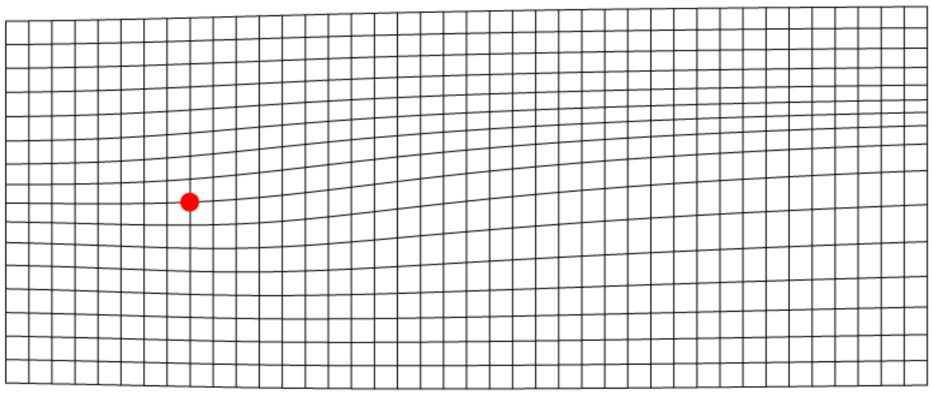
Fig 7. The new coordinate system illustrated by lines of constant $\acute{x}$ and $\acute{y}$ at hub height. The spacing is D /4 and the red dot is the hub. The extreme value C_T=2 was used the make the distortion more visible.

The output from Fuga is a table containing the normalized wake deficit $u/(C_T U^0(Z_{\text{hub}}))$ for a single, solitary rotor, and multi wake solutions are obtained by translating, scaling and superimposing instances of the tabulated single wake. In the new scheme there is an extra table containing λ/C_T which is used to tell how much to shift y before looking up in the wake table. Even if linear theory was used to derive these tables, the combined use of them yields the appropriate, non-linear result.

Equation (25) is for a single wake. When more than one turbine is present, λ should be obtained as the sum of contributions from each of them. This means that the shapes of actuator disks that are hit by wakes will be distorted in the transformed coordinates, which leads to all kinds of complications. In order to keep things simple, we propose to simply superimpose the wake deficits as functions of (x, y, z) even if superposition is only allowed for functions of $(\acute{x}, \acute{y}, \acute{z})$.

Fig. 8 shows the difference between the linear model and the new linear model.

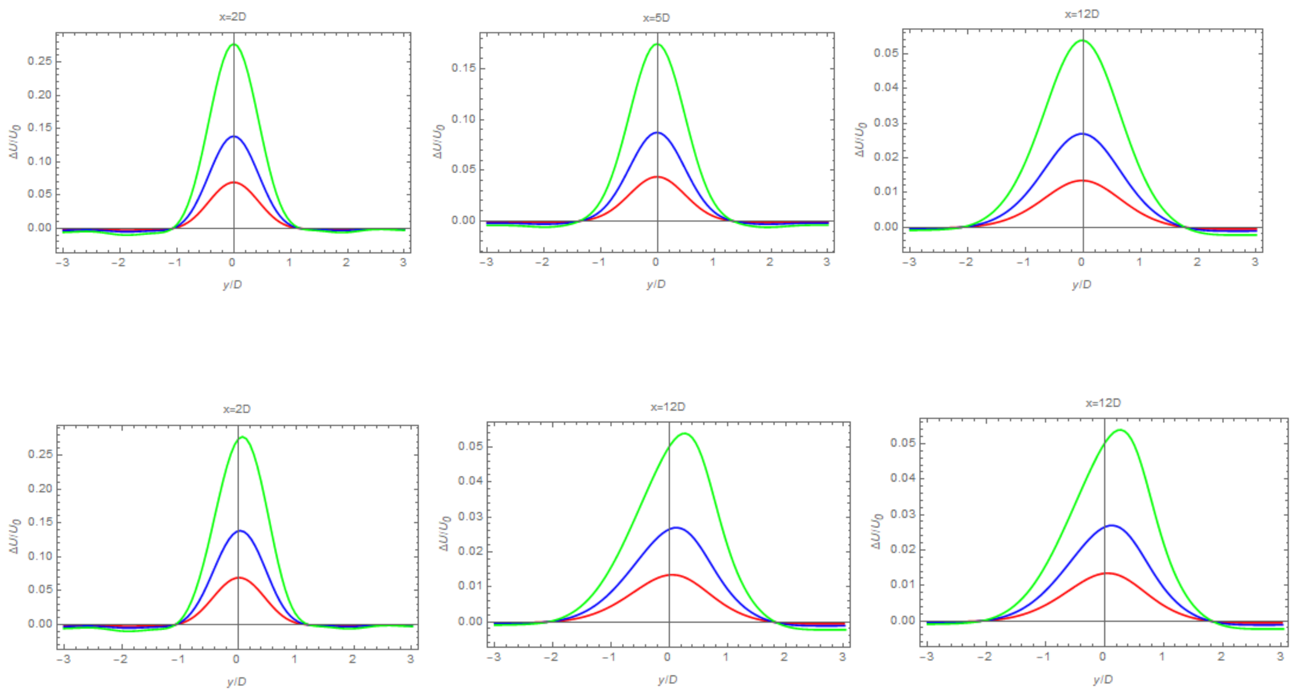

Fig 8. Velocity deficit profiles at hub height for C_T=0.2 (red), C_T=0.4 (blue) and C_T=0.8 (green) at three distances x=0.2 D, x=0.4 D and x=0.8 D, Top: conventional, linear theory. Bottom: the new method.

REFERENCES

1. Ott S, Berg J and Nielsen M (2011). Linearized CFD models for wakes. Risø-R-1772(EN). Risø National Lab. for Sustainable Energy, 39p.

2. Ott S and Nielsen M (2014). Developments of the offshore wind turbine wake model Fuga. DTU Wind Energy E-0048, 89p.

3. van der Laan MP, Sørensen NN, Réthoré P-E, Mann J, Kelly MC and Troldborg N (2015). The k- ε - fp model applied to double wind turbine wakes using different actuator disk force methods. Wind Energy, vol. 18, no. 12, pp. 2223–2240.