Comparison of steady-state analytical wake models implemented in wind farm analysis software

Rafael Mudafort[1], **Julian Quick**[2] **and Jonas Schulte**[3]

[1]National Renewable Energy Laboratory, Golden, CO, USA
[2]Technical University of Denmark, Roskilde, DK
[3]Fraunhofer Institute for Wind Energy Systems, Oldenburg, DE

E-mail: `rafael.mudafort@nrel.gov`

Abstract. A common set of mathematical wind turbine wake models are implemented in a few, well-adopted computational tools for wind farm wake modelling. Although the referenced mathematical formulations are common, implementation details may lead to differences in results. This study presents a systematic comparison of the implementation of mathematical wake models in open source, Python-based wind turbine wake modelling software, and a set of the models are directly compared. Despite aligning only the mathematical model parameters and retaining the default computational model parameters, good agreement is found across most of the model implementations, and additional agreement is expected upon further parameters alignment.

1 Introduction

With growth in the size and number of wind farms installed globally, mitigating loss in energy production due to turbine-turbine interactions through their wakes has become a major field of study with significant potential for increased efficiency and profitability [1]. Because of the physical scales that characterize wind turbine wakes, the possibility to understand these effects and develop practical methods to mitigate them directly depends on the accuracy of numerical tools [2; 3]. To that effect, decades of research into steady-state, analytical wake models for general purpose and specialized application, including wind farm flow control methods, has resulted in an expansive field of models [4]. These mathematical models balance low fidelity in physics with high algorithmic efficiency, and their implementation into software tools has enabled further study on methods to reduce wind farm wake losses. This paper presents a systematic comparison of the common mathematical models available in research software projects specifically tailored to wind farm wake modelling.

Several commercial and open-source software aimed at both the research and industrial communities implement a common set of the published mathematical wake models. Although there are a number of additional software projects related to this field, this study focuses on three well-known software within the research community: FLOw Redirection and Induction in Steady State (FLORIS) from the National Renewable Energy Laboratory (NREL) [5], Farm Optimization and eXtended yield

[1]This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Wind Energy Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

Evaluation Software (FOXES) from Fraunhofer Institute for Wind Energy Systems (IWES) [6; 7], and PyWake from the Technical University of Denmark [8] (DTU). Each of these software projects are in active development, as shown in Figure 1. They all fall within a category of software that is developed in the Python programming language and in the object-oriented programming paradigm. All three tools are relevant within the domains of wind energy yield assessment, wind farm layout design, wind farm controls design, and similar topic areas. Although each has its own overarching design objective, all three software prioritize time-to-solution performance and leverage numerical and data processing libraries typical within the Python ecosystem. All three are free, open-source software with permissive licenses and are compatible with most computer systems. The mathematical models included in each software are generally grouped into models describing the velocity of the wind in a wind turbine wake (velocity model) and models describing the magnitude of deflection of the wake (deflection model). The models available in each software are shown in Table 1. The scope of this study extends only to the models implemented in at least two of the included software.

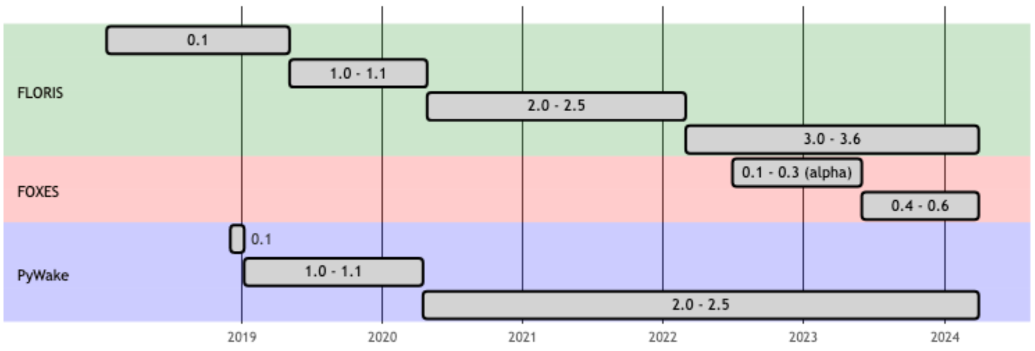

Figure 1: Release timeline for FLORIS, FOXES, and PyWake obtained from their respective source code repository release listings.

FLORIS has been in development at NREL in various forms since 2014, and the current iteration began in 2017. FLORIS is designed to support the development and implementation of new wake models, as well as wind farm energy yield calculation and wind farm layout and controls design. The software architecture is designed with explicit interfaces where new models can be integrated into the existing framework.

FOXES has been in development since 2022 at Fraunhofer IWES. It is designed as a modular tool for modelling aspects of wind farm calculations allowing users to combine existing models and easily add new models. FOXES directly integrates with optimization libraries that allow for any variable to be optimized.

PyWake has been in development at DTU since 2018. It is used to study the interaction between turbines within a wind farm and its influence on the farm's flow field and power production. It has a modular architecture providing predefined modelling blocks for calculating annual energy production.

The objective of this work is to establish the methods for characterising the similarities and differences across this class of software. Reference data from experiments, high-fidelity simulations, or literature are intentionally excluded. The focus is specifically on establishing the ensemble of results across common mathematical wake model implementations and capturing software-specific design decisions, and there is no intention to assign or imply correctness to any software's implementation of a model. For an evaluation of the wake models themselves, readers are referred to [4] and the publications specific to each model. Due to the complexity of each software project and the number of parameters available in each, it is envisioned that the current work is the beginning of an ongoing study to fully characterize the aggregate modelling capabilities within this class of software. In Section 2, this paper describes the methods and tools used to conduct the comparison of the models and algorithms in the software. Section 3 details the comparisons of the common wake models. Section 4 provides a brief analysis of the comparison results, and Section 5 summarizes the study and proposes future work.

2 Methods

This study considers the software as comprising two primary components: the mathematical formulation and the implementation in computer code, as shown in Figure 2. Approaching with the perspective of software testing, the "grey box testing" technique [22], common in software engineering, is applied to individual components of the software projects. The individual mathematical and computational models are isolated within the testing environment, and the relevant results from the test suite are compared across the included software.

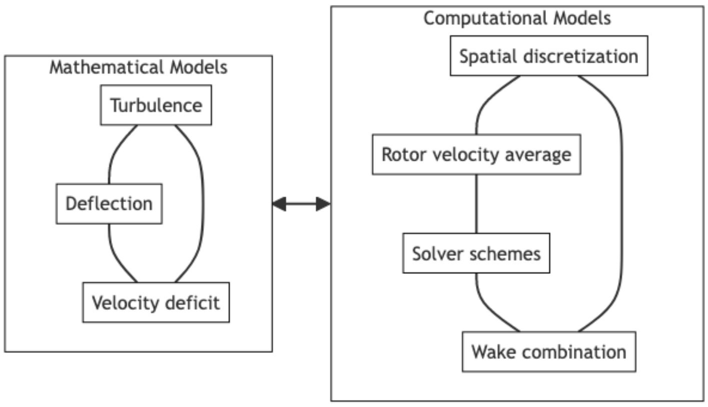

Figure 2: The primary components of the studied set of wake modelling software: mathematical models and their implementation as computational models.

This study leverages the wake model comparison framework *wcomp* [23] to express the test cases and visualize the results. *wcomp* was made available by NREL in 2023, and it provides a common interface for Python-based wake modelling projects to interface to the *windIO* ontology developed by the International Energy Agency Wind Technology Collaboration Programme (IEA Wind) Task 37 [24],

Table 1: The list of mathematical wind turbine wake velocity and deflection models, noted by "●", available in each software project included in this study. The models common to at least two software, noted in bold, are compared in Section 3.

Wake Velocity Model	FLORIS	FOXES	PyWake
Jensen 1983[9]	●	●	●
Larsen 2009[10]			●
Bastankhah / Porté-Agel 2014[11]		●	●
Bastankhah / Porté-Agel 2016[12]	●	●	
Niayifar / Porté-Agel 2016[13]			●
IEA Task37 Bastankhah 2018[14]			●
Carbajo Fuertes / Markfort / Porté-Agel 2018[15]			●
Blondel / Cathelain 2020[16]			●
Zong / Porté-Agel 2020[17]			●
Cumulative Curl 2022[18]	●		
TurbOPark (Nygaard 2022)[19]	●*	●	●
Empirical Gauss 2023	●		
Wake Deflection Model			
Jímenez 2010[20]	●		●
Bastankhah / Porté-Agel 2016[12]	●	●	
Larsen et al 2020[21]			●
Empirical Gauss 2023	●		

as shown in Figure 3.

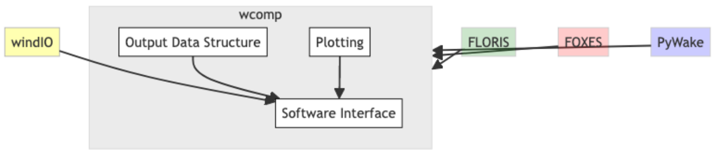

Figure 3: Architecture of the wake modelling software comparison framework *wcomp* used to integrate the software compared in this study.

The individual wake models are compared with two cases. For all cases, the wind speed is 9.8 meters per second (m/s) at the hub height, turbulence intensity is 7.5%, and air density is 1.225 kilograms per cubic meter (kg/m^3). The turbine model is the IEA Wind Task 37 15-megawatt (MW) offshore reference turbine [25], which has a rotor diameter (D) of 240 m and hub height of 150 m. The first case has a single wind turbine oriented perpendicular to the incoming wind (no yaw misalignment). The second case has four turbines placed in a line oriented with the incoming wind so that each turbine is directly behind the turbine upstream of it, and they are spaced at a distance of 5D (1200 m) apart. For the deflection model comparison, the single turbine in the one-turbine case and the leading two turbines in the four-turbine case are yawed +10° (counter-clockwise if looking at the wind farm from above toward the ground). Four one-dimension profiles are used to sample the x-component of the wind speed throughout the farm. The turbine array layouts and one-dimensional profile locations are shown in Figure 4 where:

- For all cases, a streamwise profile at hub height starts 1D upstream and extends 20D downstream of the last turbine,
- For the one-turbine case, three lateral profiles extend 2D from the rotor center in each direction and are positioned at 1D, 5D, and 10D downstream.

Additionally, an absolute value of the difference between model implementation is included in each plot.

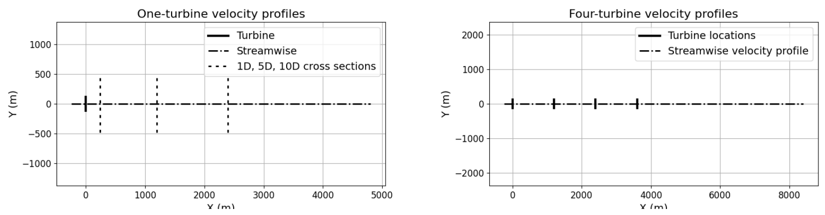

Figure 4: Turbine layout and one-dimension profile sample lines for one- and four-turbine cases.

3 Mathematical wake model comparison

This section provides a comparison of the implementation of the mathematical wake models common to at least two of the three wake modelling software included in this study, as listed in Table 1. The mathematical wake model parameters are aligned across all the cases and are noted in the sections below. However, additional computational model parameters are available as modeling decisions for the user. Two notable parameters are the method to determine the average velocity across the rotor, used to compute the thrust coefficient, and the method to combine coincident wakes. Because *wcomp* and *windIO* do not yet support the configuration of computational model parameters, they are left as the defaults to each software for this study. Note that the difference between implementations of some models may be near zero making the difference line fall on the x-axis.

3.1 Jensen 1983 wake velocity model
The model proposed by NO Jensen in 1983 assumes a top-hat shape for the wake cross section. The model is a function of the axial induction, rotor size, and distance downstream from the turbine producing the wake. The only parameter for this model is the wake growth rate, k, and it is set to 0.1. FLORIS, FOXES, and PyWake implement this model, and all three are included in the comparison in Figure 5.

3.2 Bastankhah / Porté-Agel 2014 wake velocity model
The Bastankhah / Porté-Agel 2014 model describes a self-similar Gaussian wake cross section. Since it is valid only in the far-wake region, the calculation of the length of the near-wake is also included in the model. It is a function of the thrust coefficient, rotor size, downstream distance and vertical and lateral distance from the turbine producing the wake. The only parameter for this model is the wake growth rate, k*, and it is set to 0.03. This model is implemented in FOXES and PyWake, and their results are shown in Figure 6.

3.3 Bastankhah / Porté-Agel 2016 wake velocity model
The Bastankhah / Porté-Agel 2016 model is posed as model for the wake of a yawed wind turbine, and, for the sake of comparison, it is included here as a velocity model with wind turbines operating with no yaw misalignment. In Section 3.6, it is compared again with wind turbine with yaw misalignment. This model describes a self-similar Gaussian wake cross section similar to Bastankhah / Porté-Agel 2014. It is also valid only in the far-wake region, so the calculation for the length of the near-wake region is included in the model. It is a function of the wake-center deflection, thrust coefficient, rotor size, downstream distance, and vertical and lateral distance from the turbine producing the wake. This model includes tuning parameters for the wake growth rate in the lateral and vertical directions, k_y and k_z, respectively, and two parameters to influence the onset of the far-wake region, α and β. The two wake growth rate parameters are set to equivalent values according to the empirical relationship proposed in [13], and α and β are 0.58 and 0.077, respectively. This model is implemented in FLORIS and FOXES, and their results are shown in Figure 7.

3.4 TurbOPark wake velocity model
The Turbulence Optimized Park (TurbOPark) model was proposed in Nygaard 2022, and it consists of a wind turbine wake velocity model and a wind farm blockage model. The velocity model is compared here, and the bloackage model is outside the scope of the current study. The TurbOPark model builds on an earlier top-hat model to explicitly include effects of turbulence. The wake growth rate, A, is set to 0.04. This model is implemented in FLORIS, FOXES, and PyWake, but the FLORIS implementation of TurbOPark does not support calculating the velocity for points outside of the rotor swept area, so it cannot be compared. The results for FOXES and PyWake are shown in Figure 8.

3.5 Jímenez wake deflection model
The Jímenez model for wake deflection assumes a top-hat shape for the wake cross section and calculates the wake skew angle from wake width, yaw angle, and thrust coefficient. It includes a single parameter for the linear wake growth rate, β, and it is set to 0.1. FLORIS and PyWake implement this model. Their results are shown in Figure 9.

3.6 Bastankhah / Porté-Agel 2016 wake deflection model
The Bastankhah / Porté-Agel 2016 model proposes the velocity profile for the wake behind a yawed wind turbine. See Section 3.3 for additional information. This model is implemented in FLORIS and FOXES, and their results are shown in Figure 10.

4 Discussion
The L2-Norm of the differences between implementations of common models is listed in Table 2, as measured by the one-dimensional streamwise profiles. The agreement between software implementations is very high for the top-hat models: Jensen, TurbOPark, and Jímenez. However, the Gaussian models from Bastankhah and Porté-Agel are less aligned.

The Gaussian models included in this study are only valid in the far-wake region, and the calculation for the beginning of that region is a part of the model. From Figure 7 for the Bastankhah / Porté-Agel 2016 model, it is seen that the FOXES implementation calculates the near-wake to far-wake

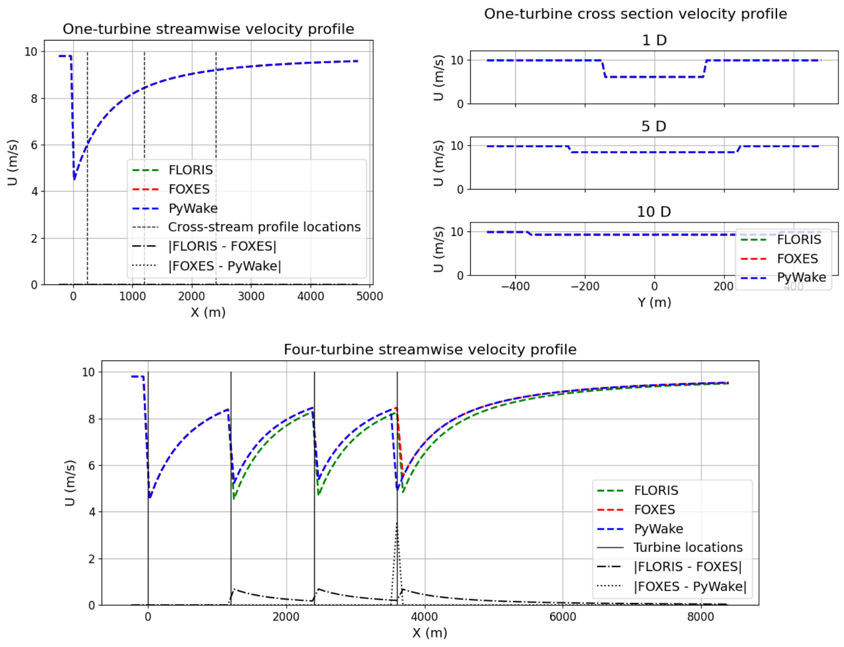

Figure 5: Jensen velocity model as implemented in FLORIS, FOXES, and PyWake compared with one-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array.

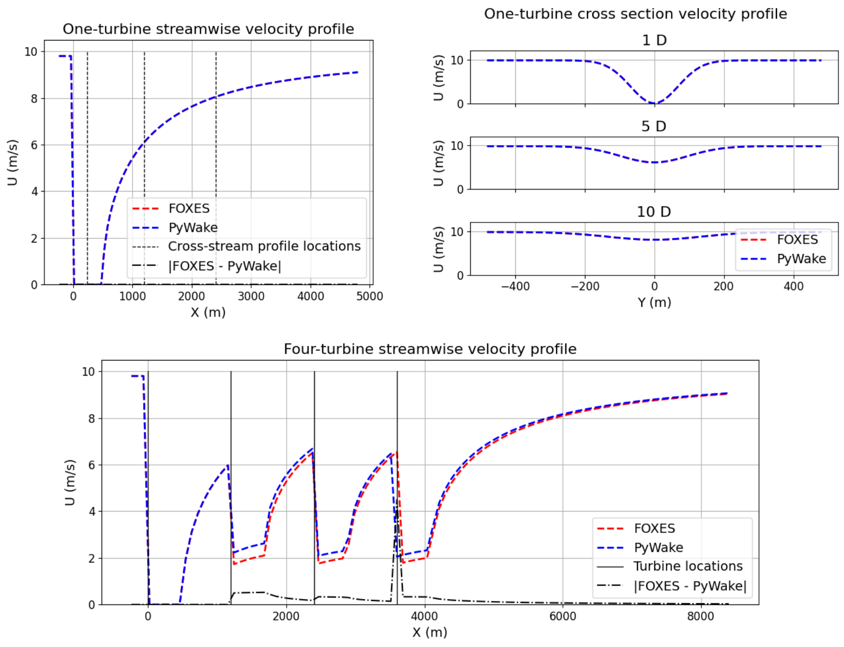

Figure 6: Bastankhah / Porté-Agel 2014 velocity model as implemented in FOXES and PyWake compared with one-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array.

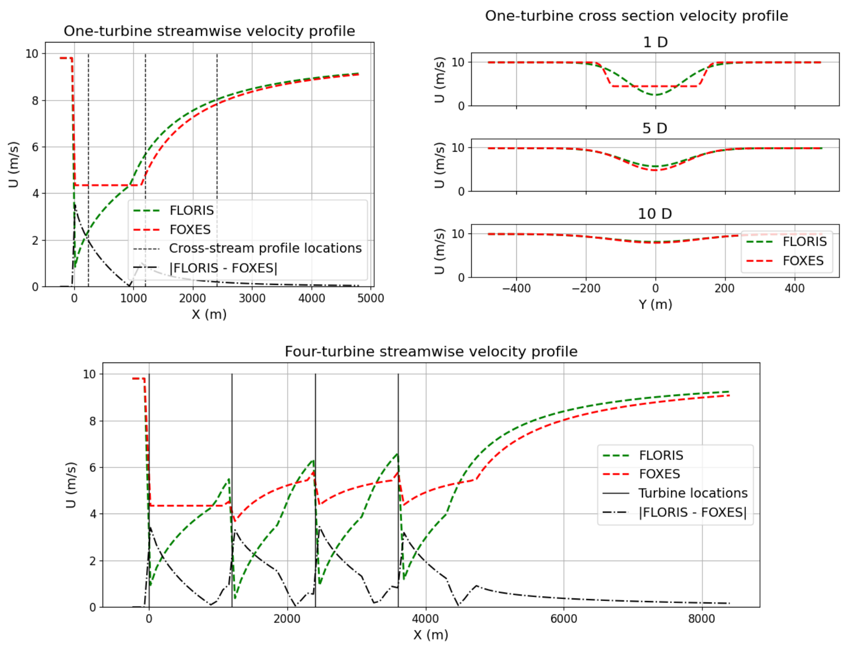

Figure 7: Bastankhah / Porté-Agel 2016 velocity model as implemented in FLORIS and FOXES compared with one-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array.

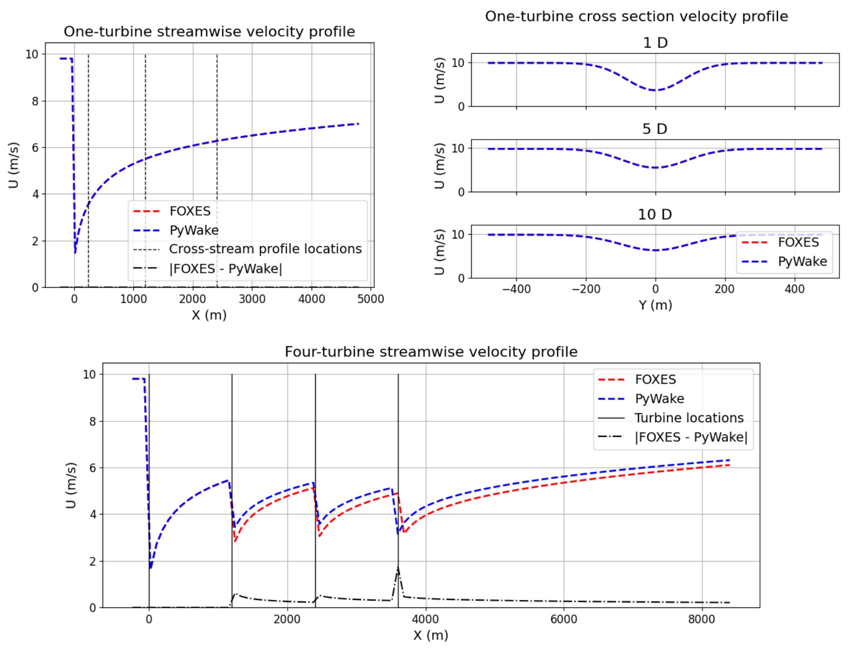

Figure 8: TurbOPark velocity model as implemented in FOXES and PyWake compared with one-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array.

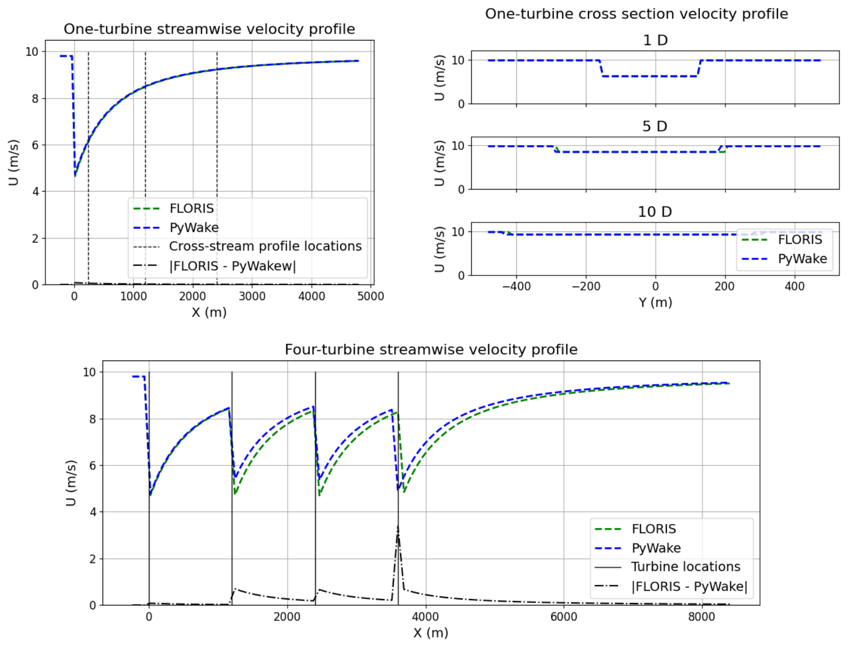

Figure 9: Jímenez wake deflection model as implemented in FLORIS and PyWake compared with one-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array. The single turbine and two leading turbines in the one- and four-turbine arrays, respectively, are yawed +10°

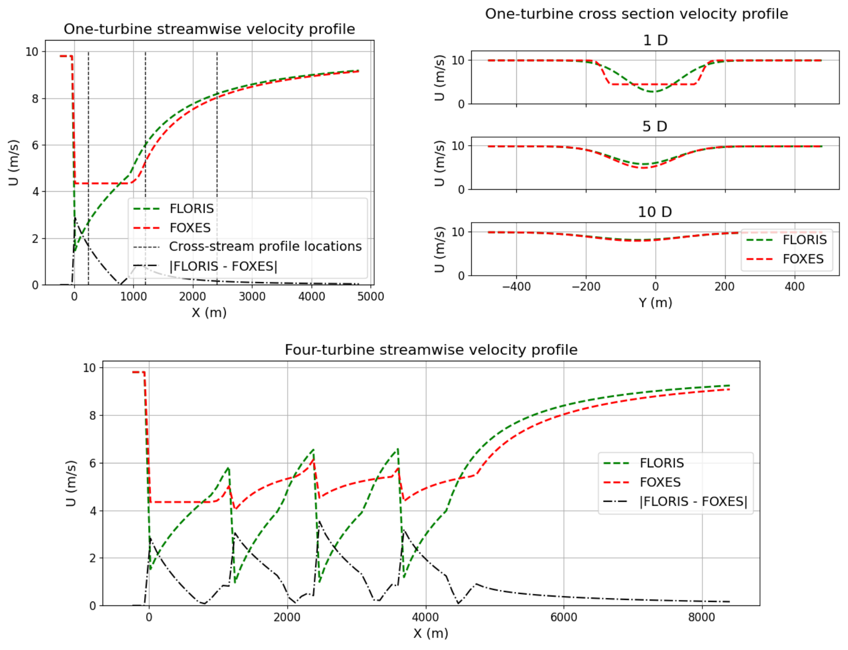

Figure 10: Bastankhah / Porté-Agel 2016 wake deflection model as implemented in FLORIS and FOXES compared with 1-dimensional profiles in the streamwise and spanwise directions for a single turbine and a streamwise one-dimensional profile for a four-turbine array. The single turbine and two leading turbines in the one- and four-turbine arrays, respectively, are yawed +10°

Table 2: The L2 norm of the difference between mathematical wake models, where implemented (indicated by a ●). The Jensen model includes FLORIS-FOXES and FOXES-PyWake norms.

Wake Velocity Model	FLORIS	FOXES	PyWake	One-Turbine L2 Norm	Four-Turbine L2 Norm
Jensen 1983	●	●	●	0.00, 0.00	2.63, 3.56
Bastankhah / Porté-Agel 2014		●	●	0.00	4.92
Bastankhah / Porté-Agel 2016	●	●		7.70	13.01
TurbOPark (Nygaard 2022)	●*	●	●	0.00	3.31
Wake Deflection Model					
Jímenez 2010	●		●	0.22	4.27
Bastankhah / Porté-Agel 2016	●	●	●	6.44	12.37

transition point at approximately 5 rotor diameters downstream, whereas the FLORIS transition occurs at approximately 4.5 rotor diameters downstream. The differences between the model implementations reduces as the wake continues to recover after the last turbine indicating that the differences in solutions would have the most impact in dense wind farm layouts. Another difference between the FLORIS and FOXES implementations on this model is the near-wake treatment. Figure 7 shows that FOXES models the near-wake with a constant velocity while FLORIS models a ramp up from the initial velocity deficit to the far-wake deficit.

The difference norms for the one-turbine cases are significantly lower than for the four-turbine cases. From Figures 5, 6, 8, and 9, it is clear that the difference is a function of the depth within a wind turbine array. This difference in the calculated results is likely due to the default methods for calculating the rotor average wind speed, modeling axial induction, and treatment of turbulence intensity induced by the turbine itself.

5 Conclusion

This work brings together three open-source research software projects from the wind farm wake modeling community to identify and characterize the ensemble of results for mathematical wake models that they implement in common. FLORIS from the National Renewable Energy Laboratory, FOXES from Fraunhofer Institute for Wind Energy Systems, and PyWake from the Technical University of Denmark are each integrated into a central framework, *wcomp*, that provides an interface to a common wind energy system ontology, *windIO*.

By providing a common ontology and the mechanisms to extract results in a consistent manner, implementations of mathematical wake models across software projects are directly compared. Overall, the wake model implementations are generally in agreement across the software, but implementation details are evident in the results. In order to accurately identify the root cause of the differences across the software, additional modeling considerations should be aligned, such as the method to average the wind speed across the rotor, grid point location and density, wind shear modeling, wake combination methods, turbulence intensity modeling, and handling for partially waked wind turbines.

This study presents the initial framework for what the authors intend to be a regular component of the wind turbine wake modeling research community. Future work in this area will include close collaborations between the developers of each software project to add additional parameterized studies, and the *wcomp* framework and *windIO* ontology will be expanded to enable configuring computational model parameters and other implementation details. Finally, the methods and results are anticipated to be presented in an online, open-source dashboard where the wind farm wake modeling community can submit new cases for comparison across the included projects and new project teams can register their software to be included in the comparisons.

6 Code Availability

Interested readers can recreate this study by installing *wcomp*; see https://rafmudaf.github.io/wcomp/. All of the figures included here are taken directly from the following Jupyter Notebooks:

- `https://github.com/rafmudaf/wcomp/blob/main/examples/torque2024_1turbine.ipynb`
- `https://github.com/rafmudaf/wcomp/blob/main/examples/torque2024_4turbine.ipynb`

References

[1] Meyers J, Bottasso C, Dykes K, Fleming P, Gebraad P, Giebel G, Göçmen T and van Wingerden J W 2022 *Wind Energy Science* **7** 2271–2306

[2] Ning A, Dykes K and Quick J 2019 URL `https://www.osti.gov/biblio/1602663`

[3] Boccolini M, Bossanyi E, Bourne S, Dombrowski A, Ferraro G, Harman K, Harrison M, Hille N, (editor) L L, Levick T, Manjock A, Mercer T, Neubert A, Ruisi R and Skeen N 2021 Wind farm control: The route to bankability Position paper DNV

[4] Porté-Agel F, Bastankhah M and Shamsoddin S 2020 *Boundary-Layer Meteorology* **174** 1–59

[5] National Renewable Energy Laboratory 2024 FLORIS Wake Modeling Framework `https://github.com/NREL/floris` v3.6

[6] Schmidt J, Vollmer L, Dörenkämper M and Stoevesandt B 2023 *Journal of Open Source Software* **8** 5464 URL `https://doi.org/10.21105/joss.05464`

[7] Foxes development group 2024 Foxes version 0.6.2 https://github.com/FraunhoferIWES/foxes

[8] Pedersen M M, Forsting A M, van der Laan P, Riva R, Romàn L A A, Risco J C, Friis-Møller M, Quick J, Christiansen J P S, Rodrigues R V, Olsen B T and Réthoré P E 2023 URL `https://gitlab.windenergy.dtu.dk/TOPFARM/PyWake`

[9] Jensen N O 1983 A note on wind generator interaction Tech. Rep. Risø-M No. 2411 Risø National Laboratory

[10] Larsen G C 2009 A simple stationary semi-analytical wake model Tech. Rep. Risoe-R No. 1713(EN) Risø National Laboratory for Sustainable Energy, Technical University of Denmark

[11] Bastankhah M and Porté-Agel F 2014 *Renewable Energy* **70** 116–123 ISSN 0960-1481 special issue on aerodynamics of offshore wind energy systems and wakes

[12] Bastankhah M and Porté-Agel F 2016 *Journal of Fluid Mechanics* **806** 506–541

[13] Niayifar A and Porté-Agel F 2016 *Energies* **9** ISSN 1996-1073

[14] IEA Wind Task 37 2018 Wake Model Description for Optimization Only Case Study

[15] Carbajo Fuertes F, Markfort C D and Porté-Agel F 2018 *Remote Sensing* **10** ISSN 2072-4292

[16] Blondel F and Cathelain M 2020 *Wind Energy Science* **5** 1225–1236

[17] Zong H and Porté-Agel F 2020 *Journal of Fluid Mechanics* **889** A8

[18] Bay C J, Fleming P, Doekemeijer B, King J, Churchfield M and Mudafort R 2023 *Wind Energy Science* **8** 401–419

[19] Nygaard N G, Steen S T, Poulsen L and Pedersen J G 2020 *Journal of Physics: Conference Series* **1618** 062072

[20] Jiménez Crespo A and Migoya E 2010 *Wind Energy* **13** 559–572

[21] Larsen G, Ott S, Liew J, van der Laan M, Simon E, Thorsen G and Jacobs P 2020 *Journal of Physics: Conference Series* **1618** 062047

[22] Lewis W E 2009 *Software Testing and Continuous Quality Improvement* (Auerbach Publications)

[23] National Renewable Energy Laboratory 2024 Wakes Compared (wcomp) `https://github.com/NREL/wcomp` v0.1

[24] Bortolotti P, Bay C, Barter G, Gaertner E, Dykes K, McWilliam M, Friis-Moller M, Molgaard Pedersen M and Zahle F 2022

[25] Gaertner E, Rinker J, Sethuraman L, Zahle F, Anderson B, Barter G E, Abbas N J, Meng F, Bortolotti P, Skrzypinski W, Scott G N, Feil R, Bredmose H, Dykes K, Shields M, Allen C and Viselli A 2020 URL `https://www.osti.gov/biblio/1603478`