FLORIS: A Brief Tutorial

Christopher Bay, Jennifer King, Paul Fleming,
Luis Martínez-Tossas, Rafael Mudafort,
Eric Simley, and Mike Lawson

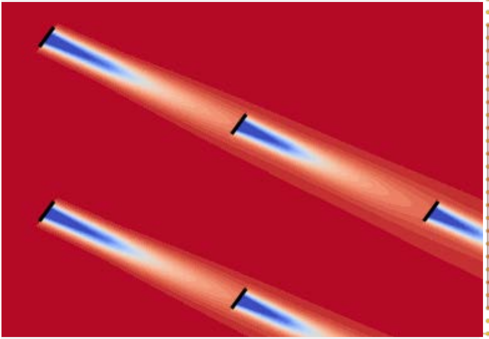
5th Wind Energy Systems Engineering Workshop
Pamplona, Spain, October 2-4, 2019

NREL/PR-5000-75661

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 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.

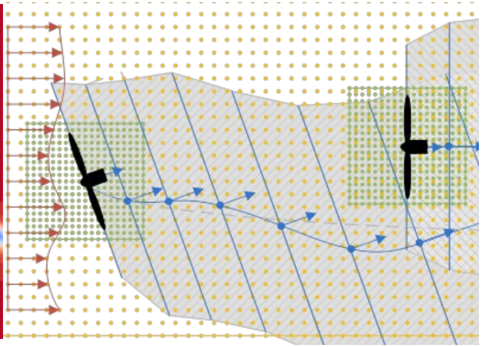
Modeling Tools at NREL

FLORIS



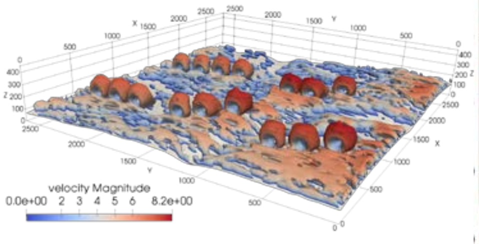
- Control-oriented model
- **Runs in fractions of seconds**
- Can be used to find optimal control settings and analyze across wind rose to estimate AEP

FAST.FARM



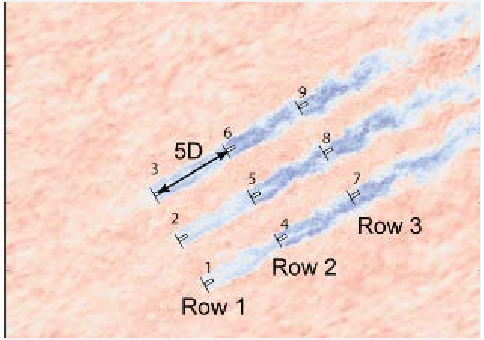
- New code which overlays DWM wakes
- Includes embedded FAST models of turbines
- **Runs on few cores**, near real time, allowing load suite analysis

WindSE



- Solves the steady/unsteady 2D/3D RANS equations
- Adjoints included for large-scale optimizations
- **Runs in serial or in parallel**, in minutes

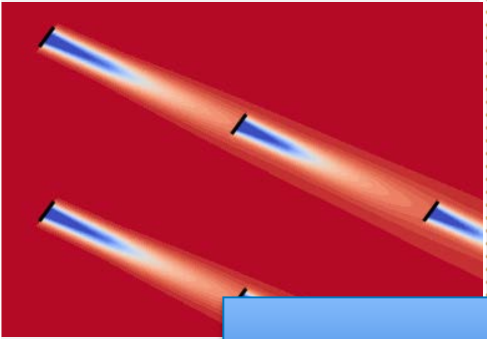
SOWFA



- Wind farm simulator based on large-eddy simulation
- Allows detailed investigation of wake physics, but **requires many cores and time to run simulations**

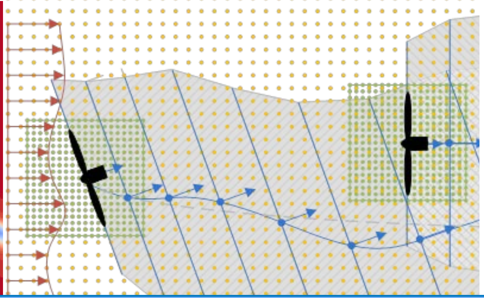
Modeling Tools at NREL

FLORIS



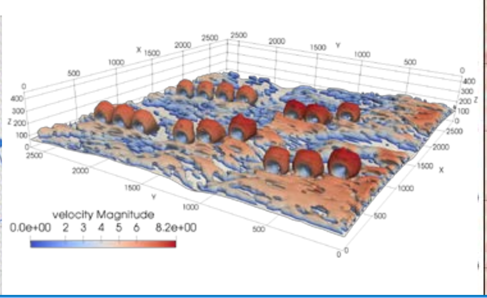
- Control-oriented model
- **Runs in fractions of seconds**
- Can be used to find optimal control settings and analyze across wind rose to estimate AEP

FAST.FARM



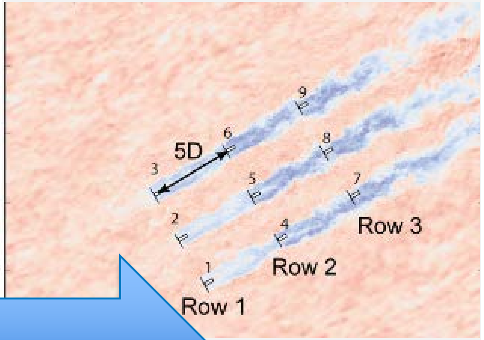
- overlays DWM wakes
- Includes embedded FAST models of turbines
- **Runs on few cores**, near real time, allowing load suite analysis

WindSE



- steady/unsteady 2D/3D RANS equations
- Adjoints included for large-scale optimizations
- **Runs in serial or in parallel**, in minutes

SOWFA

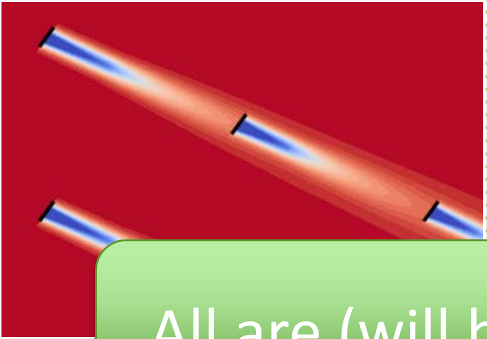


- Wind farm simulator based on large-eddy simulation
- Allows detailed investigation of wake physics, but **requires many cores and time to run simulations**

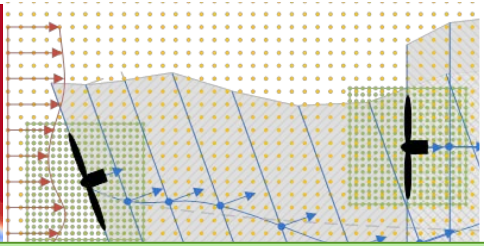
Increasing Flow Physics

Modeling Tools at NREL

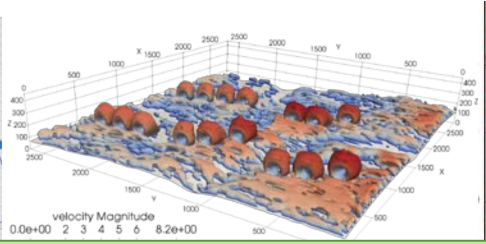
FLORIS



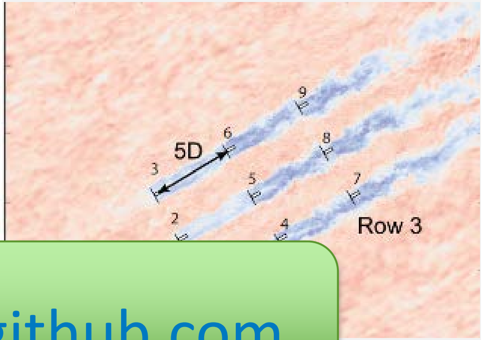
FAST.FARM



WindSE



SOWFA



All are (will be) available open source on www.github.com

FLORIS

- model
- **Runs in fractions of seconds**
- Can be used to find optimal control settings and analyze across wind rose to estimate AEP

FAST.FARM

- overlays DWM wakes
- Includes embedded FAST models of turbines
- **Runs on few cores**, near real time, allowing load suite analysis

WindSE

- steady/unsteady 2D/3D RANS equations
- Adjoints included for large-scale optimizations
- **Runs in serial or in parallel**, in minutes

SOWFA

- lator based on large-eddy simulation
- Allows detailed investigation of wake physics, but **requires many cores and time to run simulations**

FLORIS: Controls-oriented wind farm model

- Computationally inexpensive (<1s for 100 turbines)
- https://github.com/NREL/floris

Models

- Wake models
- ”Turbulence Models”
- Turbine models

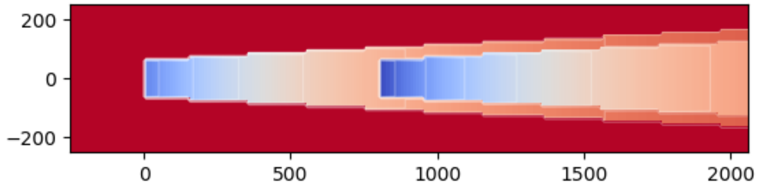
Tools

- Visualization
- Optimization
- Analysis

Wake Models

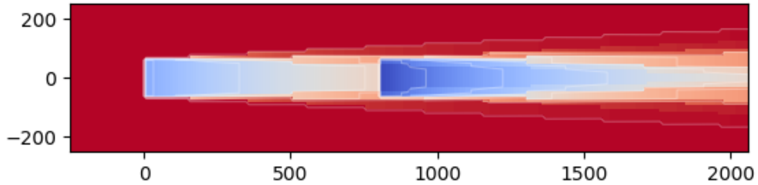
- Jensen (Park) Model – 0.0018 s

  Jensen, Niels Otto. *A note on wind generator interaction*. 1983.



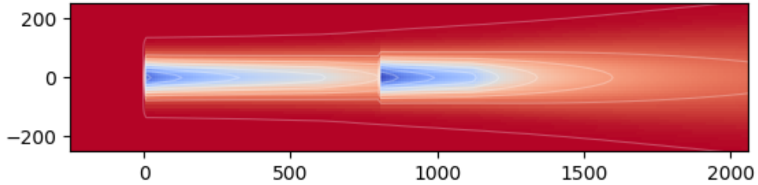
- Multi-zone wake model – 0.0019 s

  Gebraad, P. M. O., et al. Wind plant power optimization through yaw control using a parametric model for wake effects—a CFD simulation study. *2016.*



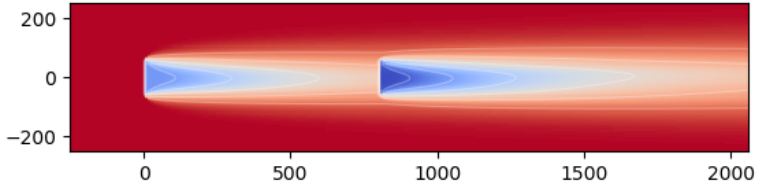
- Gaussian wake model – 0.0025 s

  Niayifar, A. and Porté-Agel, F.: A new analytical model for wind farm power prediction, 2015.
  Dilip, D. and Porté-Agel, F.: Wind Turbine Wake Mitigation through Blade Pitch Offset, 2017.
  Abkar, M. and Porté-Agel, F.: Influence of atmospheric stability on wind-turbine wakes: A large-eddy simulation study, 2015.
  Bastankhah, M. and Porté-Agel, F.: A new analytical model for wind-turbine wakes, 2014.
  Bastankhah, M. and Porté-Agel, 5 F.: Experimental and theoretical study of wind turbine wakes in yawed conditions, 2016.



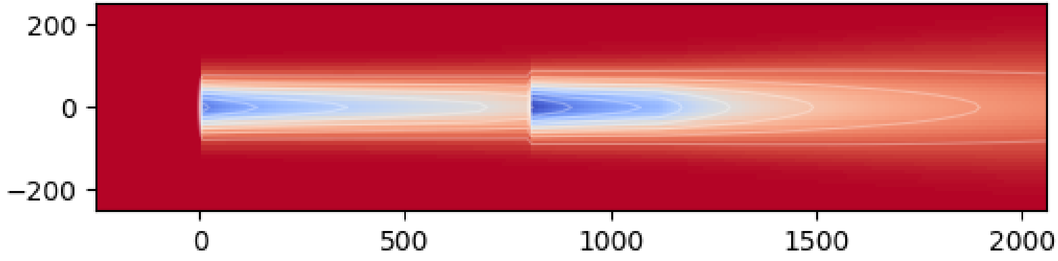
- Curl model – 1.6 s

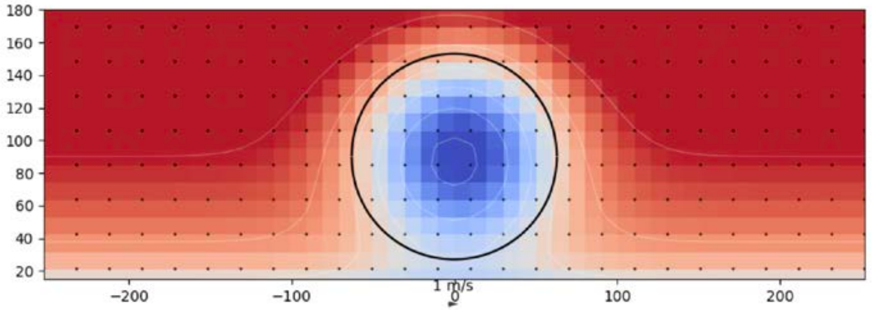
  Martínez-Tossas, L. A., Annoni, J., Fleming, P. A., and Churchfield, M. J.: The aerodynamics of the curled wake: a simplified model in view of flow control, 2019.



Wake Models - Gaussian

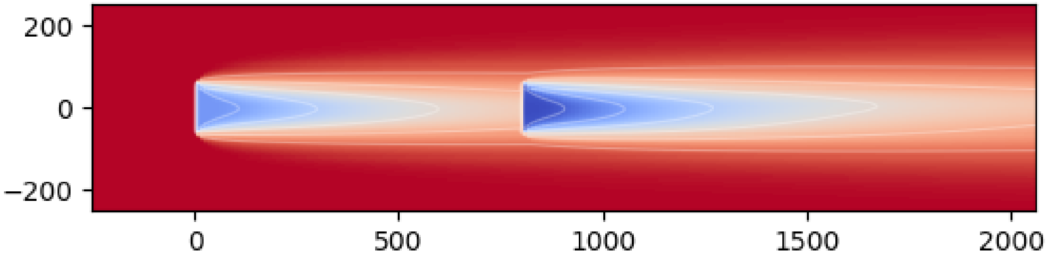
- Analytical solution to the simplified linearized Navier-Stokes equations
- Dependent on physical parameters that can be measured in the field
  - Ambient turbulence intensity
  - Shear
  - Veer
- Only 4 tuning parameters
- Good for normal turbine operation

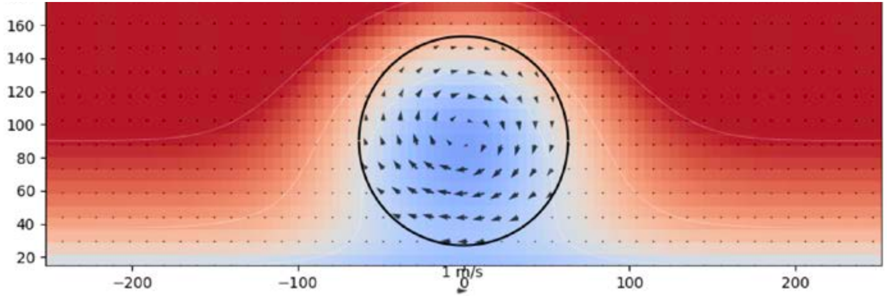




Wake Models - Curl

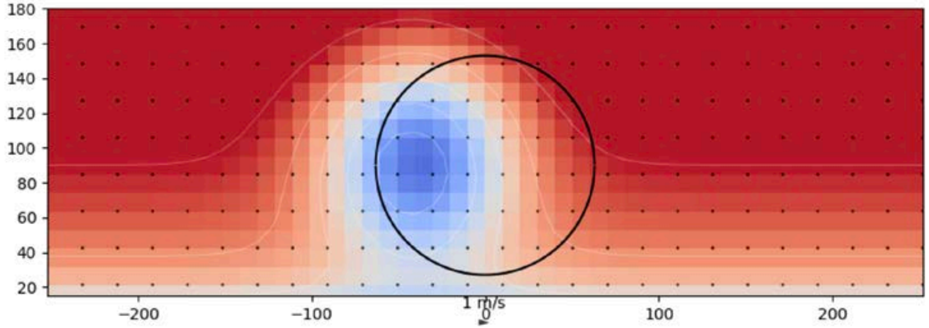
- Solves the linearized Navier-Stokes equations in time marching fashion
- Dependent on physical parameters that can be measured in the field
  - Ambient turbulence intensity
  - Shear
  - Veer
- Only 2 tuning parameters
- Good for wake steering analysis



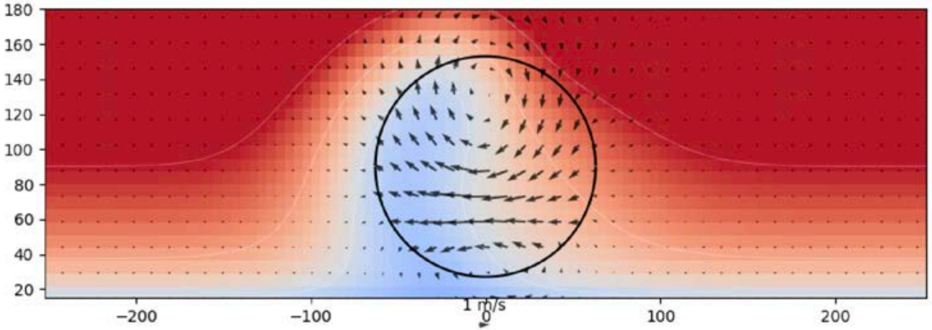


Deflection Models

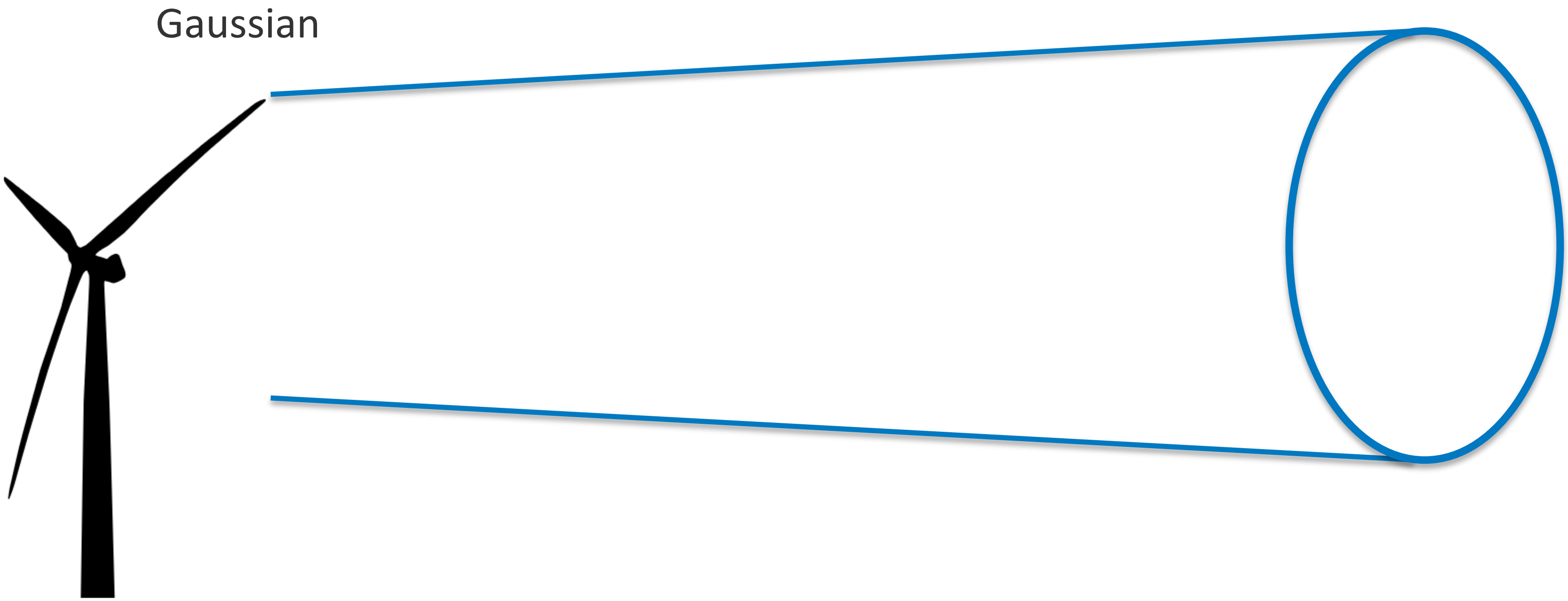
- Gaussian model deflection
  - Wake is offset from centerline



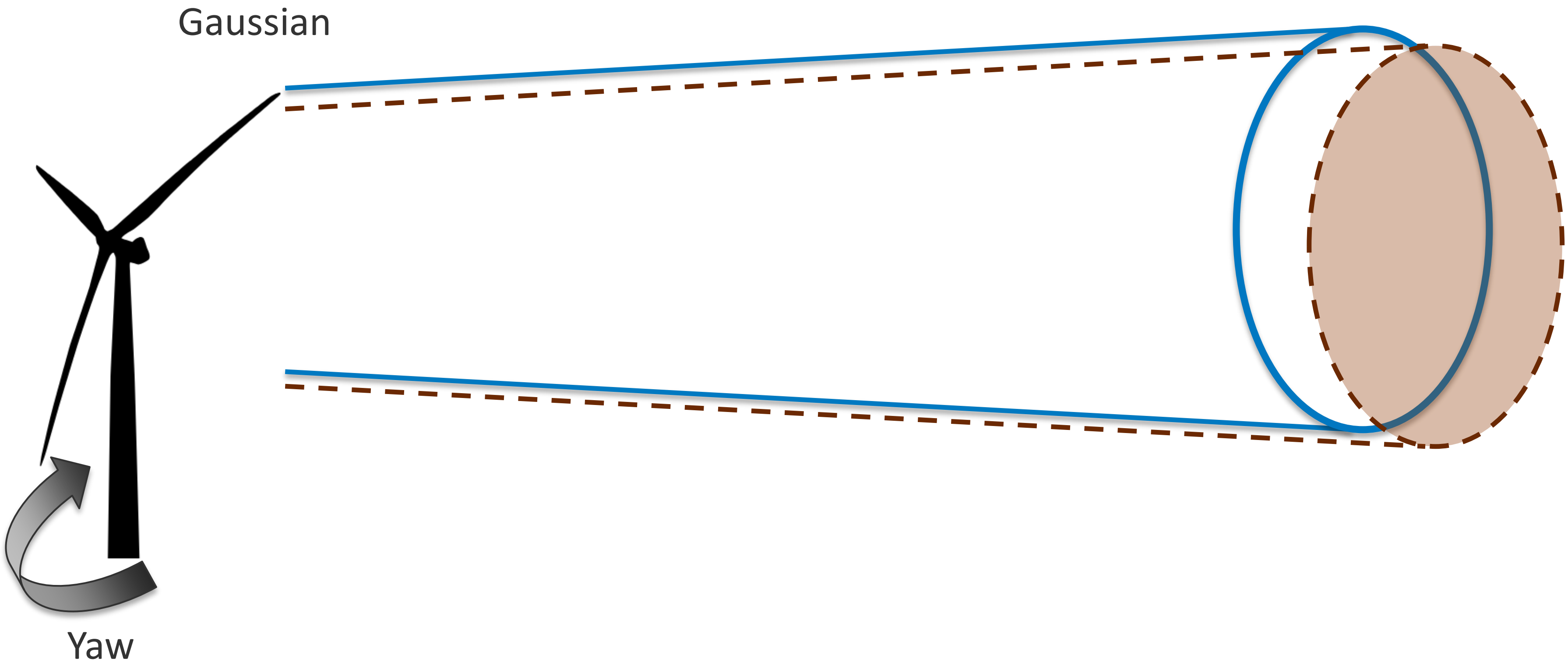
- Curl model deflection
  - Counter-rotating vortices
  - Wake rotation
  - Secondary steering



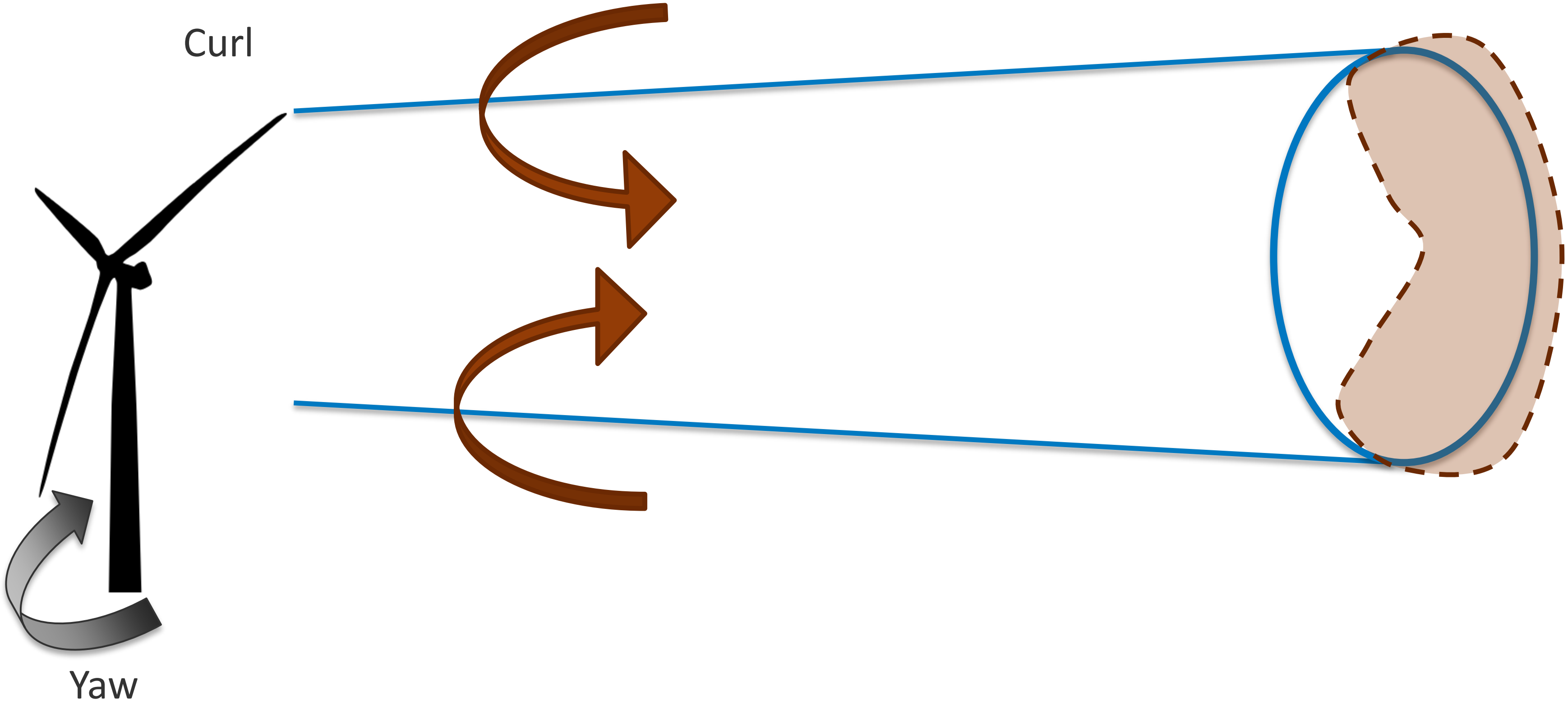
Gaussian Model



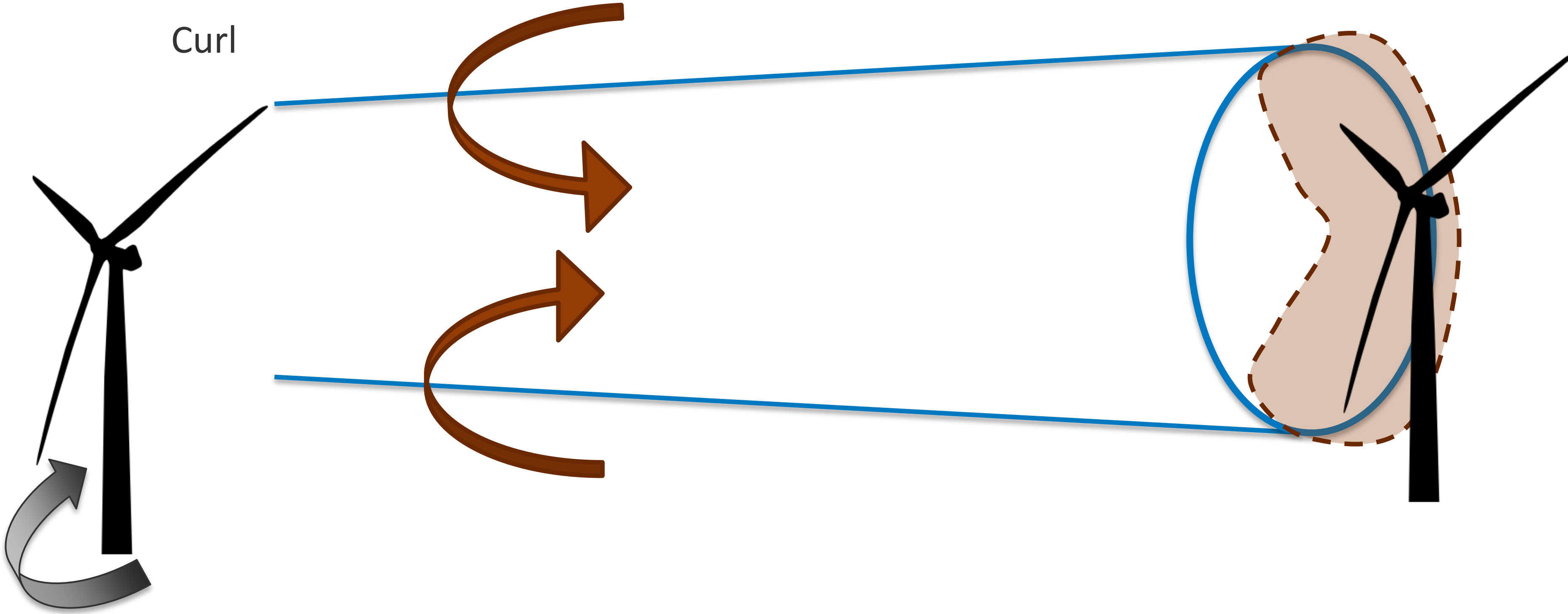
Gaussian Model



Aerodynamics of Wake Steering

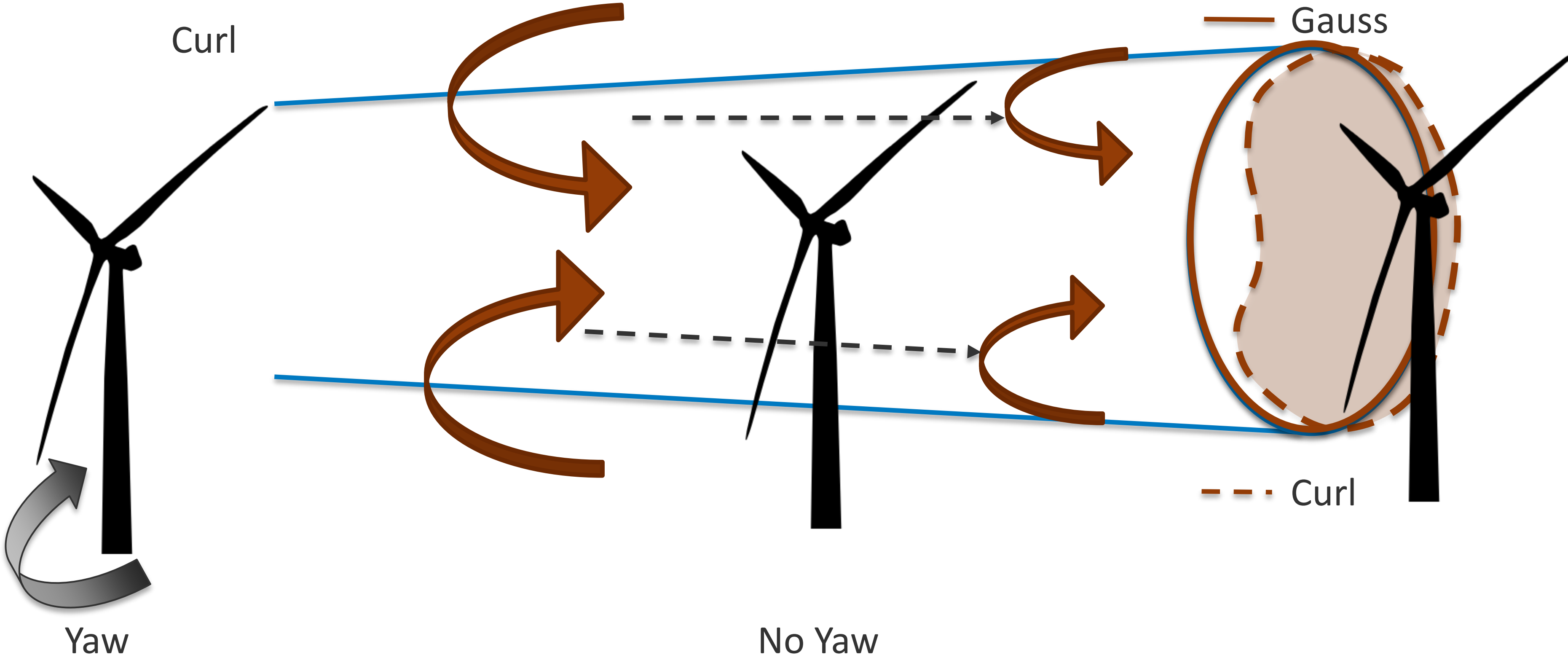


Aerodynamics of Wake Steering



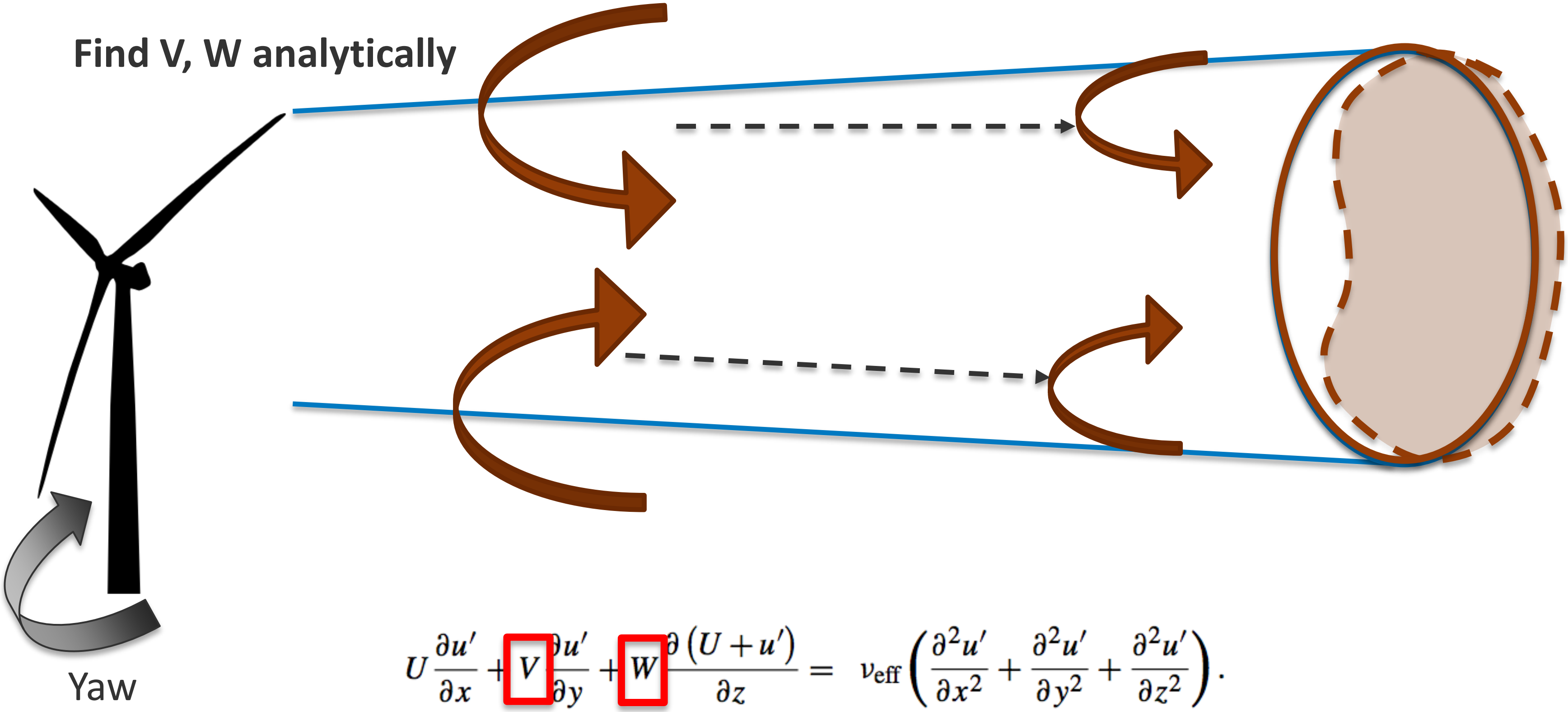
Both the Gaussian and Curl model capture
2 turbine effects very well

Aerodynamics of Wake Steering



Overview of the Curl Model

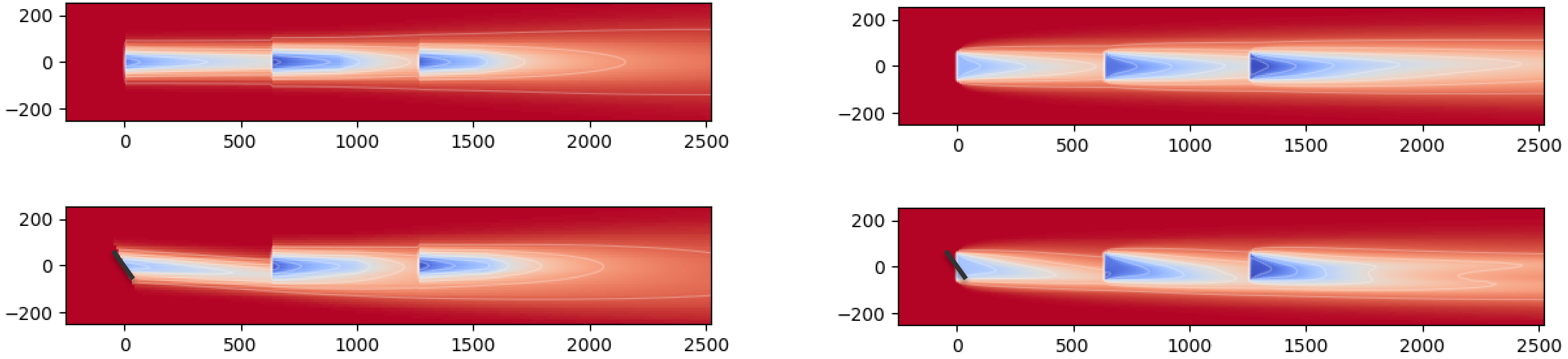
Find V, W analytically



$$U\frac{\partial u'}{\partial x} + V\frac{\partial u'}{\partial y} + W\frac{\partial (U+u')}{\partial z} = \nu_{\text{eff}}\left(\frac{\partial^2 u'}{\partial x^2} + \frac{\partial^2 u'}{\partial y^2} + \frac{\partial^2 u'}{\partial z^2}\right).$$

Martínez-Tossas, L. A., Annoni, J., Fleming, P. A., and Churchfield, M. J.: The aerodynamics of the curled wake: a simplified model in view of flow control, Wind Energ. Sci., 4, 127-138, https://doi.org/10.5194/wes-4-127-2019, 2019.

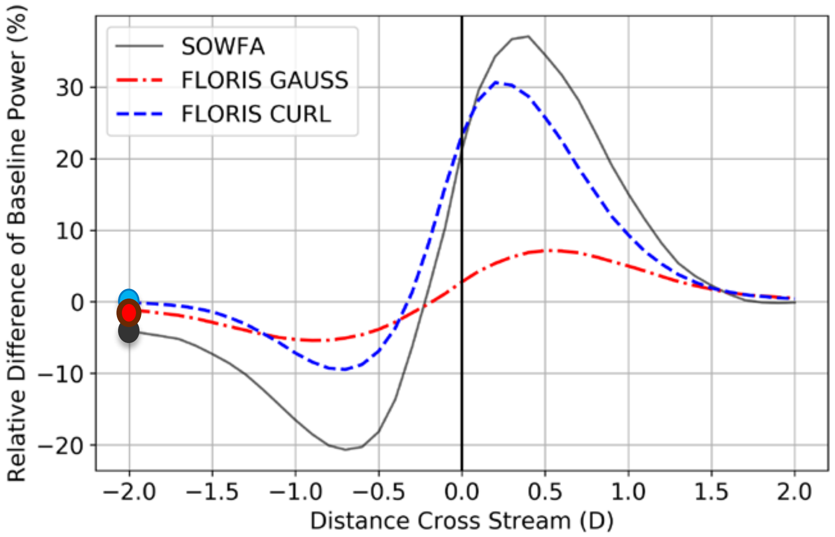
Secondary Steering

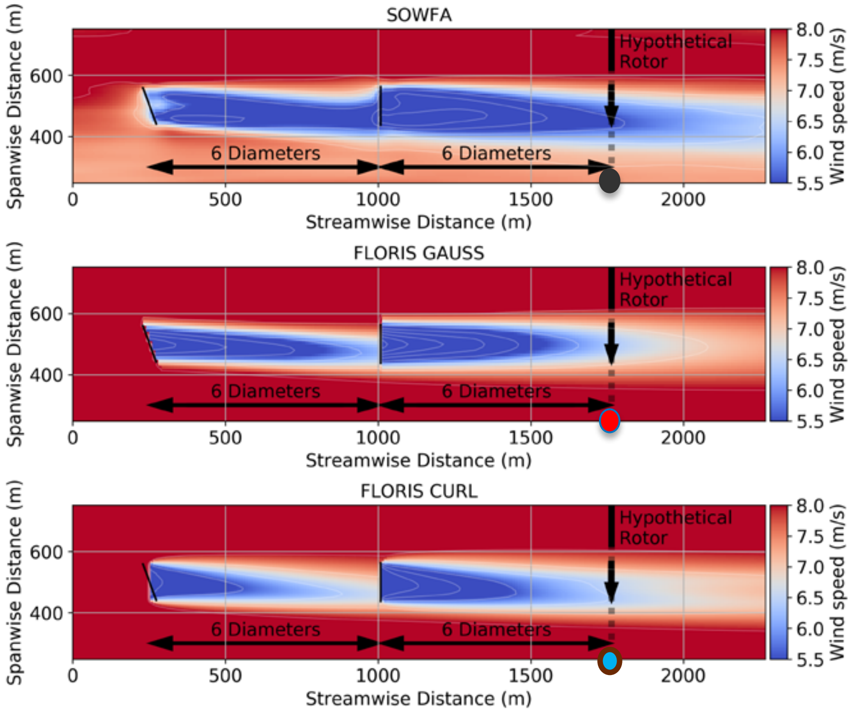


Turbines can work together to help build larger vortex-structures, developing flow control strategies throughout the farm

Compare Wake Steering

Relative power difference 6D downstream of 2nd turbine



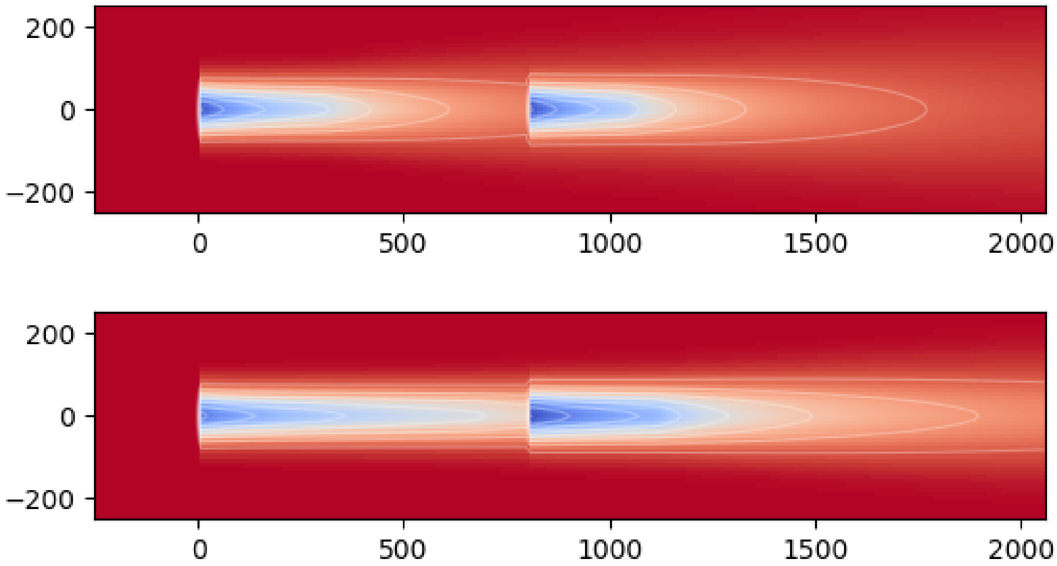


"Turbulence" Models

- **Wake expansion** dependent on ambient turbulence intensity

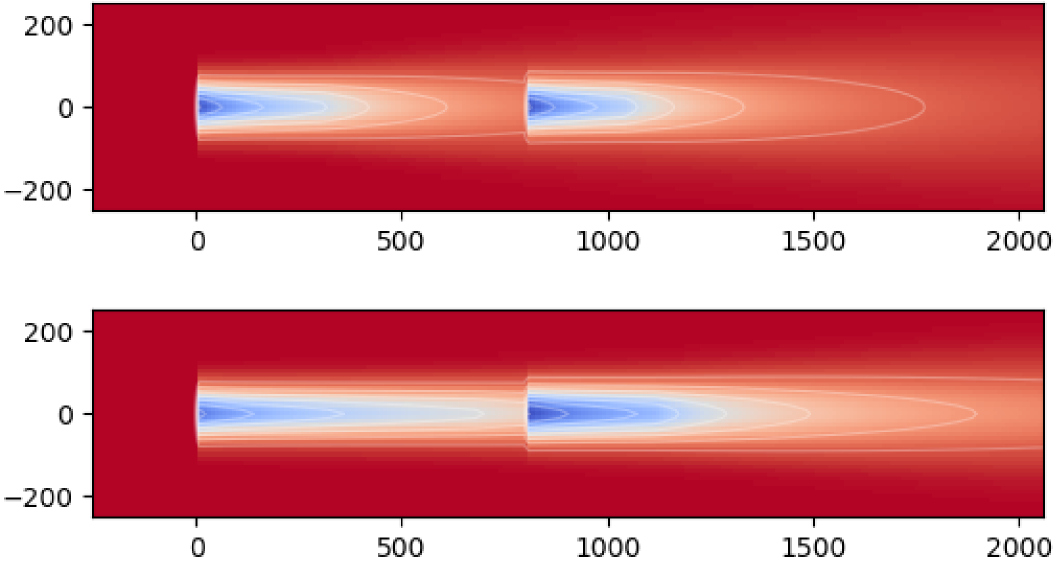
- **Added turbulence** due to turbine operation
  - As Ct increases, wake expansion increases

- Very important for investigating **deep array effects** (ongoing work)



Turbine Model - Cp/Ct Tables

- Turbine represented as **Actuator Disks**
- Generate **Cp/Ct tables** by:
  - *FAST* – Aeroelastic code
  - *CCBlade* – steady/state BEM coupled to FLORIS



Code Examples

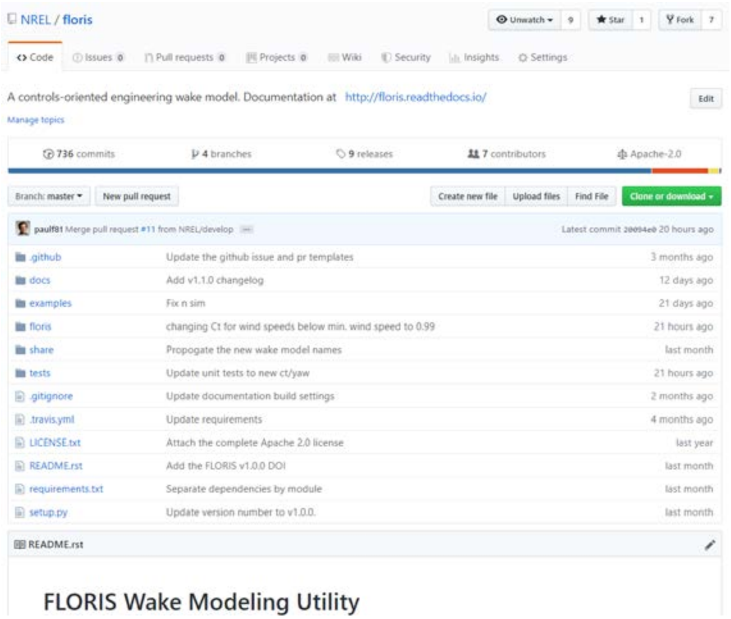
FLORIS: Open-source and Collaborative

Available at: https://github.com/NREL/floris

Divided into two packages:

- simulation:
  - Contains code for FLORIS models
- tools:
  - Modules for interacting with FLORIS models and data

Documentation and examples available at:
https://floris.readthedocs.io/en/develop/index.html



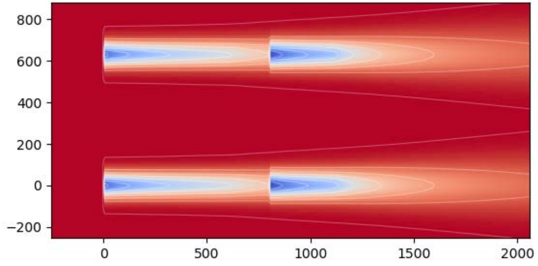
Example 0000: Open and Visualize FLORIS

Tools module allows for easy and intuitive interaction with FLORIS models.

All in python using open-source python modules.

- Line 15: import the FLORIS tools module
- Line 18: create FLORIS interface
- Line 21: calculate the wake
- Line 24: capture a horizontal cut-plane of the flow
- Line 31: use visualization module plot horizontal cut-plane

```
14  import matplotlib.pyplot as plt
15  import floris.tools as wfct
16
17  # Initialize the FLORIS interface fi
18  fi = wfct.floris_utilities.FlorisInterface("example_input.json")
19
20  # Calculate wake
21  fi.calculate_wake()
22
23  # Initialize the horizontal cut
24  hor_plane = wfct.cut_plane.HorPlane(
25      fi.get_flow_data(),
26      fi.floris.farm.turbines[0].hub_height
27  )
28
29  # Plot and show
30  fig, ax = plt.subplots()
31  wfct.visualization.visualize_cut_plane(hor_plane, ax=ax)
32  plt.show()
```



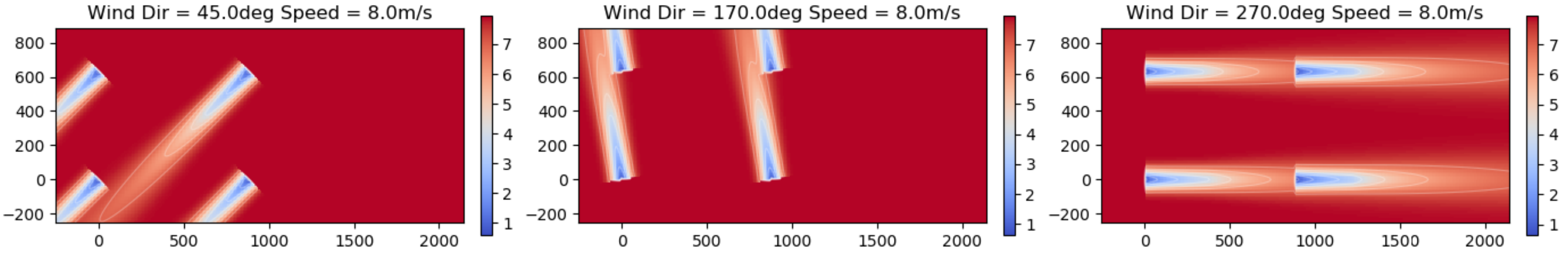
Example 0005: Changing Locations/Wind Direction

Programmatically change turbine and environmental parameters without re-loading the input file.

- Line 28: change turbine layout
- Line 63: change wind speed and direction

```
24  # set turbine locations to 4 turbines in a row - demonstrate how to change coordinates
25  D = fi.floris.farm.flow_field.turbine_map.turbines[0].rotor_diameter
26  layout_x = [0, 7*D, 0, 7*D]
27  layout_y = [0, 0, 5*D, 5*D]
28  fi.reinitialize_flow_field(layout_array=(layout_x, layout_y))
```

```
52  ws = np.linspace(6, 8, 3)
53  wd = [45.0, 170.0, 270.]
54
55  # Plot and show
56  fig, ax = plt.subplots(3, 3, figsize=(15, 15))
57  power = np.zeros((len(ws), len(wd)))
58  for i, speed in enumerate(ws):
59      for j, wdir in enumerate(wd):
60          print('Calculating wake: wind direction = ',
61                wdir, 'and wind speed = ', speed)
62
63          fi.reinitialize_flow_field(wind_speed=speed, wind_direction=wdir)
```



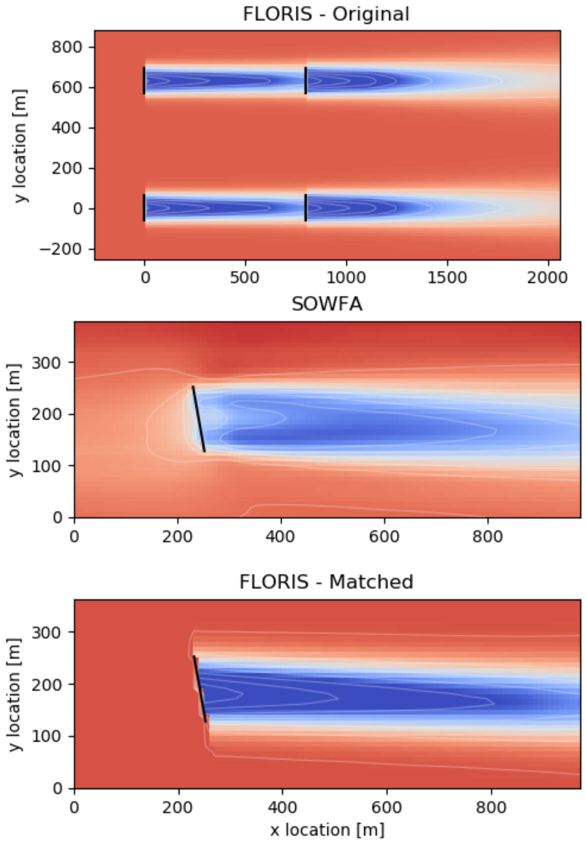
Example 0015: Compare with SOWFA

Module to load and interact with SOWFA data for analysis and comparison

- Line 27: use SOWFA interface to load SOWFA data
- Lines 55 & 61: set the relevant FLORIS model parameters to be equal to the SOWFA conditions

```
26    # Load the SOWFA case in
27    si = wfct.sowfa_utilities.SowfaInterface('sowfa_example')
```

```
54    # Set the relevant FLORIS parameters to equal the SOWFA case
55    fi.reinitialize_flow_field(wind_speed=si.precursor_wind_speed,
56                               wind_direction=si.precursor_wind_dir,
57                               layout_array=(si.layout_x, si.layout_y)
58                               )
59
60    # Set the yaw angles
61    fi.calculate_wake(yaw_angles=si.yaw_angles)
```



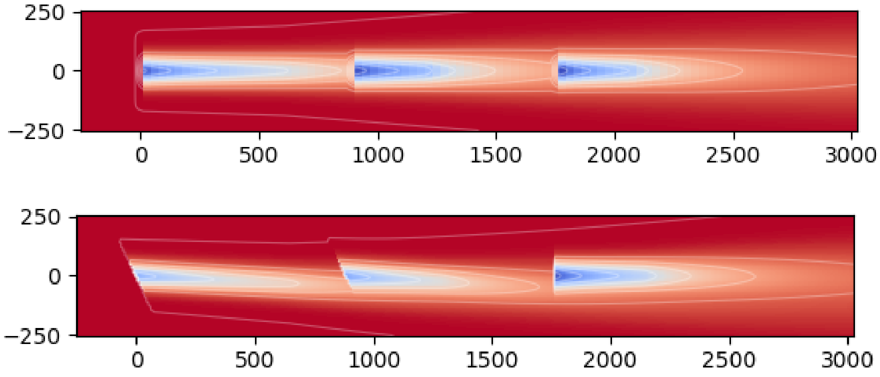
Example 0010: Optimization

Perform yaw optimizations to investigate wake steering power gains.

- Line 59: create optimization object with min. and max. yaw angles
- Line 64: perform yaw optimization

```
54  # Set bounds for allowable wake steering
55  min_yaw = 0.0
56  max_yaw = 25.0
57
58  # Instantiate the Optimization object
59  yaw_opt = YawOptimizationOneWD(fi,
60                                 minimum_yaw_angle=min_yaw,
61                                 maximum_yaw_angle=max_yaw)
62
63  # Perform optimization
64  yaw_angles = yaw_opt.optimize()
```

```
=====================================================
Optimizing wake redirection control...
Number of parameters to optimize =  3
=====================================================
yaw angles =
Turbine  0 = 24.892413743776544  deg
Turbine  1 = 24.77131815924147  deg
Turbine  2 = 1.8695304301172757e-05  deg
=========================================
Total Power Gain = 8.2%
=========================================
```



Ongoing Developments of FLORIS

- Incorporate **local effects** (currently: one wind speed/direction)
- **Deep-array effects** through better turbulence modeling
- Blade/Rotor **loads** calculations – using CCBlade from WISDEM
- Analytic **gradients** for large-scale optimizations (many turbines)
- Combinations of **optimizations –** Layout/Yaw/Thrust/Loads
- **FLORIS is a living code** – please let us know any suggestions on how we can address critical research questions in FLORIS to benefit the wind energy community.