Best Practices for Wake Model and Optimization Algorithm Selection in Wind Farm Layout Optimization.

Nicholas F. Baker[*], Andrew P. J. Stanley[†], Jared Thomas[‡], and Andrew Ning[§]
Brigham Young University, Provo, Utah 84602.
Katherine Dykes[¶]
National Renewable Energy Laboratory, Golden, Colorado 80401

This paper presents the results of two discrete case studies regarding the Wind Farm Layout Optimization (WFLO) problem. Case study (1) considers variations in optimization strategies for a simplified Bastankhah Gaussian wake model, while case study (2) studies trade offs in performance with variation in both physics model and optimization strategy selection. For (1), a supplied wake model outputs Annual Energy Production (AEP) given participant input turbine locations. For (2), participants calculate AEP using a wake model of their choice, while also implementing their preferred optimization method. Participant submissions for optimized turbine locations were then compared to Large Eddy Simulator (LES) calculations for AEP, and results were measured for both quality and accuracy.

I. Introduction

Optimizing turbine placement within a wind farm is a complex problem characterized by many local minima. The large number of inter-dependent variables involved in Wind Farm Layout Optimization (WFLO) create a design space that can quickly become intractable.

Two approaches have been taken to simplify the WFLO problem, as described by Padron et.al.[?] The first approach aims at improving the quality of individual models for wind farm attributes, i.e., aerodynamics, atmospheric physics, turbine structures, etc. As they go on to state, "The second approach is to improve the optimization problem formulation, and the algorithms [used] to solve the optimization."[?]

To better model the aerodynamics of the waked airflow region of a wind turbine, complex computational methods such as such as Direct Numerical Simulations (DNS) or Large Eddy Simulations (LES) have been developed. But the computational time these require for full simulation can be prohibitive in multi-iterative optimizations. Simplified Engineering Wake Models (EWMs) make certain limiting physics assumptions, resulting in greatly reduced computational costs.[1] Yet these simpler, less accurate approximations can sometimes lead to inefficient recommendations for turbine placement, due to what can be inaccurate assumptions in specific wind farm scenarios.

Given a single EWM, optimization methods to select ideal turbine locations are limited by characteristics of the functions governing the model. For example, EWMs that define a discontinuous wind speed behind wind turbines cannot be effectively used with gradient-based optimization methods, and models for which gradients have not been calculated are limited to gradient-free algorithms or gradient-based with finite difference derivatives. Additionally, within these limitations different optimization strategies have varying capacity to escape local minima in the pursuit of a global optimum.

To better understand the differences in EMW selection and optimization algorithm application, we have created two discrete case studies. These studies are designed to involve participants from many different research labs working on the WFLO problem. The first isolates optimization techniques for a single simplified EWM, the second observes the differences when combining variations in EWM selection and optimization method.

Explain how people haven't done this before

History/bio of IEA

history/bio of Task37

History/bio of NREL

II. Methodology

Two of the major factors contributing to superior turbine placement recommendations are 1) EWM characteristics and 2) optimization algorithm. We have therefore designed two distinct case studies in an attempt to quantify the effects of alterations in each of these variables.

[*]Masters Student, Brigham Young University Department of Mechanical Engineering
[†]Ph.D. Candidate, Brigham Young University Department of Mechanical Engineering
[‡]Ph.D. Student, Brigham Young University Department of Mechanical Engineering
[§]Assistant Professor, Brigham Young University Department of Mechanical Engineering
[¶]Senior Engineer, National Wind Technology Center

To isolate variability in the second factor (optimization method), we pre-coded a representative wake model as a control variable, and permit participants to use any optimization strategy they think will discover turbine locations which deliver the best Annual Energy Production (AEP) for the farm. This first Case Study is called the Optimization Only Case Study, and is described below in Section B.

Isolating the other factor (EWM variability) proves more complicated. An EWM's compatibility with gradient-based or gradient-free optimization methods dictate which algorithms can be applied. As such, designing a case study that restricts participants to a single optimization algorithm would unnecessarily limit the scope of EWMs studied. With the aim of acquiring as much empirical data as possible in order to determine best practices for the industry as a whole, our second case study permits not only participant selection of EWM, but also implemented optimization algorithm. It is called the Combined Physics Model/Optimization Algorithm Case Study, and is described below in Section C.

To enable production of useful data, both case studies would require a model wind farm with characteristics that are simultaneously restrictive enough to maintain simplicity, yet general enough to maintain relevance to more complex and realistic problems. The wind farm scenarios selected for the case studies are described below in Section A.

A. Common to Both Case Studies

1. Turbine

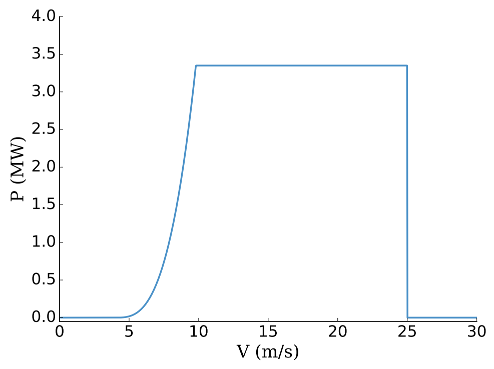
We decided to use NREL's generalized 3.35MW reference turbine in all wind farms. It's attributes are open source, and it is designed as a baseline for onshore wind turbine specifications.[2] The specifics of the turbine necessary for our simplified Gaussian wake model (used in Case Study 1) are located below:

Table 1: Attributes for NREL's 3.35MW onshore reference turbine

Rotor Diameter	130	m
Turbine Rating	3.35	MW
Cut-In Wind Speed	4	m/s
Rated Wind Speed	9.8	m/s
Cut-Out Wind Speed	25	m/s

Its power curve is defined as:

$$P(V) = \begin{cases} 0 & V < V_{\textit{cut-in}} \\ P_{\textit{rated}} \cdot \left(\frac{V - V_{\textit{cut-in}}}{V_{\textit{rated}} - V_{\textit{cut-in}}} \right)^3 & V_{\textit{cut-in}} \leq V < V_{\textit{rated}} \\ P_{\textit{rated}} & V_{\textit{rated}} \leq V < V_{\textit{cut-out}} \\ 0 & V \geq V_{\textit{cut-out}} \end{cases}$$



2. Farm Geography

To focus on optimization method and EWM variability, as well as to avoid introducing too many unecessary variables, the wind farms for all scnearios are on flat and level terrain.

Boundary Shape
As experienced by other researchers such as DUDE and OTHER DUDE, square and non-radially symmetric farm boundaries tend to give optimal turbine locations where turbines cluster at the farm corners or protrusions. For this reason, each wind farm scenario has a radially symmmetric circular boundary centered at $(0, 0)$. To simplify computations, this boundary restricts turbine hub (x, y) locations, not the outside of a turbine's blade radius. In terms of spacing, no turbine can be less than two rotor diameters from any other turbine.

Farm Diameter
Farm diameter sizing for each scenario needs to be restrictive enough to avoid simply placing all turbines on the boundary, yet also permit meaningful turbine movement by the optimizers. The following algorithm was implemented to determine both farm boundary diameters and the example layouts (described in the next section) for all farms in the Studies:

1. Place one turbine in the center of the farm
2. Organize remaining turbines in concentric circles, maintaining a minimum of 5 diameter spacing both laterally and between rings
3. If the outermost ring has less than 5 turbines:

(a) Expand penultimate ring as little as possible to fit these outermost turbines, while maintaining the lateral spacing requirement

4. Expand radius of outermost ring to the nearest 100 m.

5. Evenly expand the inner turbine rings for uniformity

Example Layouts
These layouts are displayed below graphically in Fig. 1. Though primarily for Case Study 1, they are provided to participants of both studies in `.yaml` format for 4 reasons:

1. To provide an example of the `.yaml` schema formatting, desired by NREL to descirbe turbine placement
2. To help participants visualize the different farm sizes and geography
3. To verify AEP calculation if participants implement the CAse Study 1 wake model in a different language
4. To provide participants with an optimization starting point if necessary

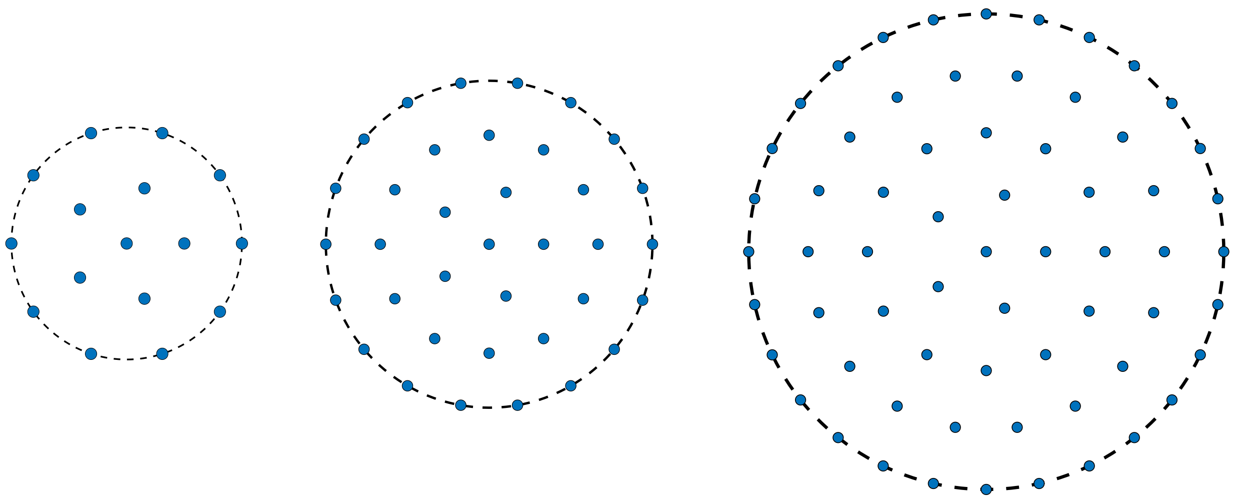


Figure 1: Example layouts for 16, 36, and 64 turbine farms

It is explicitly stated in the announcement document that these layouts are only examples. Participants are not required to use these as starting points for their optimizations, though they have the option to do so.

3. Wind Attributes

Again, to avoid introducing unecessary variables, the wind distribution frequency and wind speed are the same for all wind farm scenarios in both Case Studies.

Wind Speed
Freestream wind velocity is constant throughout the farm, at 9.8 m/s, regardless of turbine location or time of day. 9.8 m/s is used because it is the minimum wind speed for the NREL 3.35MW reference turbine to produce its rated power. Using this incoming wind velocity, turbine wakes will push air speeds down the turbine's power curve, and more local minima will be experienced by participant optimizers. If the incoming wind speed are too high (i.e. 13 m/s), some turbine layouts, though different in location, will produce the same power output, and avoid incentivizing optimizers to find the most efficient locations.

Wind Direction Frequency
In creating these Case Studies, we desired a wind direction frequency distribution (displayed graphically in a wind rose) that meets two criteria:

1. Creates many local minima
2. Permits few unique methods of reaching an optima

If there is a lack of local minima in a design space, even inefficient optimizers can find the best result. In a design space characterized by local minima, only the most robust optimization methods are able to escape these traps. For this reason, we desired a windrose that could produce many local minima.

If, on the other hand, despite the presence of local minima, many non-similar layouts produce high and similar AEP values, it would likewise not tell us much regarding the abilities of participant optimization methods. When many answers give superior results, algorithms are not incentivized to find novel or unique solutions.

Therefore to acheive both of these criteria, we experimented with 4 different patterns of wind direction frequency:

1. Bi-modal uniaxis (to simulate a canyon geography)

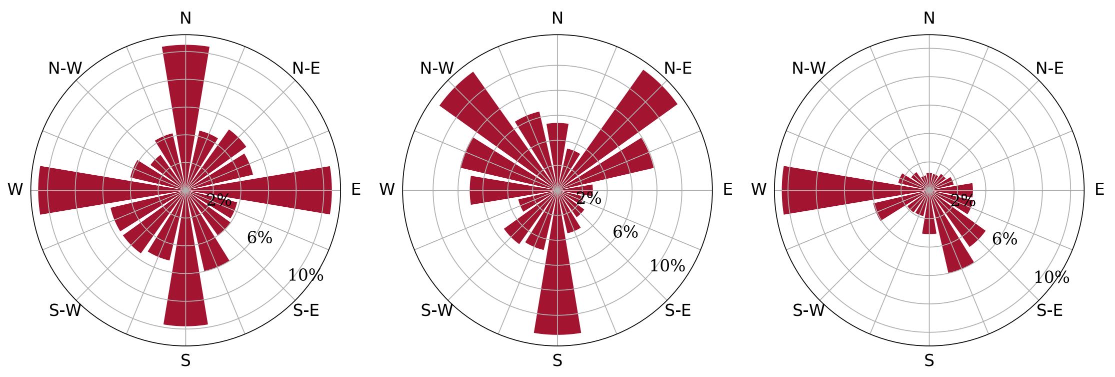
2. Quad-directional (experienced at some onshore airports)
3. Tri-direcitonal (simulating an off-shore enviornment)
4. Bi-modal off-axis (experienced in both on- and off-shore locations)

Simulations were run using the simplified Gaussian wake model and the optimization package SNOPT. We ran each wind rose through 1000 random-start optimizations. All wind roses gave generally similar optimization results with AEP distributions following a bell-curve pattern. There were slight differences, however, in that:

1. Bi-modal uniaxis gave turbine arrangements that tended towards a straight line, perpendicular to the uniaxis of the wind. Since it met our first disqualifying criteria of a lack of local minima, it was not further analyzed.
2. Quad-directional gave turbine arrangements that tended towards a grid pattern. Since this too didn't have many local minima to trap the optimizer, it was disqualified.
3. Tri-directional gave many gird-like arrangements that all had similar AEP values. Since it met our second disqualifying criteria of non-similar layouts producing similar AEP, it was not selected.
4. Bi-modal off-axis gave few results with high AEP values. We interpreted this to be indicitive of the presence of many local minima. Since this best met our selection criteria, it is the wind rose utilized for our Case Studies.

The wind roses are binned for 16 directions. The quad-directional, tri-directional, and bi-modal off-axis roses are dipectied in Fig. 2, and histograms of the 1000 random start optimizations are in Fig. 3.

Figure 2: Quad-directional, Tri-directional, and Bi-modal off-axis wind roses



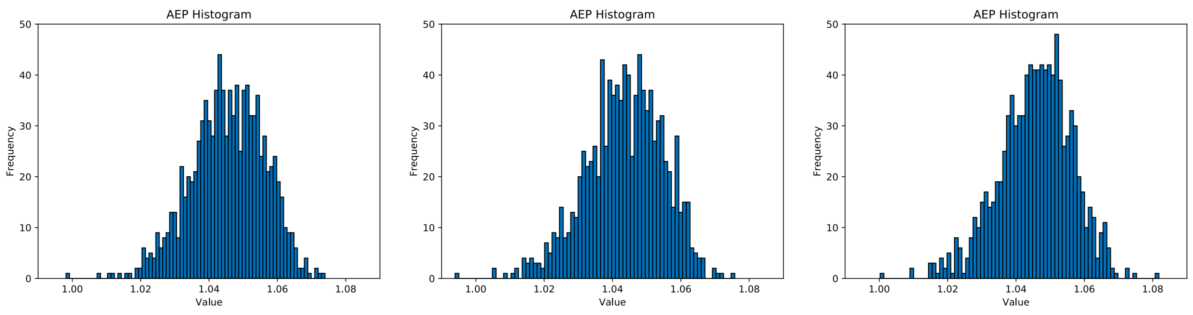


Figure 3: 1000 random start optimization runs for the Quad-, Tri-, and Bi-modal off-axis wind roses

After the above described study, we selected the bi-modal off axis wind frequency distribution for our Case Studies, shown graphically below. A greater magnitude in the radial direction from the origin indicates a higher frequency from that cardinal direction.

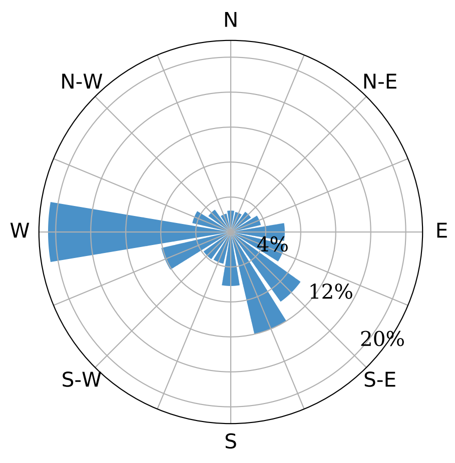


Figure 4: The wind frequency distribution for our Case Studies

4. Data File Type

One request made by leaders of IEA Task 37 was to implement their open source `.yaml` schema for all necessary data. This included:

- Farm turbine attributes
- Farm turbine locaitons
- Farm wind frequency and wind speeds

There exists a separate IEA37 working group tasked with refining the precise schema of these items, whose work is outside the scope of our specific Case Studies. Though a current work in progress, we implemented the most recent iteration and adapted them to our scenarios as necessary.

The above mentioned data schema format is supplied to participants of both cases. They are:

1. `iea37-windrose.yaml` - binned wind frequency for both Case Studies, in `.yaml` format
2. `iea37-335mw.yaml` - data for reference turbine used in both Case Studies, in `.yaml` format

Further examples for reporting turbine locations in `.yaml` schema, as well as a Python parser for all these schema is made available to Case Study participants. Since their use is more applicable to the Optimization Only Case Study, they are described in Section B.

B. Optimization Only Case Study

The purpose of this Case Study is to determine the best optimization practices for WFLO, using a single representative EWM.

1. Wake Model

A simplified version of Bastankhah's Gaussian wake model[3,4] is used, since it is compatible with both gradient-based and gradient-free methods, and is computationally inexpensive in comparison to LES and DNS methods. This wake model is described by the following equations:

$$\frac{\Delta U}{U_\infty} = \left(1 - \sqrt{1 - \frac{C_T}{8\sigma_y^2/D^2}}\right) \exp\left(-0.5\left(\frac{y-\delta}{\sigma_y}\right)^2\right) \tag{1}$$

Where $\frac{\Delta U}{U_\infty}$ is the wake velocity deficit, $C_T = \frac{8}{9}$ and is the thrust coefficient, $y - \delta$ is the distance of the point of interest from the wake center in the cross-stream horizontal direction, D is the turbine diameter, and σ_y is the standard deviation of the wake deficit in the cross-stream horizontal direction as defined in Eq. (2):

$$\sigma_y = k_y(x) + \frac{D}{\sqrt{8}} \tag{2}$$

In Eq. (2), x is the downstream distance from the turbine generating the wake to the turbine of interest, and D is the turbine diameter. k_y is determined as a function of turbulence intensity (I). In this case study turbulence intensity is treated as a very small constant of 0.075, and we therefore use a coresponding k_y of 0.0324555.[?,5]

Increasing turbulence intensity has numerous effects and draws attention away from the main purpose of this Case Study, which is to observe the differences of optimization strategies. Though these strategies may differ for a wake model with a higher turbulence intensity, this first IEA Task 37 set of Case Studies uses a very low intensity in an attempt to minimize the considered variables.

2. *Farm Sizes*

Variability in wind farm size effect optimization algorithm performance. To account for this, 3 wind farm sizes are specified in Case Study 1: 16, 36, and 64 turbines. Inclusion of 3 farm sizes is to avoid a bias towards algorithms optimized for wind farms of a specific size, and in order to observe how increased complexity correlates to convergence time and algorithm performance.

The turbine numbers are selected as perfect squares which roughly double in size. Perfect squares are used to permit even grid turbine arrangements, if desired.

Example `.yaml` schema is provided to all participants, to help understand the format with which they will need to report their optimal turbine locations. These files are:

- `iea37-ex16.yaml` - 16 turbine scenario example layout
- `iea37-ex36.yaml` - 36 turbine scenario example layout
- `iea37-ex64.yaml` - 64 turbine scenario example layout

These example layouts are depicted graphically above in Fig. 1.

3. *Supplied Code*

To enable participation in this Case Study, we created and supplied a pre-coded Python package. This package includes:

- Turbine charactersitics, wind frequency, and wind speed in NREL's `.yaml` schema described in Section 4
- Example turbine layouts for each farm size (in `.yaml` format), displayed graphically in Fig. 1
- Python parsers of the `.yaml` schema, included in the Appendix
- Python target function to calculate AEP (given `.yaml` turbine locations and farm attributes)

We selected the programming language Python since it is open source and widely used by researchers in the industry.

Participant alteration to our specific code implementation, or replication of our model in another language is permitted if needed for compatibility with participant optimization methods. This is with the understanding ,however, that final wind farm layouts will be evaluated with the original Python code package that we provided.

C. Combined Physics Model/Optimization Algorithm Case Study

The intent of this Case Study is to assess not only the optimization methods measured by Case Study 1, but also the effects that different physics model approximations have on turbine location recommendations.

Case Study 2 differs from the previous one in that 1) no wake model is provided, and 2) only a single wind farm size is to be optimized. Participants are free to chose their preferred EWM and optimization method combination. Comparison of participant results is based on:

1. **Quality:** Which participant results give the highest LES-calculated AEP.
2. **Accuracy:** Which participant wake-model-calculated AEP most closely match an LES-calculated AEP for the same turbine locations.

Unlike Case Study 1, participant reported AEP is not comparable, since different EWMs (which account for different physics phenomena) are used to calculate them. Due to this, all participant-reported optimized turbine locations will be run through the same LES for comparison. With the inherent bias each EWM has for its own optimized locations removed, reported turbine locations will be measured using the same simulation tool to compare AEP.

The LES we will use is produced by NREL and called the Simulator fOr Wind Farm Applications (SOWFA). Unfortunately, due to the computional time requirements, all participant submissions were not able to be run through SOWFA before the writing of this document. However separate from the LES calculations, results are analyzed through the cross-comparison portion of this Case Study.

D. Cross-Comparison Analysis

After the initial call for results, each participant's proposed optimal turbine locations in the standardized `.yaml` format was published to the other Combined Case Study participants. This was done with the intent that each participant will use their own wake model to calculate the AEP of the other proposed farm layouts. Though the baseline of LES gives an even playing field, from this portion of the Case Study we hope to learn if any participant's results are seen as superior by other EWMs.

E. Farm Attributes

The wind farm size for this case study is limited to 9 turbines, in order to limit the LES computation time requirements when assessing results. The previously described method under **Farm Diameter** was used to determine the boundary distance, and the wind rose and wind speed are the same as Case Study 1.

III. Results

A. Case Study 1: Optimization Only

Each participant ran the optimization algorithm and implementation of their choosing using our suppied AEP target function, or an equivalent duplicaiton in another language. Since there exists a great deal of variability in hardware, participants also reported processor speed, function calls, number of cores utilized, and amount of RAM installed their system when finding their optimized results. The AEP results and rankings are given below in Table 4

There were 10 submission for the Optimization Only Case Study. One participant submitted twice, using a different optimization method for each submission. These two submissions will be hereafter treated as if they were from different participants, and are assigned different participant numbers. For anonimity, each submission is given a number. We will referred to each submission below by this number (i.e. participant 1, ..., participant 10, etc.).

1. 16 Turbine Case

Though wall time and number of function calls varied widely, the best 4 (of 10) participant submissions used gradient-based algorithms.

Rank	AEP	Participant #	Gradients
1	418924.4064	4	Yes
2	412251.1945	8	Yes
3	414141.2938	5	Yes
4	411182.2200	1	Yes
5	409689.4417	2	No
6	408360.7813	10	Yes
7	402318.7567	3	No
8	392587.8580	7	No
9	388758.3573	6	No
10	388342.7004	9	No
example layout	366941.5712	11	N/A

Table 2: Participant results of the 16 turbine scenario

2. 36 Turbine Case

Rank	AEP	Participant #	Gradients
1	863676.2993	4	Yes
2	851631.931	10	Yes
3	849369.7863	2	No
4	846357.8142	8	Yes
5	844281.1609	1	Yes
6	828745.5992	3	No
7	813544.2105	9	No
8	777475.7827	7	No
9	776000.1425	6	No
example layout	737883.0985	11	N/A

Table 3: Prticipant results of the 36 turbine scenario

3. 64 Turbine Case

Rank	AEP	Participant #	Gradients
1	1513311.194	4	Yes
2	1506388.415	2	No
3	1480850.976	10	Yes
4	1476689.663	1	Yes
5	1455075.608	3	No
6	1425678.143	8	Yes
7	1422268.714	9	No
8	1364943.008	6	No
9	1332883.433	7	No
example layout	1294974.298	11	

Table 4: Prticipant results of the 64 turbine scenario

B. Case Study 2: Combined

Not enough time/resources for LES right now.

Cross-Comparison of results was conducted.

IV. Conclusion

Appendix

Acknowledgments

This work is funded by Brigham Young University.

V. Announcement Document

Wind Farm Layout Optimization Case Studies

IEA Task 37 on System Engineering in Wind Energy

Nicholas F. Baker, Andrew P. J. Stanley, Jared Thomas, and Andrew Ning
Brigham Young University, Provo, Utah, USA
Katherine Dykes
National Renewable Energy Laboratory, Golden, Colorado, USA

August 23, 2018

1 Introduction

Two major factors that affect wind farm layout optimization are 1) the optimization approach and 2) the wake model. This document defines two case studies designed to study these factors. One may elect to participate in either or both cases.

1. Optimization-Only Case Study: user chooses optimization approach, wake model is fixed and supplied.
2. Combined Case Study: user is free to choose both optimization approach and wake model.

Participants will (1) optimize turbine locations to maximize annual energy production, (2) submit solutions, and (3) provide details on their methodology. After all submissions are received, for the Combined Case Study participants will be expected to perform a cross comparison of other participant solutions. Data will be consolidated, processed, and made available to all participants.

2 Problem Definition

Objective

The objective of each scenario is to maximize annual energy production, which we define simply as the expected value of aerodynamic power. The wind resource for each case has a wind rose binned into 16 discrete directions, with a constant wind speed. In other words:

$$AEP = \left(\sum_{i=1}^{16} f_i P_i \right) 8760 \frac{\text{hrs}}{\text{yr}}$$

where P_i is the power produced for wind direction i, and f_i is the corresponding wind direction probability.

Design Variables

The design variables are the (x, y) locations of each turbine. All locations in this document refer to the hub location. Every turbine in the farm is identical, and explicitly defined below in **Parameters**.

Constraints

Each wind farm scenario has a fixed circular boundary centered at $(0, 0)$. All turbine (x, y) locations must remain on or within this boundary. No turbine can be less than two rotor diameters from any other turbine.

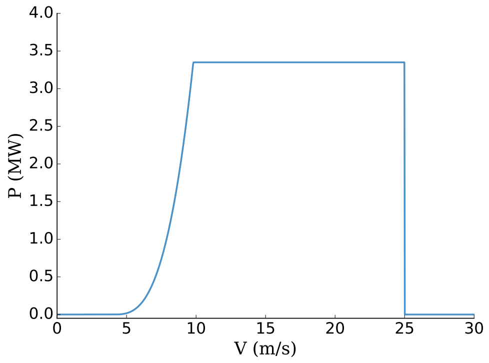
Parameters

The wind turbine is the IEA37 3.35 MW onshore reference turbine [1] with the following characteristics:

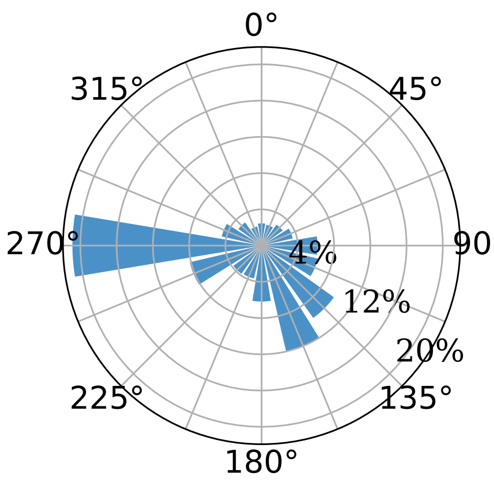
Rotor Diameter	130	m
Turbine Rating	3.35	MW
Cut-In Wind Speed	4	m/s
Rated Wind Speed	9.8	m/s
Cut-Out Wind Speed	25	m/s

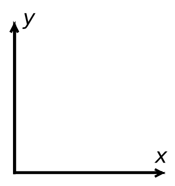
All turbine data is also contained in the enclosed `iea37-335mw.yaml`. The power curve is defined as:

$$P(V) = \begin{cases} 0 & V < V_{cut\text{-}in} \\ P_{rated} \cdot \left(\frac{V - V_{cut\text{-}in}}{V_{rated} - V_{cut\text{-}in}} \right)^3 & V_{cut\text{-}in} \leq V < V_{rated} \\ P_{rated} & V_{rated} \leq V < V_{cut\text{-}out} \\ 0 & V \geq V_{cut\text{-}out} \end{cases}$$



The farm wind speed for all scenarios is constant at 9.8 m/s. The +y axis is coincident with 0°, and the CW wind rose is defined by 16 discrete bins tabulated in `iea37-windrose.yaml`, depicted pictorially below:





2.1 Case Study 1: Optimization Only

This problem defines three different wind farm sizes, and corresponding number of turbines, intended to test scalability of your optimization approach. The three scenarios are:

1. 16 turbines, boundary radius of 1,300 m.
2. 36 turbines, boundary radius of 2,000 m.
3. 64 turbines, boundary radius of 3,000 m.

For this Case Study the user is only free to choose the optimization approach. The wake model is fixed and is a simplified version of Bastankhah's Gaussian wake model [2, 3, 4]. A Python implementation is supplied for convenience (`iea37-aepcalc.py`). Alterations to this implementation are permitted, as long as the governing physics equations are not altered. Participants may use other programming languages, but must use the same physics equations. To aid with this, the relevant equations are defined in a separate document (`iea37-wakemodel.pdf`), and example wind farm layouts with corresponding AEP values are provided in the `iea37-ex##.yaml` files to verify implementations. The example designs are only for verification, and do not need to be used as starting points in your optimization.

2.2 Case Study 2: Combined

This problem defines one scenario where the user is free to choose both the optimization algorithm and the wake model. The single wind farm scenario is nine turbines with a boundary radius of 900 m.

If needed by your wake model choice, the turbulence intensity is 0.075, and the wind shear is a power-law with a shear exponent of 0.15 using the hub height as the reference height.

3 Reporting and Evaluation

Participants will submit:

1. Optimal turbine placement solution for each scenario, using the `.yaml` format from the enclosed example layouts.
2. A survey describing your methodology and simulation environment here.

Note that for both Case Studies, your `.yaml` submissions must report both total farm AEP, and farm AEP for each binned wind direction, as in the enclosed `iea37-ex##.yaml` examples.

3.1 Case Study 1: Optimization Only

Results will be compared by running the enclosed `iea37-aepcalc.py`, which will read the submitted `.yaml` file from each participant. Submissions must adhere to the `.yaml` format in order to receive a ranking. While other implementations may be used in the optimization, all evaluations will be done with the provided `iea37-aepcalc.py` code, so it is essential that you check that your implementation is consistent.

The command-line syntax we will use to evaluate all submitted files is:

```
$python iea37-aepcalc.py iea37-yourname-opt##.yaml
```

Where:

- `iea37-yourname-opt##.yaml` will be your submitted `.yaml` of optimal turbine locations.
  - "`yourname`" is your personal or organizational name, all lowercase with no spaces or punctuation.
  - "`##`" is the scenario size, i.e. "opt16" would be for the 16-turbine scenario.

The following two files must be referenced internally by your submission, as is done by the example layouts:

- `iea37-windrose.yaml` describes the binned wind rose used in both case studies.
- `iea37-335mw.yaml` lists the turbine data for the used IEA37 3.35 MW onshore reference turbine.

3.2 Case Study 2: Combined

Because the wake models differ in this Case Study, determining a "best" solution is generally not possible. Comparisons will be made using two approaches:

1. Every participant will evaluate every other participant's solutions using their own wake model(s). It is essential that the `.yaml` format is adhered to so that cross-comparisons are painless.
2. Each solution will be compared using a higher-fidelity simulation, in this case large-eddy simulations (LES) using SOWFA. This simulation introduces its own modeling assumptions and is an imperfect way to compare, but does provide another piece of information on relative performance between approaches.

4 Enclosures

Files included with this document, needed for full participation in the Case Studies are:

- `iea37-aepcalc.py` - Python coding of AEP wake model for the Optimization Only Case Study
- `iea37-wakemodel.pdf` - description of AEP algorithm for the Optimization Only Case Study
- `iea37-windrose.yaml` - binned wind frequency for both Case Studies, in `.yaml` format
- `iea37-335mw.yaml` - data for reference turbine used in both Case Studies, in `.yaml` format
- `iea37-ex16.yaml` - 16 turbine scenario example layout
- `iea37-ex36.yaml` - 36 turbine scenario example layout
- `iea37-ex64.yaml` - 64 turbine scenario example layout

References

[1] Bortolotti, P., Dykes, K., Merz, K., Sethuraman, L., and Zahle, F., "IEA Wind Task 37 on System Engineering in Wind Energy, WP2 - Reference Wind Turbines," Tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO., May 2018.

[2] Thomas, J. J. and Ning, A., "A method for reducing multi-modality in the wind farm layout optimization problem," *Journal of Physics: Conference Series*, The Science of Making Torque from Wind, Milano, Italy, June 2018.

[3] Bastankhah, M. and Porté-Agel, F., "A new analytical model for wind-turbine wakes," *Renewable Energy*, January 2014.

[4] Bastankhah, M. and Porté-Agel, F., "Experimental and theoretical study of wind turbine wakes in yawed conditions," *J. Fluid Mech.*, Vol. 806, 2016, pp. 506–541.

VI. Python Code

```python
"""IEA Task 37 Combined Case Study AEP Calculation Code

Written by Nicholas F. Baker, PJ Stanley, and Jared Thomas (BYU FLOW lab)
Created 10 June 2018
Updated 11 Jul 2018 to include read-in of .yaml turb locs and wind freq dist.
Completed 26 Jul 2018 for commenting and release
Modified 22 Aug 2018 implementing multiple suggestions from Erik Quaeghebeur:
    - PEP 8 adherence for blank lines, length(<80 char), var names, docstring.
    - Altered multiple comments for clarity.
    - Used print_function for compatibility with Python 3.
    - Used structured datatype (coordinate) and recarray to couple x,y coords.
    - Removed unused variable 'sWindRose' (getTurbLocYAML).
    - Removed unecessary "if ... < 0" case (WindFrame).
    - Simplified calculations for sin/cos_wind_dir (WindFrame).
    - Eliminated unecessary calculation of 0 values (GaussianWake, DirPower).
    - Turbine diameter now drawn from <.yaml> (GaussianWake)
    - Used yaml.safe_load.
    - Modified .yaml reading syntax for brevity.
    - Removed some (now) unused array initializations.
"""

from __future__ import print_function   # For Python 3 compatibility
import numpy as np
import sys
import yaml                             # For reading .yaml files
from math import radians as DegToRad    # For converting degrees to radians

# Structured datatype for holding coordinate pair
coordinate = np.dtype([('x', 'f8'), ('y', 'f8')])


def WindFrame(turb_coords, wind_dir_deg):
    """Convert map coordinates to downwind/crosswind coordinates."""

    # Convert from meteorological polar system (CW, 0 deg.=N)
    # to standard polar system (CCW, 0 deg.=W)
    # Shift so North comes "along" x-axis, from left to right.
    wind_dir_deg = 270. - wind_dir_deg
    # Convert inflow wind direction from degrees to radians
    wind_dir_rad = DegToRad(wind_dir_deg)

    # Constants to use below
    cos_dir = np.cos(-wind_dir_rad)
    sin_dir = np.sin(-wind_dir_rad)
    # Convert to downwind(x) & crosswind(y) coordinates
    frame_coords = np.recarray(turb_coords.shape, coordinate)
    frame_coords.x = (turb_coords.x * cos_dir) - (turb_coords.y * sin_dir)
    frame_coords.y = (turb_coords.x * sin_dir) + (turb_coords.y * cos_dir)

    return frame_coords


def GaussianWake(frame_coords, turb_diam):
    """Return each turbine's total loss due to wake from upstream turbines"""
    # Equations and values explained in <iea37-wakemodel.pdf>
    num_turb = len(frame_coords)

    # Constant thrust coefficient
    CT = 4.0*1./3.*(1.0-1./3.)
    # Constant, relating to a turbulence intensity of 0.075
    k = 0.0324555
    # Array holding the wake deficit seen at each turbine
    loss = np.zeros(num_turb)

    for i in range(num_turb):            # Looking at each turb (Primary)
        loss_array = np.zeros(num_turb)  # Calculate the loss from all others
        for j in range(num_turb):        # Looking at all other turbs (Target)
            x = frame_coords.x[i] - frame_coords.x[j]   # Calculate the x-dist
            y = frame_coords.y[i] - frame_coords.y[j]   # And the y-offset
            if x > 0.:                   # If Primary is downwind of the Target
                sigma = k*x + turb_diam/np.sqrt(8.)  # Calculate the wake loss
                # Simplified Bastankhah Gaussian wake model
                exponent = -0.5 * (y/sigma)**2
                radical = 1. - CT/(8.*sigma**2 / turb_diam**2)
                loss_array[j] = (1.-np.sqrt(radical)) * np.exp(exponent)
            # Note that if the Target is upstream, loss is defaulted to zero
        # Total wake losses from all upstream turbs, using sqrt of sum of sqrs
        loss[i] = np.sqrt(np.sum(loss_array**2))

    return loss


def DirPower(turb_coords, wind_dir_deg, wind_speed,
             turb_diam, turb_ci, turb_co, rated_ws, rated_pwr):
    """Return the power produced by each turbine."""
    num_turb = len(turb_coords)

    # Shift coordinate frame of reference to downwind/crosswind
    frame_coords = WindFrame(turb_coords, wind_dir_deg)
    # Use the Simplified Bastankhah Gaussian wake model for wake deficits
    loss = GaussianWake(frame_coords, turb_diam)
    # Effective windspeed is freestream multiplied by wake deficits
    wind_speed_eff = wind_speed*(1.-loss)
    # By default, the turbine's power output is zero
    turb_pwr = np.zeros(num_turb)

    # Check to see if turbine produces power for experienced wind speed
    for n in range(num_turb):
        # If we're between the cut-in and rated wind speeds
        if ((turb_ci <= wind_speed_eff[n])
                and (wind_speed_eff[n] < rated_ws)):
            # Calculate the curve's power
            turb_pwr[n] = rated_pwr * ((wind_speed_eff[n]-turb_ci)
                                       / (rated_ws-turb_ci))**3
        # If we're between the rated and cut-out wind speeds
        elif ((rated_ws <= wind_speed_eff[n])
                and (wind_speed_eff[n] < turb_co)):
            # Produce the rated power
            turb_pwr[n] = rated_pwr

    # Sum the power from all turbines for this direction
    pwrDir = np.sum(turb_pwr)

    return pwrDir


def calcAEP(turb_coords, wind_freq, wind_speed, wind_dir,
            turb_diam, turb_ci, turb_co, rated_ws, rated_pwr):
    """Calculate the wind farm AEP."""
    num_bins = len(wind_freq)  # Number of bins used for our windrose

    #  Power produced by the wind farm from each wind direction
    pwr_produced = np.zeros(num_bins)
    # For each wind bin
    for i in range(num_bins):
        # Find the farm's power for the current direction
        pwr_produced[i] = DirPower(turb_coords, wind_dir[i], wind_speed,
                                   turb_diam, turb_ci, turb_co,
                                   rated_ws, rated_pwr)

    #  Convert power to AEP
    hrs_per_year = 365.*24.
    AEP = hrs_per_year * (wind_freq * pwr_produced)
    AEP /= 1.E6  # Convert to MWh

    return AEP


def getTurbLocYAML(file_name):
    """ Retrieve turbine locations and auxiliary file names from <.yaml> file.

    Auxiliary (reference) files supply wind rose and turbine attributes.
    """
    # Read in the .yaml file
    with open(file_name, 'r') as f:
        defs = yaml.safe_load(f)['definitions']

    # Rip the x- and y-coordinates (Convert from <list> to <ndarray>)
    turb_xc = np.asarray(defs['position']['items']['xc'])
    turb_yc = np.asarray(defs['position']['items']['yc'])
    turb_coords = np.recarray(turb_xc.shape, coordinate)
    turb_coords.x, turb_coords.y = turb_xc, turb_yc

    # Rip the expected AEP, used for comparison
    # AEP = defs['plant_energy']['properties']
    #            ['annual_energy_production']['default']

    # Read the auxiliary filenames for the windrose and the turbine attributes
    ref_list_turbs = defs['wind_plant']['properties']['layout']['items']
    ref_list_wr = (defs['plant_energy']['properties']
                   ['wind_resource_selection']['properties']['items'])

    # Iterate through all listed references until we find the one we want
    # The one we want is the first reference not internal to the document
    # Note: internal references use '#' as the first character
    fname_turb = next(ref['$ref']
                      for ref in ref_list_turbs if ref['$ref'][0] != '#')
    fname_wr = next(ref['$ref']
                    for ref in ref_list_wr if ref['$ref'][0] != '#')

    # Return turbine (x,y) locations, and the filenames for the others .yamls
    return turb_coords, fname_turb, fname_wr


def getWindRoseYAML(file_name):
    """Retrieve wind rose data (bins, freqs, speeds) from <.yaml> file."""
    # Read in the .yaml file
    with open(file_name, 'r') as f:
        props = yaml.safe_load(f)['definitions']['wind_inflow']['properties']

    # Rip wind directional bins, their frequency, and the farm windspeed
    # (Convert from <list> to <ndarray>)
    wind_dir = np.asarray(props['direction']['bins'])
    wind_freq = np.asarray(props['probability']['default'])
    # (Convert from <list> to <float>)
    wind_speed = float(props['speed']['default'])

    return wind_dir, wind_freq, wind_speed


def getTurbAtrbtYAML(file_name):
    '''Retreive turbine attributes from the <.yaml> file'''
    # Read in the .yaml file
    with open(file_name, 'r') as f:
        defs = yaml.safe_load(f)['definitions']
        op_props = defs['operating_mode']['properties']
        turb_props = defs['wind_turbine_lookup']['properties']
        rotor_props = defs['rotor']['properties']

    # Rip the turbine attributes
    # (Convert from <list> to <float>)
    turb_ci = float(op_props['cut_in_wind_speed']['default'])
    turb_co = float(op_props['cut_out_wind_speed']['default'])
    rated_ws = float(op_props['rated_wind_speed']['default'])
    rated_pwr = float(turb_props['power']['maximum'])
    turb_diam = float(rotor_props['radius']['default']) * 2.

    return turb_ci, turb_co, rated_ws, rated_pwr, turb_diam


if __name__ == "__main__":
    """Used for demonstration.

    An example command line syntax to run this file is:

        python iea37-aepcalc.py iea37-ex16.yaml

    For Python .yaml capability, in the terminal type "pip install pyyaml".
    """
    # Read necessary values from .yaml files
    # Get turbine locations and auxiliary <.yaml> filenames
    turb_coords, fname_turb, fname_wr = getTurbLocYAML(sys.argv[1])
    # Get the array wind sampling bins, frequency at each bin, and wind speed
    wind_dir, wind_freq, wind_speed = getWindRoseYAML(fname_wr)
    # Pull the needed turbine attributes from file
    turb_ci, turb_co, rated_ws, rated_pwr, turb_diam = getTurbAtrbtYAML(
        fname_turb)

    # Calculate the AEP from ripped values
    AEP = calcAEP(turb_coords, wind_freq, wind_speed, wind_dir,
                  turb_diam, turb_ci, turb_co, rated_ws, rated_pwr)
    # Print AEP for each binned direction, with 5 digits behind the decimal.
    print(np.array2string(AEP, precision=5, floatmode='fixed',
                          separator=', ', max_line_width=62))
    # Print AEP summed for all directions
    print(np.around(np.sum(AEP), decimals=5))
```

References

[1]Herbert-Acero, J. F., Probst, O., Réthoré, P.-E., Larsen, G. C., and Castillo-Villar, K. K., "A Review of Methodological Approaches for the Design and Optimization of Wind Farms," *Energies*, 2014, pp. 23.

[2]Bortolotti, P., Dykes, K., Merz, K., Sethuraman, L., and Zahle, F., "IEA Wind Task 37 on System Engineering in Wind Energy, WP2 - Reference Wind Turbines," Tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO., May 2018.

[3]Bastankhah, M. and Porté-Agel, F., "Experimental and theoretical study of wind turbine wakes in yawed conditions," *J. Fluid Mech.*, Vol. 806, 2016, pp. 506–541.

[4]Thomas, J. and Gebraad, P., "Improving the FLORIS Wind Plant Model for Compatibility with Gradient-Based Optimization," , pp. 14.

[5]Niayifar, A. and Porté-Agel, F., "Analytical Modeling of Wind Farms: A New Approach for Power Prediction," *Energies*, September 2016.