Wind Energ. Sci., 11, 3377–3400, 2026
https://doi.org/10.5194/wes-11-3377-2026

eawe european academy of wind energy
Open Access WIND ENERGY SCIENCE

Classification of leading-edge-erosion severity via machine learning surrogate models

Aidan Gettemy[1], Susan Minkoff[2], John Zweck[3], and Elaine Spiller[4]

[1]Department of Mathematical Sciences, University of Texas at Dallas,
800 W. Campbell Road, Richardson, TX 75080-3021, USA
[2]Department of Applied Mathematics, Computing and Data Sciences Directorate,
Brookhaven National Laboratory, Upton, NY 11973, USA
[3]Department of Mathematics, New York Institute of Technology, 1855 Broadway, New York, NY 10023, USA
[4]Mathematical and Statistical Sciences, Marquette University,
1250 W. Wisconsin Ave., Milwaukee, WI 53233, USA

Correspondence: Aidan Gettemy (aidan.gettemy@utdallas.edu)

Received: 22 December 2025 – Discussion started: 28 January 2026
Revised: 30 June 2026 – Accepted: 2 August 2026 – Published: 10 September 2026

Abstract. Leading-edge erosion is a common form of wind turbine blade deterioration that reduces aerodynamic performance, increases maintenance demands, and shortens turbine service life. Machine-learning-based structural health monitoring systems offer a promising route for detecting erosion severity, but their performance often depends on access to large labeled training datasets. For wind turbine applications, generating these datasets with full-physics simulation can be computationally expensive, motivating the use of surrogate models for efficient data generation.

In this work, we evaluate whether a Gaussian process (GP) emulator can replace full-physics simulation as a training-data generator for leading-edge-erosion classification. The surrogate is trained on a limited set of OpenFAST simulations and then used to generate large labeled datasets, essentially cost-free, for a random forest classifier. We apply a parallel partial and zero-censored emulator which extends the standard GP framework by predicting a vector of response statistics associated with aerodynamic, structural, and turbine-level outputs and incorporating output constraints to improve the physical consistency and uncertainty calibration of the predictions.

We compare two random forest classifiers: one trained directly on full simulation data and one trained on emulator-generated data. Both classifiers are evaluated on held-out full-physics simulation data across five leading-edge-erosion severity levels. The emulator-trained classifier achieves accuracy comparable to the simulator-trained classifier, demonstrating that the GP surrogate can substantially reduce the cost of training-data generation without sacrificing classification performance. These results suggest that constrained, vector-valued GP emulators can support efficient simulation-based structural health monitoring workflows and may provide a useful component of future digital-twin frameworks for wind turbine maintenance.

1 Introduction

Wind energy plays a crucial role in the worldwide transition to renewable sources of energy (Hassan et al., 2024; Bošnjaković et al., 2022). Reducing the cost of energy over the lifetime of a wind turbine (WT) requires reducing operation and maintenance (O&M) costs (Myhr et al., 2014; Park et al., 2022; Ren et al., 2021). Opportunities to reduce O&M include maintaining new offshore WTs, operating aging land-based WTs (Bilgili and Alphan, 2022; Veers et al., 2019), and monitoring large-scale WTs with flexible blades and high tip speeds (Barlas and van Kuik, 2010).

Leading-edge erosion (LEE) is a driver of O&M costs (Maniaci et al., 2022). LEE creates structural risks that may necessitate the complete replacement of WT blades (Maniaci et al., 2022). Offshore turbines face an elevated risk for LEE due to harsh environmental exposure (Shankar Verma et al., 2021b), and large-scale rotors are expected to erode faster on their blade tips due to the increased tip speeds (Carraro et al., 2022; Haus, 2020). Studies suggest erosion-related energy losses range from 2 % to 3.7 % annually (Han et al., 2018). While there are many approaches to mitigate LEE (Antoniou et al., 2022; Herring et al., 2019; Macdonald et al., 2016; Pryor et al., 2022; López et al., 2023; Shankar Verma et al., 2021a; Visbech et al., 2023), non-destructive evaluation (NDE) techniques, including visual inspection, ultrasonic testing, thermography, acoustic emission monitoring, vibration-based monitoring, and supervisory control and data acquisition (SCADA)-based signal processing (Civera and Surace, 2022), have shown promise towards reducing O&M costs by limiting turbine downtime while preventing damage from developing unnoticed. The proactive detection and mitigation of LEE minimizes energy loss and avoids full blade replacement (Campobasso et al., 2023; Tchakoua et al., 2014).

Structural health monitoring (SHM) for LEE remains challenging (Mishnaevsky, 2022). Visual inspection may only detect defects after substantial degradation has occurred (Pugh et al., 2021). Improvements in UAV-based blade surveillance (Shihavuddin et al., 2019) and gloss-based surface characterization (Leishman et al., 2022) still require devices in close proximity to WT structures in order to work. Alternatively, SCADA-based methods (Maldonado-Correa et al., 2020; Pandit et al., 2023) rely on available data gathered without the need for workers or instruments to be moved into physical proximity to the blades. Along with sensors for blade load measurements (Abdallah et al., 2015) and aerodynamic surface pressure (Duthé et al., 2021) measurements, SCADA signal processing, combined with machine learning (ML) methods, is showing promise for driving down the O&M cost of LEE (Maldonado-Correa et al., 2020; Pandit et al., 2023).

Machine learning (ML) has proven to be useful for SHM through analyzing SCADA data for turbine fault detection and blade damage (Du et al., 2020; Stetco et al., 2019; Zaher et al., 2009; Jiménez et al., 2017). ML techniques, including random forest (Fezai et al., 2020), XGBoost (Zhang et al., 2018), and neural networks (NNs) (Chen et al., 2021; Choe et al., 2021), have successfully detected faults in SHM data. Additionally, ML is being applied to SHM problems for LEE detection. Applications include the use of NN to predict energy loss from eroded blades (Campobasso et al., 2020), the coupling of computational fluid dynamics (CFD) simulators with aero-elastic modeling to simulate erosion patterns and quantify damage (Enríquez Zárate et al., 2022), transformer models to track erosion severity in turbulent conditions (Duthé et al., 2021), and deep learning to predict LEE using aerodynamic data (Abdallah et al., 2022; Carmona-Troyo et al., 2025).

Although ML is a promising avenue for detecting LEE, its effectiveness is constrained by the need to collect large volumes of high-fidelity training data (Ward et al., 2024). Data generation can impose a computational bottleneck for training and validating these systems. For instance, CFD simulations with fine-scale aerodynamic effects of erosion require computationally expensive numerical solutions to the Navier–Stokes equations, which is not feasible for a large dataset generation (Langel et al., 2017). Even reduced-order modeling approaches are prohibitively costly when simulating erosion under diverse atmospheric conditions (Carraro et al., 2022). While aero-elastic simulations offer a more tractable alternative, studies demonstrate that the impact of blade damage is affected by wind speed, turbulence intensity, and air density, necessitating extensive dataset coverage of operational scenarios for reliable model generalization (Pandit et al., 2023; Papi et al., 2020). In this study, we address a central limitation of ML-based SHM by reducing the computational cost of generating sufficiently large labeled datasets from full-physics aero-elastic models. We show that a computationally inexpensive GP emulator can be trained on a limited set of simulations and then used to generate large datasets to train LEE classifiers while preserving classification performance when evaluated on held-out ground truth simulation data.

Surrogate modeling is commonly employed to address complexity and scalability challenges in WT engineering by providing a computationally efficient approach to damage tracking and predictive maintenance (Clark and Clark, 2022; Golparvar et al., 2021; Murcia et al., 2018; Singh et al., 2024). Surrogate modeling approaches encompass several techniques, but Gaussian process (GP) emulators are the most popular choice (Murcia et al., 2018; Singh et al., 2024). GPs are frequently used for uncertainty quantification, calibration, power or load prediction, SHM, and reliability analysis (Golparvar et al., 2021; Clark and Clark, 2022; Rogers et al., 2020; Avendano-Valencia et al., 2020; Abdallah et al., 2019). GP applications include multi-fidelity kriging, damage-equivalent load estimation, and the fusion of synthetic and real data (Abdallah et al., 2019; Avendaño-Valencia et al., 2021; Haghi et al., 2024). However, the existing GP literature primarily focuses on predicting scalar or

low-dimensional quantities of interest. Emulation for SHM requires physically consistent, higher-dimensional outputs for downstream tasks such as damage classification. With the standard approach, emulating a WT simulator that generates higher-dimensional vector-valued output from multiple sensor channels requires separately fitting and storing multiple standard scalar GPs, one for each output dimension. In this paper we use partial parallel emulation to significantly accelerate the prediction of vector-valued outputs (Gu and Berger, 2016). In addition, for outputs with constraints, such as generator power, a standard GP is unrealistic because its output is unbounded. Spiller et al. (2023) developed the zero-censored GP emulator to enforce range-limited outputs. We show that the combination of parallel partial and zero-censored emulation (PPzGP) (Gu and Berger, 2016; Spiller et al., 2023) improves the fidelity and computational efficiency of higher-dimensional WT emulation models.

Our primary contribution in this paper is the introduction of a framework to reduce the computational cost of training a SHM monitoring algorithm. Specifically, we present the first application of the PPzGP emulation methodology to wind turbine modeling and the classification of LEE. While this work involves simplifications to the physics and operating conditions, including assuming steady-state wind, the approach could be extended to more physically realistic scenarios, including turbulent inflow. We compare the performance of LEE classifiers trained on datasets generated from the emulator to those obtained using OpenFAST simulations. Our PPzGP emulator reduces the computational cost of acquiring training data and improves accuracy compared to training the classifier using OpenFAST simulations. The PPzGP is also significantly more efficient than training multiple standard scalar GP emulators for each output quantity of interest. Once fit, the cost of using the GP emulator to generate SHM classifier training data is essentially free. In our experiments, the classification results show generated data can be used without a significant loss of accuracy (emulator-trained classifiers: 0.88 macro-F1; OpenFAST trained classifiers: 0.82 macro-F1). The computational savings are significant. Including the cost of running the simulator to produce the training data for the emulator, the overall computational time is drastically reduced. Performance improvements are greatest for the intermediate erosion classes, implying that the larger emulator-generated training set is especially beneficial for resolving the more ambiguous LEE class boundaries.

2 Methods

In this section, we describe the workflow used to test whether emulator-generated data can replace direct simulations for training LEE classifiers. We use OpenFAST (Golparvar et al., 2021; Jonkman et al., 2024; Moynihan et al., 2022; Murcia et al., 2018) to simulate the NREL 5 MW reference turbine under varying environmental conditions and stochastic LEE states. We use global sensitivity analysis (Morris, 1991) to identify influential simulator inputs. Finally, we train a vector-valued GP emulator (Smith, 2024) with parallel partial emulation (Gu and Berger, 2016) and zero-censored treatment (Spiller et al., 2023) of range-limited outputs, generate large emulator-based training datasets, and compare the resulting random forest (Breiman, 2001) classifier performance with that of the simulation-trained classifier. Both classifiers are evaluated on held-out simulations so that the full aeroelastic model provides the reference test data.

2.1 OpenFAST

The OpenFAST wind turbine simulator is developed and maintained by the National Laboratory of the Rockies (NLR) for a wide range of aerodynamic and structural applications. OpenFAST simulates the response of a wind turbine to environmental conditions by linking specialized structural and physical modules, including those for wind flow, aerodynamics, structural dynamics, and servo-dynamics (Jonkman, 2013; Rinker et al., 2020). OpenFAST has been applied to supply realistic aero-servo-elastic data for LEE detection, including the generation of blade tip accelerometer data (Enríquez Zárate et al., 2022) and lift or drag sensor data (Duthé et al., 2021).

Because of its versatility and well-established capabilities, we use OpenFAST to generate realistic LEE data, relying on the coupling of four OpenFAST modules: `AeroDyn`, `ElastoDyn`, `ServoDyn`, and `InflowWind`. These modules contribute to the simulation of blade forces, including lift and drag, blade loads, generator power, and the wind environment. The `InflowWind` module calculates the wind vectors around the turbine while taking into account the turbine's size and hub height. `InflowWind` can simulate wind fields with a variety of properties. In this study we use uniform wind fields, include wind shear, and specify wind direction. Aerodynamic forces and blade loads, handled by the module `AeroDyn`, are particularly important for LEE. `AeroDyn` uses blade element momentum theory (BEM) (Jonkman et al., 2015) to calculate aerodynamic forces on the rotor by breaking the blade into discrete regions or elements. These elements are assigned a lift or drag coefficient according to their instantaneous angle of attack with the oncoming air. The relationship between the angle of attack and lift or drag is modeled using aerodynamic polar curves. OpenFAST uses a look-up table based on the polar curve to calculate torque and thrust forces on the axial shaft of the wind turbine (Ning, 2014; López et al., 2023). `ServoDyn` regulates the electronic systems of the wind turbine using structural motions, loads, and wind measurements; calculates the power generation; and determines control responses. In this study, the nacelle is fixed, but we enable the servo-controller to pitch the blades and regulate the rotor, in order to maximize power below rated wind speed and maintain rated power

Table 1. OpenFAST NREL 5 MW reference WT (RWT) upwind three-blade WT simulation model parameters.

Simulation parameter	Description
Rotor, blade length, hub diameter, hub Height	{126.0, 61.5, 3.0, 90.0 m}
Cut-in, rated, cut-out wind speed	$\{3.0\,\mathrm{m\,s^{-1}}, 11.4\,\mathrm{m\,s^{-1}}, 25.0\,\mathrm{m\,s^{-1}}\}$
Simulation time step	6.25 ms
Simulation total time	180 s
Wind type	Uniform wind files

above rated wind speed. The default wind turbine controller model has variable speed and collective pitch control, which regulates the power generated as a function of the wind speed by pitching the wind turbine blades and controlling the rotor torque to maximize power below the rated wind speed (Abbas et al., 2022). Aerodynamic changes due to LEE result in changes in the dynamics of the blade. These effects are modeled using the `ElastoDyn` module, which calculates the blade and tower displacement, velocity, and acceleration and allows for the simulation of accelerometers at the tips of the blades and loads at the root of the blades. `ElastoDyn` uses several different input parameters, including the blade geometry, the mass or inertia of blade elements, and the stiffness of elements.

2.2 Leading-edge-erosion data generation

All datasets in this study were generated with the 5 MW reference wind turbine in OpenFAST using the `AeroDyn`, `ElastoDyn`, `ServoDyn`, and `InflowWind` modules (Jonkman et al., 2009). The wind speed, direction, air density, and shear (see Table 1) are held fixed during each run.

The wind speed ranges from the lowest speed at which the turbine generates power to the speed where the blades are stalled to prevent damage (cut-in to cut-out wind speeds) for the 5 MW reference wind turbine (see Table 1) (Jonkman et al., 2009). Because air density has a large impact on power (Golparvar et al., 2021), we use a wide range of air densities, based on a conservative lower bound expected in cold, low-humidity conditions at around sea level up to realistic high temperatures for turbines in onshore environments (Liang et al., 2020). We model wind speed, V, as a function of height, h, above the ground using the wind shear power law (Platt et al., 2016) $V(h) = V_{\mathrm{r}}\left(\frac{h}{h_{\mathrm{r}}}\right)^{\nu}$, where V_{r} is the reference wind speed, h_{r} is the hub height (see Table 1), and ν is the shear parameter. The nominal wind shear parameter depends on the topographic surroundings of the turbine. We set a value of $\nu = 0.2$ as the default wind shear parameter (Liu et al., 2024). In Table 2 we summarize the ranges of the

Table 2. Environmental variable ranges.

Input	Range	Units
Wind direction	[−15, 15]	°
Air density	[1.10, 1.42]	$\mathrm{kg\,m^{-3}}$
Wind speed	[3, 25]	$\mathrm{m\,s^{-1}}$
Wind shear coefficient	[0, 0.5]	(–)

environmental parameters. To ensure that the turbine reaches equilibrium with the environmental conditions, we ran the simulations for 180 s and discarded the first two-thirds of simulation times to avoid transient effects (Golparvar et al., 2021).

2.2.1 Leading-edge-erosion model

LEE is primarily driven by material stresses due to the collision of particles (e.g., rain drops) with the blade surface (Bech et al., 2018; Pryor et al., 2022). This has the effect of damaging the blade coating, ultimately leading to surface roughening and the growth of pits and gouges (Pryor et al., 2022). As the blade is roughened, the transition between laminar and turbulent airflow shifts, (Panthi and Iungo, 2023), reducing lift and increasing drag (Gaudern, 2014). Rather than using computationally expensive CFD simulations to model airflow over rough surfaces, for the results in this paper we adapt a phenomenological LEE model developed by Duthé et al. (2021) that is straightforward to implement within the `AeroDyn` module of OpenFAST. With this simplified model, the effect that erosion has on the aerodynamic properties of the blade edge is represented via a spatial perturbation of the lift and drag polars as a function of the angle of attack. Duthé et al. (2021) based these perturbations on wind tunnel experiments (Han et al., 2018) in which erosion defects were simulated using roughened tape applied to the blade (Zidane et al., 2016).

As in Duthé et al. (2021), we quantify the severity of erosion using a finite ordered set of damage classes, which encode the overall erosion level of the blade. These classes provide a practical link between observed surface degradation, aerodynamic performance loss, and potential remedial actions (Han et al., 2018; Sareen et al., 2014; Maniaci et al., 2022).

Although we do not model the time evolution of erosion, the idea is that erosion severity will be greater for older blades that have experienced more impact events. In addition, it is reasonable to assume that, to first order, the amount of erosion depends linearly on position along the blade. This erosion is expected to initiate and progress more rapidly in blade regions that are further from the hub (Bech et al., 2018; Maniaci et al., 2022; Schramm et al., 2017; Verma et al., 2020). Finally, to take into account the inherent variability in

erosion, we model the spatial dependence of erosion stochastically (Duthé et al., 2021).

We now describe the details of this erosion model. For this study, we use five erosion severity classes parameterized by $\alpha \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$. We divide the blade into six erosion regions via a grouping of the `AeroDyn` nodes defined in the NREL 5MW RWT (see Table 1; Jonkman et al., 2009) summarized in Table 3 and shown in Fig. 1. We let $e_i \in [0, 1]$ denote the erosion level of the ith blade region, where $e_i = 0$ corresponds to a clean surface, and $e_i = 1$ corresponds to the maximum modeled erosion state. We model the erosion vector as a correlated random field,

$$\mathbf{e} = (e_1, \cdots, e_6) \sim \mathrm{MVN}(\boldsymbol{\mu}(\alpha), \boldsymbol{\Sigma}(\alpha)),$$

where both the mean and covariance depend on the erosion severity class, α. To incorporate the linear relationship between blade velocity and radial distance into the model, we define the mean erosion level by

$$\boldsymbol{\mu}(\alpha) = (\alpha x_1, \ldots, \alpha x_6),$$

where x_i is the normalized radial distance from the hub to the midpoint of the ith blade region. The covariance matrix is defined by

$$\boldsymbol{\Sigma}(\alpha)_{i,j} = \left(\frac{\min(x_i, x_j)\alpha}{10}\right)^2 \exp\left(\frac{-(x_i - x_j)^2}{x_i + x_j}\right).$$

This heuristic covariance model imposes stronger correlation between the erosion level in nearby blade regions while allowing erosion variability to increase both toward the blade tip and with erosion severity. The increase in erosion variability with erosion severity accounts for the fact that severe erosion can be due to a range of physical states, including coating loss, roughness patches, pits, gouges, and localized defects, whereas mild erosion is comparatively more uniform (Maniaci et al., 2022). This model takes values sampled from the normal distribution that lie in the interval $[0, 1]$, replacing values that are < 0 by 0 and values > 1 by 1.

Once we have sampled the erosion levels, e_i, we model the aerodynamic effect of erosion through a simplified spatial perturbation of the lift and drag polars. This approach is consistent with prior erosion models that interpolate or perturb aerodynamic polars according to local damage severity (Maniaci et al., 2022; Abdallah et al., 2015). Based on reported severe-erosion effects, including lift losses up to 53 % (Han et al., 2018) and drag increases up to 500 % (Sareen et al., 2014), we scale the lift and drag coefficients in the ith blade region according to $c_{\mathrm{L},i} = 1 - 0.53e_i$ and $c_{\mathrm{D},i} = 1 + 4e_i$.

This simplified erosion model has several limitations. First, our model lacks realistic temporal resolution of erosion. More detailed stochastic models represent erosion as a random accumulation process, where damage arrival and severity depend on quantities such as blade radius, wind speed, precipitation, and exposure history (Duthé et al., 2021). In addition, experimental descriptions of liquid droplet erosion often identify a multi-stage progression, including an incubation period with little apparent mass loss and a nonlinear acceleration phase, followed by an approximately linear damage–growth regime (Law and Koutsos, 2020). By contrast, the present model assigns blades directly to ordered erosion classes and does not attempt to reproduce these intermediate growth dynamics.

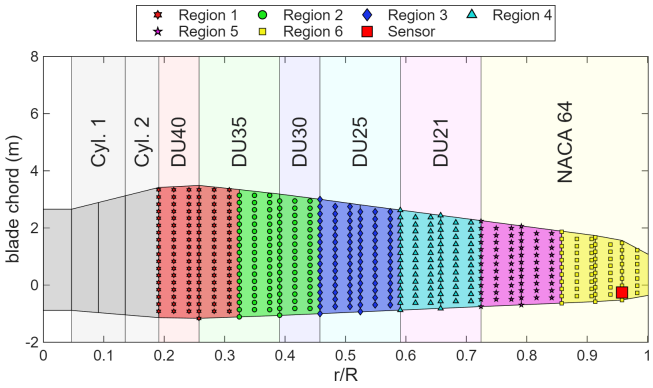

Figure 1. Diagram of erosion regions described in Table 3. The red square shows the location of the aerodynamic node where lift and drag sensor data are measured. The y axis measures r/R, the relative proportion of the total blade length; see Table 3.

Second, we use a simplified aerodynamic perturbation model which interpolates between clean and maximally eroded behavior using scalar modifications to the lift and drag coefficients. In reality, surface roughness can alter the shape of lift and drag polars in an angle-of-attack-dependent and nonlinear manner. For example, Abdallah et al. (2015) model eroded lift polars using multiple aerodynamic parameters, while Duthé et al. (2021) show that erosion effects may be negligible over some angle-of-attack ranges. A more detailed modeling approach would explicitly modify the blade surface geometry, compute the resulting aerodynamic polars using CFD, and then pass the modified polars to a turbine-level simulator (Enríquez Zárate et al., 2022).

Finally, our use of five erosion classes necessarily coarsens a gradual physical process. This representation does not resolve subtle differences between early-stage erosion states or the full diversity of possible damage patterns along the blade. However, our simplified stochastic parameterization is not intended to model the complete time evolution of LEE. Rather, it is designed to generate physically motivated families of aerodynamic perturbations for evaluating whether emulator-generated data can support erosion-severity classification. For this purpose, the model preserves several important qualitative features. Namely, (1) erosion damage increases monotonically with class severity, (2) erosion accumulation is weighted toward the tip, and (3) the induced aerodynamic changes reduce lift and increase drag within reported ranges. Since practical erosion assessment often re-

Table 3. Blade erosion regions, defined by the aerodynamic nodes of the 5 MW NREL reference turbine. Node 18 is used to track aerodynamic data.

Region	Aerodynamic nodes	Distance from hub (m)	Region length (m)	Airfoils included
1	5–6	10.25	8.20	Du40, Du35
2	7–8	18.45	12.30	Du35, Du30
3	9–10	30.75	8.20	Du25
4	11–12	38.95	8.20	Du21
5	13–14	47.15	7.50	NACA64
6	15–19	54.67	6.83	NACA64

lies on coarse-level observed blade deterioration, which is then connected to performance loss and maintenance decisions (Maniaci et al., 2022), this class-based proxy provides a useful proof-of-concept setting for testing the proposed emulator-based workflow.

2.2.2 Sensors

We use five sensors to monitor erosion damage: generator power, aerodynamic lift and drag, blade tip acceleration, and blade root force moment (Duthé et al., 2021; Enríquez Zárate et al., 2022). Although lift and drag measurements are not commonly available, new sensors under development (Barber et al., 2022) measure aerodynamic pressure over an aerofoil section to calculate the dynamic lift coefficient, which can be used to detect changes in airflow patterns due to surface roughness (Abdallah et al., 2022). Similar studies have used these data to track the progression of LEE using OpenFAST software (Abdallah et al., 2022; Duthé et al., 2021). Note that we only track lift and drag at the tip of the blade, `AeroDyn Node 18`, since it is the location where erosion will accumulate most rapidly (Duthé et al., 2021).

Since aerodynamic changes alter blade kinematics, we track flap-wise acceleration of the tip of the blade, used by Du et al. (2020) and Enríquez Zárate et al. (2022), and the edge-wise moment of the root of the blade, used by Moynihan et al. (2022) and Sheibat-Othman et al. (2015).

We use feature extraction to reduce the dimension of the time series outputs from our OpenFAST simulation, a vector with 160 entries per second of simulation time per sensor. Following a similar approach to Enríquez Zárate et al. (2022), we use statistical moments of each of the five output time series, specifically the mean, standard deviation, skew, and kurtosis (Kaewniam et al., 2022), rather than the time series itself.

Enríquez Zárate et al. (2022) also extracted richer features from time series data such as higher-order crossings, mobility, and complexity features, but in Sect. 4 in Figs. 8 and 9 we show high classification accuracy without additional feature extraction. A plausible reason for this may be that (1) we look at a relatively coarse set of erosion levels that naturally have distinct features because they do not naturally overlap, and (2) we are using bulk output statistical features which contain a large amount of information for steady-state wind flow. In Table 4 we summarize the simulator outputs, along with their variable names in OpenFAST and their units. These quantities are the inputs to the random forest classifier which we use to determine erosion accumulation damage.

2.3 Global sensitivity analysis

Global sensitivity analysis (GSA) identifies inputs which drive significant variation in the simulator outputs (Smith, 2024). By using GSA to fix non-influential inputs, fewer simulations are required to train an accurate emulator. Morris (1991) proposed the elementary effect method (Morris screening) as an efficient GSA method for high-dimensional, nonlinear, and computationally expensive models. Velarde et al. (2019) used elementary effect analysis to determine the relative importance of input parameters for analyzing the foundation loads of offshore wind turbines.

To apply the elementary effect analysis, the input space, $\boldsymbol{\mathcal{U}}$, is first scaled to the p-dimensional unit hypercube, which is partitioned into a grid with spacing Δ. A random sample of r initial input points, $\boldsymbol{u}_0^{(j)}$ for $j = 1, \ldots, r$, is taken from these grid points. Starting at each of these random points, a sequence of p points is generated by applying p unit changes to the input dimensions in a random order to create the elementary effect experiment design. For each component index l of $\boldsymbol{u}$ and each of the r trajectories indexed by j, the method calculates the elementary effect using a finite difference:

$$\mathrm{EE}_{l,j}(\mathbf{u}) = [f(\boldsymbol{u} + \mathbf{e}_l) - f(\boldsymbol{u})]/\Delta. \quad (1)$$

The mean and the standard deviation of the elementary effects, $\{\mathrm{EE}_{l,1}, \ldots, \mathrm{EE}_{l,r}\}$, are calculated for each dimension l. To avoid the possibility of large positive and negative elementary effects canceling out, we use $\mu_l^* = \frac{1}{r}\sum_{j=1}^{r}|\mathrm{EE}_{l,j}|$ instead of the mean, μ_l. The standard deviation, which requires the average of elementary effects, μ_l, is given by $\sigma_l = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(\mathrm{EE}_{l,j} - \mu_l)^2}$ (Iwanaga et al., 2022). A large mean elementary effect, μ_l^*, indicates that the l^{th} input has a large overall effect, while a large σ_l indicates strong nonlinear effects or interactions with the other inputs (Cropp and Braddock, 2002). A plot of pairs (μ_l^*, σ_l) for each input $l \in \{1, \ldots, p\}$ is used to determine the relative importance

Table 4. Simulated sensor outputs.

OpenFAST variable	Description	Symbol	Unit
`TipALxB1`	Blade local flapwise absolute tip acceleration	a_{tip}	m s^{-2}
`B1N6Cd`	Drag coefficient at the blade 1 region 6 sensor	C_{D}	(–)
`B1N6Cl`	Lift coefficient at outermost region	C_{L}	(–)
`GenPwr`	Generator power	P_{gen}	MW
`RootMxb1`	Blade root edgewise moment	M_{root}	kN m

of the inputs (Campolongo et al., 2007; Morris, 1991). We use the elementary effect tool provided in the Python library `SALib` (Herman and Usher, 2017; Iwanaga et al., 2022).

2.4 Gaussian process emulation

In this section we review Gaussian process (GP) emulation. For several decades GP emulation has been applied to problems from domains as varied as thermal energy storage (Currin, 1988), electrical circuits (Welch et al., 1992), and the design of chemical experiments (Sacks et al., 1989). GP emulators were introduced to reduce the computational cost of repeatedly evaluating complex numerical models. They are especially useful when the computational cost of a single simulation is high and when an extremely large number of simulations are required. The GP emulator is trained on a limited number of simulations run at design points. Once trained, the GP acts as an interpolator between design points in input space so that evaluating the output quantity of interest (QoI) at untested inputs is essentially free. Therefore, it is feasible to generate much larger training datasets for subsequent machine learning algorithms via an emulator than would be possible using the full simulator. The GP also provides uncertainty quantification for evaluations at new inputs via credible intervals which assess the accuracy of the emulator as a surrogate model for the full physical simulator.

For wind turbine condition monitoring, building training sets for damage classification requires evaluating the simulator hundreds to thousands of times, which can be computationally prohibitive. Steady-state simulations in OpenFAST take on the order of minutes on a typical laptop/desktop computer, while turbulent wind and wave simulations in offshore environments take much longer. GPs are good approximations of simulators that vary smoothly as a function of inputs, which is the case for wind turbines (Rinker, 2016). Further, prediction and uncertainty estimation are robust in the presence of limited data (as compared to neural networks, for example) (Myren and Lawrence, 2021).

2.4.1 Scalar Gaussian process emulation

We begin with a discussion of the basic GP formalism and then describe the extensions we employ for emulation of wind turbines. For scalar-valued output, the wind turbine simulator is treated as a function, f, from the space of input parameters, $\boldsymbol{u} \in \boldsymbol{\mathcal{U}} \subset \mathbb{R}^p$, to the output quantity of interest, $v \in \mathcal{V} \subset \mathbb{R}$. For wind turbine erosion modeling, inputs include (for example) the wind velocity and the blade damage level, with generator power as an output. The unknown deterministic function $f : \boldsymbol{\mathcal{U}} \to \mathcal{V}$ is then modeled as a draw from a random Gaussian process (GP), $\tilde{f}$, with the property that for any finite subset, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \in \boldsymbol{\mathcal{U}}$, the random variables, $\{\tilde{f}(\boldsymbol{u}_1), \ldots, \tilde{f}(\boldsymbol{u}_n)\} \in \mathcal{V}$, follow a multivariate Gaussian distribution (Rasmussen, 2003). The modeling begins with the assumption of a GP prior with a mean trend $\mu : \boldsymbol{\mathcal{U}} \to \mathcal{V}$ and covariance $C : \boldsymbol{\mathcal{U}} \times \boldsymbol{\mathcal{U}} \to \mathbb{R}$. The mean trend is of the form $\mu(\boldsymbol{u}) = \boldsymbol{h}^T(\boldsymbol{u})\boldsymbol{\theta}$ for some choice of q basis functions (often taken to be a constant or linear function of the input parameters), $\boldsymbol{h}(\boldsymbol{u}) = (h_1(\boldsymbol{u}), \ldots, h_q(\boldsymbol{u}))^T$, and regression coefficients, $\boldsymbol{\theta}$. The covariance is of the form $C(\boldsymbol{u}_1, \boldsymbol{u}_2) = \sigma^2 c(\boldsymbol{u}_1, \boldsymbol{u}_2)$, where σ^2 is the variance (output scaling) of the GP, and $c : \boldsymbol{\mathcal{U}} \times \boldsymbol{\mathcal{U}} \to \mathbb{R}$ is the correlation function, which is specified by the choice of a kernel (Rasmussen, 2003; Santner et al., 2003). In this work we use the Matérn 5/2 kernel (Stein, 1999, Gu et al., 2018, Spiller et al., 2023) given by $c(\boldsymbol{u}_1, \boldsymbol{u}_2) = \prod_{k=1}^{p}(1 + \sqrt{5}d_k + \frac{5}{3}d_k^2)\exp(-\sqrt{5}d_k)$. Here $d_k = \frac{1}{\gamma_k}|u_{1,k} - u_{2,k}|$ is a normalized distance between the kth entries of the vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. The correlation coefficients, $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)^T$, determine how rapidly the emulated output changes with respect to each input dimension. Letting $\mathbf{U}_i^{\text{D}}$ $(i = 1, \ldots, n)$ represent the design of training inputs (e.g., different settings of wind speed and blade damage), the corresponding response vector of outputs (power generated for each setting) is given by $\mathbf{v}^{\text{D}} = (v_1^{\text{D}}, \ldots, v_n^{\text{D}})^T$. We denote the training input/response data together as $\mathcal{D}_{\text{GP}} = \{(\mathbf{U}_1^{\text{D}}, v_1^{\text{D}}), \ldots, (\mathbf{U}_n^{\text{D}}, v_n^{\text{D}})\}$.

The GP emulator's predictive posterior distribution evaluated at a new input, $v^* \sim \tilde{f}(\boldsymbol{u}^*)$, is conditioned on the training design/response, $\mathcal{D}_{\text{GP}}$. It follows Student's t-distribution with $n - q$ degrees of freedom (Gu et al., 2018),

$$v^* \mid \mathbf{v}^{\text{D}}, \mathbf{U}^{\text{D}}, \boldsymbol{\gamma}, \boldsymbol{\theta}, \sigma^2 \sim \text{St}(m(\boldsymbol{u}^*), \sigma^2 c''(\boldsymbol{u}^*), n - q). \tag{2}$$

The predictive mean, $m(\boldsymbol{u}^*)$, and predictive variance, $\sigma^2 c''(\boldsymbol{u}^*)$, are given by

$$m(\boldsymbol{u}^*) = \boldsymbol{h}^T(\boldsymbol{u}^*)\boldsymbol{\theta} + \boldsymbol{r}^T(\boldsymbol{u}^*)\mathbf{R}^{-1}(\mathbf{v}^{\text{D}} - \mathbf{H}\boldsymbol{\theta}), \tag{3}$$

$$c''(\boldsymbol{u}^*) = 1 - \boldsymbol{r}^T(\boldsymbol{u}^*)\mathbf{R}^{-1}\boldsymbol{r}(\boldsymbol{u}^*) + \mathbf{w}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{w}^T, \tag{4}$$

where $\mathbf{R}$ is an $n \times n$ matrix of correlations between input design points, $\mathbf{R}_{i,j} = c(\mathbf{U}_i^{\text{D}}, \mathbf{U}_j^{\text{D}})$. Likewise $\boldsymbol{r}$ is a vector of cor-

relations between the new input and each of the inputs in the design, $\boldsymbol{r}_i(\boldsymbol{u}^*) = c(\boldsymbol{u}^*, \mathbf{U}_i^{\mathrm{D}})$ and $\mathbf{w} = (\boldsymbol{r}^T(\boldsymbol{u}^*)\mathbf{R}^{-1}\mathbf{H} - \boldsymbol{h}^T(\boldsymbol{u}^*))$. The GP predictive mean given in Eq. (3) is a best linear unbiased predictor (Santner et al., 2003). Also note that the rule of thumb for the minimum size of a GP design is $10\times$ the number of input dimensions ($10 \times p$) (Berger and Smith, 2019).

To specify $\tilde{f}$, we need estimates of $\sigma^2, \boldsymbol{\theta}$, and $\boldsymbol{\gamma}$. Finding good estimates of the correlation lengths is key to fitting a GP emulator. We use the R package `RobustGASP` (Gu et al., 2019), which considers the marginal posterior density for $\boldsymbol{\gamma}$ to obtain the maximum a posteriori (MAP) estimate of $\boldsymbol{\gamma}$ (Gu et al., 2018). With $\boldsymbol{\gamma}$ in hand, we can estimate the trend parameters and the scalar variance, respectively, as

$$\boldsymbol{\theta} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{v}^{\mathrm{D}}, \text{and} \tag{5}$$

$$\sigma^2 = (n-q)^{-1}(\mathbf{v}^{\mathrm{D}} - \mathbf{H}\boldsymbol{\theta})^T\mathbf{R}^{-1}(\mathbf{v}^{\mathrm{D}} - \mathbf{H}\boldsymbol{\theta}). \tag{6}$$

In the next two sections we introduce parallel partial emulation to handle vector-valued QoIs and zero-censored emulation to handle QoIs with range constraints.

2.4.2 Parallel partial emulation

For simulators with vector-valued output, practitioners following the standard scalar GP methodology typically either train a separate GP for each component of the vector (Bayarri et al., 2015) or use dimension reduction on the output vector before fitting GPs (Higdon et al., 2008). The former approach can be computationally prohibitive, while the latter emulator is no longer an interpolator of the data. Instead, we use a parallel partial Gaussian process emulator (PPE) which approximates the entire output vector $f : \boldsymbol{\mathcal{U}} \to \mathbb{R}^s$ via a single emulator (Gu and Berger, 2016; Dolski et al., 2024). The PPE starts with the assumption that output components can be treated independently, each with their own scalar variance, σ_j^2, and trend parameters, $\boldsymbol{\theta}_j$, $j = 1, \ldots, s$. Yet all output components share a common correlation structure with respect to dependence on input scenarios.

For a design of n input scenarios, the vector-valued responses are now collected in an $n \times s$ matrix $\mathbf{V}^{\mathrm{D}}$, where the jth column, $\mathbf{V}_j^{\mathrm{D}}$, corresponds to all n responses of the jth output component in the training data. Much like the scalar GP, the PPE approximates the jth component of $\mathbf{v}^* = \tilde{f}(\boldsymbol{u}^*)$ at an untested input $\boldsymbol{u}^*$ as a sample from the Student t-distribution with $n - q$ degrees of freedom given by

$$\mathbf{v}_j^* \mid \mathbf{V}_j^{\mathrm{D}}, \mathbf{U}^{\mathrm{D}}, \boldsymbol{\gamma}, \boldsymbol{\theta}_j, \sigma_j^2 \;\sim\; \mathrm{St}(m_j(\boldsymbol{u}^*), \sigma_j^2 c''(\boldsymbol{u}^*), n-q). \tag{7}$$

Here $\boldsymbol{\theta}_j$ is calculated by replacing $\mathbf{v}^{\mathrm{D}}$ by $\mathbf{V}_j^{\mathrm{D}}$ in Eq. (5) and where $m_j(\boldsymbol{u}^*)$ and σ_j^2 are calculated by replacing $\mathbf{v}^{\mathrm{D}}$ by $\mathbf{V}_j^{\mathrm{D}}$ and $\boldsymbol{\theta}$ by $\boldsymbol{\theta}_j$ in Eqs. (3) and (6), respectively.

As each component of the output vector shares a common correlational structure, $\mathbf{R}$ is the only matrix which must be inverted. Thus, the cost of training and predicting with a PPE is comparable to that of a scalar GP.

2.4.3 Zero-censored emulation

Computer model QoIs are often required to be positive or to take values within a given interval, as is the case for wind turbine data. For example, for the NREL reference turbine, the generator power cannot exceed 5 MW, and the standard deviation of any random variable (sensor data) must remain positive. Restrictions on the range of f pose a challenge for GP emulation because Gaussian processes have full support. That is, a GP's predictive outputs take values in the interval from $(-\infty, \infty)$. Further, this severe form of non-stationarity (outputs varying smoothly as inputs change versus outputs not varying at all as inputs change) violates standard GP assumptions. To mitigate the difficulty of fitting a GP to a range-limited QoI, Spiller et al. (2023) proposed the *zero-censored* Gaussian process emulator (zGP).

Regardless of whether the scalar output v is in reality bounded above, $v \in (-\infty, v_{\max}]$, or bounded below, $v \in [v_{\min}, \infty)$, by applying a linear transformation, we can assume that the transformed output is bounded below by zero. The main objective of the zGP is to replace all of the zero outputs with negative values that are consistent with a GP fit only to the positive outputs. To start the process, zero-output training data are initialized with negative values (this can be done with a deterministic rule or by sampling GPs fit to positive data; see Algorithm 2 in Spiller et al., 2023, for more detail). Then for each input in the design that led to a zero output, a GP is fit to all other (imputed negative and positive) output responses. At the left-out design point, this GP's truncated normal distribution (on $(-\infty, 0)$) is sampled. This sample replaces the previous negative sample for that design point in a Gibbs-sampled Markov chain Monte Carlo (MCMC) scheme. The process is then repeated for each of the other design points that led to zero outputs to complete one step in the MCMC chain.

We observe that this imputation process (which can be expensive but is only done once as preprocessing) converges after a few thousand iterations. One hundred samples are kept from this chain (accounting for burn-in and thinning) for each input that originally led to a zero output. For each such input, we take the average of the hundred negative MCMC samples as the final imputed negative response. We then use the imputed negative- and original positive-output-response QoIs to fit a GP. For prediction at an untested input, we take the GP's prediction if positive or zero if the GP's prediction is negative.

For vector-valued outputs, the zGP is a preprocessing step which is independently applied to each component of the output vector that is range-constrained. This process can be sped up via parallelization. The resulting imputed values are used for the training of a PPE (see Seidman, 2025).

The parallel partial and zero-censored Gaussian process emulator (PPzGP) has several advantages that make it an effective emulator for wind turbine simulation. First, it predicts multiple outputs simultaneously, eliminating the need for

multiple, independent surrogate models, and second, it satisfies known range constraints on sensor outputs. While scalar GP emulators have been used to analyze WT power production (Golparvar et al., 2021), wake steering (Gori et al., 2024), blade loads (Clark and Clark, 2022), and mono-pile reliability (Morató et al., 2019), the results in this paper are the first where zero-censored and parallel partial emulation have been combined for WT modeling.

2.5 Random forest classification algorithm

Since erosion damage is challenging to directly observe on wind turbines in the field, the goal is to classify the damage level from the observed multi-modal sensor data. The level of damage due to LEE is stratified into several levels of erosion severity. Random forests have been used for many problems ranging from biology (Boulesteix et al., 2012) and healthcare (Esmaily et al., 2018) to filtering sensor data (Buschjäger and Morik, 2017) and remote sensing (Belgiu and Drăguţ, 2016). Random forests have been applied to diagnose wind turbine faults in real data (Fezai et al., 2020) and to detect irregular sensor patterns due to blade or gearbox damage (Zhang et al., 2018). Random forests handle mixed continuous and categorical input vectors, require low computational cost, provide internal error estimates on untested inputs, and even provide an ordering of the importance level of each of its inputs (Díaz-Uriarte and Alvarez de Andrés, 2006).

Random forests are built from individual decision trees. A decision tree for classification starts with the root node containing the full training dataset. It then bins data into smaller nodes (called leaves). Random forest classifiers use ensembles of decision trees which bin data into sets that have minimum variance. The predictions from the ensemble of decision trees are averaged for the final prediction. Points not used in building a particular decision tree (called out-of-bag samples) can be used to evaluate feature importance. In this work, we use the random forest implementation in `fitcensemble` from MATLAB 2025a (The MathWorks Inc., 2024).

3 Workflow outline

In Fig. 2, we show the workflow we used to compare two classifiers for leading-edge erosion, one trained directly on simulated data and the other trained on emulated data. In Table 5 we summarize the five datasets used in this study and their roles. We use the first dataset for the global sensitivity analysis in Sect. 2.3 to determine which of the simulation inputs have a significant effect on the sensor outputs. The statistical moments of the sensor outputs from the simulation model, which are described in Sect. 2.2.2, are used as inputs to the classifier. In order to eliminate potential bias, we perform feature ranking and selection on an independent dataset in Sect. 4.2 to rank the importance of the sensor outputs for discriminating between erosion classes and select a shorter list of predictors in order to improve classifier accuracy. In Sect. 4.3, we use the third dataset to train and test the emulator. We use the emulator to generate datasets of various sizes to evaluate the performance of the emulator-trained classifier. These datasets are shown in row 4 of Table 5. In Sect. 4.4, we use the fifth dataset as the ground truth against which to compare the emulator-trained classifiers' performance. Classifier hyperparameter tuning is done within each of the repeated 5-fold cross-validation sets. Predictions are made on the testing portion of each split, which is not seen by the model during hyperparameter tuning. The emulator-trained classification models do not train on simulation data.

4 Results

4.1 Global sensitivity analysis

The elementary effect study was carried out on the means of the five sensor output time series described in Sect. 2.2. The elementary effect analysis results show that wind shear has the least influence among the inputs tested, while wind speed has the most for each of the output QoIs. In Fig. 3 we show the normalized elementary effect plots for each sensor output. For each graph, inputs with a large μ^* have a strong, linear effect on the output, while those with a large σ interact nonlinearly with other inputs. We use a grid width of $\Delta = 1/9$ in the normalized input dimensions and $r = 25$ trajectories for a total of 150 simulations.

As wind shear has almost no effect on any of the output QoIs we considered, we fixed wind shear at the nominal value of 0.2. However, the limited impact of wind shear may be due to our assumption of steady-state wind conditions. Whether wind shear is influential in the turbulent case would have to be analyzed. Since μ^* is far larger for wind speed than for air density, erosion, and wind direction, this sensitivity analysis supports the need for data from across a broad range of wind speeds for training erosion detection models (Papi et al., 2020). Importantly, the rankings of air density, erosion, and wind direction vary depending on the QoI. For instance, air density impacts generator power and root moment more than other non-wind-speed inputs. Air density is known to influence turbine generator power under the same constant wind speed conditions (Golparvar et al., 2021), as it affects the power available in the wind by increasing air mass interacting with the blades. (Specifically, $P = \frac{1}{2}\rho A v^3 C_\mathrm{p}$, where ρ is air density, A is the swept area, v is wind velocity, and C_p is the power coefficient.) Higher air density enhances lift force generation in blades, amplifying root bending moments and increasing power output. Wind direction, which is the relative angle of the wind to the nacelle direction, is the third-most influential input for blade tip acceleration, but the fourth for every other output. In summary, the GSA study reveals that lift and drag channels are more sensitive to erosion than the other outputs we tracked (Abdallah et al., 2022; Carmona-Troyo et al., 2025;

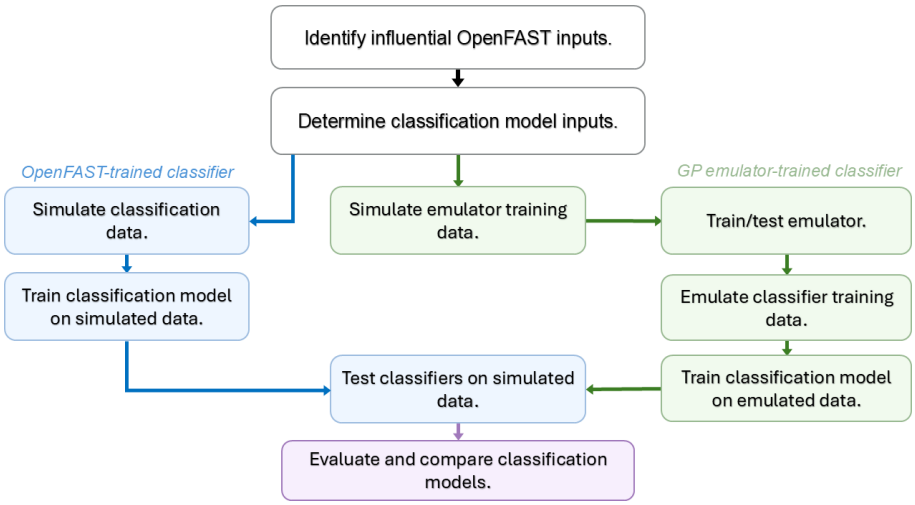

Figure 2. Workflow for comparing a simulator-trained classifier with a GP-emulator-trained classifier for leading-edge-erosion classification.

Table 5. Dataset description.

Number	Dataset	Total size	Source	Used for
1	Global sensitivity dataset	150	OpenFAST	Model input screening
2	Classifier feature selection dataset	500	OpenFAST	Selecting classifier input variables
3	Emulator-training dataset	210	OpenFAST	Training emulators
4	Emulator-generated dataset	500/1000/5000/10 000/50 000	GP emulator	Training emulator-trained classifier
5	Classifier testing dataset	600	OpenFAST	Hyperparameter tuning and final classifier evaluation

Duthé et al., 2021). As a result, we were left with a total of 9 inputs for both the simulator and emulator.

4.2 Classifier input selection

Classification results often improve when the dimension of the input space is reduced. In this subsection, we describe how we selected the inputs to the LEE classification problem from the sensor outputs. We considered 23 potential damage predictors, including wind speed, wind direction, and air density along with the mean, standard deviation, skew, and kurtosis of the QoIs listed in Table 4. To perform this reduction we use a separate OpenFAST-generated dataset and a random forest classifier to rank the potential predictors by how effectively they differentiate damage classes. To reduce bias, we repeat the predictor ranking calculation on 10 different 5-fold cross-validation splits of the simulated data. These training data include 100 samples from each erosion class obtained using a Latin hypercube design (Helton and Davis, 2003).

We show the ranking of the damage predictors in Fig. 4. The variability in the score assigned to each predictor is relatively small, which indicates that the rank of each damage predictor is stable across model fits. The truncation limit for inclusion of a sensor output was set to be 85 % of the sum over the predictor importance of all of the inputs. This ranking is also consistent with the elementary effect sensitivity study. The most important sensor outputs include the mean and standard deviation of the drag coefficient and the mean lift coefficient. Additional selected predictors include the skewness and standard deviation of blade tip acceleration, the mean blade root moment, the standard deviation of the lift coefficient, and the mean generator power.

4.3 Emulator evaluation

4.3.1 Comparing scalar-GP emulation and PPzGP emulation

We trained and evaluated the GP emulators on the third dataset in Table 5. These simulations were generated using OpenFAST with 50 samples for each of the four eroded blade classes and 10 for the clean blade class. The training dataset varies the three environmental inputs identified by the GSA using Latin hypercube sampling over the ranges in Table 6. The erosion level in blade sections 1–6 are varied according to values given in Table 6, which we selected to ensure coverage of the stochastic erosion classes from Sect. 4.4.

First, we compare the training efficiency and prediction accuracy of scalar-GP emulation to that of standard parallel

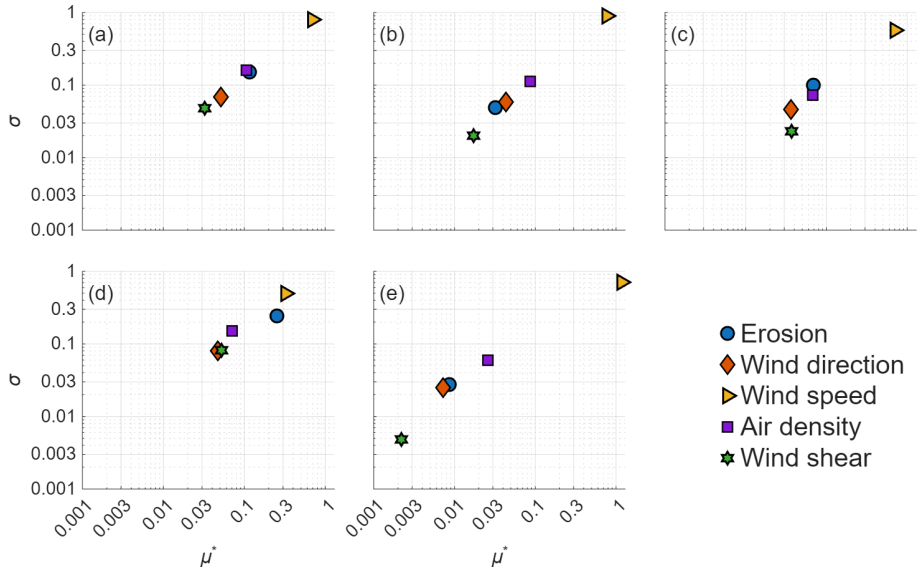

Figure 3. Elementary effect plots normalized for variable magnitude on a log vs. log scale for the mean of the QoIs; **(a)** blade root moment (M_{root}), **(b)** blade tip acceleration (a_{tip}), **(c)** lift coefficient (C_{L}), **(d)** drag coefficient (C_{D}), and **(e)** generator power (P_{gen}). The same symbols are used in each subfigure for the five inputs: erosion severity, wind direction, wind speed, air density, and wind shear.

Table 6. Erosion level ranges for the six blade regions and five erosion severity classes in the training data for the emulation studies.

Severity class (α)	e_1	e_2	e_3	e_4	e_5	e_6
$\alpha_1 = 0.00$	{0}	{0}	{0}	{0}	{0}	{0}
$\alpha_2 = 0.25$	[0.01, 0.02]	[0.04, 0.07]	[0.07, 0.11]	[0.12, 0.15]	[0.14, 0.19]	[0.18, 0.25]
$\alpha_3 = 0.50$	[0.03, 0.04]	[0.09, 0.13]	[0.15, 0.22]	[0.21, 0.31]	[0.29, 0.39]	[0.36, 0.51]
$\alpha_4 = 0.75$	[0.04, 0.07]	[0.13, 0.20]	[0.23, 0.32]	[0.32, 0.46]	[0.42, 0.59]	[0.55, 0.77]
$\alpha_5 = 1.00$	[0.06, 0.09]	[0.17, 0.26]	[0.30, 0.44]	[0.43, 0.61]	[0.57, 0.78]	[0.73, 1.00]

Table 7. Computational cost of the standard emulators and PPEs. Values are reported as mean ±95 % confidence interval. Prediction times correspond to generating 10 000 predictions.

Emulator	Training time	Prediction time
GP	52.3 s ± 1.0 s	8.0 s ± 0.2 s
zGP	48.2 s ± 1.0 s	7.6 s ± 0.1 s
PPGP	5.8 s ± 0.6 s	0.9 s ± < 0.1 s
PPzGP	3.7 s ± 0.5 s	0.9 s ± < 0.1 s

partial Gaussian process (PPGP) emulation, with and without the additional preprocessing step of zero-censored emulation. We test the emulators and compute 95 % confidence levels using five repeated 5-fold cross-validation sets of the OpenFAST dataset. Each split allocates 168 training points proportionally across erosion severity classes, with the remaining 42 data points reserved for testing.

In Table 7 we show that the time taken to train the parallel partial emulator is 8 times less than with 8-fold scalar emulation, which is to be expected because the output vector has eight components. In Table 8 we show the normalized root mean square error (NRMSE), which, to allow for comparisons between outputs with different scales, is defined to be the root mean square error divided by the output range. The NRMSE is slightly less with the PPzGP emulator than with the standard GP emulator, which is trained to predict each output separately, except for the standard deviation of the lift coefficient, σ-C_{L}, and the mean of the generator power, μ-P_{gen}.

In Table 9, we show the average percentage of predictions falling within the 95 % credible intervals of the PPzGP emulator for each of the eight sensor quantities. The 95 % GP credible interval contains the true function value with a rate of 0.95. If the GP model is calibrated well, under repeated sampling, the empirical coverage of the credible interval will also converge to 95 %. We observe coverage between 84 % and 95 % for the PPzGP. We attribute this to model misspecification (choice of kernel, violations of the stationarity assumption, etc.) and effects of sampling noise in the out QoI moment calculations, which we do not model directly.

In Table 10 we compare the performance of the standard GP and PPzGP emulators across the selected sensor out-

Table 8. Normalized root mean square error (NRMSE) for each emulator and sensor quantity selected in Fig. 4. Values are reported as mean ± 95 % confidence interval as the percentage of the output range.

Emulator	μ-C_D	σ-C_D	μ-C_L	γ-a_{tip}	μ-M_{root}	σ-C_L	σ-a_{tip}	μ-P_{gen}
GP	(6.5 ± 1.3) %	(12.1 ± 1.0) %	(2.6 ± 0.3) %	(13.4 ± 1.5) %	(9.2 ± 0.4) %	(6.9 ± 1.2) %	(11.1 ± 0.9) %	(1.7 ± 0.2) %
zGP	(6.5 ± 1.3) %	(11.8 ± 1.1) %	(2.6 ± 0.3) %	(13.4 ± 1.5) %	(9.2 ± 0.4) %	(6.8 ± 1.0) %	(9.5 ± 0.7) %	(1.7 ± 0.1) %
PPGP	(6.3 ± 1.1) %	(12.6 ± 1.4) %	(2.5 ± 0.2) %	(13.4 ± 1.6) %	(8.8 ± 0.5) %	(7.6 ± 0.9) %	(9.3 ± 0.7) %	(2.0 ± 0.2) %
PPzGP	(6.1 ± 1.1) %	(11.8 ± 1.2) %	(2.6 ± 0.2) %	(12.7 ± 1.5) %	(8.5 ± 0.5) %	(7.3 ± 0.9) %	(9.0 ± 0.7) %	(2.1 ± 0.2) %

Table 9. Empirical coverage in percent of the nominal 95 % credible intervals for each output.

Emulator	μ-C_D	σ-C_D	μ-C_L	γ-a_{tip}	μ-M_{root}	σ-C_L	σ-a_{tip}	μ-P_{gen}
GP	(84.7 ± 3.5) %	(85.4 ± 2.4) %	(85.7 ± 2.5) %	(86.1 ± 2.0) %	(78.6 ± 2.5) %	(85.3 ± 3.0) %	(79.4 ± 2.7) %	(81.9 ± 2.6) %
zGP	(84.8 ± 3.5) %	(86.4 ± 1.8) %	(85.7 ± 2.5) %	(86.1 ± 2.0) %	(78.6 ± 2.5) %	(86.6 ± 2.6) %	(83.7 ± 2.2) %	(84.8 ± 2.5) %
PPGP	(90.6 ± 2.2) %	(87.6 ± 2.6) %	(91.1 ± 1.7) %	(87.3 ± 2.1) %	(82.2 ± 2.6) %	(89.3 ± 2.6) %	(83.3 ± 2.7) %	(94.5 ± 1.7) %
PPzGP	(92.8 ± 2.2) %	(89.1 ± 2.6) %	(93.0 ± 1.7) %	(88.0 ± 1.8) %	(84.1 ± 2.8) %	(89.5 ± 2.5) %	(84.8 ± 2.6) %	(95.2 ± 1.5) %

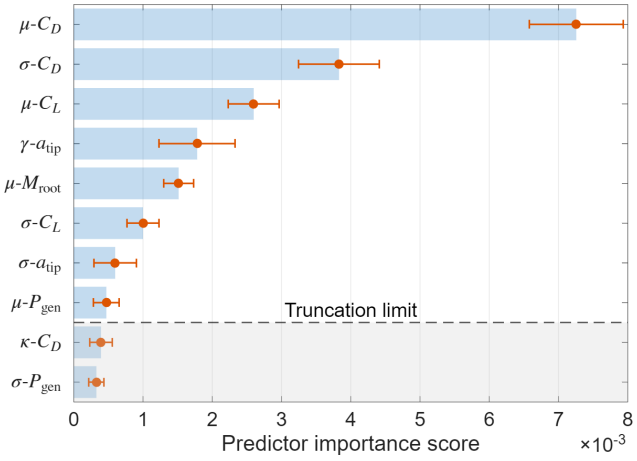

Figure 4. Average predictor importance score via 10 5-fold repeated cross-validation studies. Error bars represent the standard deviation of the score over the 50 train/test splits. The variables denote the mean (μ), standard deviation (σ), skew (γ), and/or kurtosis (κ) of the drag coefficient (C_D), the lift coefficient (C_L), the blade tip acceleration (a_{tip}), the blade root moment (M_{root}), and the generator power (P_{gen}).

puts using paired 95 % confidence intervals for differences in NRMSE and empirical coverage. A positive difference in NRMSE indicates the PPzGP emulator has a lower error, while a negative difference in coverage implies the PPzGP has better coverage. These results show that PPzGP emulation provides improved uncertainty calibration for all sensor outputs while maintaining prediction accuracy comparable to the standard GP. For most outputs, the paired NRMSE confidence intervals include zero, indicating that the mean prediction errors of the two emulators are not significantly different. In contrast, the coverage intervals are mostly negative, indicating that the PPzGP intervals are generally closer to the nominal 95 % level than those of the standard GP. This suggests that incorporating physical output constraints can improve emulator uncertainty quantification without degrading mean prediction accuracy. These benefits are obtained alongside the reduced training time and storage requirements of the PPzGP formulation.

Table 10. Paired differences between the standard GP and PPzGP emulators. Values are 95 % confidence intervals of the paired differences. Differences are computed as standard GP minus PPzGP.

Sensor output	NRMSE difference	Coverage difference
$\mu - C_D$	[−0.2 %, 1.0 %]	[−10.8 %, −5.3 %]
$\sigma - C_D$	[−0.6 %, 1.0 %]	[−6.4 %, −1.0 %]
$\mu - C_L$	[−0.1 %, 0.2 %]	[−9.3 %, −5.4 %]
$\gamma - a_{tip}$	[−0.1 %, 1.0 %]	[−4.1 %, 0.3 %]
$\mu - M_{root}$	[0.2 %, 1.1 %]	[−8.1 %, −3.0 %]
$\sigma - C_L$	[−1.1 %, 0.3 %]	[−6.6 %, −1.8 %]
$\sigma - a_{tip}$	[1.6 %, 2.7 %]	[−8.0 %, −2.7 %]
$\mu - P_{gen}$	[−0.5 %, −0.2 %]	[−15.6 %, −11.1 %]

4.3.2 PPzGP emulator fitting

In Fig. 5, we compare the PPzGP emulator predictions (blue) with the simulated outputs (red), along with the associated 95 % credible intervals for four sensor outputs for one test split. For each output, the majority of true values fall within the credible intervals, indicating good emulator calibration. The generator power predictions exhibit narrow credible intervals, and the zGP ensures that the mean power does not exceed the maximum rated power of 5 MW. The statistical constraint that the blade tip acceleration standard deviation is non-negative is also enforced.

In Table 11, we distinguish between the cost of a single simulation, training, and generating samples with the GP emulator. All timings are wall-clock times measured on a machine with an Intel(R) Core™ Ultra 7 155H processor

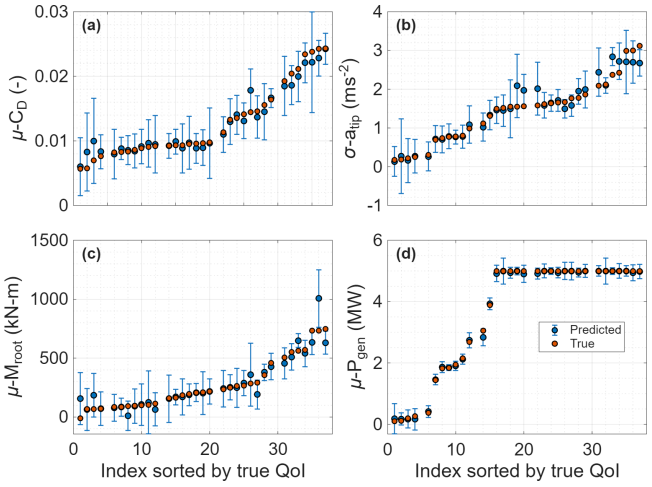

Figure 5. Predictions, ordered by simulated (true) value, and 95 % credible intervals of the PPzGP emulator model for four of eight outputs: **(a)** mean of drag sensor, **(b)** standard deviation of blade tip acceleration, **(c)** mean of root moment, and **(d)** mean of generator power.

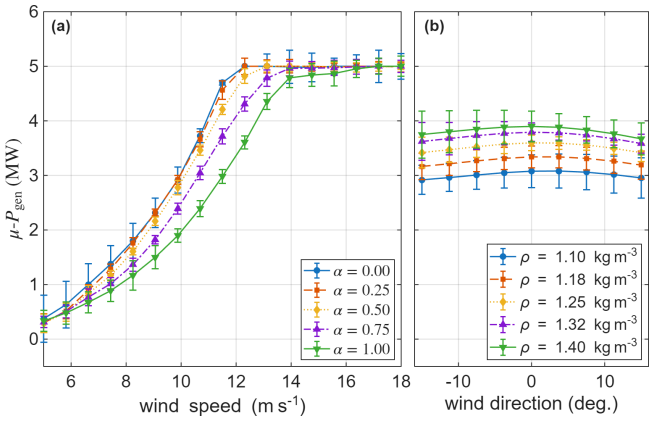

Figure 6. **(a)** Mean generator power as a function of wind speed for five levels of LEE. **(b)** Mean generator power as a function of wind direction for different air densities at fixed wind speed with clean blades.

Table 11. Wall-clock computational cost for OpenFAST simulation, emulator training, emulator prediction, and full classifier-training data generation. Prediction times are reported for generating 10 000 samples.

Action	Wall-clock time (s)
Single OpenFAST simulation	28.4 ± 0.3
PPzGP training	3.67 ± 0.56
Standard GP training	52.3 ± 1.0
PPzGP prediction	0.93 ± 0.01
Standard GP prediction	7.6 ± 0.4
Direct OpenFAST generation, 10 000 samples	$\approx 2.84 \times 10^5$
Full PPzGP workflow, 10 000 samples	$\approx 6.2 \times 10^3$

(3.80 GHz) and 32 GB RAM using OpenFAST-v3.5.0. Simulator timings were estimated from 25 sequential OpenFAST runs at randomly selected inputs. Emulator timings were estimated from five repeated 5-fold splits on 210 data points; prediction timings correspond to generating 10 000 emulator samples and were repeated 25 times.

Our OpenFAST simulation requires 28.4 ± 0.3 s, whereas generating 10 000 PPzGP samples requires 0.93 ± 0.01 s, corresponding to approximately 9.3×10^{-5} s per emulator sample. Thus, evaluating the fitted PPzGP is roughly 3×10^5 times faster than running OpenFAST once. This prediction-only speedup does not include the cost of constructing the emulator. The full PPzGP workflow includes the 210 OpenFAST simulations used to train the emulator, zero-censored preprocessing for range-limited outputs, PPzGP fitting, and generation of the emulator-based classifier-training data. Using the measured OpenFAST timing, the simulator portion of this workflow costs approximately 5964 s. Zero-censored preprocessing is the most expensive part of fitting the PPzGP, but it can be sped up by preparing each output in parallel, approximately 280 s on our laptop. It takes 3.7 s for PPzGP fitting and 0.93 s to generate 10 000 samples; the full surrogate workflow requires approximately 6.2×10^3 s. In contrast, directly generating 10 000 OpenFAST samples would require approximately 2.84×10^5 s, giving a full-workflow speedup of about 46×. The prediction-only speedup and full-workflow speedup indicate that querying the fitted emulator is extremely fast, and the full workflow remains dominated by the initial OpenFAST simulations needed to train the emulator.

In Fig. 6 we show that the PPzGP emulator captures the essential wind turbine dynamics represented by the full simulator. For this analysis, erosion is set deterministically for the values of α given in the legend with $\mathbf{e} = (\alpha x_1, \ldots, \alpha x_6)$. In the left panel, we show generator power as a function of wind speed across five erosion severity classes. The PPzGP accurately reproduces the power curve, with higher erosion levels requiring greater wind speeds to achieve an equivalent power relative to the clean blades. The non-eroded power-curve aligns with findings from Golparvar et al. (2021), who used a GP to estimate the first and second statistical moments of generator power for a different turbine model. The PPzGP provides vector-valued predictions across multiple sensor outputs. In the right panel we demonstrate that the model captures two key physical trends: the power increases with air density and decreases when the rotor is misaligned with the wind. These relationships are consistent with prior studies (Hulsman et al., 2022) and are also evident in Fig. 6. Power loss due to erosion is proportionally highest at lower power levels, remains significant up to the rated wind speed, and disappears beyond that threshold (Campobasso et al.,

2023). However, as we see in Fig. 5, model limitations exist. The prediction performance deteriorates in extreme loading conditions, particularly for high values of tip acceleration and root moment mean. These deficiencies likely stem from a lack of training data in more extreme turbine operational states.

4.4 Classification comparison

In this section we present our main results comparing two LEE classifiers using the random forest algorithm. The first, which we refer to as the simulator-trained classifier, is trained directly on the fifth dataset (OpenFAST) in Table 5. This dataset includes 120 samples from each of the five erosion severity classes where the wind speed, wind direction, and air density are varied within each class using a Latin hypercube design with ranges in Table 2. The second, which we refer to as the emulator-trained classifier, is trained using data predicted by the PPzGP emulator. For the PPzGP emulator, we compared performance using emulated datasets of several different sizes. To eliminate bias from favorable splits, we use 10 repeated 5-fold cross-validation experiments with samples balanced across the damage classes. For both classifiers, we use the same sensor outputs identified using an independent dataset in Sect. 4.2.

Within each cross-validation split, 500 simulated samples are used to train the simulation-based classifier and tune the random forest hyperparameters. Using a further 5-fold validation split within each training split to avoid overfitting, we optimize the number of learning cycles, corresponding to the number of trees in the ensemble; the maximum number of splits, which controls individual tree complexity; and the minimum leaf size, which regularizes the terminal-node size and helps limit overfitting. The emulator-trained classifier is trained using generated datasets with total sizes of 500, 1000, 5000, 10 000, and 50 000 data points. We use the emulated data to train a single model using the same hyperparameter tuning procedure. We evaluate classifier performance using accuracy, balanced accuracy, macro-F1 score, receiver operating characteristic (ROC) curves, area under the curve (AUC), and confusion matrices. Accuracy reports the overall fraction of correct predictions. The F1 score is defined by $\mathrm{F1} = 2\frac{pr}{p+r}$, where the precision, p, is the percentage of correct predictions out of all predictions for a given class, and the recall, r, is the percentage of correct predictions out of all true samples for a class. Balanced accuracy and macro-F1 give equal weight to each erosion class and are therefore useful when class-wise performance differs. ROC curves show the tradeoff between true-positive and false-positive rates, and the AUC summarizes class separability, with larger values indicating better performance. To compare simulator-trained and emulator-trained classifiers, paired confidence intervals are computed on model predictions from the same held-out test split from the repeated cross-validation studies.

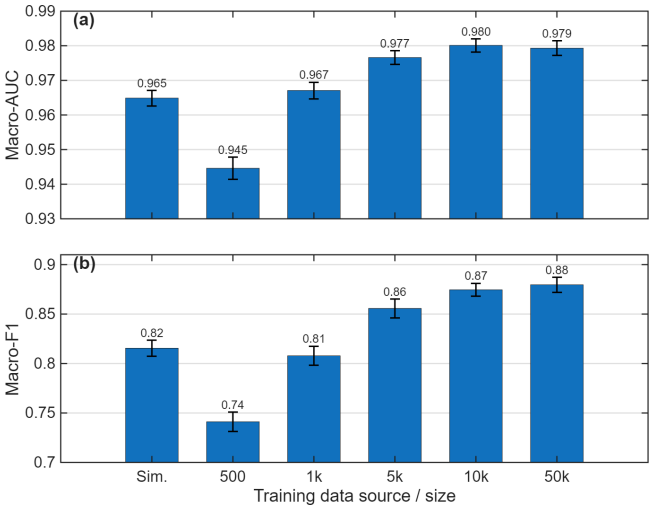

Figure 7. Comparison of classifier performance as a function of training data source and size. Panel **(a)** shows macro-AUC, and panel **(b)** shows macro-F1 for the simulator-trained classifier and emulator-trained classifiers using 500, 1000, 5000, 10 000, and 50 000 emulated training samples. Error bars indicate 95 % confidence intervals across repeated cross-validation splits.

In Fig. 7 we show that increasing the size of the emulated training dataset improves the downstream performance of the LEE classifier. As assessed using macro-AUC and macro-F1 metrics, with 500 samples the performance of the emulator-trained classifier is worse than that of the simulator-trained classifier; with 1000 samples the performance is comparable; and with 5000, 10 000, and 50 000 samples the emulator-trained classifier outperforms the simulator-trained classifier.

In Table 12 we show the paired differences between the emulator-trained and simulator-trained classifier scores for each emulated training-set size. Positive values indicate that the emulator-trained classifier performs better on the same held-out simulated test sets, while confidence intervals that do not contain zero indicate statistically meaningful differences across repeated splits. The results show a clear dependence on emulated training-set size. With only 500 emulated samples, the emulator-trained classifier performs worse than the simulator-trained classifier across all three metrics. With 1000 samples, the differences are small, indicating comparable performance. With 5000, 10 000, and 50 000 samples, the paired differences are positive for the macro-AUC, balanced accuracy, and macro-F1, showing that the larger emulator-generated datasets improve downstream classifier performance. These results suggest that the main benefit of the emulator-based workflow is not that each emulated point exactly reproduces a simulator point, but that the emulator can generate substantially larger training datasets at negligible additional computational cost, improving aggregate classifier performance.

Table 12. Paired differences in classifier performance between emulator-trained and simulator-trained classifiers for different emulated training-set sizes. Values are reported as mean paired difference with 95 % confidence interval. Differences are reported as points, computed as 100× the raw metric difference. Positive values indicate better performance for the emulator-trained classifier.

Emulated training	Macro-AUC		Balanced accuracy		Macro-F1	
size	Diff. [95 % CI]	p value	Diff. [95 % CI]	p value	Diff. [95 % CI]	p value
500	−2.0 [−2.3, −1.8]	$< 10^{-14}$	−7.4 [−8.4, −6.4]	$< 10^{-14}$	−7.4 [−8.5, −6.4]	$< 10^{-14}$
1000	0.2 [< 0.0, 0.4]	5.20×10^{-2}	−0.9 [−1.9, 0.1]	8.16×10^{-2}	−0.8 [−1.8, 0.3]	1.38×10^{-1}
5000	1.2 [1.0, 1.3]	$< 10^{-14}$	3.8 [2.9, 4.7]	7.42×10^{-11}	4.0 [3.1, 4.9]	8.02×10^{-12}
10 000	1.5 [1.3, 1.7]	$< 10^{-14}$	5.6 [4.8, 6.4]	$< 10^{-14}$	5.9 [5.1, 6.8]	$< 10^{-14}$
50 000	1.4 [1.2, 1.6]	$< 10^{-14}$	6.1 [5.2, 7.0]	$< 10^{-14}$	6.4 [5.5, 7.3]	$< 10^{-14}$

Table 13. Per-class AUC comparison between simulator-trained and emulator-trained classifiers. The AUC difference is computed as emulator-trained minus simulator-trained, with a 95 % confidence interval for the paired difference. Differences are reported as points, computed as 100× the raw metric difference.

Class	Emulator-trained AUC	Simulator-trained AUC	AUC Diff.
$\alpha = 0$	1.00	1.00	0.01 [< 0.01, 0.02]
$\alpha = 0.25$	1.00	0.99	0.31 [0.17, 0.46]
$\alpha = 0.50$	0.97	0.96	0.95 [0.50, 1.41]
$\alpha = 0.75$	0.95	0.90	5.22 [4.53, 5.91]
$\alpha = 1.00$	0.98	0.97	1.13 [0.81, 1.44]

In Table 13 we show that the AUC performance of the best emulator-trained classifier is comparable to or slightly better than that of the simulator-trained classifier across all erosion classes. Overall, the simulator-trained classifier achieved a mean accuracy of 81.9 % with a 95 % confidence interval of [81.1 %, 82.7 %], while the emulator-trained classifier achieved a higher mean accuracy of 87.5 % with a 95 % confidence interval of [86.8 %, 88.2 %].

A per-class comparison indicates that the emulator-trained classifier preserves performance on the clean class while improving accuracy for all eroded classes (see Table 14). The strongest improvement occurs for the 0.75 erosion class, where per-class accuracy increases by 5.6 percentage points, precision by 11.7 percentage points, and recall by 19.1 percentage points. For the fully eroded class, the emulator-trained classifier also improves per-class accuracy, precision, and recall by 3.4, 9.2, and 7.7 percentage points, respectively. The only notable tradeoff is a small decrease in recall for the mild erosion class, 0.25, despite improved precision and accuracy for that class. The largest improvements occur for the intermediate erosion classes, suggesting that the larger emulator-generated training set is especially helpful for distinguishing the more difficult class boundaries.

The normalized confusion matrices in Fig. 8 show that both classifiers are fairly accurate, predicting the correct label with a rate greater than 0.60. Most notably the emulator-trained classifier shows slightly stronger performance for the intermediate and severe erosion classes, suggesting that the larger emulator-generated training set improves class-wise separation in the regions where misclassification is most likely.

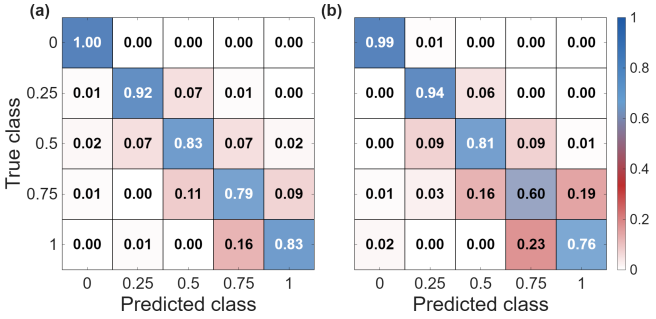

Figure 8. Normalized confusion matrices comparing the best emulator-trained classifier and the simulator-trained classifier. Panel **(a)** shows the emulator-trained classifier with 10 000 training points, and panel **(b)** shows the simulator-trained classifier. Rows indicate the true erosion class, columns indicate the predicted erosion class, and diagonal entries correspond to correct classifications.

The ROC curves in Fig. 9 provide further support for the result in Fig. 8. Both classifiers exhibit high AUC values across all erosion classes, but the emulator-trained classifier provides noticeably stronger separation for the intermediate damage classes. This improvement is especially clear for classes $\alpha = 0.5$ and $\alpha = 0.75$, where the simulator-trained classifier has the lowest AUC values, indicating that the larger emulator-generated training set helps resolve the most challenging class boundaries.

Table 14. Class-wise differences between models, emulator-trained minus simulator-trained. Differences are reported in points, defined as 100× the raw metric value. Each entry reports difference (p value).

Metric	0	0.25	0.50	0.75	1.00
Per-class accuracy	0.0 (1.000)	0.8 (0.013)	1.5 ($< 10^{-3}$)	5.6 ($< 10^{-14}$)	3.4 ($< 10^{-12}$)
Precision	−0.7 (0.272)	5.0 ($< 10^{-5}$)	4.4 (0.001)	11.7 ($< 10^{-12}$)	9.2 ($< 10^{-9}$)
Recall	0.7 (0.002)	−2.2 (0.048)	2.8 (0.009)	19.1 ($< 10^{-14}$)	7.7 ($< 10^{-8}$)

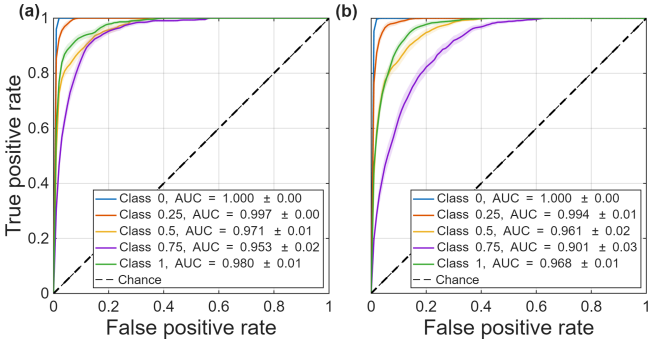

Figure 9. One-versus-rest ROC curves for the best emulator-trained classifier and the simulator-trained classifier. Panel **(a)** shows the emulator-trained classifier using 10 000 emulated training samples, while panel **(b)** shows the simulator-trained classifier. Curves are shown for each erosion class, with shaded regions indicating variability across repeated cross-validation splits and reported AUC values given as mean ± standard deviation.

Table 15. Per-class recall for the emulator-trained classifier with 10 000 generated samples and the simulator-trained classifier. Values are reported as mean recall with 95 % confidence intervals.

True erosion severity class	Emulator-trained recall [95 % CI]	Simulator-trained recall [95 % CI]
$\alpha = 0.00$	1.00 [1.00, 1.00]	0.99 [0.99, 1.00]
$\alpha = 0.25$	0.92 [0.90, 0.93]	0.94 [0.92, 0.95]
$\alpha = 0.50$	0.83 [0.81, 0.85]	0.81 [0.78, 0.83]
$\alpha = 0.75$	0.79 [0.77, 0.81]	0.60 [0.57, 0.63]
$\alpha = 1.00$	0.83 [0.81, 0.85]	0.76 [0.73, 0.78]

While the classification accuracy improvement is modest, the computational savings are significant. Once trained the PPzGP emulator generates over 10 000 data points in less than a second. These data points are similar enough to true simulated data as to allow for comparable, or slightly improved, classifier accuracy.

Because underpredicting erosion severity is more consequential than overpredicting it, in Table 15 we show the per-class recall, which is of particular importance for the most severe erosion class. One-quarter of the most severely eroded blades were classified incorrectly using the simulator-trained classifier. In contrast, the emulator-trained classifier has a recall of 83 % in the most severely eroded case. However, both of these results are high if the risk of missing a fault is severe.

Below we perform a risk-sensitive thresholding experiment. Figure 8 shows that both classifiers can underpredict severe damage, including cases where the simulator-trained classifier assigns fully eroded blades to the clean class. To mitigate this behavior, we modify the decision rule for the severe erosion class, $\alpha = 1.00$, by assigning a sample to the severe class whenever its severe-class probability exceeds a threshold, τ. Lowering τ increases the number of severe-damage alarms, reducing severe false negatives while increasing the false-alarm rate. We choose τ by minimizing an asymmetric confusion cost that penalizes underprediction 4 times more heavily than overprediction:

$$C_{ij} = \begin{cases} 0, & i = j, \\ 4|i-j|, & j < i, \\ |i-j|, & j > i, \end{cases} \tag{8}$$

where $(j < i)$ corresponds to underprediction of erosion severity, and $(j > i)$ corresponds to overprediction. The average weighted confusion cost is then defined by $\overline{C} = \frac{1}{N}\sum_{i,j} n_{ij} C_{ij}$, where n_{ij} is the number of samples with true class i predicted as class j, and N is the total number of test samples. This asymmetric weighting reflects the maintenance-risk perspective that failing to detect severe erosion is more consequential than conservatively overpredicting the erosion severity.

In Fig. 10 we show that decreasing τ reduces the severe false-negative rate but increases the false-alarm rate for both classifiers, mostly for $\alpha = 0.75$. The weighted confusion cost is minimized at a larger threshold for the emulator-trained model, with a correspondingly lower weighted cost than the simulator-trained model.

In Fig. 11 we show the row-normalized confusion matrices obtained using the cost-minimizing thresholds identified in Fig. 10. Compared with the default decision rule ($\tau = 0.5$), the tuned classifiers predict severe damage more often. This conservative tuning reduces the rate of severe underprediction for both the simulator-trained and emulator-trained classifiers, although it also increases overpredictions. Figure 11b shows a noticeable reduction in recall for the $\alpha = 0.75$ class under the tuned rule compared to Fig.8b, while the emulator-trained classifier maintains more consistent recall across the severe classes. Overall, these results show that the emulator-trained classifier can be tuned effectively to reduce high-risk

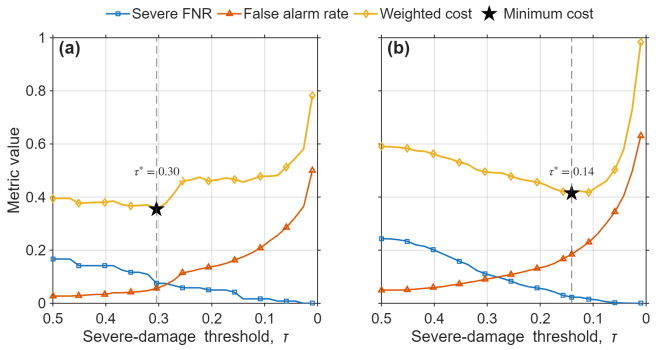

Figure 10. Risk-sensitive threshold tuning for severe-damage detection. The severe-damage alarm threshold τ is varied for the emulator-trained classifier **(a)** and simulator-trained classifier **(b)**. The dashed vertical line and star marker indicate the threshold τ^* that minimizes the weighted confusion cost for each classifier.

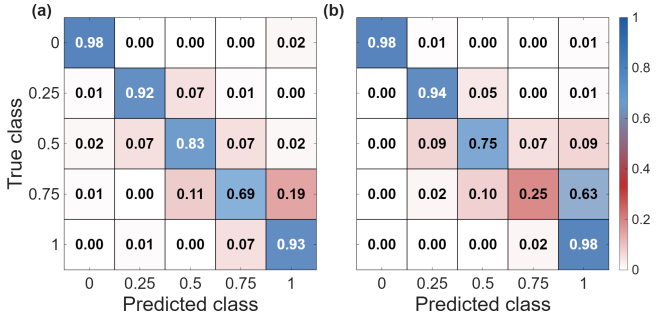

Figure 11. Row-normalized confusion matrices for the emulator-trained classifier **(a)** and simulator-trained classifier **(b)** at the cost-minimizing severe-damage threshold τ^* in Fig. 10.

severe false negatives, with a risk-reduction tradeoff comparable to that of the simulator-trained classifier.

4.4.1 Robustness to sensor noise and bias

To assess whether the high classification performance in the baseline study is due to an overly clean and easily separable dataset, we performed an additional noisy-sensor classification experiment. Sensor noise was modeled by adding both a sample-level Gaussian bias and time-varying Gaussian noise to each classifier input channel. The bias standard deviation was set to 10 % of the average absolute mean sensor value, and the pointwise noise standard deviation was set to 10 % of the corresponding time series standard deviation. We kept the same sensor outputs described in Table 4. We then repeat the classifier input ranking using a noisy version of dataset 2 from Table 5 and the selection procedure from Sect. 4.2, also shown in Fig. 4, retaining $\mu - C_D$, $\sigma - C_D$, $\gamma - a_{tip}$, $\sigma - C_L$, $\mu - C_L$, $\mu - M_{root}$, $\mu - a_{tip}$, $\kappa - C_D$, and $\kappa - a_{tip}$. The emulator is then trained on a noisy version of its original training dataset. We compare the simulator-trained classifier to the emulator-trained classifier. We fit the emulator using a noisy version of dataset 3 from Table 5 and generated 10 000 data points for training the classifier. We use a noisy version of dataset 5 from Table 5 for 10 repeated 5-fold cross-validation studies.

In Table 16 we show that noisy sensing changes the class-wise error distribution. Compared to Table 15, all recall values are lower as expected. In comparison to the simulator-trained model on noisy data, the emulator-trained classifier has lower recall for the lowest erosion classes. However, recall improves for the intermediate classes. Recall for the most severe class remains statistically comparable between the two classifiers.

4.4.2 Classification without direct aerodynamic sensing

To further assess the dependence of the classifier on direct aerodynamic sensing, we repeat the classification study after removing the lift and drag coefficient channels. In addition to M_{root}, a_{tip}, and P_{gen}, we consider an expanded set of structural outputs in Table 17. These channels are not directly tied to the imposed aerodynamic erosion perturbation. We calculated six additional features shown in Table 18 from the time series data and their first difference (Enríquez Zárate et al., 2022). Two features were based on frequency response using the spectrum of the time series data (Tavares et al., 2022). For a signal $x(t)$ with power spectral density $S_{xx}(f)$, the band power over a frequency interval $[f_1, f_2]$ is

$$P_{[f_1,f_2]} = \int_{f_1}^{f_2} S_{xx}(f)\,\mathrm{d}f.$$

In this study, we selected frequency bands using the range of rotor speeds observed in the data to capture energy near rotor-related frequencies. For the three-bladed turbine, both the once-per-revolution band (1P) and blade-passing band (3P) were considered.

We repeated the predictor selection process from Sect. 4.2 four times, ranking and selecting the top 14 quantities from an initial list of 108: $\gamma - a_{x,tip}$, $\mu - M_{x,root}$, $\mu - F_{y,root}$, $\rho_1 - a_{z,tip}$, $BP_{3P} - M_{z,root}$, $NFD - a_{z,tip}$, $\rho_1 - M_{y,tower}$, $\rho_1 - M_{z,shaft}$, $\kappa - M_{z,shaft}$, $\rho_1 - M_{y,shaft}$, $BP_{3P} - M_{y,tower}$, $\rho_1 - M_{x,tower}$, $\kappa - M_{z,root}$, and $\kappa - a_{x,tip}$. The two highest-ranked predictors, $\gamma - a_{x,tip}$ and $\mu - M_{x,root}$, were also ranked highly in the primary study, suggesting their importance even without the aerodynamic sensor channels.

We train the emulator on a separate dataset with 210 samples and then use 10 repeated 5-fold cross-validations and inner-loop hyperparameter tuning to compare the simulator-trained and emulator-trained classifiers with dataset 5 from Table 5. In this experiment, we used the emulator to generate a dataset of 50 000 samples.

In Fig. 12 we show that, after removing the aerodynamic lift and drag channels, classifier performance decreases substantially relative to the full-sensing case shown in Fig. 8,

Table 16. Class-wise recall comparison between simulator-trained and emulator-trained classifiers under noisy sensing. Average recall values are reported along with paired differences, computed as emulator-trained minus simulator-trained. Differences are reported as points, computed as 100× the raw metric difference.

Erosion class	Emulator-trained recall	Simulator-trained recall	Diff. [95 % CI]	p value
$\alpha = 0.00$	0.91	0.97	$-6.5[-8.8, -4.2]$	7.87×10^{-7}
$\alpha = 0.25$	0.78	0.84	$-6.1[-8.6, -3.5]$	1.63×10^{-5}
$\alpha = 0.50$	0.75	0.69	$5.6[2.6, 8.5]$	3.74×10^{-4}
$\alpha = 0.75$	0.57	0.51	$5.9[3.1, 8.8]$	1.18×10^{-4}
$\alpha = 1.00$	0.78	0.78	$-0.6\ [-3.4, 2.2]$	6.78×10^{-1}

Table 17. Additional OpenFAST output channels used in the reduced-sensing experiment. Note that we relabel `RootMxb1`:M_{root} from Table 4 as $M_{x,\text{root}}$.

OpenFAST variable	Description	Symbol	Unit
`RootFyb1`	Blade root edgewise shear force	$F_{y,\text{root}}$	kN
`RootFzc1`	Blade root axial force	$F_{z,\text{root}}$	kN
`RootMzc1`	Blade root pitching moment	$M_{z,\text{root}}$	kN m
`TwrBsMxt`	Tower-base side-to-side moment	$M_{x,\text{tower}}$	kN m
`TwrBsMyt`	Tower-base fore–aft moment	$M_{y,\text{tower}}$	kN m
`LSSTipMys`	Low-speed shaft tip bending moment	$M_{y,\text{shaft}}$	kN m
`LSSTipMzs`	Low-speed shaft tip bending moment	$M_{z,\text{shaft}}$	kN m
`TipALzb1`	Blade tip acceleration	$a_{z,\text{tip}}$	m s^{-2}
`RtSpeed`	Rotor speed	Ω_{rotor}	rpm

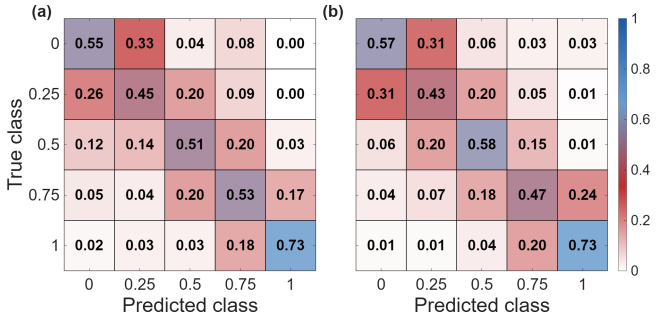

Figure 12. Confusion matrices for classification using non-aerodynamic sensor channels with emulator-trained classifier **(a)** and simulator-trained classifier **(b)**.

confirming that with less access to aerodynamics-based sensors the classification problem is more difficult. However, the simulator-trained and emulator-trained classifiers have comparable class-wise behavior. Nonetheless, the confusion matrices are dominated by their diagonal entries, indicating that misclassifications typically occur between adjacent erosion classes rather than between widely separated damage states. The severe erosion class remains identifiable, with both classifiers correctly classifying 73 % of $\alpha = 1.00$ samples. Consistent with the confusion matrices, in Table 19 we observe nearly identical balanced accuracy and macro-F1 values for the two classifiers, indicating that the emulator-trained classifier remains statistically comparable to the simulator-trained classifier in the case of no direct aerodynamic sensors.

5 Discussion

In this paper we provide a proof-of-concept study for the use of wind turbine emulators to generate data for SHM. Even though turbulent inflow makes distinguishing between different levels of LEE more difficult, we believe it is possible to extend the proposed modeling framework to more realistic operating conditions, including turbulent flow, controller transitions, and yaw adjustments. The OpenFAST turbulence model is parametrized by moments such as average wind speed and the mean and covariance of turbulence intensity, which could be used as inputs to an emulator. More complex operating conditions would require much longer OpenFAST simulations (on the order of tens of minutes rather than seconds) to extract reliable statistical features from time series information. In steady-state simulations, we achieved good results with a comparatively small dataset. However, in order to fit a GP on complex simulations, more simulations will likely be required to achieve high emulator accuracy, which may require different emulator fitting strategies (Clark and Clark, 2022). However, in the case of long simulation runs, the speedup from using an emulator would be even greater than in the current study.

The main motivation for development of the emulator is to reduce the computational cost of running a digital twin for

Table 18. Additional time series descriptors used in the reduced-sensing experiment.

Feature	Symbol	Definition
RMS	$\mathrm{RMS}(x)$	$\sqrt{n^{-1}\sum_i x_i^2}$
Normalized first difference	$\mathrm{NFD}(x)$	$\mathrm{RMS}(\Delta x)/\mathrm{RMS}(x)$
1P bandpower	$\mathrm{BP}_{1\mathrm{P}}(x)$	Spectral power in $[0.104, 0.222]$ Hz.
3P bandpower	$\mathrm{BP}_{3\mathrm{P}}(x)$	Spectral power in $3 \times [0.104, 0.222]$ Hz.
Lag-1 difference	$\mathrm{L1D}(x)$	$(n-1)^{-1}\sum_i \lvert x_{i+1} - x_i \rvert$
Lag-1 autocorrelation	$\rho_1(x)$	$\mathrm{corr}(x_1, \ldots, x_{n-1}; x_2, \ldots, x_n)$

Table 19. Comparison of simulator-trained and emulator-trained classifiers under the reduced-sensing experiment. Values are reported as mean classifier score with standard deviation. Paired differences are computed as emulator-trained minus simulator-trained and are reported with 95 % confidence intervals. Differences are reported as points, computed as 100× the raw metric difference.

Metric	Simulator-trained	Emulator-trained	Diff. [95 % CI]	p value
Balanced accuracy	0.56 ± 0.05	0.56 ± 0.04	$-0.2[-1.9, 1.6]$	0.85
Macro-F1	0.55 ± 0.05	0.56 ± 0.04	$0.2[-1.6, 2.0]$	0.86

monitoring LEE. We envisage that the PPzGP emulator in this paper would serve as the kernel of a probabilistic, graph-based digital twin for tracking damage progression over time (Kapteyn et al., 2021). Areas in which this study could be extended include accounting for how real turbine monitoring data are affected by sensor noise, controller behavior, atmospheric variability, and maintenance history. Further, in this work we do not track erosion progression over time. Closing the gap would require additional development of the surrogate model to include effects like turbulence and validation of an improved surrogate model by comparison with field data (SCADA data and blade inspections) (Duthé et al., 2021; Visbech et al., 2023). Secondly, our erosion proxy does not capture the nonlinear effects on aerodynamic lift and drag polar as a function of erosion damage. Thus, while the emulator fitting results are promising, they should be interpreted within the context of our heuristic erosion model. Future work should incorporate more realistic nonlinear effects on erosion dynamics.

6 Conclusions

This study compares LEE classification using both a full physical simulator-trained random forest classifier and one trained by sampling a computationally efficient emulator. We show that these two damage classifiers provide similar classification results, as evaluated through a variety of ML metrics at predicting levels of LEE damage on WT blades in three proof-of-concept studies. The main contribution of this work is the use of the PPzGP emulator as a computationally efficient surrogate model for enabling damage classification tasks. The PPzGP can be trained on a relatively small set of full physical simulations. It can be adapted to handle arbitrarily large numbers of output QoIs without increasing the overall training cost, and the zero-censored emulator incorporates physical turbine constraints. Taken together, the advantages of this emulation-based framework lead to a reduction in computational cost and could be integrated into real-time SHM systems, such as those that comprise a digital twin.

Code and data availability. Code used to run the experiments and the data used for model training and testing are available on GitHub and at Zenodo (https://doi.org/10.5281/zenodo.16729170, https://zenodo.org/records/21877368; Gettemy, 2026).

Author contributions. Conceptualization: AG, SM, and JZ. Simulation studies and investigation: AG. Methodology: AG, SM, and JZ. Writing (original draft preparation): AG, SM, JZ. Writing (review and editing): AG, SM, JZ, ES.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. The authors bear the ultimate responsibility for providing appropriate place names. Views expressed in the text are those of the authors and do not necessarily reflect the views of the publisher.

Acknowledgements. We are grateful to the anonymous reviewers for their suggestions to improve the paper. Susan Minkoff would like to acknowledge support from the sponsors of the UT Dallas

3D + 4D Seismic FWI Research Consortium. We thank Todd Griffith for the initial suggestion for the project and guidance on wind turbine engineering. We are grateful to Ipsita Mishra for sharing her expertise with running OpenFAST as well as useful discussions on wind turbine modeling in general.

Financial support. This research has been supported by the National Science Foundation (grant no. 2401945). Brookhaven National Laboratory is supported by the US Department of Energy's Office of Science under contract no. DE-SC0012704. Aidan Gettemy was supported on this project by the National Science Foundation (grant no. 2401945).

Review statement. This paper was edited by Julie Teuwen and reviewed by two anonymous referees.

References

Abbas, N. J., Zalkind, D. S., Pao, L., and Wright, A.: A reference open-source controller for fixed and floating offshore wind turbines, Wind Energ. Sci., 7, 53–73, https://doi.org/10.5194/wes-7-53-2022, 2022.

Abdallah, I., Natarajan, A., and Sørensen, J. D.: Impact of uncertainty in airfoil characteristics on wind turbine extreme loads, Renew. Energ., 75, 283–300, https://doi.org/10.1016/j.renene.2014.10.009, 2015.

Abdallah, I., Lataniotis, C., and Sudret, B.: Parametric hierarchical kriging for multi-fidelity aero-servo-elastic simulators – Application to extreme loads on wind turbines, Probabilist. Eng. Mech., 55, 67–77, 2019.

Abdallah, I., Duthé, G., Barber, S., and Chatzi, E.: Identifying evolving leading edge erosion by tracking clusters of lift coefficients, J. Phys. Conf. Ser., 2265, 032089, https://doi.org/10.1088/1742-6596/2265/3/032089, 2022.

Antoniou, A., Dyer, K., Finnegan, W., Herring, R., Holst, B., Bech, J. I., Katsivalis, I., Kutlualp, T., Teuwen, J. J., et al.: Multilayer leading edge protection systems of wind turbine blades: a review of material technology and damage modelling, in: 20th European Conference on Composite Materials: Composites Meet Sustainability, EPFL Lausanne, Composite Construction Laboratory, Lausanne, Switzerland, 97–104, https://doi.org/10.5075/epfl-298799_978-2-9701614-0-0, 2022.

Avendano-Valencia, L. D., Chatzi, E. N., and Tcherniak, D.: Gaussian process models for mitigation of operational variability in the structural health monitoring of wind turbines, Mech. Syst. Signal Pr., 142, 106686, https://doi.org/10.1016/j.ymssp.2020.106686, 2020.

Avendaño-Valencia, L. D., Abdallah, I., and Chatzi, E.: Virtual fatigue diagnostics of wake-affected wind turbine via Gaussian Process Regression, Renew. Energ., 170, 539–561, 2021.

Barber, S., Deparday, J., Marykovskiy, Y., Chatzi, E., Abdallah, I., Duthé, G., Magno, M., Polonelli, T., Fischer, R., and Müller, H.: Development of a wireless, non-intrusive, MEMS-based pressure and acoustic measurement system for large-scale operating wind turbine blades, Wind Energ. Sci., 7, 1383–1398, https://doi.org/10.5194/wes-7-1383-2022, 2022.

Barlas, T. K. and van Kuik, G. A.: Review of state of the art in smart rotor control research for wind turbines, Prog. Aerosp. Sci., 46, 1–27, https://doi.org/10.1016/j.paerosci.2009.08.002, 2010.

Bayarri, M. J., Berger, J. O., Calder, E. S., Patra, A. K., Pitman, E. B., Spiller, E. T., and Wolpert, R. L.: A Methodology for Quantifying Volcanic Hazards, International Journal of Uncertainty Quantification, 5, 297–325, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2015011451, 2015.

Bech, J. I., Hasager, C. B., and Bak, C.: Extending the life of wind turbine blade leading edges by reducing the tip speed during extreme precipitation events, Wind Energ. Sci., 3, 729–748, https://doi.org/10.5194/wes-3-729-2018, 2018.

Belgiu, M. and Drăguţ, L.: Random forest in remote sensing: A review of applications and future directions, ISPRS J. Photogramm., 114, 24–31, https://doi.org/10.1016/j.isprsjprs.2016.01.011, 2016.

Berger, J. O. and Smith, L. A.: On the statistical formalism of uncertainty quantification, Annu. Rev. Stat. Appl., 6, 433–460, https://doi.org/10.1146/annurev-statistics-030718-105232, 2019.

Bilgili, M. and Alphan, H.: Global growth in offshore wind turbine technology, Clean Technol. Envir., 24, 2215–2227, https://doi.org/10.1007/s10098-022-02314-0, 2022.

Bošnjaković, M., Katinić, M., Santa, R., and Marić, D.: Wind turbine technology trends, Appl. Sci., 12, 8653, https://doi.org/10.3390/app12178653, 2022.

Boulesteix, A.-L., Janitza, S., Kruppa, J., and König, I. R.: Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics, WIRES Data Min. Knowl., 2, 493–507, https://doi.org/10.1002/widm.1072, 2012.

Breiman, L.: Random forests, Mach. Learn., 45, 5–32, https://doi.org/10.1023/A:1010933404324, 2001.

Buschjäger, S. and Morik, K.: Decision tree and random forest implementations for fast filtering of sensor data, IEEE T. Circuits-I., 65, 209–222, https://doi.org/10.1109/TCSI.2017.2710627, 2017.

Campobasso, M. S., Cavazzini, A., and Minisci, E.: Rapid estimate of wind turbine energy loss due to blade leading edge delamination using artificial neural networks, J. Turbomach., 142, 071002, https://doi.org/10.1115/1.4047186, 2020.

Campobasso, M. S., Castorrini, A., Ortolani, A., and Minisci, E.: Probabilistic analysis of wind turbine performance degradation due to blade erosion accounting for uncertainty of damage geometry, Renew. Sust. Energ. Rev., 178, 113254, https://doi.org/10.1016/j.rser.2023.113254, 2023.

Campolongo, F., Cariboni, J., and Saltelli, A.: An effective screening design for sensitivity analysis of large models, Environ. Modell. Softw., 22, 1509–1518, https://doi.org/10.1016/j.envsoft.2006.10.004, 2007.

Carmona-Troyo, J. A., Trujillo, L., Enríquez-Zárate, J., Hernandez, D. E., and Cárdenas-Florido, L. A.: Classification of Damage on Wind Turbine Blades Using Automatic Machine Learning and Pressure Coefficient, Expert Syst., 42, e70024, https://doi.org/10.1111/exsy.70024, 2025.

Carraro, M., De Vanna, F., Zweiri, F., Benini, E., Heidari, A., and Hadavinia, H.: CFD modeling of wind turbine blades with eroded leading edge, Fluids, 7, 302, https://doi.org/10.3390/fluids7090302, 2022.

Chen, H., Liu, H., Chu, X., Liu, Q., and Xue, D.: Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network, Renew. Energ., 172, 829–840, https://doi.org/10.1016/j.renene.2021.03.078, 2021.

Choe, D.-E., Kim, H.-C., and Kim, M.-H.: Sequence-based modeling of deep learning with LSTM and GRU networks for structural damage detection of floating offshore wind turbine blades, Renew. Energ., 174, 218–235, https://doi.org/10.1016/j.renene.2021.04.025, 2021.

Civera, M. and Surace, C.: Non-destructive techniques for the condition and structural health monitoring of wind turbines: A literature review of the last 20 years, Sensors, 22, 1627, https://doi.org/10.3390/s22041627, 2022.

Clark, A. C. and Clark, C. E.: Employing Bayesian Quadrature to Improve Fitting of Surrogate Models to Wind Turbine Loads, J. Phys. Conf. Ser., 2265, 042045, https://doi.org/10.1088/1742-6596/2265/4/042045, 2022.

Cropp, R. A. and Braddock, R. D.: The new Morris method: an efficient second-order screening method, Reliab. Eng. Syst. Safe., 78, 77–83, https://doi.org/10.1016/S0951-8320(02)00109-6, 2002.

Currin, C.: A Bayesian approach to the design and analysis of computer experiments, Tech. rep., Oak Ridge National Lab.(ORNL), Oak Ridge, TN, United States, https://doi.org/10.2172/814584, 1988.

Díaz-Uriarte, R. and Alvarez de Andrés, S.: Gene selection and classification of microarray data using random forest, BMC Bioinformatics, 7, 1–13, https://doi.org/10.1186/1471-2105-7-3, 2006.

Dolski, T., Spiller, E. T., and Minkoff, S. E.: Gaussian process emulation for high-dimensional coupled systems, Technometrics, 1–15, https://doi.org/10.1080/00401706.2024.2322651, 2024.

Du, Y., Zhou, S., Jing, X., Peng, Y., Wu, H., and Kwok, N.: Damage detection techniques for wind turbine blades: a review, Mech. Syst. Signal Pr., 141, 106445, https://doi.org/10.1016/j.ymssp.2019.106445, 2020.

Duthé, G., Abdallah, I., Barber, S., and Chatzi, E.: Modeling and monitoring erosion of the leading edge of wind turbine blades, Energies, 14, 7262, https://doi.org/10.3390/en14217262, 2021.

Enríquez Zárate, J., Gómez López, M. d. l. Á., Carmona Troyo, J. A., and Trujillo, L.: Analysis and detection of erosion in wind turbine blades, Math. Comput. Appl., 27, 5, https://doi.org/10.3390/mca27010005, 2022.

Esmaily, H., Tayefi, M., Doosti, H., Ghayour-Mobarhan, M., Nezami, H., and Amirabadizadeh, A.: A comparison between decision tree and random forest in determining the risk factors associated with type 2 diabetes, Journal of Research in Health Sciences, 18, 412, https://pmc.ncbi.nlm.nih.gov/articles/PMC7204421/ (last access: 28 November 2025), 2018.

Fezai, R., Dhibi, K., Mansouri, M., Trabelsi, M., Hajji, M., Bouzrara, K., Nounou, H., and Nounou, M.: Effective random forest-based fault detection and diagnosis for wind energy conversion systems, IEEE Sens. J., 21, 6914–6921, https://doi.org/10.1109/JSEN.2020.3037237, 2020.

Gaudern, N.: A practical study of the aerodynamic impact of wind turbine blade leading edge erosion, J. Phys. Conf. Ser., 524, 012031, https://doi.org/10.1088/1742-6596/524/1/012031, 2014.

Gettemy, A.: Classification of Leading Edge Erosion Severity Via Machine Learning Surrogate Models, Zenodo [data set] and [code], https://doi.org/10.5281/zenodo.21877368, 2026.

Golparvar, B., Papadopoulos, P., Ezzat, A. A., and Wang, R.-Q.: A surrogate-model-based approach for estimating the first and second-order moments of offshore wind power, Appl. Energ., 299, 117286, https://doi.org/10.1016/j.apenergy.2021.117286, 2021.

Gori, F., Laizet, S., and Wynn, A.: Wind farm power maximisation via wake steering: a gaussian process-based yaw-dependent parameter tuning approach, Wind Energy, 27, 1545–1562, https://doi.org/10.1002/we.2953, 2024.

Gu, M. and Berger, J. O.: Parallel partial Gaussian process emulation for computer models with massive output, Ann. Appl. Stat., 1317–1347, https://doi.org/10.1214/16-AOAS934, 2016.

Gu, M., Wang, X., and Berger, J. O.: Robust Gaussian stochastic process emulation, Ann. Stat., 46, 3038–3066, https://doi.org/10.1214/17-AOS1648, 2018.

Gu, M., Palomo, J., and Berger, J. O.: RobustGaSP: Robust Gaussian Stochastic Process Emulation in R, R J., 11, 112–136, https://doi.org/10.32614/RJ-2019-011, 2019.

Haghi, R., Stagg, C., and Crawford, C.: Wind Turbine damage equivalent load assessment using Gaussian process regression combining measurement and synthetic data, Energies, 17, 346, https://doi.org/10.3390/en17020346, 2024.

Han, W., Kim, J., and Kim, B.: Effects of contamination and erosion at the leading edge of blade tip airfoils on the annual energy production of wind turbines, Renew. Energ., 115, 817–823, https://doi.org/10.1016/j.renene.2017.09.002, 2018.

Hassan, Q., Viktor, P., Al-Musawi, T. J., Ali, B. M., Algburi, S., Alzoubi, H. M., Al-Jiboory, A. K., Sameen, A. Z., Salman, H. M., and Jaszczur, M.: The renewable energy role in the global energy transformations, Renew. Energ. Focus, 48, 100545, https://doi.org/10.1016/j.ref.2024.100545, 2024.

Haus, L. C.: New methods for digital twin modelling of wave and wind energy systems, Master's thesis, University of Texas at Dallas, https://hdl.handle.net/10735.1/9016 (last access: 12 September 2025), 2020.

Helton, J. C. and Davis, F. J.: Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems, Reliab. Eng. Syst. Safe., 81, 23–69, https://doi.org/10.1016/S0951-8320(03)00058-9, 2003.

Herman, J. and Usher, W.: SALib: An open-source python library for sensitivity analysis, The Journal of Open Source Software, 2, 97, https://doi.org/10.21105/joss.00097, 2017.

Herring, R., Dyer, K., Martin, F., and Ward, C.: The increasing importance of leading edge erosion and a review of existing protection solutions, Renew. Sust. Energ. Rev., 115, 109382, https://doi.org/10.1016/j.rser.2019.109382, 2019.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M.: Computer model calibration using high-dimensional output, J. Am. Stat. Assoc., 103, 570–583, https://doi.org/10.1198/016214507000000888, 2008.

Hulsman, P., Sucameli, C., Petrović, V., Rott, A., Gerds, A., and Kühn, M.: Turbine power loss during yaw-misaligned free field tests at different atmospheric conditions, J. Phys. Conf. Ser., 2265, 032074, https://doi.org/10.1088/1742-6596/2265/3/032074, 2022.

Iwanaga, T., Usher, W., and Herman, J.: Toward SALib 2.0: advancing the accessibility and interpretability of global sensitivity analyses, Socio-Environmental Systems Modelling, 4, 18155, https://doi.org/10.18174/sesmo.18155, 2022.

Jiménez, A. A., Muñoz, C. Q. G., and Márquez, F. P. G.: Machine learning for wind turbine blades maintenance management, Energies, 11, 1–16, https://doi.org/10.3390/en11010013, 2017.

Jonkman, B., Platt, A., Mudafort, R. M., Branlard, E., Sprague, M., Ross, H., jjonkman, HaymanConsulting, Slaughter, D., Hall, M., Vijayakumar, G., Buhl, M., Russell9798, Bortolotti, P., reos rcrozier, Ananthan, S., RyanDavies19, S., M., Rood, J., rdamiani, nrmendoza, sinolonghai, pschuenemann, ashesh2512, kshaler, Housner, S., psakievich, Wang, L., Bendl, K., and Carmo, L.: OpenFAST/openfast: v3.5.3, Zenodo [code], https://doi.org/10.5281/zenodo.6324287, 2024.

Jonkman, J.: The new modularization framework for the FAST wind turbine CAE tool, in: 51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, Grapevine (Dallas/Ft. Worth Region), Texas, 7–10 January 2013, 202, https://doi.org/10.2514/6.2013-202, 2013.

Jonkman, J., Butterfield, S., Musial, W., and Scott, G.: Definition of a 5-MW reference wind turbine for offshore system development, Tech. rep., National Renewable Energy Lab. (NREL), Golden, CO (United States), https://doi.org/10.2172/947422, 2009.

Jonkman, J. M., Hayman, G. J., Jonkman, B. J., Damiani, R. R., and Murray, R. E.: AeroDyn v15 user's guide and theory manual, NREL Draft Report, 46, https://www.nlr.gov/docs/libraries/wind-docs/aerodyn-manual.pdf?sfvrsn=51d961db_1 (last access: 16 June 2026), 2015.

Kaewniam, P., Cao, M., Alkayem, N. F., Li, D., and Manoach, E.: Recent advances in damage detection of wind turbine blades: a state-of-the-art review, Renew. Sust. Energ. Rev., 167, 112723, https://doi.org/10.1016/j.rser.2022.112723, 2022.

Kapteyn, M. G., Pretorius, J. V., and Willcox, K. E.: A probabilistic graphical model foundation for enabling predictive digital twins at scale, Nat. Computational Science, 1, 337–347, https://doi.org/10.1038/s43588-021-00069-0, 2021.

Langel, C. M., Chow, R. C., Van Dam, C., and Maniaci, D. C.: RANS based methodology for predicting the influence of leading edge erosion on airfoil performance, Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), https://doi.org/10.2172/1404827, 2017.

Law, H. and Koutsos, V.: Leading edge erosion of wind turbines: Effect of solid airborne particles and rain on operational wind farms, Wind Energy, 23, 1955–1965, https://doi.org/10.1002/we.2540, 2020.

Leishman, G., Nash, D., Yang, L., and Dyer, K.: A novel approach for wind turbine blade erosion characterization: an investigation using surface gloss measurement, Coatings, 12, 928, https://doi.org/10.3390/coatings12070928, 2022.

Liang, Y., Ji, X., Wu, C., He, J., and Qin, Z.: Estimation of the influences of air density on wind energy assessment: a case study from China, Energ. Convers. Manage., 224, 113371, https://doi.org/10.1016/j.enconman.2020.113371, 2020.

Liu, H., Chen, G., Hua, Z., Zhang, J., and Wang, Q.: Wind shear model considering atmospheric stability to improve accuracy of wind resource assessment, Processes, 12, 954, https://doi.org/10.3390/pr12050954, 2024.

López, J. C., Kolios, A., Wang, L., and Chiachio, M.: A wind turbine blade leading edge rain erosion computational framework, Renew. Energ., 203, 131–141, https://doi.org/10.1016/j.renene.2022.12.050, 2023.

Macdonald, H., Infield, D., Nash, D. H., and Stack, M. M.: Mapping hail meteorological observations for prediction of erosion in wind turbines, Wind Energy, 19, 777–784, https://doi.org/10.1002/we.1854, 2016.

Maldonado-Correa, J., Martín-Martínez, S., Artigao, E., and Gómez-Lázaro, E.: Using SCADA data for wind turbine condition monitoring: a systematic literature review, Energies, 13, 3132, https://doi.org/10.3390/en13123132, 2020.

Maniaci, D. C., MacDonald, H., Paquette, J., and Clarke, R. J.: Leading Edge Erosion Classification System, Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), https://doi.org/10.2172/2432094, 2022.

Mishnaevsky, L.: Root causes and mechanisms of failure of wind turbine blades: overview, Materials, 15, 2959, https://doi.org/10.3390/ma15092959, 2022.

Morató, A., Sriramula, S., and Krishnan, N.: Kriging models for aero-elastic simulations and reliability analysis of offshore wind turbine support structures, Ships Offshore Struc., 14, 545–558, https://doi.org/10.1080/17445302.2018.1522738, 2019.

Morris, M. D.: Factorial sampling plans for preliminary computational experiments, Technometrics, 33, 161–174, https://doi.org/10.1080/00401706.1991.10484804, 1991.

Moynihan, B., Moaveni, B., Liberatore, S., and Hines, E.: Estimation of blade forces in wind turbines using blade root strain measurements with OpenFAST verification, Renew. Energ., 184, 662–676, https://doi.org/10.1016/j.renene.2021.11.094, 2022.

Murcia, J. P., Réthoré, P.-E., Dimitrov, N., Natarajan, A., Sørensen, J. D., Graf, P., and Kim, T.: Uncertainty propagation through an aeroelastic wind turbine model using polynomial surrogates, Renew. Energ., 119, 910–922, https://doi.org/10.1016/j.renene.2017.07.070, 2018.

Myhr, A., Bjerkseter, C., Ågotnes, A., and Nygaard, T. A.: Levelised cost of energy for offshore floating wind turbines in a life cycle perspective, Renew. Energ., 66, 714–728, https://doi.org/10.1016/j.renene.2014.01.017, 2014.

Myren, S. and Lawrence, E.: A comparison of Gaussian processes and neural networks for computer model emulation and calibration, Stat. Anal. Data Min., 14, 606–623, https://doi.org/10.1002/sam.11507, 2021.

Ning, S. A.: A simple solution method for the blade element momentum equations with guaranteed convergence, Wind Energy, 17, 1327–1345, https://doi.org/10.1002/we.1636, 2014.

Pandit, R., Astolfi, D., Hong, J., Infield, D., and Santos, M.: SCADA data for wind turbine data-driven condition/performance monitoring: a review on state-of-art, challenges and future trends, Wind Engineering, 47, 422–441, https://doi.org/10.1177/0309524X221124031, 2023.

Panthi, K. and Iungo, G. V.: Quantification of wind turbine energy loss due to leading-edge erosion through infrared-camera imaging, numerical simulations, and assessment against SCADA and meteorological data, Wind Energy, 26, 266–282, https://doi.org/10.1002/we.2798, 2023.

Papi, F., Cappugi, L., Perez-Becker, S., and Bianchini, A.: Numerical modeling of the effects of leading-edge erosion and trailing-edge damage on wind turbine loads and performance, J. Eng. Gas Turb. Power, 142, 111005, https://doi.org/10.1115/1.4048451, 2020.

Park, J., Kim, C., Dinh, M.-C., and Park, M.: Design of a condition monitoring system for wind turbines, Energies, 15, 464, https://doi.org/10.3390/en15020464, 2022.

Platt, A., Jonkman, B., and Jonkman, J.: InflowWind user's guide, Technical report, National Renewable Energy Laboratory, https://www.nlr.gov/docs/libraries/wind-docs/aerodyn-manual.pdf?sfvrsn=51d961db_1 (last access: 16 June 2026), 2016.

Pryor, S. C., Barthelmie, R. J., Cadence, J., Dellwik, E., Hasager, C. B., Kral, S. T., Reuder, J., Rodgers, M., and Veraart, M.: Atmospheric drivers of wind turbine blade leading edge erosion: review and recommendations for future research, Energies, 15, 8553, https://doi.org/10.3390/en15228553, 2022.

Pugh, K., Nash, J., Reaburn, G., and Stack, M.: On analytical tools for assessing the raindrop erosion of wind turbine blades, Renew. Sust. Energ. Rev., 137, 110 611, https://doi.org/10.1016/j.rser.2020.110611, 2021.

Rasmussen, C. E.: Gaussian processes in machine learning, in: Summer school on machine learning, Springer, 63–71, https://doi.org/10.1007/978-3-540-28650-9_4, 2003.

Ren, Z., Verma, A. S., Li, Y., Teuwen, J. J., and Jiang, Z.: Offshore wind turbine operations and maintenance: a state-of-the-art review, Renew. Sust. Energ. Rev., 144, 110886, https://doi.org/10.1016/j.rser.2021.110886, 2021.

Rinker, J., Gaertner, E., Zahle, F., Skrzypiński, W., Abbas, N., Bredmose, H., Barter, G., and Dykes, K.: Comparison of loads from HAWC2 and OpenFAST for the IEA wind 15 mw reference wind turbine, J. Phys. Conf. Ser., 1618, 052052, https://doi.org/10.1088/1742-6596/1618/5/052052, 2020.

Rinker, J. M.: Calculating the sensitivity of wind turbine loads to wind inputs using response surfaces, J. Phys. Conf. Ser., 753, 032057, https://doi.org/10.1088/1742-6596/753/3/032057, 2016.

Rogers, T., Gardner, P., Dervilis, N., Worden, K., Maguire, A., Papatheou, E., and Cross, E.: Probabilistic modelling of wind turbine power curves with application of heteroscedastic Gaussian process regression, Renew. Energ., 148, 1124–1136, 2020.

Sacks, J., Schiller, S. B., and Welch, W. J.: Designs for computer experiments, Technometrics, 31, 41–47, https://doi.org/10.1080/00401706.1989.10488474, 1989.

Santner, T. J., Williams, B. J., Notz, W. I., and Williams, B. J.: The design and analysis of computer experiments, vol. 1 of Springer Series in Statistics, Springer, 2 edn., https://doi.org/10.1007/978-1-4939-8847-1, 2003.

Sareen, A., Sapre, C. A., and Selig, M. S.: Effects of leading edge erosion on wind turbine blade performance, Wind Energy, 17, 1531–1542, https://doi.org/10.1002/we.1649, 2014.

Schramm, M., Rahimi, H., Stoevesandt, B., and Tangager, K.: The influence of eroded blades on wind turbine performance using numerical simulations, Energies, 10, 1420, https://doi.org/10.3390/en10091420, 2017.

Seidman, J.: SideofMan/zGP: zGP in R v1.0.0, Zenodo [code], https://doi.org/10.5281/zenodo.17956672, 2025.

Shankar Verma, A., Jiang, Z., Ren, Z., Caboni, M., Verhoef, H., van der Mijle-Meijer, H., Castro, S. G., and Teuwen, J. J.: A probabilistic long-term framework for site-specific erosion analysis of wind turbine blades: A case study of 31 Dutch sites, Wind Energy, 24, 1315–1336, https://doi.org/10.1002/we.2634, 2021a.

Shankar Verma, A., Jiang, Z., Ren, Z., Hu, W., and Teuwen, J. J.: Effects of onshore and offshore environmental parameters on the leading edge erosion of wind turbine blades: a comparative study, J. Offshore Mech. Arct., 143, 042001, https://doi.org/10.1115/1.4049248, 2021b.

Sheibat-Othman, N., Othman, S., Tayari, R., Sakly, A., Odgaard, P. F., and Larsen, L. F.: Estimation of the wind turbine yaw error by support vector machines, IFAC-PapersOnLine, 48, 339–344, https://doi.org/10.1016/j.ifacol.2015.12.401, 2015.

Shihavuddin, A., Chen, X., Fedorov, V., Nymark Christensen, A., Andre Brogaard Riis, N., Branner, K., Bjorholm Dahl, A., and Reinhold Paulsen, R.: Wind turbine surface damage detection by deep learning aided drone inspection analysis, Energies, 12, 676, https://doi.org/10.3390/en12040676, 2019.

Singh, D., Dwight, R., and Viré, A.: Probabilistic surrogate modeling of damage equivalent loads on onshore and offshore wind turbines using mixture density networks, Wind Energ. Sci., 9, 1885–1904, https://doi.org/10.5194/wes-9-1885-2024, 2024.

Smith, R. C.: Uncertainty quantification: theory, implementation, and applications, SIAM, https://doi.org/10.1137/1.9781611977844, 2024.

Spiller, E. T., Wolpert, R. L., Tierz, P., and Asher, T. G.: The Zero Problem: Gaussian Process Emulators for Range-Constrained Computer Models, SIAM/ASA Journal on Uncertainty Quantification, 11, 540–566, https://doi.org/10.1137/21M1467420, 2023.

Stein, M. L.: Interpolation of spatial data: some theory for kriging, Springer Science & Business Media, https://doi.org/10.1007/978-1-4612-1494-6, 1999.

Stetco, A., Dinmohammadi, F., Zhao, X., Robu, V., Flynn, D., Barnes, M., Keane, J., and Nenadic, G.: Machine learning methods for wind turbine condition monitoring: a review, Renew. Energ., 133, 620–635, https://doi.org/10.1016/j.renene.2018.10.047, 2019.

Tavares, A., Lopes, B., Di Lorenzo, E., Cornelis, B., Peeters, B., Desmet, W., and Gryllias, K.: Machine learning techniques for damage detection in wind turbine blades, in: European Workshop on Structural Health Monitoring, Springer, 176–189, https://doi.org/10.1007/978-3-031-07254-3_18, 2022.

Tchakoua, P., Wamkeue, R., Ouhrouche, M., Slaoui-Hasnaoui, F., Tameghe, T. A., and Ekemb, G.: Wind turbine condition monitoring: State-of-the-art review, new trends, and future challenges, Energies, 7, 2595–2630, https://doi.org/10.3390/en7042595, 2014.

The MathWorks Inc.: Statistics and machine learning toolbox, https://www.mathworks.com/help/stats/index.html (last access: 26 June 2026), 2024.

Veers, P., Dykes, K., Lantz, E., Barth, S., Bottasso, C. L., Carlson, O., Clifton, A., Green, J., Green, P., Holttinen, H., et al.: Grand challenges in the science of wind energy, Science, 366, eaau2027, https://doi.org/10.1126/science.aau2027, 2019.

Velarde, J., Kramhøft, C., and Sørensen, J. D.: Global sensitivity analysis of offshore wind turbine foundation fatigue loads, Renew. Energ., 140, 177–189, https://doi.org/10.1016/j.renene.2019.03.055, 2019.

Verma, A. S., Castro, S. G., Jiang, Z., and Teuwen, J. J.: Numerical investigation of rain droplet impact on offshore wind turbine blades under different rainfall conditions: A parametric study, Compos. Struct., 241, 112096, https://doi.org/10.1016/j.compstruct.2020.112096, 2020.

Visbech, J., Göçmen, T., Hasager, C. B., Shkalov, H., Handberg, M., and Nielsen, K. P.: Introducing a data-driven

approach to predict site-specific leading-edge erosion from mesoscale weather simulations, Wind Energ. Sci., 8, 173–191, https://doi.org/10.5194/wes-8-173-2023, 2023.

Ward, T., Jenab, K., Ortega-Moody, J., and Staub, S.: A comprehensive review of machine learning techniques for condition-based maintenance, International Journal of Prognostics and Health Management, 15, https://doi.org/10.36001/ijphm.2024.v15i2.3850, 2024.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D.: Screening, predicting, and computer experiments, Technometrics, 34, 15–25, https://doi.org/10.2307/1269548, 1992.

Zaher, A., McArthur, S., Infield, D., and Patel, Y.: Online wind turbine fault detection through automated SCADA data analysis, Wind Energy, 12, 574–593, https://doi.org/10.1002/we.319, 2009.

Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., and Si, Y.: A data-driven design for fault detection of wind turbines using random forests and XGboost, Ieee Access, 6, 21020–21031, https://doi.org/10.1109/ACCESS.2018.2818678, 2018.

Zidane, I. F., Saqr, K. M., Swadener, G., Ma, X., and Shehadeh, M. F.: On the role of surface roughness in the aerodynamic performance and energy conversion of horizontal wind turbine blades: a review, Int. J. Energ. Res., 40, 2054–2077, https://doi.org/10.1002/er.3580, 2016.