Serial-Refine Method for Fast Wake-Steering Yaw Optimization

Paul A. Fleming, Andrew P. J. Stanley, Christopher J. Bay, Jennifer King, Eric Simley, Bart M. Doekemeijer, Rafael Mudafort

National Wind Technology Center, National Renewable Energy Laboratory, Golden, CO, 80401, USA

E-mail: paul.fleming@nrel.gov
Keywords: Wake Steering; Optimization; FLORIS; Wind Farm Control

Abstract. In this paper we present the Serial-Refine method for quickly finding the optimal yaw angles in wake steering. The method optimizes turbine angles serially from upstream to downstream using a small number of candidate angles. The presented results show that Serial-Refine finds solutions that are at least as good as former conventional optimization approaches but that require much less computation time.

1. Introduction

Wake steering is a form of wind farm control in which turbines use intentional yaw misalignment to deflect and alter their wakes such that the total farm production is increased through reduced wake losses. The turbine yaw offsets to be applied to each turbine for a particular wind speed and direction are typically found via optimizations of an engineering model of wakes and wake steering, such as the open-source FLOw Redirection and Induction in Steady State (FLORIS) model [1]. In field experiments to date, the yaw offsets are computed ahead of time for every wind direction and wind speed combination and collected in a lookup table for use in on-line control. Turbulence intensity can be included as a third variable.

Typically, this optimization is accomplished using gradient-based or nonlinear and nonconvex optimization methods. Within the FLORIS framework for example, tools are included for identifying the optimal yaw angles that rely on either SciPy's sequential least squares programming (SLSQP) solver [2] or pyOptSparse [3]. Both of those methods are generalized global and nonconvex optimization methods that perform a wide search over the space of yaw angles. This optimization problem can become very time consuming to solve because the number of dimensions over which to optimize is equal to the number of turbines in the wind farm. With wind farms often containing tens of to more than a hundred wind turbines, dimensionality is a nontrivial issue. A hundred-turbine wind farm optimization over the entire wind rose of directions and speeds may take hours to days of computing time on computer clusters.

The speed of optimization has been acceptable in the past because the optimal yaw offsets are precomputed and the number of turbines involved is not large (for example, a field test of five turbines [4]). However, as the size of wind farms continues to grow, there is an increasing need for optimization methods that find optimal yaw angles quickly and that scale well with the number of turbines.

Table 1. Default options for the SLSQP optimization algorithm in FLORIS

Option	**Value**
Initial guess for optimal yaw angles	$[0.0°, 0.0°, \ldots, 0.0°]$
Tolerance for normalized cost function	10^{-12}
Maximum number of iterations	50
Step size used for numerical approximation of Jacobian	0.1

An example of where the problem of dimensionality becomes apparent is in the coupled design problem of layout and wind farm control, in which the layout of a wind farm is codesigned with the wind farm controller. In other words, the effect of the wind farm controller is accounted for in the wind farm design (e.g., [5]). This requires a nested optimization in which the optimal yaw angles are calculated for an entire farm over the entire wind rose during each evaluation of a wind farm layout in the higher-level layout optimization problem. A second example application that requires fast discovery of optimal yaw angles is on-line, real-time wind farm control, in which the engineering model is continually reestimated, and the yaw angles are likewise reoptimized using the updated model [6]. Such a real-time control algorithm would typically happen on the time scale of seconds to minutes. Thirdly, large-scale studies over many farms and over years of atmospheric conditions would make use of improved optimization time to reduce computation cost in time and processing power, which can otherwise be extensive [7].

Stanley et al. [8] introduce a new concept for yaw optimization meant to more quickly converge to the optimal yaw angles compared to global gradient-based optimization methods. The algorithm in [8] sorts the turbines according to the freestream wind direction and then considers for each turbine in order, a Boolean comparison between a zero yaw angle and a maximum yaw angle (e.g., 20°). This much simpler optimization method makes the assumption that the optimal yaw angle of a turbine has no correlation with the yaw angles of the turbines in its wake. The computational cost for this method scales approximately linearly with the number of turbines rather than exponentially. Stanley et al. empirically confirmed that this method captures the majority of the gain using many fewer calculations.

In this paper, we propose an adaptation of the method by Stanley et al. [8] called the "Serial-Refine" (SR) method. The Serial-Refine method allows the discovery of angles that provide comparable or better power uplifts than the default gradient-based and nonlinear optimization methods currently provided with FLORIS. Moreover, the computational cost is one to multiple orders of magnitude lower because the SR method scales linearly with the number of turbines rather than exponentially.

2. Yaw-Angle Optimization

As described, designing a wake-steering controller typically involves finding the set of yaw angles that yield the highest power production, according to the engineering model being optimized, for a given set of turbines and for a given wind speed and direction (as well as for other potential conditions such as turbulence intensity).

The FLORIS software framework includes a variety of wake models and wake-steering optimization algorithms. Currently, the default wake-steering optimization algorithm in FLORIS leverages the SLSQP gradient-based optimization method provided by the SciPy Python library [2]. The default convergence criteria of this method are listed in Table 1. The SLSQP optimization method has been used to design several past field-implemented wake-steering controllers [4, 9].

In the remainder of this article, we compare the Serial-Refine method to the default SLSQP optimization method in FLORIS. It is important to note that this work is not meant to criticize

the SLSQP method but rather to compare the serialized optimization framework against a known and effective global gradient-based optimization method. It is also important to note that the results and performance of the SLSQP method are dependent on the parameters picked in Table 1. However, we believe any multivariable, global gradient-based optimization must incur computation costs similar to the default settings we have adopted through experience.

2.1. Serial-Refine Method

We define the Serial-Refine method for wake-steering optimization in Algorithm 1.

Algorithm 1 Serial-Refine Algorithm.
The turbines must be sorted in order from upstream to downstream prior to running the algorithm

$yaw_{opt} \leftarrow [0°, 0°, \ldots, 0°]$ ▷ Initialize optimal yaw vector to 0° for all turbines
$power_{opt} \leftarrow computeFarmPower(yaw_{opt})$ ▷ Initialize optimal power
for $i \leftarrow 1, N_{turbine} - 1$ **do** ▷ First Pass, note don't optimize last turbine
  $yaw_{test} \leftarrow yaw_{opt}$
  for $yawAngle \leftarrow [-30°, -15°, 0°, 15°, 30°]$ **do**
    $yaw_{test}[i] \leftarrow yawAngle$
    $power_{test} \leftarrow computeFarmPower(yaw_{test})$
    if $power_{test} \geq power_{opt}$ **then**
      $power_{opt} \leftarrow power_{test}$
      $yaw_{opt} \leftarrow yaw_{test}$
    end if
  end for
end for
for $i \leftarrow 1, N_{turbine} - 1$ **do** ▷ Refine Pass
  $yaw_{test} \leftarrow yaw_{opt}$
  for $yawAngle \leftarrow yaw_{opt}[i] + [-7.5°, -3.75°, 0°, 3.75°, 7.5°]$ **do**
    $yaw_{test}[i] \leftarrow yawAngle$
    $power_{test} \leftarrow computeFarmPower(yaw_{test})$
    if $power_{test} \geq power_{opt}$ **then**
      $power_{opt} \leftarrow power_{test}$
      $yaw_{opt} \leftarrow yaw_{test}$
    end if
  end for
end for
return yaw_{opt}

The Serial-Refine method works as follows: As in [8], the turbines are first sorted by most upstream to most downstream. The algorithm then passes through each turbine twice. On the first ("Serial") pass, we step through the turbines one by one, evaluating N_{Yaw} different yaw angles for each (default $N_{Yaw} = 5$) and proceeding with the angle that yields the highest wind farm power production. On the second ("Refine") pass, we again step through the turbines one by one, evaluating N_{Yaw} different yaw angles for each. The evaluated yaw angles are now in proximity of the previously found yaw angle. For example, if the yaw angle limits in the first pass are [-30°, 30°], the first pass might consider the yaw angles [-30°, -15°, 0°, 15°, 30°] for each turbine. Then, the second pass will consider the angle found in the first pass, plus an offset range of [-7.5°, -3.75°, 0°, 3.75°, 7.5°]. Figure 1 shows the optimal yaw angles selected for an aligned row of five turbines for both the SR and SLSQP optimization methods (left), and the yaw angles explored for each turbine by the SR method (right).

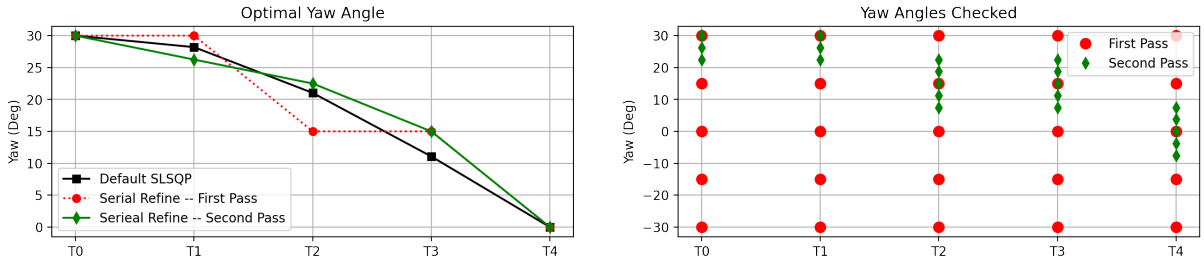

Figure 1. Left: Optimal angles for a row of five turbines found using the default SLSQP method versus those found in the first and second passes of SR. Right: Angles explored for each turbine on each pass in SR.

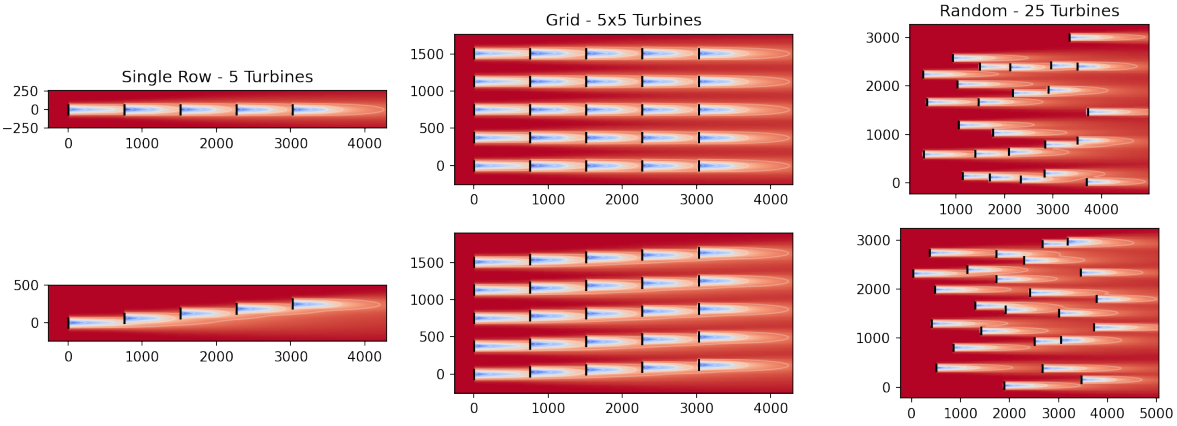

Figure 2. The three optimization scenarios considered in this work: (Left) a single row of turbines with varying angle to the wind, (Middle) a grid of turbines with varying angle, and (Right) randomized wind farm layouts. All x and y lengths are given in meters.

3. Case Studies

We now consider a set of case studies in which we compare the effectiveness of Serial-Refine against the default SLSQP-based algorithm. For several farm types, the optimal yaws were found using both approaches, and the quality of the solutions (i.e., the resulting uplift in wind farm power production) as well as the time to reach those solutions are compared.

The three farm types considered are shown in Fig. 2 and consist of a row of turbines, a regular grid of turbines, and a randomized set of wind turbines. In each case, solutions were found for several wind directions (in the case of the row and grid) or for different randomized wind farm layouts in the random farm case. The optimizations were further performed for different numbers of turbines in each case. Finally, three candidate settings for the SR algorithm were considered: $N_{Yaw} = 3$, $N_{Yaw} = 5$, and $N_{Yaw} = 7$, where N_{Yaw} is the number of evaluated angles per turbine and per pass. In all scenarios, the yaw angles were bounded to [-30°, 30°].

3.1. Row Optimizations

The results for the cases using a row of turbines are given in Fig. 3. The left-most subplot compares the wind farm power production with the yaw angles found by the SR method and those found by the SLSQP method. Whereas the SR yields a lower wind farm power production

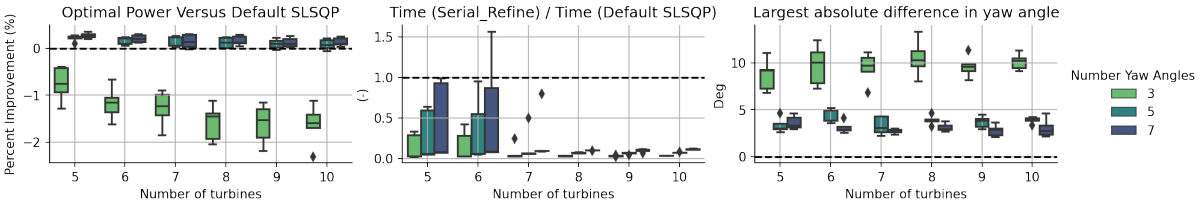

Figure 3. Results for the case of row optimizations using boxplots to summarize the statistics across wind directions. (Left) Percent improvement in final power. (Middle) The ratio of time taken by the SR method to the time taken by the default SLSQP method. (Right) The maximum difference in yaw angle.

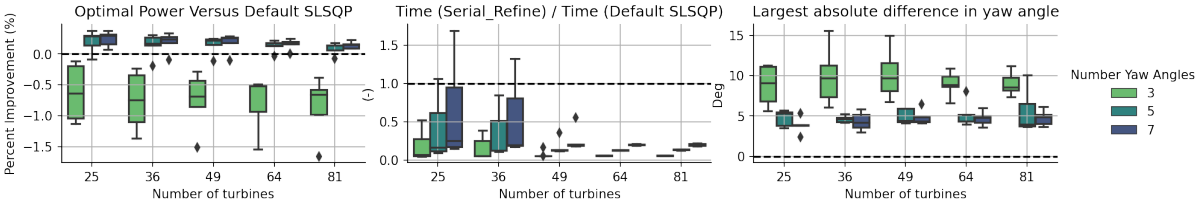

Figure 4. Results for the case of grid optimizations using boxplots to summarize the statistics across wind directions. (Left) Percent improvement in final power. (Middle) The ratio of time taken by the SR method to the time taken by the default SLSQP method. (Right) The maximum difference in yaw angle.

when $N_{Yaw} = 3$, both $N_{Yaw} = 5$ and $N_{Yaw} = 7$ allow the SR method to outperform the SLSQP method. Considering the change in time to find the solution (the middle subplot), we see a substantial improvement in performance. For example, the SR solutions with $N_{Yaw} = 5$ take only one-tenth the time of the SLSQP method to find the optimal solution for each case for the seven-turbine array. Finally, the right-most subplot compares the largest difference in yaw angle observed and finds a result similar to the SLSQP method for the $N_{Yaw} = 5$ and $N_{Yaw} = 7$ SR optimizations, confirming that $N_{Yaw} = 3$ might be too coarse.

3.2. Grid Optimizations

We next consider the results for the grid of turbines in Fig. 4. Note that in this case the number of turbines simulated grows much larger, with 81 (9 x 9) turbines being considered in one case. This result in many ways matches and underscores that of the row. Using $N_{Yaw} = 5$ or $N_{Yaw} = 7$ allows Serial-Refine to converge to a better solution than SLSQP. In terms of time performance, all versions of Serial-Refine are faster than the default SLSQP approach. However, the use of $N_{Yaw} = 5$ is significantly faster than $N_{Yaw} = 7$ while being roughly equivalent in quality of solution, suggesting a declining benefit of adding additional resolution above $N_{Yaw} = 5$ across cases studied so far.

3.3. Random Wind Farm Optimizations

Finally, we consider the results for optimizing the yaw angles of the randomly laid-out wind farms in Fig. 5. The pattern observed for the earlier cases is repeated here: Serial-Refine with $N_{Yaw} = 5$ or $N_{Yaw} = 7$ yaw angles produces results that improve upon those of the default SLSQP method by similar amounts, while $N_{Yaw} = 5$ is substantially faster. It is useful to note that in this case the yaw angle solution found by SR is meaningfully improved versus the solution found by SLSQP. For example, for the case of the 81-turbine random layout considered,

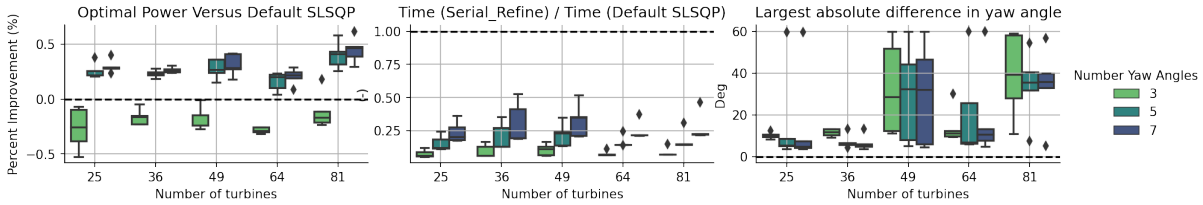

Figure 5. Results for the case of random layout optimizations using boxplots to summarize the statistics across random instantiations. (Left) Percent improvement in final power. (Middle) The ratio of time taken by the SR method to the time taken by the default SLSQP method. (Right) The maximum difference in yaw angle.

the median gain from the no-yaw baseline rose from 4.27% (default SLSQP) to 4.53% (SR with $N_{Yaw} = 5$).

4. Conclusions

This paper presented the new Serial-Refine algorithm for quickly identifying optimal yaw angles for wake steering through the optimization of an engineering model of wakes and wake steering. Compared to the default SLSQP implementation included in the FLORIS software framework, which has been used in past campaigns and research studies, Serial-Refine is shown to be around 10 times faster while producing slightly better final optimization results.

It is important to note that changes to these options would change the baseline we compare to, but we believe the principle will hold that a multivariable, gradient-based optimization using finite differences will in any case require more wake calculations than checking a fixed number of angles in series.

Based on the success of these results, we now include Serial-Refine as the default yaw optimization in the newly released FLORIS v3 [10].

Acknowledgements

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

References

[1] National Renewable Energy Laboratory 2022 FLORIS Wake Modeling Utility URL `https://github.com/NREL/floris`
[2] SciPy SLSQP `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html` accessed: 2021-09-16
[3] Wu N, Kenway G, Mader C A, Jasa J and Martins J R R A 2020 *Journal of Open Source Software* **5** 2564
[4] Fleming P, King J, Simley E, Roadman J, Scholbrock A, Murphy P, Lundquist J K, Moriarty P, Fleming K, van Dam J, Bay C, Mudafort R, Jager D, Skopek J, Scott M, Ryan B, Guernsey C and Brake D 2020 *Wind Energy Science* **5** 945–958
[5] Gebraad P, Thomas J J, Ning A, Fleming P and Dykes K 2017 *Wind Energy* **20** 97–107
[6] Doekemeijer B M, van der Hoek D and van Wingerden J W 2020 *Renewable Energy* **156** 719–730 ISSN 0960-1481 URL `https://www.sciencedirect.com/science/article/pii/S0960148120305358`

[7] Bensason D, Simley E, Roberts O, Fleming P, Debnath M, King J, Bay C and Mudafort R 2021 *Journal of Renewable and Sustainable Energy* **13** 033303 (*Preprint* https://doi.org/10.1063/5.0039325) URL https://doi.org/10.1063/5.0039325
[8] Stanley A P J, Bay C, Mudafort R and Fleming P 2022 *Wind Energy Science* **7** 741–757 URL https://wes.copernicus.org/articles/7/741/2022/
[9] Simley E, Fleming P and King J 2019 Field validation of wake steering control with wind direction variability *NAWEA* (Amherst, Massachusetts)
[10] National Renewable Energy Laboratory 2022 FLORIS v3.0rc1 URL https://www.nrel.gov/wind/assets/pdfs/floris-version-3-0rc1.pdf