NREL
Transforming ENERGY

FLORIS

Michael (Misha) Sinner
NAWEA/WindTech 2024
New Brunswick, New Jersey







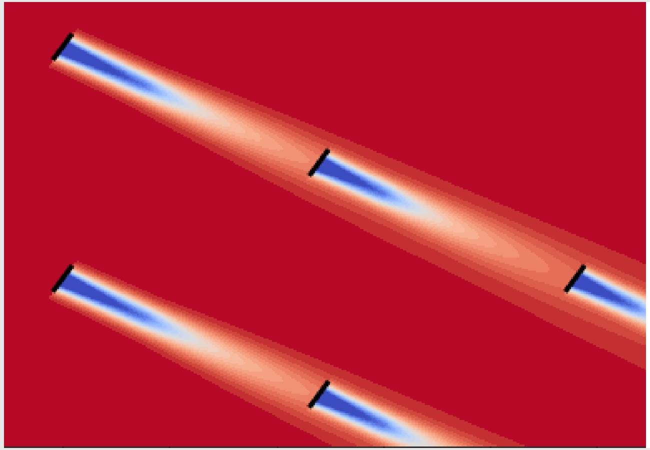

Wake models

Turbine models

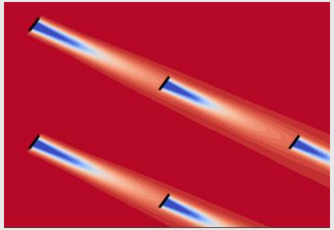
Wind data

Design tools

FLORIS



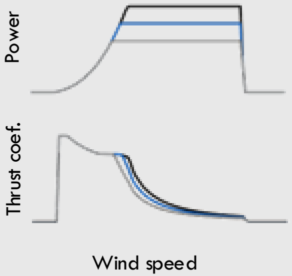
Wake models



Flow velocity deficit models

- Jensen
- Gauss-Curl Hybrid
- Cumulative Curl
- TurbOPark
- Empirical Gaussian

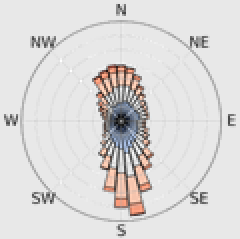
Turbine models



Actuator disks with power, thrust coefficient curves

- Yaw misaligned
- Derating
- Peak shaving
- Active wake mixing
- Shut off

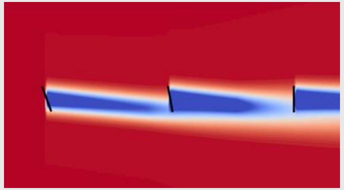
Wind data



Vectorized input wind conditions

- Wind rose
- Time series
- Flow heterogeneity
- Data readers

Design tools



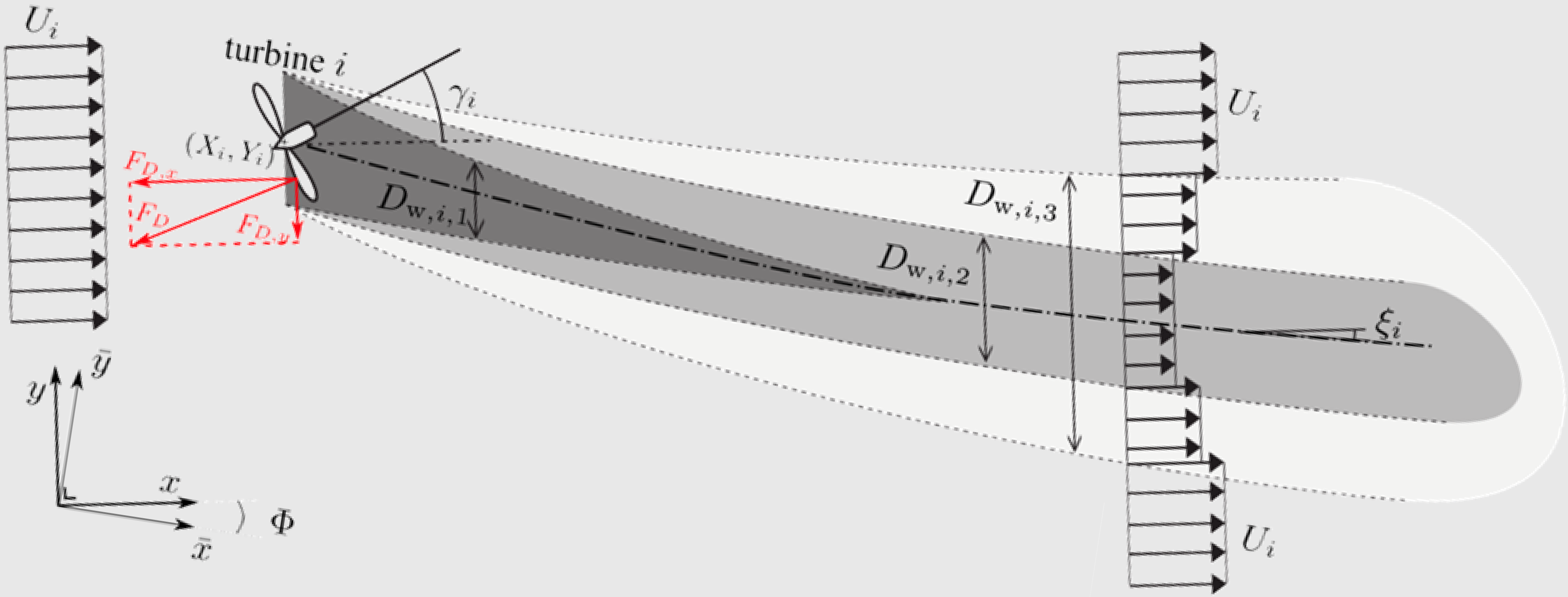
Optimization tools to help in the design and control of wind farms

- Yaw optimization
- Layout optimization

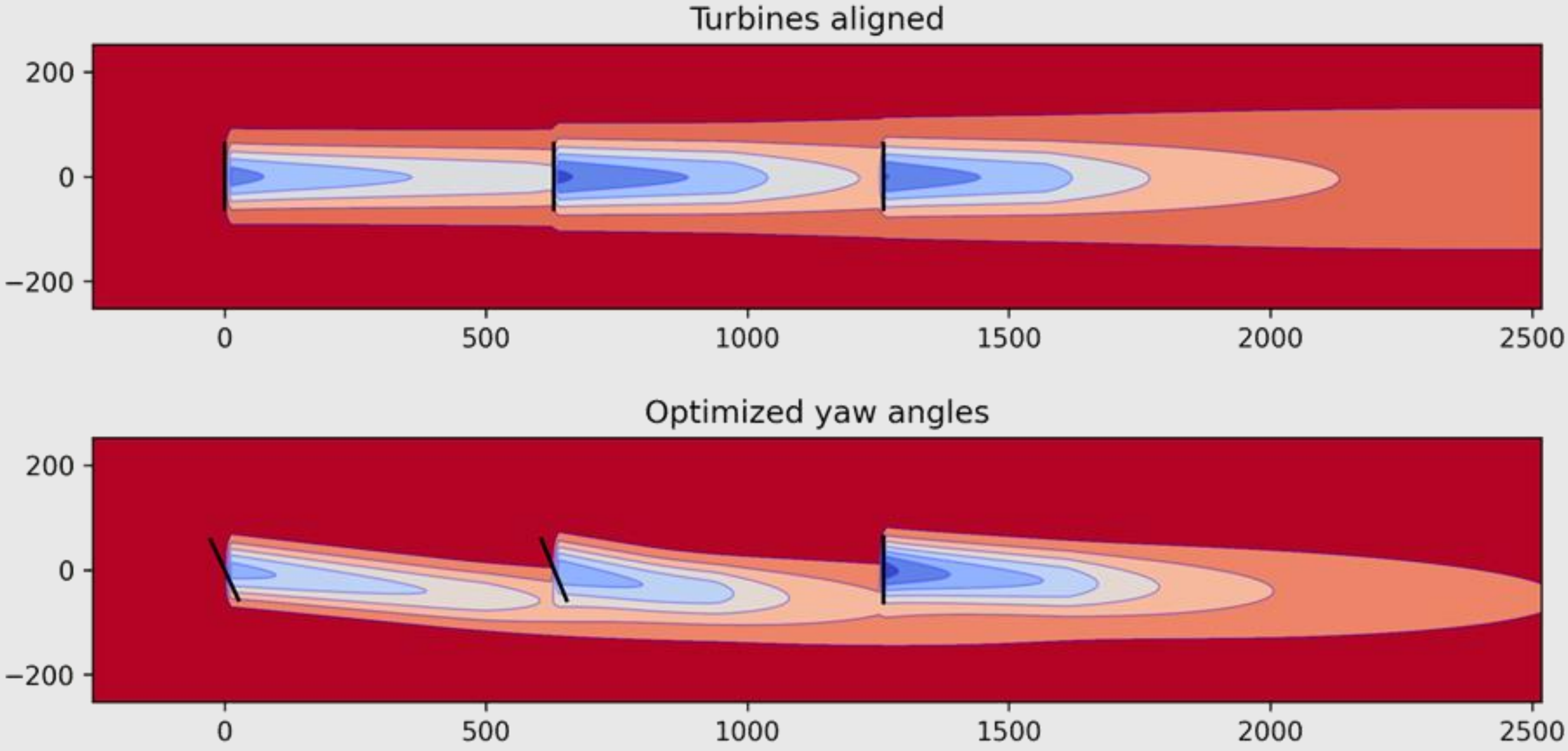
FLORIS has many use cases

- Originally developed to simulate wake steering
- Controller development for experimental campaigns
- Tools added to perform layout design
- Integration into hybrid plant simulation and design tools
- Analysis techniques developed into standalone repositories (FLASC)

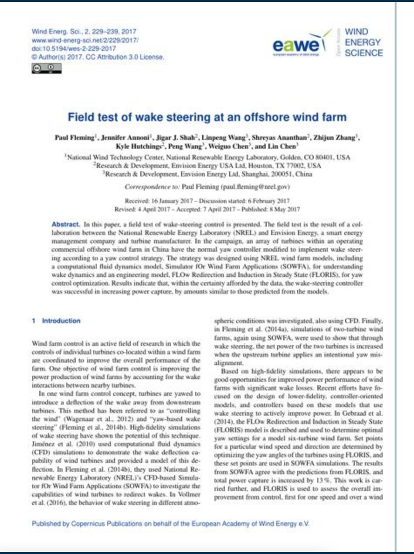
What is wake steering?

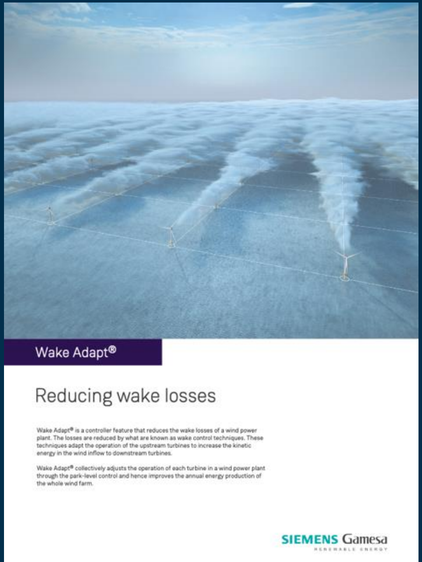


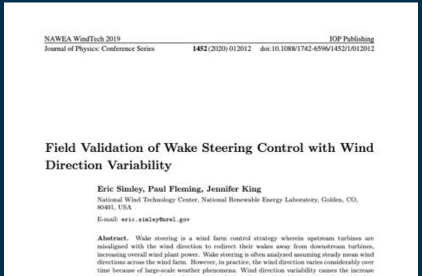
Gebraad, P. M. O., *et al*. Wind plant power optimization through yaw control using a parametric model for wake effects—a CFD simulation study. *Wind Energy*, 2016.

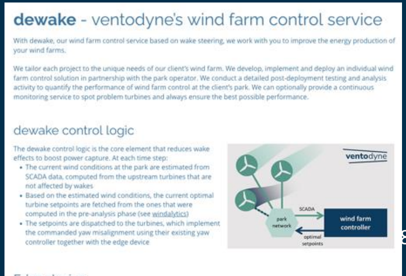

Turbines aligned
200
0
−200
0
500
1000
1500
2000
2500
Optimized yaw angles
200
0
−200
0
500
1000
1500
2000
2500

Wake steering is now out in the wild



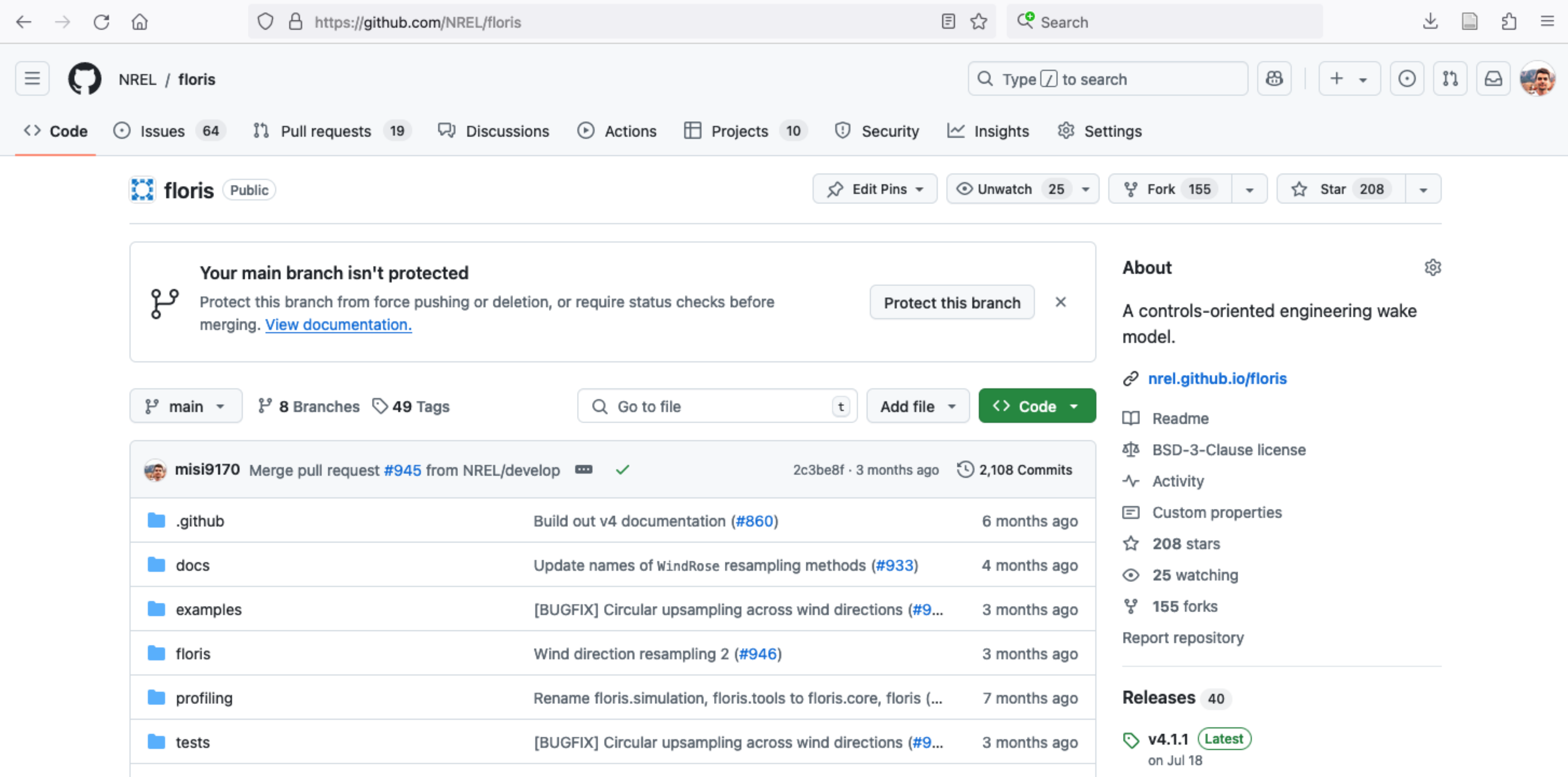






FLORIS software

FLORIS is available on github.com



… and can be readily cloned

```
> git clone https://github.com/NREL/floris
> pip install -e floris
```

or installed directly from PyPI

```
> pip install floris
```

FLORIS documentation

Documentation at nrel.github.io/floris/

Search ⌘ + K

FLORIS Wake Modeling & Wind Farm Controls

Getting Started

- Installation
- Switching from FLORIS v3 to v4

User Reference

- Introductory Concepts
- Wind Data Objects
- FLORIS Models
- Advanced Concepts
- Heterogeneous Map
- Floating Wind Turbine Modeling
- Turbine Library Interface
- Turbine Operation Models

light/dark

Contents

- Quick Start
- Engaging on GitHub

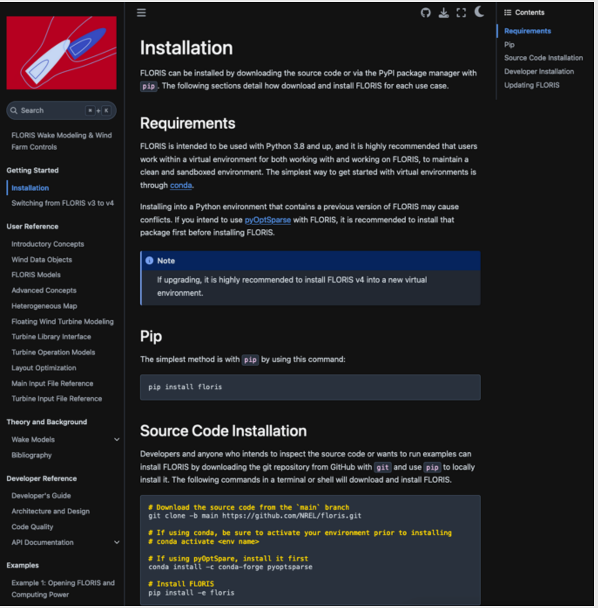
FLORIS Wake Modeling & Wind Farm Controls

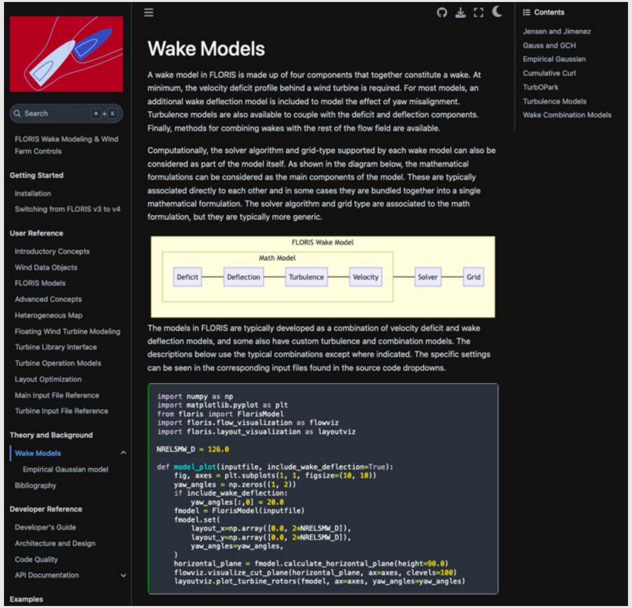
FLORIS is a controls-focused wind farm simulation software incorporating steady-state engineering wake models into a performance-focused Python framework. The software is in active development and engagement with the development team is highly encouraged. If you are interested in using FLORIS to conduct studies of a wind farm or extending FLORIS to include your own wake model, please join the conversation in GitHub Discussions!

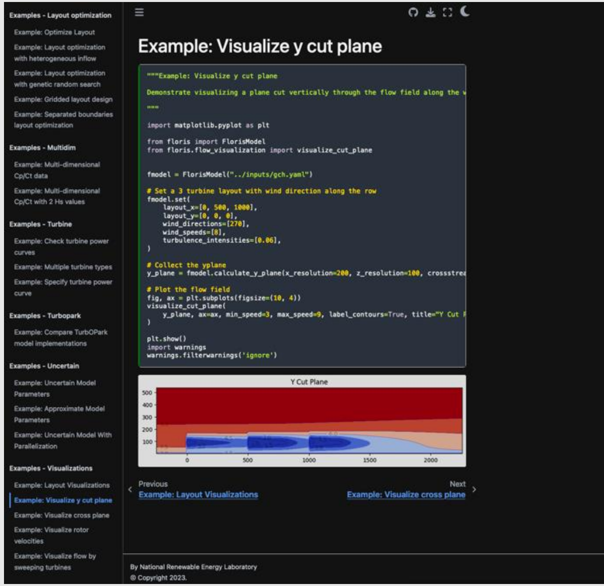
Quick Start

FLORIS is a Python package run on the command line typically by providing an input file with an initial configuration. It can be installed with `pip install floris` (see Installation). The typical entry point is `FlorisModel` which accepts the path to the input file as an argument. From there, changes can be made to the initial configuration through the `FlorisModel.set()` routine, and the simulation is executed with `FlorisModel.run()`.

```
from floris import FlorisModel
fmodel = FlorisModel("path/to/input.yaml")
fmodel.set(
    wind_directions=[i for i in range(10)],
    wind_speeds=[8.0]*10,
    turbulence_intensities=[0.06]*10
)
fmodel.run()
```


Installation
FLORIS can be installed by downloading the source code or via the PyPI package manager with pip. The following sections detail how download and install FLORIS for each use case.
Requirements
FLORIS is intended to be used with Python 3.8 and up, and it is highly recommended that users work within a virtual environment for both working with and working on FLORIS, to maintain a clean and sandboxed environment. The simplest way to get started with virtual environments is through conda.
Installing into a Python environment that contains a previous version of FLORIS may cause conflicts. If you intend to use pyOptSparse with FLORIS, it is recommended to install that package first before installing FLORIS.
Note
If upgrading, it is highly recommended to install FLORIS v4 into a new virtual environment.
Pip
The simplest method is with pip by using this command:
pip install floris
Source Code Installation
Developers and anyone who intends to inspect the source code or wants to run examples can install FLORIS by downloading the git repository from GitHub with git and use pip to locally install it. The following commands in a terminal or shell will download and install FLORIS.
Download the source code from the `main` branch
git clone -b main https://github.com/NREL/floris.git
If using conda, be sure to activate your environment prior to installing
conda activate <env name>
If using pyOptSpare, install it first
conda install -c conda-forge pyoptsparse
Install FLORIS
pip install -e floris
Contents
Requirements
Pip
Source Code Installation
Developer Installation
Updating FLORIS

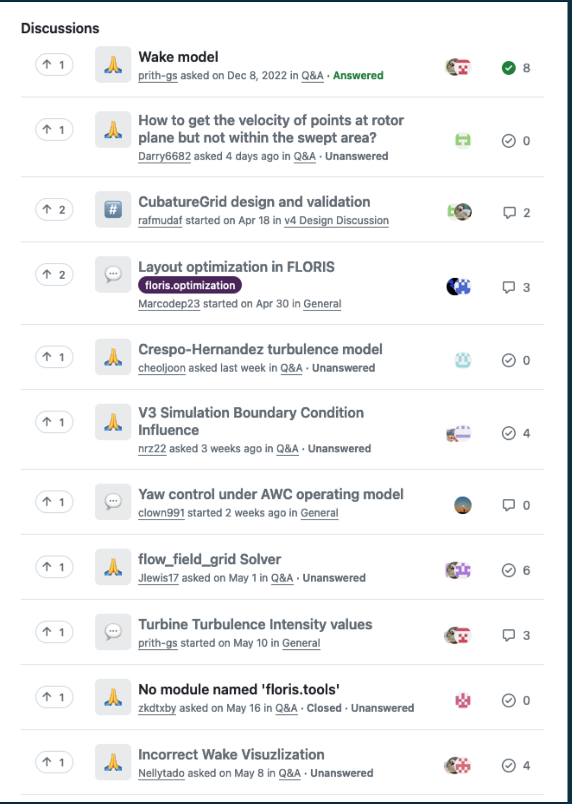

Wake Models
A wake model in FLORIS is made up of four components that together constitute a wake. At minimum, the velocity deficit profile behind a wind turbine is required. For most models, an additional wake deflection model is included to model the effect of yaw misalignment. Turbulence models are also available to couple with the deficit and deflection components. Finally, methods for combining wakes with the rest of the flow field are available.
Computationally, the solver algorithm and grid-type supported by each wake model can also be considered as part of the model itself. As shown in the diagram below, the mathematical formulations can be considered as the main components of the model. These are typically associated directly to each other and in some cases they are bundled together into a single mathematical formulation. The solver algorithm and grid type are associated to the math formulation, but they are typically more generic.
FLORIS Wake Model
Math Model
Deficit
Deflection
Turbulence
Velocity
Solver
Grid
The models in FLORIS are typically developed as a combination of velocity deficit and wake deflection models, and some also have custom turbulence and combination models. The descriptions below use the typical combinations except where indicated. The specific settings can be seen in the corresponding input files found in the source code dropdowns.
Contents
Jensen and Jimenez
Gauss and GCH
Empirical Gaussian
Cumulative Curl
TurbOPark
Turbulence Models
Wake Combination Models


Example: Visualize y cut plane
"""Example: Visualize y cut plane
Demonstrate visualizing a plane cut vertically through the flow field along the w
import matplotlib.pyplot as plt
from floris import FlorisModel
from floris.flow_visualization import visualize_cut_plane
Y Cut Plane
Previous
Example: Layout Visualizations
Next
Example: Visualize cross plane
By National Renewable Energy Laboratory
© Copyright 2023.

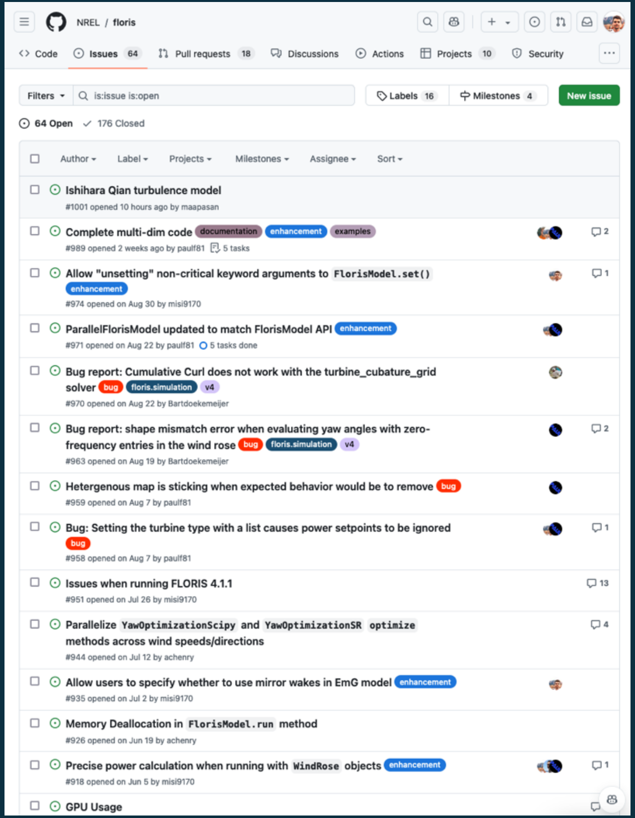
Open-source software community

We support and encourage interaction on github

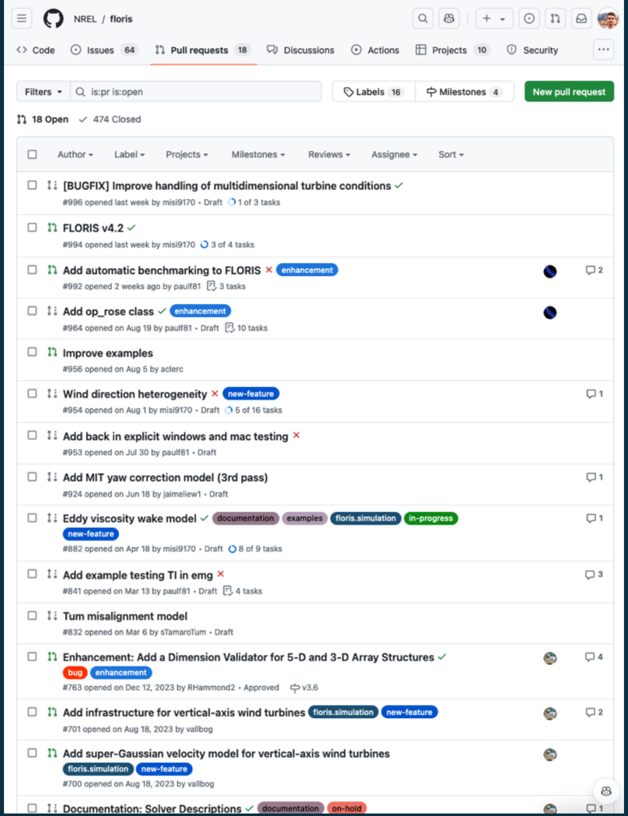
Discussion forum



Issues



Pull requests



Basic FLORIS usage

```python
import numpy as np
from floris import FlorisModel, TimeSeries

# Load the Floris model
fmodel = FlorisModel("inputs/gch.yaml")

# Set up inflow wind conditions
time_series = TimeSeries(
    wind_directions=270 + 30 * np.random.randn(100),
    wind_speeds=8 + 2 * np.random.randn(100),
    turbulence_intensities=0.06 + 0.02 * np.random.randn(100),
)

# Set the wind conditions for the model
fmodel.set(wind_data=time_series)

# Run the calculations
fmodel.run()

# Extract turbine and farm powers
turbine_powers = fmodel.get_turbine_powers() / 1000.0
farm_power = fmodel.get_farm_power() / 1000.0

print(turbine_powers.shape)
print(farm_power.shape)

# # Output:
# (100, 3)
# (100,)
```

Input file contains wake model parameters and specifies turbine to use

Wind data objects (`TimeSeries`, `WindRose`, `WindTIRose`, etc) conveniently package inflow conditions

Set inflow conditions, farm layout, control setpoints, etc. (replaces `reinitialize()`)

Execute solve, takes no inputs (replaces `calculate_wake()`)

Extract outputs after solve

```yaml

name: GCH
description: Three turbines using Gauss Curl Hybrid model
floris_version: v4

logging:
  console:
    enable: true
    level: WARNING
  file:
    enable: false
    level: WARNING

solver:
  type: turbine_grid
  turbine_grid_points: 3

farm:
  layout_x:
  - 0.0
  - 630.0
  layout_y:
  - 0.0
  - 0.0
  turbine_type:
  - nrel_5MW
  - iea_10MW

flow_field:
  air_density: 1.225
  reference_wind_height: 90.0 # Since multiple defined turbines, must s
  turbulence_intensities:
  - 0.06
  wind_directions:
  - 270.0
  wind_shear: 0.12
  wind_speeds:
  - 8.0
  wind_veer: 0.0

wake:
  model_strings:
    combination_model: sosfs
    deflection_model: gauss
    turbulence_model: crespo_hernandez
    velocity_model: gauss

  enable_secondary_steering: false
  enable_yaw_added_recovery: false
  enable_transverse_velocities: false
  enable_active_wake_mixing: false

```

Documentation

Logging

Grid points

Farm details

Inflow details

Wake model selection

Deflection parameters

Deficit parameters

Turbulence parameters

```yaml
  wake_deflection_parameters:
    gauss:
      ad: 0.0
      alpha: 0.58
      bd: 0.0
      beta: 0.077
      dm: 1.0
      ka: 0.38
      kb: 0.004
    jimenez:
      ad: 0.0
      bd: 0.0
      kd: 0.05

  wake_velocity_parameters:
    cc:
      a_s: 0.179367259
      b_s: 0.0118889215
      c_s1: 0.0563691592
      c_s2: 0.13290157
      a_f: 3.11
      b_f: -0.68
      c_f: 2.41
      alpha_mod: 1.0
    gauss:
      alpha: 0.58
      beta: 0.077
      ka: 0.38
      kb: 0.004
    jensen:
      we: 0.05

  wake_turbulence_parameters:
    crespo_hernandez:
      initial: 0.1
      constant: 0.5
      ai: 0.8
      downstream: -0.32

```

Anything can be set dynamically, too!

```yaml
# Data based on:
# https://github.com/IEAWindTask37/IEA-15-240-RWT/blob/master/
# IEA-15-240-RWT_tabular.xlsx
# Note: Small power variations above rated removed.
# Generator efficiency of 100% used.
turbine_type: 'iea_15MW'
hub_height: 150.0
rotor_diameter: 242.24
TSR: 8.0
operation_model: cosine-loss
power_thrust_table:
  ref_air_density: 1.225
  ref_tilt: 6.0
  cosine_loss_exponent_yaw: 1.88
  cosine_loss_exponent_tilt: 1.88
  helix_a: 1.809
  helix_power_b: 4.828e-03
  helix_power_c: 4.017e-11
  helix_thrust_b: 1.390e-03
  helix_thrust_c: 5.084e-04
  power:
    - 0.000000
    - 0.000000
    - 42.733312
    - 292.585981
    - 607.966543
    - 981.097693
    - 1401.98084
    - 1858.67086
    - 2337.575997
    - 2824.097302
    - 3303.06456
    - 3759.432328
    - 4178.637714
    - 4547.19121
    - 4855.342682
    - 5091.537139
    - 5248.453137
    - 5320.793207
    - 5335.345498
    - 5437.90563
    - 5631.253025
    - 5920.980626
    - 6315.115602
    - 6824.470067
    - 7462.846389
    - 8238.359448
    - 9167.96703
```

Documentation

Physical characteristics

Operation model

Power/thrust curve metadata

Power/thrust curve definition

New in FLORIS v4

Updated procedure for setting up and running

```
> fmodel = FlorisModel('inputs/gch.yaml')
> fmodel.set(wind_data=time_series)
> fmodel.run()
```

`set()` **replaces** `reinitialize()`

`run()` **replaces** `calculate_wake()`

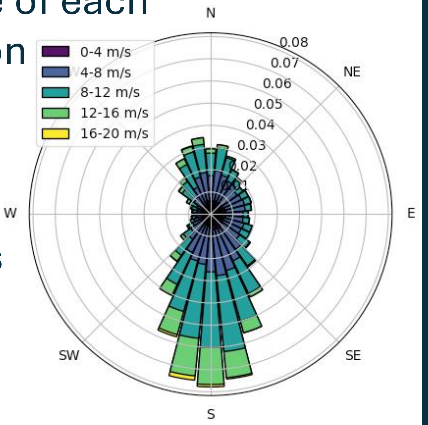
New underlying data structure

- Wind conditions collapsed into a single `findex` dimension, rather than `wind_directions x wind_speeds`
- Enables arbitrary changes per `findex`, for example, varying turbulence intensity
- Similar to running in `time_series=True` in v3

`WindData` objects

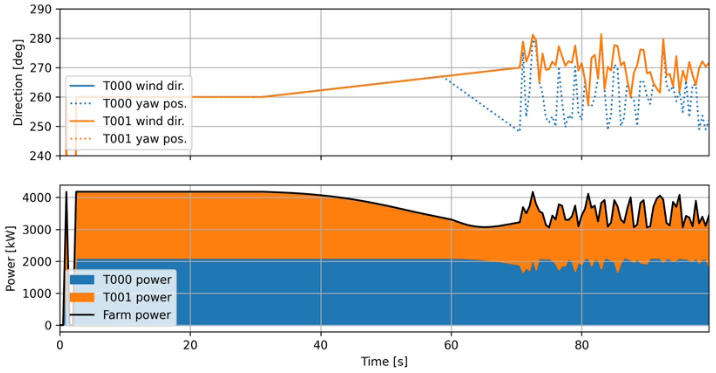
`WindRose`

- For running over a grid of wind speed, wind direction combinations
- Specify frequency of occurrence of each combination
- Useful for AEP evaluations



`TimeSeries`

- For running a series of unique wind conditions
- Useful for playing through observations

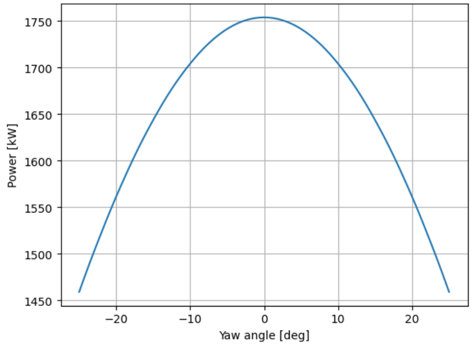


Turbine operation models

Allows flexible definition of the turbine actuator disk and how it operates

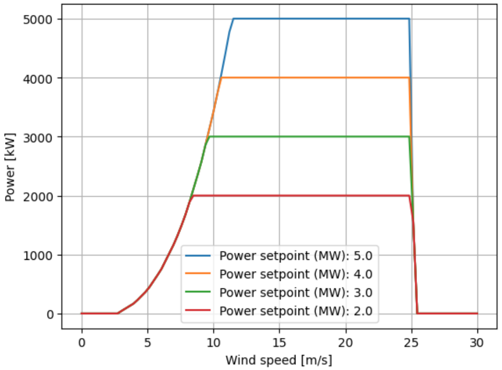
Cosine loss

Default, loses power to yaw according to cosine exponential model



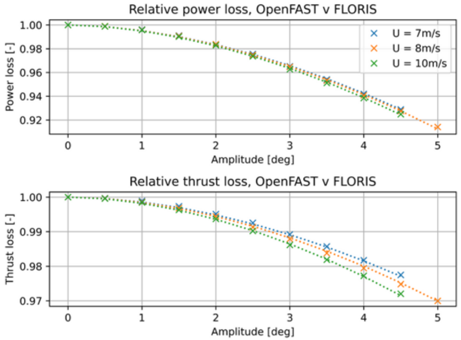
Simple derating

Approximate model for how turbines behave in derated operation



Active wake control

Models how turbines perform with Helix wake mixing



Symmetry between `FlorisModel`s

`FlorisModel`

Basic class for running FLORIS calculations

`UncertainFlorisModel`

Class for running calculations under uncertainty

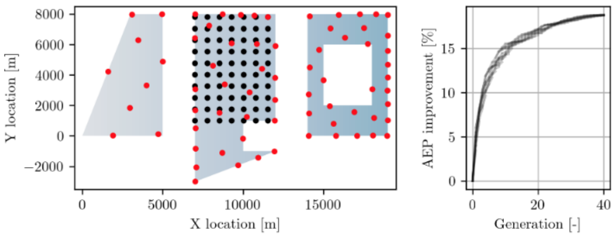
`ParFlorisModel`

Class for running calculations with parallel computing

New wind farm layout optimizers

Random Search

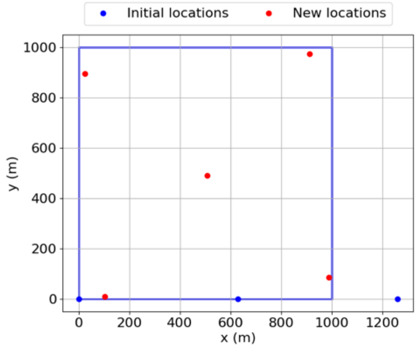
- Randomly perturbs turbine locations to look for improved layouts
- Robust to initial condition, complex boundaries
- Genetic parallelization



https://iopscience.iop.org/article/10.1088/1742-6596/2767/3/032036

Gridded

- Maximizes number of turbines that fit into a boundary in a gridded pattern
- Useful for generating a suboptimal gridded layout



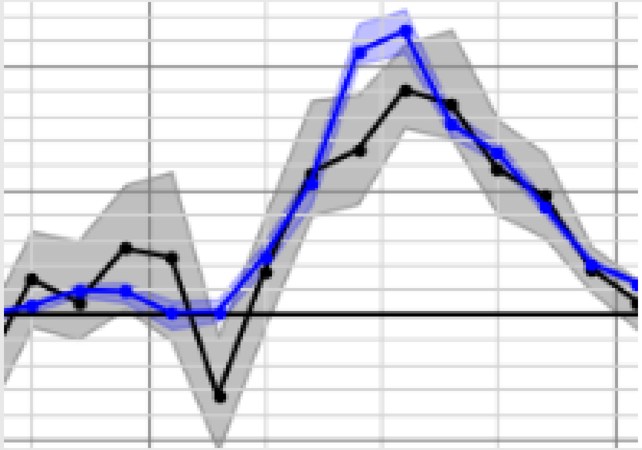
SCADA filtering

Bias correction

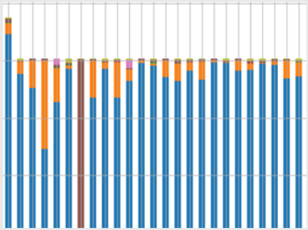
Uplift analysis

Model fitting

FLASC



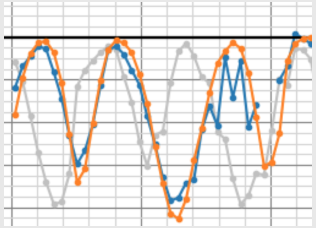
SCADA filtering



Filtering and outlier detection for power curves

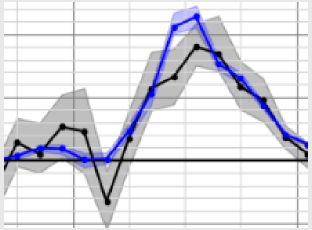
- Abnormal conditions
- Abnormal operation
- Stuck sensors

Bias correction



Correction of northing bias (yaw encoder bias) via wake position comparison

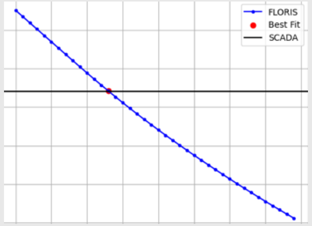
Uplift analysis



Comparison of power and energy production between two or more test cases

- Energy ratio
- Total uplift

Model fitting



Parameter fitting for FLORIS turbine and wake models to SCADA records

- EmG parameters
- Wind dir. variability
- Yaw cosine exponent

FLORIS examples

Thank you

www.nrel.gov

michael.sinner@nrel.gov

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. 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. A portion of this research was performed using computational resources sponsored by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

NREL Transforming ENERGY