2023-1

Sparsity for Gradient-Based Optimization of Wind Farm Layouts

Benjamin T. Varela
Brigham Young University - Provo, ben.varela@me.com

Andrew Ning
Brigham Young University - Provo, aning@byu.edu

Original Publication Citation

Varela, B. T. and Ning, A., "Sparsity for Gradient-Based Optimization of Wind Farm Layouts," AIAA SCITECH Forum, National Harbor, MD, Jan 2023. doi: /10.2514/6.2023-1543

Sparsity for Gradient-Based Optimization of Wind Farm Layouts

Benjamin T. Varela ∗ and Andrew Ning † ‡
Brigham Young University, Provo, Utah, 84602 United States

Optimizing wind farm layouts is an important step in designing an efficient wind farm. Optimizing wind farm layouts is also a difficult task due to computation times increasing with the number of turbines present in the farm. The most computationally expensive part of gradient-based optimization is calculating the gradient. In order to reduce the expense of gradient calculation, we performed a study on the use of sparsity in wind farm layout optimization. This paper presents the findings of the sparsity study and provides a method to use sparsity in wind farm layout optimization. We tested this sparsity method by optimizing multiple farms with sparse methods and compared the results to optimizations of the same farms using traditional methods. By using the sparse method to optimize multiple farms, we found that the objective results were comparable between sparse and traditional methods and that sparse methods were 4 times faster than traditional methods on average. We expect more speedups with improved methodology and larger wind farms. By using sparse methods, it is possible to solve the wind farm layout optimization problem more efficiently, thus allowing for a more thorough study of the wind farm layout design space without excessive computational costs. Further work is required to refine the method and prepare for testing on real-world wind farm layout applications.

I. Introduction

Due to the world's constant need for energy, the use of wind turbines has grown significantly. The need for more renewable energy has given rise to "wind farms," a collection of turbines located in a small area. Whereas more turbines typically mean more energy produced, the total energy produced by a wind farm is not simply a factor of the number of turbines in the farm, but also depends on where the turbines are placed [1]. Turbines placed side by side may experience the same wind speeds, but turbines behind other turbines sit in the wake of the turbines in front and produce less energy because they experience lower wind speeds [2]. Whereas avoiding waked turbines is simple for a single wind direction, most wind farms experience different wind directions over time. In order to avoid the problem of waked turbines, it is important to carefully consider the relative placement of each turbine in order to maximize the farm's energy production. The process of finding the best turbine placement is referred to as wind farm layout optimization.

Gradient-free optimization methods are quite popular for solving wind farm layout optimization problems because they can traverse multi-modal design spaces, but as potential wind farms get larger and more variables are introduced, the performance of gradient-free methods is reduced [3]. Due to the increasing complexity of the wind farm layout optimization problem, there has been recent interest in gradient-based optimization methods. This interest in gradient-based methods has led to the development of methods to handle the multi-modality of wind farm layout optimization problems [4].

Gradient-based optimization methods require the gradient of the objective function. Because there is no practical analytical method for calculating the gradient for wind farm layout problems, for traditional methods it is necessary to compute the gradient numerically at every optimization iteration. The repeated computation of the gradients is the most expensive part of wind farm layout optimization. The reduction of computational expense is the reason for this study in which we exploited sparsity to compute the gradient more efficiently.

Using sparsity is not a new idea, and there has been significant study into methods for leveraging sparsity efficiently for nonlinear systems of equations and optimizations [5, 6]. Leveraging sparsity in gradient-based optimization reduces the time required to compute Jacobians [7].

The wind farm layout optimization problem is an excellent candidate for sparsity because the energy produced by a turbine in a wind farm is only affected by the wake of a few turbines in front of it. Because turbine energy is affected by a limited number of other turbines, there are many turbine-turbine interactions that can be ignored. By ignoring these

∗Ph.D. Candidate, Mechanical Engineering.
†Associate Professor, Mechanical Engineering, AIAA Associate Fellow
‡Joint Appointment, National Renewable Energy Laboratory

turbine-turbine interactions, sparsity is introduced and can be leveraged in calculating the energy Jacobian. The energy Jacobian can then be used to calculate the gradient.

Stanley and Ning report wind farm layout optimizations with 100 turbines using gradient-based methods taking anywhere from 1,000 to 100,000 function calls to converge [8]. Even though 100 turbines is not an unreasonable number of turbines, with so many function calls it can be expensive, computationally, to optimize. The sparsity method we use here can significantly reduce the amount of time overall by minimizing the amount of time each function call requires.

One limitation to using sparsity for wind farm layout optimization is that the turbine-turbine interactions change during the optimization process. In order to account for these changes, it is necessary to modify which turbine-turbine interactions can be ignored during optimization. Updating these interactions repeatedly would effectively eliminate the effect of sparse methods. However, for most sparse problems, the sparsity pattern could be fixed or be updated only occasionally while still maintaining the intended accuracy. That said, the wind farms used in this work are relatively small. We expect that better results could be obtained by using larger farms. In other words, for this small case, the amount of sparsity is small, but as the wind farm size grows, the meaningful interactions will become a smaller percentage of all possible interactions.

Our goal is to develop a method to use sparsity to improve the performance of gradient-based methods when applied to the wind farm layout optimization problem.

II. Methods

Annual energy production (AEP) is the total energy produced by a wind farm in one year. The farm AEP is the weighted sum of the energy of the farm at every wind state (Eq. 1). In Eq. 1, p_s represents the probability of a given wind state occurring during a one-year period. By differentiation of Eq. 1, the gradient of the AEP becomes the sum of the gradients of the energy of the farm at every wind state (Eq. 2). Similarly, the energy at any given wind state is the sum of the energy for each turbine at that wind state, so the gradient of the energy for a wind state is the sum of the gradients of each turbine at that given wind state (Eq. 3).

$$AEP = \sum_{\text{states}} E_s * p_s \tag{1}$$

$$\nabla(AEP) = \sum_{\text{states}} \nabla(E_s) * p_s \tag{2}$$

$$\nabla(E_s) * p_s = \sum_{\text{turbines}} \nabla(e_t) * p_s \tag{3}$$

The gradients of the energy produced for each turbine can be represented in a matrix that we call the energy Jacobian. For example, a wind farm with 6 turbines in which only the x and y positions of each turbine are considered as design variables would have an energy Jacobian with 6 rows (1 for each turbine energy) and 12 columns (one for each design variable). Figure 1 shows a small wind farm undergoing a single wind state.

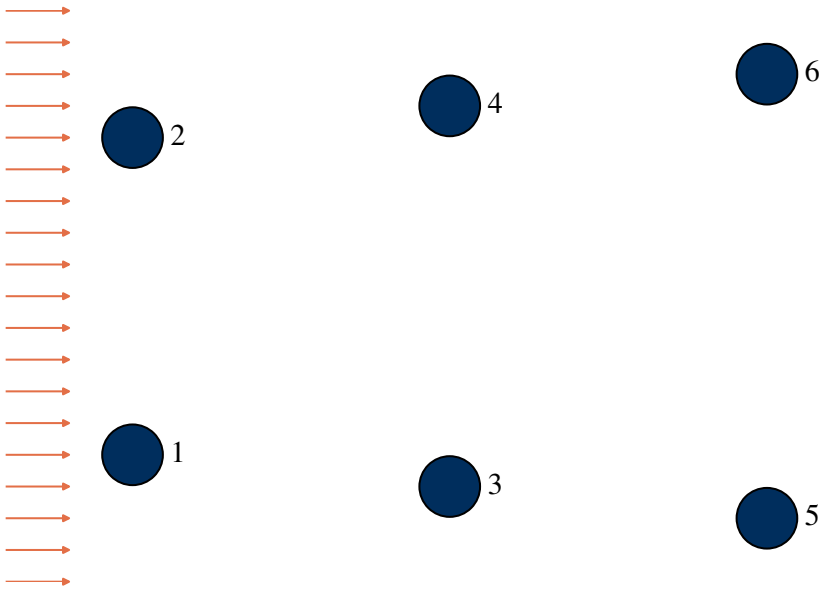

Fig. 1 This small farm is used for description of the energy Jacobian and graph coloring, where x/D refers to the x position normalized by the rotor diameter. The arrows indicate the direction of the freestream wind.

The following energy Jacobian (Eq. 4) is the energy Jacobian for the small wind farm in Fig. 1, using only one wind direction, with rounded values for convenience. Values below machine precision have been removed. The small values that would appear in the Jacobian are caused by the fact that Gaussian wake models never reach an influence of zero. Even if turbines are spaced extremely far apart, Gaussian wake models will still calculate some influence between them.

All the values shown are scaled so that the AEP of the farm is between 0 and 1 which is a common practice in optimization. The first element in row 1 represents how the x position of turbine 1 affects the energy output of turbine 1, the second element in row 1 represents how the x position of turbine 2 affects the energy or turbine 1, the seventh element in row 1 represents how the y position of turbine 1 affects the energy output of turbine 1, and so on for all the other elements. This configuration defines each row as $\nabla(e_t)$, the gradient of the energy of a turbine in a single wind state. As a note, the first two rows are all zeros because turbines 1 and 2 are in freestream wind and small movements will not change their energy outputs.

$$
\begin{array}{c}
 \\
\nabla(e_1) \\
\nabla(e_2) \\
\nabla(e_3) \\
\nabla(e_4) \\
\nabla(e_5) \\
\nabla(e_6)
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & y_1 & y_2 & y_3 & y_4 & y_5 & y_6
\end{array} \\
\left[\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{-6e-6} & 0 & \text{6e-6} & 0 & 0 & 0 & \text{2e-2} & 0 & \text{-2e-2} & 0 & 0 & 0 \\
0 & \text{-6e-6} & 0 & \text{6e-6} & 0 & 0 & 0 & \text{-2e-2} & 0 & \text{2e-2} & 0 & 0 \\
\text{5e-5} & 0 & \text{-6e-6} & 0 & \text{-5e-5} & 0 & \text{-2e-4} & 0 & \text{2e-2} & 0 & \text{-2e-2} & 0 \\
0 & \text{5e-5} & 0 & \text{-6e-6} & 0 & \text{-5e-5} & 0 & \text{2e-4} & 0 & \text{-2e-2} & 0 & \text{2e-2}
\end{array}\right]
\end{array}
\tag{4}
$$

As shown in Eq. 3, the summation of each column results in $\nabla(E_s)$, the gradient of the energy produced by the wind farm for a single wind state. For every wind state applied to a farm, there exists an energy Jacobian and therefore a $\nabla(E_s)$. Eq. 2 shows that the summation of these $\nabla(E_s)$ provides the gradient of the energy produced by the farm. These gradients, accounting for each wind direction, are the gradients needed for wind farm layout optimization.

A. Sparsity and Coloring

A sparse Jacobian is a Jacobian that contains mostly elements equal to zero. For optimization, a zero in the Jacobian means that an output is independent of an input. Because of this relationship, a sparse Jacobian can be compressed by combining columns that do not have nonzero elements in the same row position, such as x_5 and x_6 in Eq. 4. Another compression method is combining rows that do not have nonzero elements in the same column position. In order to calculate a Jacobian numerically, a function must be called repeatedly. For forward algorithmic differentiation and finite

difference methods, the number of function calls required to calculate the Jacobian depends on the number of columns (inputs) in the Jacobian. For reverse algorithmic differentiation, the number of function calls depends on the number of rows (outputs). In this work, we used forward algorithmic differentiation and combined columns to compress Jacobians.

In the examples discussed so far, we calculated the values in the Jacobian beforehand without using sparsity. In practical applications, such as optimization, the Jacobian values change at each iteration, but the placement of nonzero elements often remains the same. By recording the placement of the nonzero elements, we produced a sparsity pattern. The sparsity pattern for Eq. 4 is shown in Eq. 5.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{5}$$

Determining which columns or rows can be combined is a process often referred to as coloring. In this work, the number of colors in a Jacobian refers to how many columns exist in the Jacobian after it has been compressed. An example of coloring and the resulting compression for Eq. 5 is included in Eq. 6.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{6}$$

For a given Jacobian, there are likely to be several different possible coloring schemes that use the same number of colors and can be used for sparse Jacobian calculation. The main objective of coloring is to use as few colors as possible.

Equation 6 shows that the energy Jacobian 4 can be reduced from twelve columns to six columns. This reduction in columns by half of the original columns means that the computational expense for calculating the energy Jacobian with sparsity and coloring is half the expense of calculating it without sparsity and coloring.

The SparseDiffTools [9] package in Julia [10] contains methods for computing a "color vector" that assigns each column a number in place of the color. The ForwardDiff [11] package in Julia contains methods to compute sparse Jacobians using the "color vector" from SparseDiffTools.

B. Optimization Problem Setup

In this work, we performed the optimization of the wind farm with the help of FLOWFarm [4], an open-source wind farm tool designed with gradient-based optimization in mind. From FLOWFarm, we used the Cumulative Curl wake model [12] to determine wake velocities.

We selected the Cumulative Curl model because it was developed to address deep array effects. Deep array effects describe how wake recovery is affected when many wakes interact far downstream [12].

The Cumulative Curl model implements deep array effects by calculating the combined wake effect at each turbine. This is in contrast to other methods of wind farm modeling, which calculate all the velocity deficits with a wake deficit model such as the Bastankah model [13], and then a wake combination model such as the Katic model [14], or the linear local-velocity superposition model [15] superimposes the deficits. The Cumulative Curl model is a single model that replaces both the wake deficit model and wake combination model.

The optimization problem was set up as a wind farm with a circular boundary constraint. Figure 2 shows one of these farms. We constrained the turbines to stay within the red boundary and keep sufficient space between turbines. Figure 2 shows a wind farm with 95 turbines. There is a central turbine with five rings of turbines around the central turbine. The radius of the first ring of turbines is 5.1 turbine diameters, the radius of the second ring is 10.2 turbine diameters and so on for all five rings. The number of turbines per ring is the maximum number of turbines that can exist on that ring while keeping each turbine spaced at least 5.1 turbine diameters apart. The radius of the boundary for the farm is equal to the radius of the fifth ring.

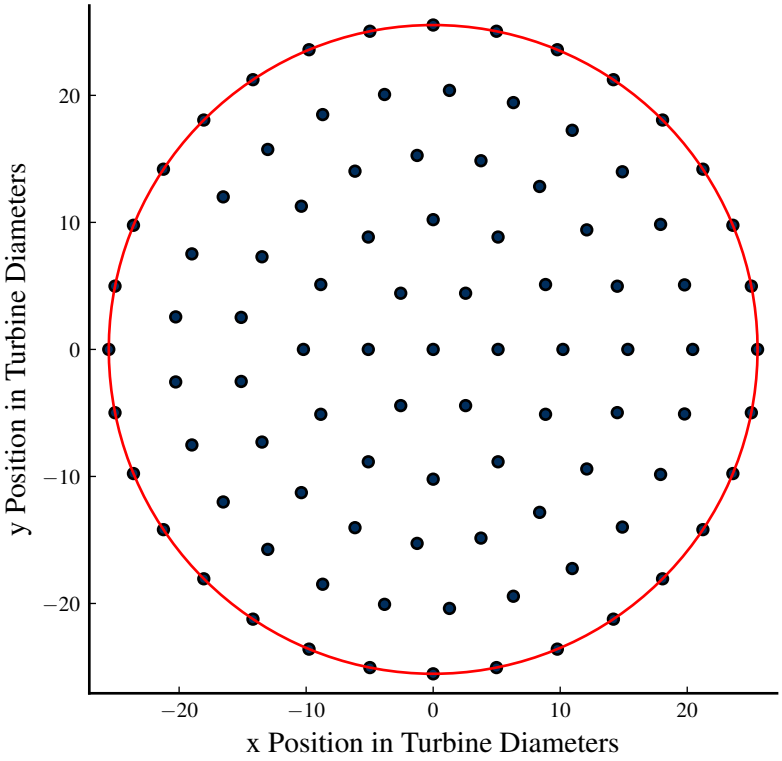

Fig. 2 This figure shows a round wind farm with 95 turbines, each represented by a dot. The turbines are positioned in rings, evenly spaced. The red circle is the boundary of the farm, placed at the fifth ring of turbines.

All round farms are built with the same parameters but use a different number of turbine rings.

Although the examples above demonstrate single wind direction scenarios, it is necessary to use multiple wind directions to create a realistic wind farm layout optimization problem. We used the Nantucket wind rose [16] in 12 directions with each wind speed set to 8 m/s. The Nantucket wind rose is described in Table 1.

Table 1 A description of the Nantucket wind rose that has been divided into 12 directions with all wind speeds set to 8.0 m/s.

Wind Direction (deg)	Wind Speed (m/s)	Probability
10	8.0	7.7%
40	8.0	6.7%
70	8.0	4.4%
100	8.0	4.6%
130	8.0	5.5%
160	8.0	6.7%
190	8.0	9.1%
220	8.0	12.9%
250	8.0	11.7%
280	8.0	9.5%
310	8.0	9.2%
340	8.0	8.2%

C. Implementation of Sparse Jacobians in Optimization

In order to calculate a sparsity pattern for the wind farm, we chose to calculate the full energy Jacobian for the given wind state and drop insignificant values as discussed earlier. We then used the resultant sparse energy Jacobian to define the sparsity pattern and determine the coloring. Calculating the full energy Jacobian is the normal process to calculating the AEP gradient, and so calculating a sparsity pattern from the full energy Jacobian is no faster than just using the full energy Jacobian to get the gradient. The advantage in calculating the sparsity pattern is that the sparsity pattern can be reused for multiple iterations without calculating the full energy Jacobian.

Although the sparsity pattern can be reused for multiple iterations, as the turbines move, the sparsity pattern becomes inaccurate and must be updated. Calculating a new sparsity pattern is described in the previous paragraph. In order to determine when the pattern needs to be updated, we developed a heuristic that depends on the values of the energy Jacobians. A threshold is crucial because it determines which values in the Jacobians are considered significant. A higher threshold means more values are removed from the full energy Jacobian, which decreases computation time for the sparse energy Jacobian. Therefore, the heuristic we selected is when 10 % of the calculated values are at least 2 orders of magnitude below the threshold, the pattern needs to be updated. When multiple wind states are present, this heuristic applies to each wind state's sparsity pattern individually, meaning that it is possible to update the sparsity pattern for a single wind state without updating the sparsity patterns for all wind states.

In order to obtain faster computation, we also chose to include an adaptive threshold. By using a higher threshold at the beginning of an optimization results in patterns with more sparsity, so Jacobians can be calculated more quickly. By using a lower threshold at the end of an optimization the solution can be calculated more accurately. The heuristics for the adaptive threshold were selected by trial and error to achieve shortened optimization times without sacrificing the objective result. We started the adaptive threshold at 1e-8 and decreased the threshold by two orders of magnitude each time the number of sparsity pattern updates reached 5/6 the number of wind states. When the threshold reaches machine precision (1e-16), it is kept constant for the rest of the optimization. We used a change of 2 orders of magnitude so that the threshold would reach machine precision before the end of the optimization.

This is more easily illustrated with an example. For a farm with 12 wind states there are 12 sparsity patterns, and the heuristic for modifying the threshold is 10 pattern updates. The first time the function to compute the AEP gradient is called the threshold is 1e-8. Because no pattern updates have occurred, the threshold remains 1e-8. Because this is the first function call, no sparsity patterns have been determined, so all 12 patterns update and the gradient is computed. On the second function call there have been 12 pattern updates, so the threshold drops to 1e-10 and the pattern updates value is reset to 0.

With these heuristics the threshold reaches machine precision before the end of the optimization but allows the first few iterations to use patterns that are more sparse, so the gradient is computed faster.

III. Results and Discussion

In this section, we first compare the accuracy of this sparse method with that of the non-sparse (or dense) method for different sized farms. Different wind directions complicate matters significantly, because with each wind direction there is a unique Jacobian, with its own colors and different sparsity pattern. So for simplicity in presentation, these first results focus on just a single direction. Next, we optimize a wind farm using 12 wind directions and show that although both methods have similar wake loss, the computational expense for the sparse method is much less than for the dense method.

A. Gradient Accuracy

In order to determine the gradient accuracy, we calculated the AEP gradient for farms of various sizes using the full Jacobian without sparsity and compared to the gradient of the AEP calculated with sparsity. We called any calculations done without sparsity "dense methods" and calculations done with sparsity "sparse methods." The results of the comparison between sparse method gradients and dense method gradients are shown in Fig. 3. The error in the gradient was calculated by subtracting the gradient calculated with sparse methods from the gradient calculated with dense methods, resulting in an error vector. The maximum error plotted in Fig. 3 was the value from the error vector with the largest magnitude. The percent colors were calculated by dividing the number of colors in the compressed energy Jacobian by the number of columns in the original Jacobian. The number of columns was used because the number of columns is equivalent to the number of colors a dense method would use in calculating the Jacobian. The number of colors in the compressed energy Jacobian was changed by changing the threshold used to determine which values in the Jacobian to drop. Higher thresholds drop more values and produce a Jacobian with fewer colors.

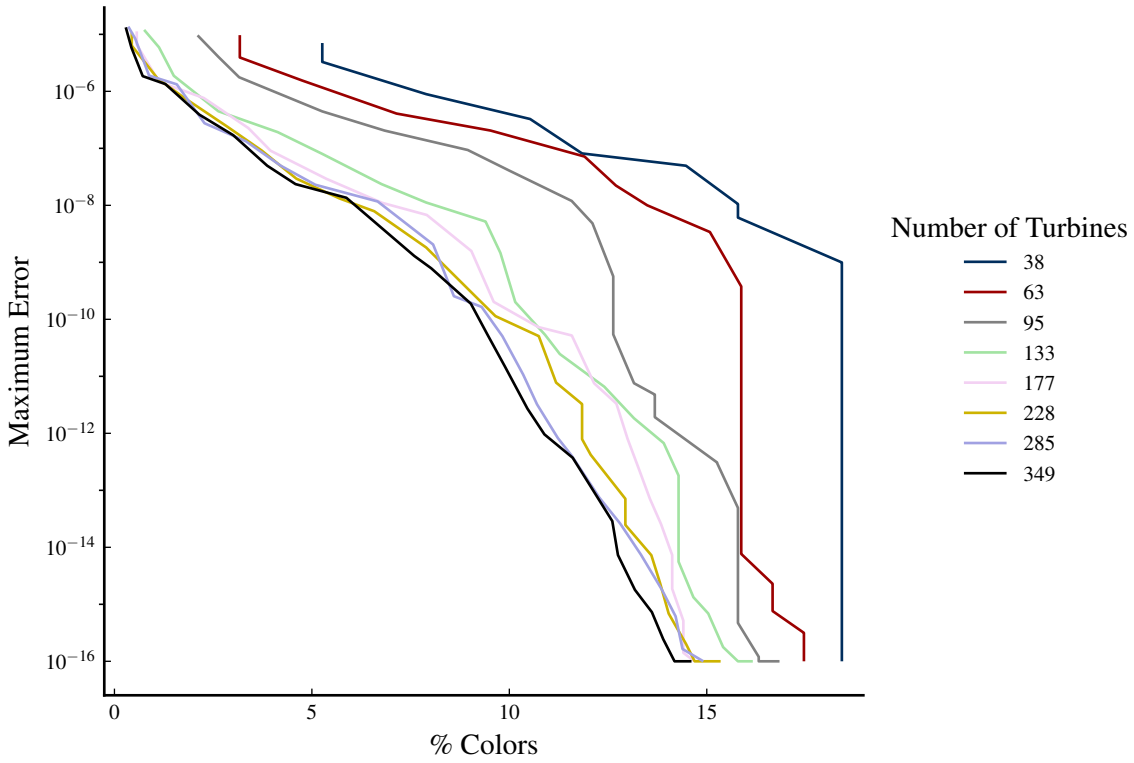

Fig. 3 A graphical representation of the error introduced into the gradient by removing values from the energy Jacobian. The "% Colors" is the percent of colors that exist in the compressed energy Jacobian compared to the maximum colors in the original Jacobian.

From Fig. 3, it is clear that at approximately 15% of the maximum colors the *AEP* gradient reaches machine precision. This 15% colors means that the original energy Jacobian can be compressed to 15% of its original size. Although reaching machine precision is not required, this 15% approximation provides a gradient that is ideal for precise optimization. In other words, 15% of the colors would provide a gradient that is effectively equivalent to the gradient provided by dense methods. Only using 15% of the colors should also mean that computation time should approach 15% of the time taken to calculate the gradient with the full energy Jacobian.

Figure 4 plots the same data as Fig. 3 but instead shows how the percent of colors changes based on the desired accuracy of the gradient and the size of the farm. From Fig. 4 it is clear that significantly fewer colors are required for a gradient with lower accuracy, regardless of the size of the wind farm.

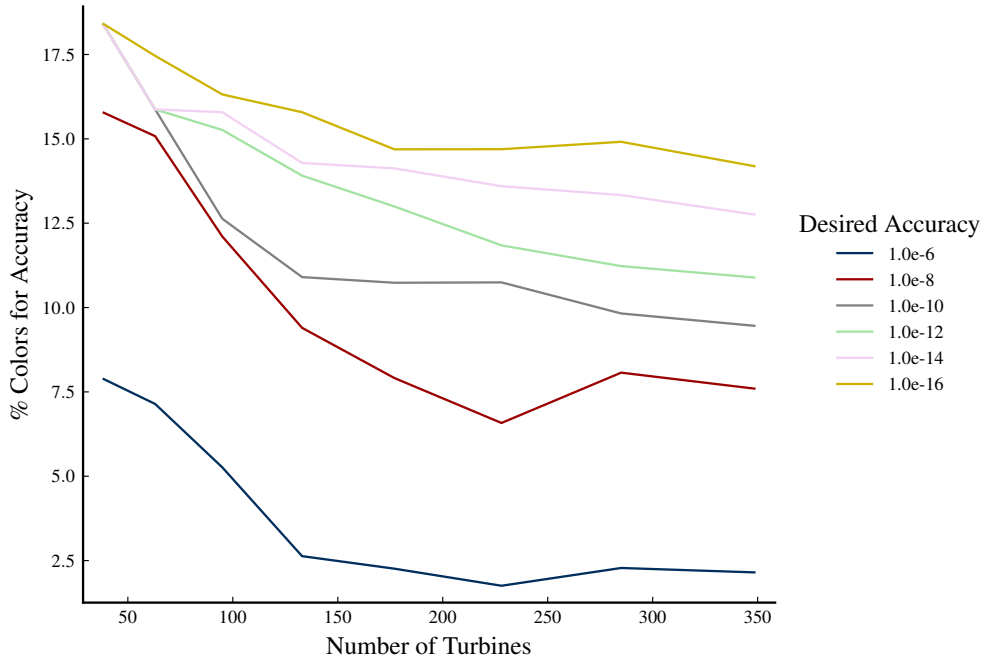

Fig. 4 This figure shows what percentage of Jacobian colors is required for a wind farm Jacobian to achieve an AEP gradient that has a maximum error less than the desired accuracy.

B. Optimization Results

One way to measure the efficiency of a wind farm layout is to measure how much energy is lost due to turbines being in the wake of other turbines. We calculated the wake loss with Eq. 7,

$$L_w = 100\left(1 - \frac{AEP}{AEP_{\text{ideal}}}\right), \tag{7}$$

where AEP_{ideal} is the AEP of the wind farm if all turbines were in freestream wind. Figure 5 uses this definition to compare the effectiveness of using sparse methods to dense methods. In order to create variation, we used the farm shown in Fig. 2 with 95 turbines. For each optimization, each turbine was randomly moved up to two rotor diameters in x and two rotor diameters in y to create different starting layouts. We then optimized each layout using both sparse and dense methods 10 times. Figure 5 shows the amount of wake loss (L_w) reduction from the initial wind farm layout to the final optimized solution as shown in Eq. 8.

$$L_{w\ \text{reduction}} = L_{w\ \text{initial}} - L_{w\ \text{optimized}} \tag{8}$$

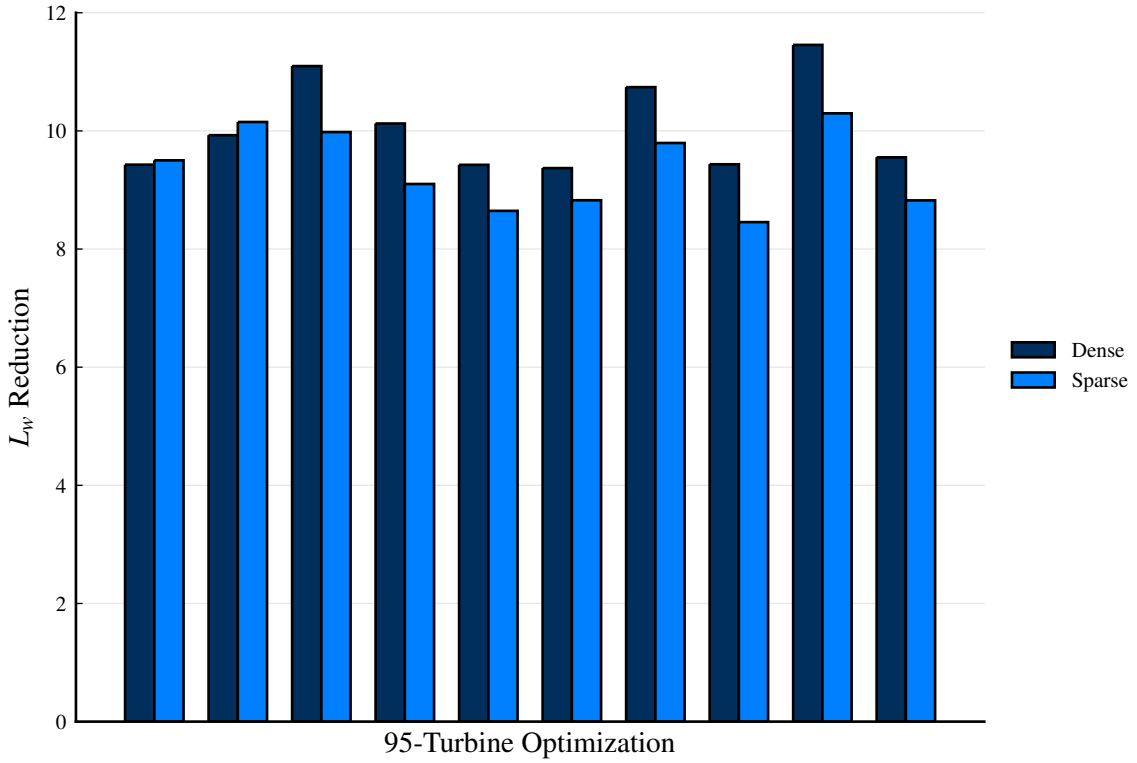

Fig. 5 Comparing the improvement in wake loss (L_w) between dense and sparse methods across 10 different optimizations for a 95-turbine round farm.

As shown in Fig. 5, the results of the optimizations regarding L_w were similar. The dense method has a slightly higher L_w reduction in most cases with an average of 0.7% more L_w reduction, which is expected because it uses a more accurate gradient. Also, likely due to the multi-modal nature of the design space, the dense and sparse methods differ in the amount of L_w reduction because they find different local minima. To make the sparse method yield results more similar to the dense method, using a lower threshold when determining the sparsity patterns or adjusting how the threshold changes during the optimization could be beneficial.

Whereas the results for sparse and dense methods are comparable, the computation time varies significantly. Figure 6 shows how the timings compare between the sparse and dense method.

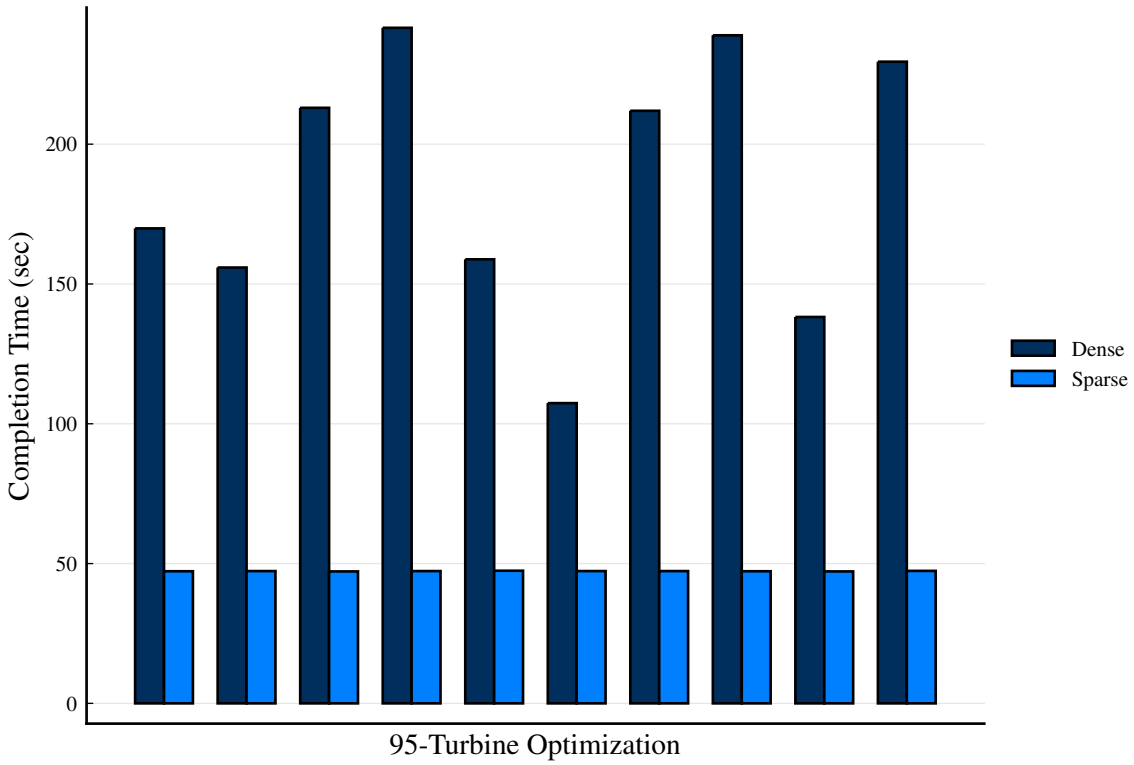

Fig. 6 Comparing the time taken for the same optimizations shown in Fig. 5 to converge for both the sparse and dense optimization methods.

Figure 6 clearly shows that the sparse method is both computationally less expensive and more consistent in the amount of computation time than the dense method. All the optimizations for the 95 turbine farm completed within 47 to 48 seconds. With this level of consistency, it is possible to estimate the time required for extensive multi-start optimizations. On average, the dense methods took 4 times longer to complete than the sparse methods. Although this is not the most significant speedup with wind farm layout optimization, we believe that longer optimizations would likely show greater improvement in the speeds of sparse methods.

IV. Conclusion

As shown above, the sparse Jacobian method presented here is able to achieve accurate results, comparable to dense methods. Approximately 15% of the original number of colors in the dense calculations are needed for this sparse method to obtain the same maximum error as the dense method in the optimization process.

The sparse Jacobian method reformulates the AEP gradient of a wind farm layout optimization into a collection of energy Jacobians so that sparsity can be applied in calculating the AEP gradient. By applying sparsity in calculating the AEP gradient, wind farm layout optimization problems can be solved more quickly than with traditional dense methods. We tested this by applying sparse methods in the optimization of three different wind farm layout problems and compared the results to optimizations with dense methods. By comparing both the time taken to complete the optimization and the improvement to the wind farm layout, we found that sparse methods maintained accuracy while increasing the speed of the optimizations by a factor of 4 on average, but with more significant speedups expected as the methodology is improved, or for larger farms.

Further study is needed to increase the efficiency of the sparse Jacobian method for wind farm layout optimization. The pattern update rules we used can be refined to allow fewer updates to sparsity patterns, thus increasing optimization speed. The wind farm layout optimization problems we used are simple and not practical in full-scale application. Further testing on real-world wind farm scenarios is required before we can confirm the full effectiveness of the sparse Jacobian method for wind farm layout optimization.

Acknowledgments

This work was authored in part 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] Sorensen, P., and Nielsen, T., "Recalibrating wind turbine wake model parameters—validating the wake model performance for large offshore wind farms," *European Wind Energy Conference and Exhibition, EWEA*, 2006.

[2] Beyer, H. G., Lange, B., and Waldl, H.-P., "Modelling tools for wind farm upgrading," *European Union Wind Energy Conference, AIAA*, 1996.

[3] Rios, L. M., and Sahinidis, N. V., "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, Vol. 56, No. 3, 2013, pp. 1247–1293. https://doi.org/10.1007/s10898-012-9951-y, URL https://doi.org/10.1007/s10898-012-9951-y.

[4] Thomas, J., McOmber, S., and Ning, A., "Wake expansion continuation: Multi-modality reduction in the wind farm layout optimization problem," *Wind Energy*, Vol. 25, 2022. https://doi.org/10.1002/we.2692.

[5] Coleman, T. F., and Moré, J. J., "Estimation of Sparse Jacobian Matrices and Graph Coloring Blems," *SIAM Journal on Numerical Analysis*, Vol. 20, No. 1, 1983, pp. 187–209. https://doi.org/10.1137/0720013, URL https://doi.org/10.1137/0720013.

[6] Coleman, T. F., and Verma, A., "The Efficient Computation of Sparse Jacobian Matrices Using Automatic Differentiation," *SIAM Journal on Scientific Computing*, Vol. 19, No. 4, 1998, pp. 1210–1233. https://doi.org/10.1137/S1064827595295349, URL https://doi.org/10.1137/S1064827595295349.

[7] Martins, J. R. R. A., and Ning, A., *Engineering Design Optimization*, 9781108833417, Cambridge University Press, 2022.

[8] Stanley, A., and Ning, A., "Massive simplification of the wind farm layout optimization problem," *Wind Energy Science*, Vol. 4, 2019, pp. 663–676. https://doi.org/10.5194/wes-4-663-2019.

[9] Rackauckas, C., and Mishra, P., "SparseDiffTools.jl," https://github.com/JuliaDiff/SparseDiffTools.jl, 2022.

[10] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, Vol. 59, No. 1, 2017, pp. 65–98. https://doi.org/10.1137/141000671, URL https://doi.org/10.1137/141000671.

[11] Revels, J., Lubin, M., and Papamarkou, T., "Forward-Mode Automatic Differentiation in Julia," *CoRR*, Vol. abs/1607.07892, 2016. URL http://arxiv.org/abs/1607.07892.

[12] Bay, C. J., Fleming, P., Doekemeijer, B., King, J., Churchfield, M., and Mudafort, R., "Addressing deep array effects and impacts to wake steering with the cumulative-curl wake model," *Wind Energy Science Discussions*, Vol. 2022, 2022, pp. 1–28. https://doi.org/10.5194/wes-2022-17, URL https://wes.copernicus.org/preprints/wes-2022-17/.

[13] Bastankhah, M., and Porté-Agel, F., "Experimental and theoretical study of wind turbine wakes in yawed conditions," *Journal of Fluid Mechanics*, Vol. 806, 2016, pp. 506–541. https://doi.org/10.1017/jfm.2016.595.

[14] Katic, I., Højstrup, J., and Jensen, N. O., "A simple model for cluster efficiency," *European wind energy association conference and exhibition*, Vol. 1, A. Raguzzi Rome, Italy, 1986, pp. 407–410.

[15] Niayifar, A., and Porté-Agel, F., "Analytical Modeling of Wind Farms: A New Approach for Power Prediction," *Energies*, Vol. 9, No. 9, 2016. https://doi.org/10.3390/en9090741, URL https://www.mdpi.com/1996-1073/9/9/741.

[16] Center, W. R. C., "Station wind rose: Nantucket, Dec," https:// wrcc.dri.edu/cgi-bin/wea_windrose.pl?laKACK., 2012.