DiffWake: A General Differentiable Wind Farm Solver in JAX

Maria Bånkestad[1,2,3*], **Leon Sütfeld**[1], **Aleksis Pirinen**[1,2,3], **Hamidreza Abedi**[1]
[1]RISE Research Institutes of Sweden
[2]Climate AI Nordics
[3]Swedish Centre for Impacts of Climate Extremes

Abstract

We introduce *DiffWake*, an end-to-end differentiable and curl-conserving flow solver for wind farms implemented in JAX. It preserves the core physics of wake formation and recovery while enabling exact gradient backpropagation through wake, thrust, and power calculations on GPUs. Unlike traditional engineering models with fixed empirical parameters, DiffWake offers a general, differentiable foundation for wind farm simulation that supports optimization, calibration, and machine learning-enhanced modeling. We demonstrate its use for (i) layout optimization under spatial constraints and (ii) probabilistic turbulence calibration from SCADA data using a lightweight neural network. Together, these results demonstrate a unified framework linking physical modeling with machine learning for accurate and scalable wind farm simulation. [1]

1 Introduction

Accurate modeling of wind farms requires capturing complex airflow interactions, known as wakes (see Fig. 1), between turbines [1, 2]. High-fidelity computational fluid dynamics (CFD) methods, such as Large-Eddy Simulation (LES) or Reynolds-Averaged Navier–Stokes (RANS), can resolve wake dynamics in detail but are far too computationally expensive for use in optimization or control [1]. Simplified engineering models, therefore, provide analytic approximations that balance physical realism and computational efficiency, yet they remain limited in accuracy under diverse atmospheric and layout conditions.

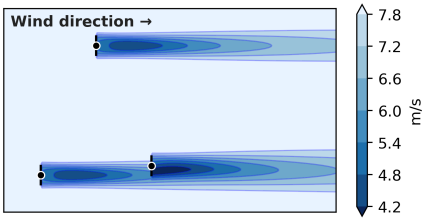

Figure 1: Wake interactions among turbines in the horizontal plane at hub height.

This motivates hybrid formulations that retain physical consistency in the engineering models while leveraging machine learning for parameter inference and model improvement [3, 4]. However, most existing engineering wake models remain based on hand-derived equations and CPU-bound solvers, which restrict scalability and compatibility with modern ML workflows [2, 5]. Gradient-based optimization has been explored through analytic or adjoint sensitivities [6, 7, 8] and, more recently, differentiable rotor-averaged formulations [9], yet fully differentiable implementations suitable for GPU acceleration are still lacking.

To address these gaps, we present *DiffWake*—a fully differentiable, JIT-compiled wind farm solver that preserves the core physics of wake, thrust, and power interactions while exposing exact gradients

*Corresponding author: maria.bankestad@ri.se
[1]Code is available at github.com/mariabankestad/DiffWake , along with a PyTorch version.

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025.

throughout the computation. The model bridges established analytical wake formulations with modern machine-learning tools, enabling direct inference of physical parameters from data and gradient-based optimization of layouts and control strategies. Built in JAX [10], DiffWake provides, to our knowledge, the first differentiable implementation of the cumulative–curl (CC) wake model [11, 12], a momentum-conserving extension of the Gaussian deficit formulation, using compiled tensor operations for efficient reverse-mode differentiation and GPU acceleration. We demonstrate its utility in two representative applications: (i) gradient-based layout optimization under spatial constraints and (ii) probabilistic calibration of turbulence intensity from SCADA data[2].

2 Background and method

Automatic differentiation (AD, the general form of backpropagation) [13, 14] computes exact (machine-precision) derivatives by recording elementary operations during execution. In JAX [10], these computational traces are just-in-time (JIT) compiled into efficient GPU kernels, allowing complex iterative solvers to remain both numerically stable and differentiable. This enables the rewriting of legacy numerical models—such as wake simulators—as differentiable modules, thereby eliminating finite-difference approximations and facilitating gradient-based inference and optimization.

Engineering wake models describe the mean flow velocity $u(x, y, z)$ and its reduction behind turbines, known as the velocity deficit. Downstream of a turbine, the flow can be viewed as a freestream component plus a deficit that decays laterally and vertically due to turbulence. A common formulation is the Gaussian wake model [15], which defines the local deficit as $u_{\mathrm{def}}(x, y, z) = U_\infty - u(x, y, z)$ and approximates its cross-section by a Gaussian profile:

$$u_{\mathrm{def}}(x, y, z) \approx \Delta U_0 \exp\left[-\tfrac{1}{2}\left(\frac{y - \delta(x)}{\sigma_y}\right)^2 - \tfrac{1}{2}\left(\frac{z - z_h}{\sigma_z}\right)^2\right], \tag{1}$$

where ΔU_0 is the maximum (centerline) deficit scaling with thrust and local turbulence intensity (TI), z_h is the hub height, and $\delta(x)$ is the lateral deflection (zero for aligned wakes). The wake widths σ_y and σ_z grow approximately linearly with downstream distance.

The cumulative–curl (CC) model [11] extends the Gaussian formulation by replacing its empirical deficit with a curl-based velocity field that enforces momentum and circulation conservation. Each turbine induces a local velocity perturbation Δu_t from this formulation, and the total downstream velocity is

$$u(x, y, z) = U_\infty - \sum_{t=1}^{N_t} \Delta u_t(x, y, z; C_{\mathrm{T}}, \gamma_{\mathrm{yaw}}, I), \tag{2}$$

where U_∞ is the inflow velocity, N_t the number of turbines, C_{T} the thrust coefficient, and I the turbulence intensity.

Differentiable implementation. DiffWake re-implements the cumulative–curl wake equations in JAX using pure tensor operations for full reproducibility and differentiability. Wake propagation is performed within a single compiled `lax.fori_loop`, preserving causality—each turbine influences only downstream ones—while supporting batched evaluation across wind conditions. This design yields exact gradients of farm power with respect to turbine positions, yaw angles, and model parameters, enabling stable, GPU-accelerated gradient-based optimization.

Optimization and inference. For parameter inference, DiffWake acts as a differentiable forward model, $\hat{P}_i = \mathrm{DiffWake}(c_i, \theta)$, where c_i denotes fixed simulation inputs or context (e.g., inflow conditions or turbine states) and θ are the optimized parameters such as empirical coefficients, layouts, or yaw angles. Gradients $\nabla_\theta P$ are computed automatically and can therefore be used directly in optimizers such as L-BFGS [16] or Adam [17]. Because the entire solver compiles into a single XLA graph [18], gradient evaluations are numerically stable and typically orders of magnitude faster than finite-difference methods. Our implementation follows the NumPy-based CC model in FLORIS [19] for physical consistency, but is entirely re-implemented in JAX for differentiability and GPU execution.

[2]SCADA data are turbine-level operational measurements such as wind speed, direction, and power used for monitoring and control.

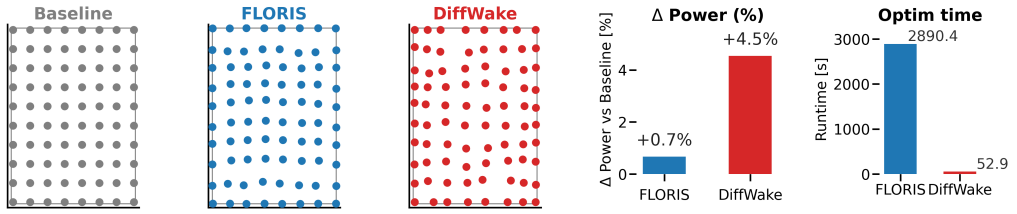

Figure 2: Final wind farm layouts (left), power increase (center-right), and optimization runtime (far-right) for the Baseline, FLORIS (SciPy/SLSQP), and DiffWake (L-BFGS) methods. FLORIS terminated after five iterations due to numerical instability, whereas DiffWake converged in 73 iterations with much lower time per step. Power is the wind-rose–weighted mean over 66 wind conditions. While results may vary slightly with initialization, DiffWake remained stable and completed in 53 s versus 2,890 s for FLORIS, which did not converge.

3 Experiments

We design two experiments to demonstrate how DiffWake can be used in different ways: (i) as a differentiable simulator for wind farm layout optimization, and (ii) as a tool for probabilistic calibration of turbulence intensity. These two use cases highlight complementary aspects of our approach: (i) gradient-based design and (ii) physics-consistent uncertainty modeling.

3.1 Wind farm layout optimization

Optimizing turbine layouts is a long-standing challenge in wind energy. Closely spaced turbines experience wake losses, while overly sparse layouts waste area and cabling costs. The goal is to maximize expected farm power under realistic wind conditions [20, 19], but power depends nonlinearly on wake interactions and site-specific wind statistics. We embed the differentiable DiffWake solver directly within the optimization, providing exact power gradients with respect to turbine coordinates for use in gradient-based optimizers such as L-BFGS.

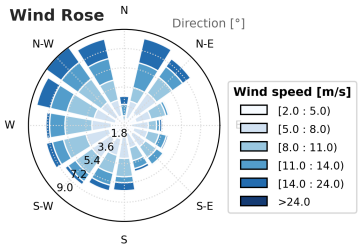

Figure 3: Wind rose at the Horns Rev site.

We use the Horns Rev offshore wind farm as a case study, fixing the number of turbines, turbine type, and layout bounds [21]. Wind directions, speeds, and weights are drawn from the measured wind rose (Fig. 3), so the objective reflects annual production under observed conditions. Turbulence intensity is held fixed at I_0. The optimization problem is

$$\max_{\mathbf{X}=\{\mathbf{x}_n\}_{n=1}^{N}} \sum_{k=1}^{K} w_k \, P(\mathbf{X}; c_k), \qquad c_k = (\mathrm{wd}_k, \mathrm{ws}_k, I_0)$$

$$\text{s.t.} \;\; \mathbf{x}_n \in [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}], \quad n = 1, \ldots, N,$$

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \geq d_{\min}, \qquad d_{\min} = 2D, \;\; 1 \leq i < j \leq N.$$

where $P(\mathbf{X}; c_k)$ is the total farm power predicted by DiffWake under wind condition c_k, and w_k are wind-rose weights. This objective corresponds to maximizing the expected annual energy production (AEP) subject to spatial and operational constraints. Coordinates are parameterized in an unconstrained space and mapped into the layout box via a $\tanh$ transform, while minimum spacing is enforced through a differentiable softplus penalty. Optimization uses L-BFGS with a Strong–Wolfe line search [22, 16].

Results. Fig. 2 compares optimized turbine layouts and power for the baseline, FLORIS, and DiffWake (L-BFGS) solvers. The FLORIS–SciPy implementation frequently failed to converge, as its finite-difference gradients are noisy and numerically unstable, often leading to nonphysical wake velocities and premature termination. The FLORIS baseline (v4.5, cumulative–curl model) utilized identical physical parameters and adjusted solver tolerances according to the official documentation, ensuring a fair setup. DiffWake, by contrast, converged reliably to higher-power layouts and ran over $50\times$ faster, benefiting from exact, stable gradients computed via automatic differentiation.

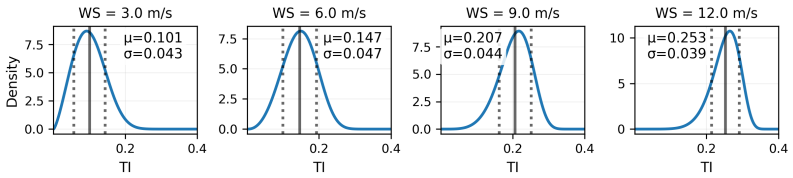

Figure 4: Learned TI distributions for different wind speeds at fixed day, hour, and direction. Each panel displays the predicted probability density, with the mean (solid) and standard deviation (dotted).

3.2 Data-driven turbulence modeling

Turbulence intensity (TI) significantly impacts wake recovery: higher TI enhances mixing and accelerates wake decay, while lower TI enables wakes to persist. Yet SCADA-derived TI is often unreliable—missing, smoothed over 10-minute intervals, or biased by nacelle effects. We utilize the publicly available Smarteole SCADA dataset [23], which comprises 10-minute measurements of wind speed, direction, power, and turbulence. Engineers often rely on IEC classes [24] or site-specific tables [25], but these overlook real atmospheric variability [26, 27, 28].

We model turbulence as a bounded random variable,

$$I = I_{\min} + (I_{\max} - I_{\min})Z, \;\; Z \sim \mathrm{Kumar}(a_\theta(x), b_\theta(x)),$$

where $a_\theta(x), b_\theta(x) > 0$ are predicted by a neural network from features x (wind direction, speed, hour, and day-of-year). The Kumaraswamy distribution [29], though less familiar than the Beta, has similar flexibility and support on $[0, 1]$ but provides closed-form CDF and inverse CDF expressions, making it computationally efficient for sampling and likelihood evaluation. This formulation captures the stochastic variability and uncertainty of TI while remaining physically bounded and numerically stable. Embedded in DiffWake, it enables end-to-end calibration from turbine power observations under the governing wake physics. Training and likelihood details are provided in Appendix A.

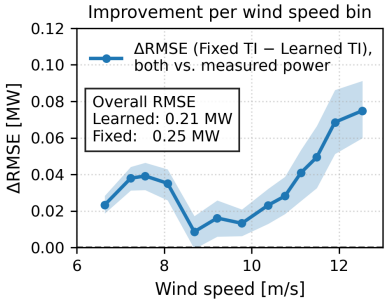

Figure 5: Change in RMSE versus wind speed when replacing the fixed TI with the learned model, both evaluated against measured power. The improvement increases with wind speed, indicating that the learned TI captures the stronger and more variable turbulence more effectively at higher inflow speeds. Shaded areas indicate 95% bootstrap confidence intervals, and the empirical wind speed distribution in the dataset weights the overall RMSE values.

Results. The inferred TI distributions (Fig. 4) increase with wind speed as expected, since stronger inflow enhances shear and velocity fluctuations [25]. Fig. 5 shows RMSE differences in a turbine power output prediction task between the fixed and learned TI models. The learned model reduces the overall RMSE from 0.25 to 0.21, with the most significant gains at low and high wind speeds. Both models perform similarly near 9 m/s, where the baseline TI (0.09) is already adequate. Although the absolute improvement is modest, which is expected since TI affects power only indirectly through wake recovery, the result demonstrates a fully differentiable, physics-consistent calibration that generalizes to other CC parameters (k_y, k_z, x_{nw}, C_{T}) for uncertainty-aware model tuning and optimization.

4 Discussion and conclusions

We introduced DiffWake, a fully differentiable, curl-conserving flow solver for wind farm modeling. Implemented in JAX, it combines analytical wake physics with automatic differentiation and GPU acceleration to provide efficient gradients through wake, thrust, and power computations. This enables gradient-based inference and design at realistic scales. Across two applications—turbulence calibration and layout optimization—DiffWake illustrates how differentiable solvers can unify physical modeling and machine learning. Beyond these examples, DiffWake provides a general foundation for hybrid modeling, supporting the quantification of uncertainty and integration with surrogate models. The same principles extend to other flow and energy systems, highlighting its potential as a broadly applicable differentiable framework.

Acknowledgments and Disclosure of Funding

This work was funded by the Swedish Energy Agency (through grant No. 2023-204936). The computations and data storage were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

[1] Benjamin Sanderse, Sander P. van der Pijl, and Barry Koren. Review of computational fluid dynamics for wind turbine wake aerodynamics. *Wind Energy*, 2011.

[2] Richard J. A. M. Stevens and Charles Meneveau. Flow structure and turbulence in wind farms. *Annual Review of Fluid Mechanics*, 2017.

[3] Jae H. Lee, Hao Gao, and Michael Döllinger. Editorial: Integrating machine learning with physics-based modeling of physiological systems. *Frontiers in Physiology*, 16, 2025.

[4] T. Göçmen, J.-W. van Wingerden, P. A. Fleming, et al. Data-driven wind farm flow control and challenges towards practical implementation. *Renewable and Sustainable Energy Reviews*, 199:114115, 2025.

[5] Bart M. Doekemeijer, Paul A. Fleming, and Jan-Willem van Wingerden. A tutorial on the FLOw redirection and induction in steady state (FLORIS) framework for wind farm control. *Wind Energy Science*, 5(2):287–300, 2020.

[6] Aaron Klein, Paul Fleming, and Katherine Dykes. A continuously differentiable turbine layout optimization. In *Journal of Physics: Conference Series*, 2016.

[7] Jennifer King, Katherine Dykes, Peter Graf, Peter Hamlington, and Julie Lundquist. Adjoint optimization of wind farm layouts for systems engineering applications. In *Journal of Physics: Conference Series*, 2016.

[8] David Criado Risco, Jesper Rasmussen, and Søren Ott. Gradient-based wind farm layout optimization with inclusion and exclusion zones. *Wind Energy Science*, 2024.

[9] Abdullah Ali, Søren Ott, and Anders B. Pedersen. Direct integration of non-axisymmetric gaussian wind-turbine wake profiles for differentiable rotor-averaged models. *Wind Energy Science*, 2025.

[10] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, et al. JAX: Composable transformations of python+numpy programs. `https://github.com/google/jax`, 2018.

[11] L. A. Martínez-Tossas, J. King, M. J. Churchfield, and F. Sotiropoulos. The curl model: A three-dimensional and momentum-conserving approach for wind turbine wakes. *Wind Energy Science*, 2019.

[12] Christopher J. Bay, Paul Fleming, Bart Doekemeijer, Jennifer King, Matthew Churchfield, and Rafael Mudafort. Addressing deep array effects and impacts to wake steering with the cumulative-curl wake model. *Wind Energy Science*, 2023.

[13] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey A. Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 2018.

[14] Charles C. Margossian. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2019.

[15] M. Bastankhah and F. Porté-Agel. Experimental and theoretical study of wind turbine wakes in yawed conditions. *Journal of Fluid Mechanics*, 2016.

[16] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Google. XLA: Accelerated linear algebra. `https://www.tensorflow.org/xla`, 2017.

[19] Pieter Gebraad, Matthew Churchfield, and Paul Fleming. Wind plant power optimization through yaw control using a parametric model for wake effects—a CFD simulation study. *Wind Energy*, 2017.

[20] Sten Truelsen Frandsen, Rebecca Jane Barthelmie, and Sara C. Pryor. Analysis of turbulence intensity and mean wind speed offshore at the horns rev site. *Wind Energy*, 2006.

[21] R. J. Barthelmie et al. Modelling and measuring flow and wind turbine wakes at the horns rev offshore wind farm. *Boundary-Layer Meteorology*, 2009.

[22] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 1980.

[23] Smarteole Consortium. Smarteole wind farm SCADA dataset (field test 1), 2020.

[24] IEC 61400-1: Wind Energy Generation Systems – Part 1: Design Requirements, 2019.

[25] Sten Frandsen. Turbulence and turbulence-generated structural loading in wind turbine clusters. Technical report, Risø National Laboratory, 2007.

[26] Alfredo Peña, Sven-Erik Gryning, and Jakob Mann. On the length-scale of the wind profile. *Quarterly Journal of the Royal Meteorological Society*, 2010.

[27] Ameya Sathe, Jakob Mann, Thanasis Barlas, Wim Bierbooms, and Gerard van Bussel. Influence of atmospheric stability on wind turbine loads. *Wind Energy*, 2011.

[28] Fernando Porté-Agel, Majid Bastankhah, and Saeed Shamsoddin. Wind-turbine and wind-farm flows: A review. *Boundary-Layer Meteorology*, 2020.

[29] Ponnambalam Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 1980.

[30] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

Appendix

Here we present some supplementary information on the experiments conducted in this paper. All experiments run on JAX 0.7.1, Python 3.12.3, and CUDA 12.6. GPU results use an NVIDIA A100.

Appendix A: Probabilistic turbulence model details

We parameterize turbulence intensity (TI) as a bounded random variable $I \in [I_{\min}, I_{\max}]$ using the Kumaraswamy family, chosen for its simple closed-form inverse CDF and reparameterizable sampling:

$$Z(u; a, b) = (1 - (1 - u)^{1/b})^{1/a}, \qquad u \sim \mathcal{U}(0, 1),$$
$$I = I_{\min} + (I_{\max} - I_{\min})Z, \quad Z \sim \mathrm{Kumar}(a_\theta(x), b_\theta(x)).$$

The neural network outputs positive shape parameters $a_\theta(x), b_\theta(x)$ from input features

$$x = (\sin(\mathrm{wd}), \cos(\mathrm{wd}), \mathrm{ws}, \sin(\mathrm{h}), \cos(\mathrm{h}), \sin(\mathrm{doy}), \cos(\mathrm{doy})),$$

which ensures periodicity in wind direction, hour, and day-of-year. This parameterization allows backpropagation through stochastic sampling using the reparameterization trick [30, 31].

Likelihood and training. For each SCADA observation P_i with context c_i (layout, inflow, turbine states), the differentiable DiffWake solver maps a sampled turbulence $I_{i,s}$ to predicted power:

$$\hat{P}_{i,s} = \mathrm{DiffWake}(c_i, I_{i,s}), \qquad P_i \mid I_{i,s}, c_i \sim \mathcal{N}(\hat{P}_{i,s}, \sigma^2),$$

where $\sigma > 0$ captures measurement and model noise. The marginal likelihood is approximated via Monte-Carlo sampling over S draws from $p_\theta(I \mid x_i)$, yielding the loss:

$$\mathcal{L}(\theta, \sigma) = -\frac{1}{N} \sum_i \log \Big[\tfrac{1}{S} \sum_s \mathcal{N}(P_i \mid \hat{P}_{i,s}, \sigma^2) \Big] + \lambda \, \mathrm{KL}\big(p_\theta(I \mid x_i) \,\|\, p_0(I)\big),$$

where $p_0(I) = \mathrm{Kumar}(a_0 = 1, b_0 = 3.75)$ serves as a weak prior that mildly favors lower I values while remaining broad, reflecting that strong turbulence is rare and often corresponds to transient conditions. All parameters (θ, σ) are optimized with Adam [17].