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
[2]Computing and Data Sciences Directorate, Brookhaven National Laboratory, 2 Center St, Upton, NY 11973
[3]Department of Mathematics, New York Institute of Technology, 1855 Broadway, New York, NY 10023
[4]Mathematical and Statistical Sciences, Marquette University, 1250 W. Wisconsin Ave., Milwaukee, WI 53233

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

Abstract.

~~As the number and size of wind turbines has increased, manual observation and maintenance of the turbines has become increasingly dangerous and time consuming for human operators. One key form of turbine deterioration is leading-edge erosion which degrades the blade laminate over time. This erosion is caused by environmental factors such as blowing sand, rain, and bug accumulation. Blade damage reduces aerodynamic efficiency and shortens the operational lifespan of wind turbines, motivating the need for~~

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 ~~. Ideally one would like to use a digital twin which couples a physical device (the turbine) with a computer model by bidirectional passage of information between the physical and digital twins. In a digital twin, sensor data from the turbine continually updates the computer model which then predicts the state of the system for future maintenance and operation decisions, potentially eliminating the need for frequent manual inspections. Machine learning-based classifiers trained on simulation data accurately detect damage, but require large training data sets, highlighting the need for computationally efficient alternatives to full physics simulation. A Gaussian process~~ 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) ~~surrogate model can be trained from a small set of full simulation datapoints. Once trained, GP's make predictions very fast (1000 times faster than a simulator evaluation) while also providing information about the uncertainty in the emulator prediction relative to the full physical simulator. The GP emulator methodology we employ includes two extensions to~~ 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 ~~. First, the output quantity of interest is vector-valued (rather than a scalar). In our case the vector contains statistics of relevant outputs such as lift, drag, generator power, etc. Second, the range of the outputsare constrained to fit specifications of the blade (so are not defined over the usual full-space domain required of Gaussian distributions). In this~~

~~work we test~~ 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~~developed to quantify levels of leading edge erosion. The classifiers differ in whether they are trained~~ : one trained directly on full simulation data ~~or data from the GP surrogate. We find that the classifier trained on surrogate datais as accurate as the classifier trained on full simulation data . Using the surrogate-generated dataset, the classifier distinguishes between five~~ 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~~with 87% accuracy , surpassing the simulation-trained classifier's accuracy of 83%~~. 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 ~~highlight the promise of using GP surrogates to train classifiers for leading edge erosion, a key component of a digital twin~~ 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 ~~wider international~~ worldwide transition to renewable ~~energy, with the European Union, China, and the United States planning to increase their supply of renewable power . By 2050, wind energy could supply up to a third of global electricity demand . The increase in the number of offshore wind farms will contribute to lowering the cost of wind generated electricity by taking advantage of stronger, steadier winds over the ocean and utilizing larger rotor diameters . However, the wind energy sector faces several challenges to delivering affordable, reliable power such as blade durability, large-scale turbine monitoring, and aging wind turbine infrastructure . Offshore turbines must endure a higher level of environmental damage than turbines on land, and both on and offshore turbines require higher levels of maintenance to address wear-related performance declines as they age. These factors contribute to operational expenses which represent a significant portion of power production costs .~~ 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).

~~A major challenge in wind turbine maintenance is leading edge erosion . Rain, hail, and airborne particles roughen the blade's surface , shifting the boundary between laminar and turbulent airflow and compromising aerodynamic efficiency by reducing lift and increasing drag .~~ Leading edge erosion ~~forms as small pits merge over time into larger gouges, potentially leading to delamination . This erosion of the laminate~~ (LEE) is a driver of O&M costs (Maniaci et al., 2022). LEE creates structural risks that may ~~require~~ necessitate the complete replacement of ~~the blades of older turbines~~ WT blades (Maniaci et al., 2022).

Furthermore, newer rotors have faster erosion rates due to the increased blade tip wind speed inherent in longer blade designs . Offshore turbines face an elevated risk for leading-edge erosion 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). These effects begin to accumulate within months of a blade's installation, with erosion accumulating faster at the tip . Because erosion depends on coating fatigue and environmental conditions, predicting damage levels remains challenging . Approaches to mitigate leading edge erosion include improving blade resistance , mapping environmental exposure to anticipate wear patterns , and developing predictive models to estimate operation and maintenance costs . Since inspections require shutdowns and wind farms are increasingly difficult to access, especially in offshore conditions, continuous monitoringand damage detection are essential for erosion damage management that 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 down time 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).

Detecting leading-edge erosion remains challenging due to manufacturing defects (including defects in the blade coating) and environmental factors such as variation in precipitation intensity and wind speed Structural health monitoring (SHM) for LEE remains challenging (Mishnaevsky, 2022). Conventional detection methods such as visual inspection usually Visual inspection may only detect defects after substantial degradation has occurred (Pugh et al., 2021). Newer detection methods including Improvements in UAV-based blade surveillance (Shihavuddin et al., 2019) and gloss-based surface characterization (Leishman et al., 2022) may enhance the precision of erosion assessment but are not always simple to implement. Alternatively, Supervisory Control and Data Acquisition (SCADA) Data 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) is a readily available source of data , and new sensors , such as rely on available data gathered without the need for workers or instruments to be moved into physical proximity with the blades. Along with sensors for blade load measurements (Abdallah et al., 2015) and aerodynamic surface pressure (Duthé et al., 2021) , allow for continuous erosion monitoring. 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).

In wind energy, machine learning has been used to analyze Machine learning (ML) has proven to be useful for SHM through analyzing SCADA data for turbine fault detection and blade damage assessment . Machine learning has proven effective in enhancing damage identification . Various methods(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 (NN) (Chen et al., 2021; Choe et al., 2021), have successfully detected faults using condition monitoring data. Machine learning has also been applied to problems related to leading edge erosion detection. () used a neural network to predict the loss of annual

energy production from eroded blades. Computational 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) coupled simulators with aero-elastic modeling (OpenFAST) has been used to to simulate erosion patterns and quantify damage using machine learning (Enríquez Zárate et al., 2022). Erosion has also been modeled as a stochastic process, employing , transformer models to track erosion severity in turbulent conditions via aerodynamic pressure coefficients (Duthé et al., 2021). Deep learning methods have been applied to erosion prediction , and deep learning to predict LEE using aerodynamic data . Finally, machine learning models incorporating a variety of damage predictors including pressure, temperature, humidity, and wind speed have shown the advantage of using a variety of input channels for erosion detection in condition monitoring . (Abdallah et al., 2022; Carmona-Troyo et al., 2025).

Machine learning presents Although ML is a promising avenue for detecting leading-edge erosion, yet LEE, its effectiveness is constrained by the scarcity need collect large volumes of high-fidelity training data (Ward et al., 2024). Operational turbine datasets are often noisy due to sensor faults and are not generally made publicly available by manufacturers and operators. To mitigate these limitations, numerical models are commonly used to generate synthetic training data for validating new condition monitoring approaches . Since erosion is governed by stochastic environmental inputs, training robust detection algorithms from physically realistic simulations remains Data generation can impose a computational bottleneck , limiting the scalability of condition monitoring for training and validating these systems. For instance, CFD captures the simulations which fine-scale aerodynamic effects of erosion but requires require computationally expensive numerical solutions to the Navier–Stokes equations, rendering them infeasible at full turbine scales which is not feasible for a large dataset generation (Langel et al., 2017). Reduced-order approaches such as large eddy simulation and Reynolds-averaged Navier–Stokes alleviate some of the computational burden, but even simplified two-dimensional models remain Even reduced-order modeling approaches are prohibitively costly when simulating erosion under diverse atmospheric conditions (Carraro et al., 2022). Aero-elastic While aero-elastic simulations offer a more tractable alternative, producing realistic SCADA and sensor data, but erosion classification remains sensitive to environmental variability. Studies studies demonstrate that the impact of blade damage is modulated 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). Generating comprehensive datasets with 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 modelsfor machine learning introduces additional computational bottlenecks , particularly when these models are coupled with simulations for collision-induced damage from rain or hail . . 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 addresses 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). A promising direction for leading edge erosion detection involves integrating embedded sensors on blades with predictive damage models and control mechanisms within a digital twin system . Digital twins are virtual replicas of physical turbines which integrate real-time sensor data,

physics-based models, and control mechanisms to enable predictive maintenance and automated decision-making . Potential applications include offshore wind turbine behavior modeling, global sensitivity assessments, and predictive maintenance planningSurrogate 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, real time data integration and prediction require models which can be run extremely rapidly. The surrogate model is an integral component of the digital twin framework because it enables the accurate tracking of damage states that would otherwise be too costly to simulate. 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. (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.

In this paper we present a novel application of Gaussian process emulation for generating training data to classify leading edge erosion damage. Our Gaussian process-based surrogate emulates multiple quantities of interestwith high predictive accuracy in significantly reduced time over full model simulation. 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 also is 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 emulator results show classification results show generated data can be used without a significant loss of accuracy (1) low normalized root mean squared error (<10%) and high credible interval coverage (>88%) on multi-modal sensor data ; (2) improved classification accuracy for random forest classification when trained on emulated data compared to simulated data (87% vs. 83%); and (3) 1000+× faster predictions compared to aerodynamic simulations. While the accuracy improvement is modest, the emulator-trained classifiers: 0.88 Macro-F1 vs. OpenFAST trained classifiers: 0.82 Macro-F1). The computational savings are significant. Even accounting for Including the cost of running the simulator to produce the

training data for the emulator, the overall computational time is drastically reducedas the GP does not require a large number of training points. By demonstrating that emulated data provides training quality comparable to full simulations, we establish the feasibility of surrogate models for leading edge erosion severity tracking. Efficient surrogates allow for the exploration of more complex turbine condition monitoring scenarios and new methods for maintenance optimization, valuable as the kernel of a digital twin for wind turbines. . 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 present a workflow for using emulation in wind turbine condition monitoringdescribe 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) for aero-elastic simulations. We further apply the Morris screening method (a to simulate the NREL 5 MW reference turbine under varying environmental conditions and stochastic LEE states. We use global sensitivity analysis tool) to identify key input parameters for surrogate model construction. For surrogate modeling, we use the Gaussian Process (GP) (Morris, 1991) to identify influential simulator inputs. Finally, we train a vector-valued GP emulator (Smith, 2024)) emulator to create an effective multi-modal surrogate for wind turbine simulations. We emphasize two extensions to the GP emulator that improve the emulator's fidelity and are novel in the context of wind energy modeling: with parallel partial emulation (Gu and Berger, 2016) and the range-censored emulation zero-censored treatment (Spiller et al., 2023) (for of range-limited outputs). We use a random forest classification algorithm , generate large emulator-based training datasets, and compare the resulting random forest (Breiman, 2001) for damage detection. Lastly we provide details on the construction of our experimental dataset which represents diverse environmental scenarios and erosion patterns for training and validation. 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 Renewable Energy Laboratory (NRELLaboratory 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, aero-dynamics , structural dynamics, and servo-dynamics (Jonkman, 2013; Rinker et al., 2020).

OpenFAST links computational grids, aerodynamic forces, and control systems, which has facilitated its adoption in a broad set of onshore and offshore wind turbine reference model studies. () couple the `AeroDyn` and `ServoDyn` modules to benchmark new controllers which maximize power for floating offshore wind turbines. Furthermore, OpenFAST can OpenFAST has been applied to supply realistic aero-servo-elastic data for leading edge erosion LEE detection, including the generation of blade-tip accelerometer data (Enríquez Zárate et al., 2022) and lift/drag sensor data (Duthé et al., 2021).

Because of its versatility and well-established capabilities, we use OpenFAST to generate realistic ~~leading edge erosion~~ LEE data, relying on the coupling of four ~~of the 14 main~~ OpenFAST modules, `AeroDyn`, `ElastoDyn`, `ServoDyn`, and ~~`Inflow Wind`~~`InflowWind`. ~~We selected these modules because they~~ 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. ~~For example,~~ In this study we use uniform wind fields include wind shear ~~(where the wind close to the ground has a different velocity than wind at the top of the wind turbine tower) and can simulate sudden changes in wind speed and wind direction.~~ and specify wind direction. ~~The simulation of aerodynamic~~ Aerodynamic forces and blade loads, handled by the module ~~`Aerodyn`~~`AeroDyn`, ~~is~~ are particularly important for ~~leading edge erosion.~~ LEE. ~~`Aerodyn`~~`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 ~~the aerodynamic polar curve~~aerodynamic polar curves. OpenFAST uses a look-up table based on the polar curve to ~~determine the lift and drag coefficient of each blade element as a function of time, enabling the realization of~~ calculate torque and thrust forces on the axial shaft of the wind turbine ~~which then determine the generator power~~ (Ning, 2014; López et al., 2023). `ServoDyn` regulates the electronic systems of the wind turbine using structural motions, loads, and wind measurements ~~for inputs~~ and calculates the power generation ~~. It controls the actuators of the wind turbine model, which pitch the blades, regulate the rotor speed, and rotate the nacelle. In these simulations~~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 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 ~~leading edge erosion~~ 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 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

Table 1. OpenFAST NREL 5MW Reference WT (RWT) upwind 3 blade WT simulation model parameters.

Simulation Parameter	Description
Rotor, Blade length, Hub Diameter, Hub Height	$\{126.0\,\mathrm{m}, 61.5\,\mathrm{m}, 3.0\,\mathrm{m}, 90.0\,\mathrm{m}\}$
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

Table 2. Environmental variable ranges

Input	Range	Units
Wind Direction	[-15, 15]	deg.
Air Density	[1.10, 1.42]	$\mathrm{kg\,m^{-3}}$
Wind Speed	[3, 25]	$\mathrm{m\,s^{-1}}$
Wind Shear Coefficient	[0, 0.5]	(-)

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 environmental parameters. To ensure that the turbine reaches equilibrium with the environmental conditions, we ran the simulations for 180 seconds 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 of 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 6 erosion regions via a grouping of the NREL 5MW RWT, see Table 1, (Jonkman et al., 2009) `AeroDyn` nodes, as summarized in Table 3 and shown in Fig. 1. We let $e_i \in [0,1]$ denote the erosion level of the i-th 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,

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

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 i-th 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 i-th blade region according to $c_{L,i} = 1 - 0.53e_i$ and $c_{D,i} = 1 + 4e_i$.

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

Region	Aerodyn Nodes	Distance from Hub	Region Length	Airfoils Included
1	5-6	10.25 m	8.20 m	Du40, Du35
2	7-8	18.45 m	12.30 m	Du35, Du30
3	9-10	30.75 m	8.20 m	Du25
4	11-12	38.95 m	8.20 m	Du21
5	13-14	47.15 m	7.50 m	NACA64
6	15-19	54.67 m	6.83 m	NACA64

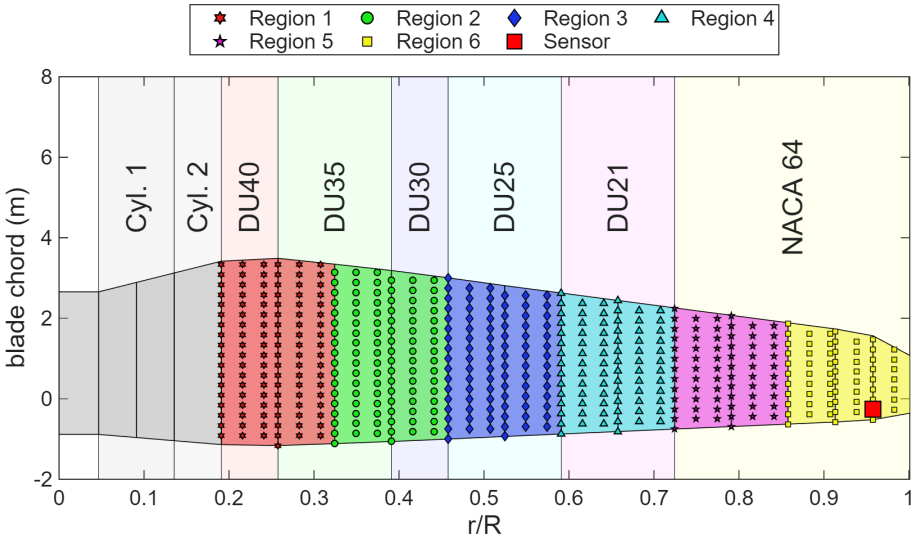

Figure 1. Diagram of erosion regions described in Table 3. The red square shows the location of the aerodynamic node where lift and drag sensor data is measured. The y axis measures r/R, the relative proportion of the total blade length, see Table 3.

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, a nonlinear acceleration phase, which is 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.

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 relies 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 aero-foil 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 this 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.; Enríquez Zárate et al. (2020; 2022), and the edge-wise moment of the root of the blade, used by Moynihan et al.; Sheibat-Othman et al. (2022; 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 Section 4 in Fig. 8 and Fig. 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.

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 the blade 1 region 6 sensor	C_L	(-)
`GenPwr`	Generator power	P_{gen}	MW
`RootMxb1`	Blade-root edgewise moment	M_{root}	kN m

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 effects method (Morris screening), as an efficient GSA method for high dimensional, non-linear and computationally expensive models. Velarde et al. (2019) used elementary effects analysis to determine the relative importance of input parameters for analyzing the foundation loads of offshore wind turbines.

To apply the elementary effects 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, $\mathbf{u}_0^{(j)}$ for $j = 1,...,r$, 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 input dimensions in a random order to create the elementary effects experiment design. For each component index l of $\mathbf{u}$ and each of the r trajectories indexed by j, the method calculates the elementary effect using a finite difference

$$EE_{l,j}(\mathbf{u}) = [f(\mathbf{u} + \mathbf{e}_l) - f(\mathbf{u})]/\Delta. \tag{1}$$

The mean and the standard deviation of the elementary effects, $\{EE_{l,1},...,EE_{l,r}\}$, are calculated for each dimension l. To avoid the possibility of large positive and negative elementary effects canceling out, we use $\mu_l^* = \frac{1}{r}\sum_{j=1}^{r}|EE_{l,j}|$, instead of the mean, μ_l. The standard deviation, which requires the average of elementary effects, μ_l, is given by $\sigma_l = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(EE_{l,j} - \mu_l)^2}$ (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 $l \in \{1,...,p\}$ is used to determine the relative importance of the inputs (Campolongo et al., 2007; Morris, 1991). We use the elementary effects 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 surrogate modeling~~. 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). ~~Gaussian process~~ 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~~ Steady-state simulations in OpenFAST take on the order of minutes on a typical laptop/desktop computer, while turbulent wind and wave simulations in off-shore environments take much longer. ~~GP's~~ 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~~QoI~~. 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, $\{\boldsymbol{u}_1, ..., \boldsymbol{u}_n\} \in \boldsymbol{\mathcal{U}}$, the random variables, $\{\tilde{f}(\boldsymbol{u}_1), ..., \tilde{f}(\boldsymbol{u}_n)\} \in \mathcal{V}$, 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 $\mu(\boldsymbol{u}) = \mathbf{h}^T(\boldsymbol{u})\boldsymbol{\theta}$ for some choice of q basis functions (often taken to be a constant or linear function of the input parameters), $\mathbf{h}(\boldsymbol{u}) = (h_1(\boldsymbol{u}), ..., h_q(\boldsymbol{u}))^T$, 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), and 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 k^{th} entries of the vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. The correlation coefficients, $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_p)^T$, determine how rapidly the emulated output changes with respect to each input dimension. Letting $\boldsymbol{U}_i^{\text{D}}$ $(i = 1, \ldots, n)$ 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 $\boldsymbol{v}^{\text{D}} = (v_1^{\text{D}}, ..., v_n^{\text{D}})^T$. We denote the training input/response data together as $\mathcal{D}_{\text{GP}} = \{(\boldsymbol{U}_1^{\text{D}}, v_1^{\text{D}}), ..., (\boldsymbol{U}_n^{\text{D}}, v_n^{\text{D}})\}$.

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),

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

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

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

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

where $\mathbf{R}$ is an $n \times n$ matrix of correlations between input design points, $\mathbf{R}_{i,j} = c(\boldsymbol{U}_i^{\text{D}}, \boldsymbol{U}_j^{\text{D}})$. Likewise $\mathbf{r}$ is a vector of correlations between the new input and each of the inputs in the design, $\mathbf{r}_i(\boldsymbol{u}^*) = c(\boldsymbol{u}^*, \boldsymbol{U}_i^{\text{D}})$, and $\boldsymbol{w} = \left(\boldsymbol{r}^T(\boldsymbol{u}^*)\boldsymbol{R}^{-1}\boldsymbol{H} - \boldsymbol{h}^T(\boldsymbol{u}^*)\right)$. 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,

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

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

In the next two sections we introduce parallel partial emulation to handle vector-valued ~~QoI's and range-limited~~ QoIs and zero-limited emulation to handle ~~QoI's~~ 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 $\mathbf{V}^{\text{D}}$ where the j^{th} column, $\mathbf{V}_j^{\text{D}}$, corresponds to all n responses of the j^{th} output component in the training data. Much like the scalar GP, the PPE approximates the j^{th} component of $\boldsymbol{v}^* = \tilde{f}(\boldsymbol{u}^*)$ at an untested input $\boldsymbol{u}^*$ as a sample from the Student t-distribution with $n - q$ degrees of freedom given by

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

Here $\boldsymbol{\theta}_j$ is calculated by replacing $\boldsymbol{v}^{\text{D}}$ by $\mathbf{V}_j^{\text{D}}$ in Equation 5, and where $m_j(\boldsymbol{u}^*)$ and σ_j^2 are calculated by replacing $\boldsymbol{v}^{\text{D}}$ by $\mathbf{V}_j^{\text{D}}$ and $\boldsymbol{\theta}$ by $\boldsymbol{\theta}_j$ in Equations 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~~ GPs 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 is 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~~ GPs 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 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 ~~QoI's~~ QoIs to fit a GP. For prediction at an untested input, we take the ~~GP's~~ GPs prediction if positive, or zero if the ~~GP's~~ GPs prediction is negative.

For vector-valued outputs, the zGP is a pre-processing 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 zero-censored Gaussian process emulator (PPzGP) has several advantages that make it an effective ~~surrogate model~~ 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 ~~as surrogate models for analyzing wind turbine~~ to analyze WT power production (Golparvar et al., 2021), ~~maximizing power with~~ wake steering (Gori et al., 2024), blade loads (Clark and Clark, 2022), and mono-pile reliability ~~analysis~~ (Morató et al., 2019), the results in this paper are the first where ~~range-limited~~ zero-censored and parallel partial emulation have been combined for ~~wind turbine~~ WT modeling. ~~The surrogate PPzGP model is orders of magnitude faster to interrogate than the full simulator, which means that it is tractable for generating large training data sets for damage classification.~~

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

To apply the elementary effects 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, $\mathbf{u}_0^{(j)}$ for $j = 1, ..., r$, 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 vectors in the entrants in a random order to create the elementary effects experiment design.

This design provides a more comprehensive sensitivity analysis than a one-factor-at-a-time (OAT) approach . Although it is less precise than variance-based global sensitivity analysis methods such as Sobol indices , the Morris screening requires far fewer model evaluations. To explicitly quantify pairwise parameter interactions, methods like Sobol indices require a number of simulations which scales quadratically with the number of parameters, but the computational cost of the Morris screening scales linearly .

For each component index l of $\mathbf{u}$ and each of the r trajectories indexed by j, the method calculates the elementary effect using a finite difference

$$EE_{l,j}(\mathbf{u}) = [f(\mathbf{u} + \mathbf{e}_l) - f(\mathbf{u})]/\Delta.$$

The mean and the standard deviation of the elementary effects, $\{EE_{l,1}, ..., EE_{l,r}\}$, are calculated for each dimension l. To avoid the possibility of large positive and negative elementary effects canceling out, we use $\mu_l^* = \frac{1}{r}\sum_{j=1}^{r} |EE_{l,j}|$, instead of the mean, μ_l. The standard deviation, which requires the average of elementary effects, μ_l, is given by $\sigma_l = \sqrt{\frac{1}{r}\sum_{j=1}^{r}(EE_{l,j} - \mu_l)^2}$. 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 . A plot of pairs (μ_l^*, σ_l) for each input $l \in \{1, ..., p\}$ is used to determine the relative importance of the inputs . We use the elementary effects tool provided in the Python Library `SALib` .

2.5 Random forest classification algorithm

The full simulator and the PPzGP surrogate model map environmental and damage-related input parameters to an output vector that includes generator power, blade tip acceleration, and lift. 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 leading edge erosion LEE is stratified into several levels of erosion severity, $\{\alpha_1, ..., \alpha_k\}$. 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 ~~leafs~~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 ~~MatLab's random forest (`fitcensemble`) implementation and associated hyper-parameter optimization framework (~~the random forest implementation in ~~`OptimizeHyperparameters`~~`fitcensemble`~~) to select the tree depth and the number of independent learners~~ from MATLAB 2025a. (The MathWorks Inc., 2024).

3 Workflow outline

~~All datasets in this study were generated with the 5 MW reference wind turbine in OpenFAST using the `Aerodyn`, `Elastodyn`, `Servodyn`, and `Inflow Wind` modules .We assume constant wind speed, direction, air density, and shear (see Table 1). 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 . **Simulation Parameter** **Description** Power Rating 5 MW Rotor Orientation, Configuration Upwind, 3 Blades Rotor, Hub Diameter 126 m, 3 m Hub Height 90 m Cut-in, Rated, Cut-out wind speed $\{3, 11.4, 25\}$ ms^{-1} Simulation Time Step 6.25 ms Simulation Total Time 180 s Wind Type Uniform Wind Files We varied four environmental inputs, wind direction, wind speed, air density, and wind shear. During each simulation, the turbine nacelle is fixed. The wind direction refers to the yaw angle measured from the nacelle. The wind speed ranges from the lowest speed 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) . Because air density has a large impact on power , 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 . We model wind speed, V, as a function of height, h, above the ground using the wind shear power law $V(h) = V_{\mathrm{ref}} \left(\frac{h}{h_{\mathrm{ref}}}\right)^{\nu}$ where V_r is the reference wind speed, h_{ref} 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 .~~

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 ~~2~~ 5 we summarize the ~~ranges of the environmental parameters we varied our inputs over for the design. To ensure that the turbine reaches equilibrium with the environmental conditions, we ran the simulations for 180 seconds and discarded the first two-thirds of simulation times to avoid transient effects . **Input** **Range** **Units** Wind Direction -15, 15degrees Air Density 1.1, 1.42kgm^{-3} Wind Speed 3, 25ms^{-1} Wind Shear Coefficient 0, 0.5(-)~~

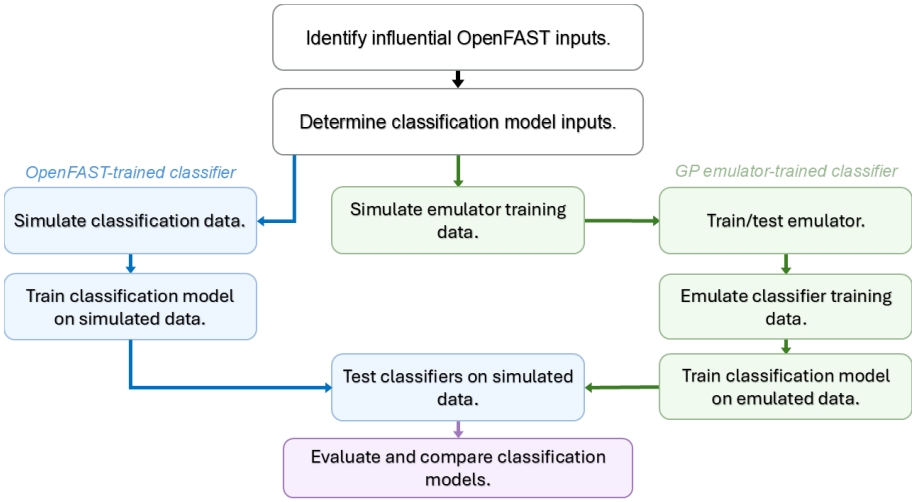

Figure 2. Workflow for comparing a simulator-trained classifier with a GP-emulator-trained classifier for leading-edge erosion classification.

We discretize the blade surface into six regions, defined by the data in Table 3 and shown in Figure 1. In the columns of Table 3 we show which `AeroDyn` blade nodes belong to each region, the location of the region along the blade, the length of the region, and the airfoil classes assigned to the region (which we kept the same as the NREL reference wind turbine). We use a stochastic modelfor leading edge erosion in which the severity of damage falls into one of several classes (). () model erosion and () model blade damage and structural damage by grouping the aerodynamic elements used by OpenFAST into different blade regions. Both () and () capture aerodynamic changes due to leading edge erosion by altering the lift and drag coefficients of the airfoils in different regions along the blade.

In our leading edge erosion model, we group blades into five erosion severity classes defined by $\alpha = \{0.00, 0.25, 0.50, 0.75, 1.00\}$. The erosion level, $e_i \in [0, 1]$, of the i-th blade segment is a constant such that if $e_i = 0$, the blade element is "clean" and if $e_i = 1$ the blade element is at its maximal eroded state. Random erosion level vectors, $\mathbf{e}$, are sampled from a multivariate normal distribution whose mean vector and covariance matrix are given in terms of the erosion severity class, α,

$$\mathbf{e} \sim MVN(\boldsymbol{\mu}(\alpha), \boldsymbol{\Sigma}(\alpha)).$$

We approximate the aerodynamic effects of leading edge erosion on the blade by imposing a relationship between the blade region erosion level and the lift and drag coefficients. () found that the most severe case of erosion led to a lift loss of 53% while saw an increase in drag of up to 500% for the most severely eroded airfoils. We, therefore, use the heuristic that the aerodynamic effects of erosion scale the lift coefficient in the i-th blade region by $c_{L,i} = 1 - 0.53e_i$ and five datasets used in this study and their roles. We use the first dataset for the global sensitivity analysis in Section 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 were described in Section 2.2.2, are used as inputs to the drag coefficient by $c_{D,i} = 1 + 4e_i$. Since erosion accumulates faster and causes more damage at the tip of the blade , we set the mean erosion level vector to be $\boldsymbol{\mu}(\alpha) = (\alpha x_1, ..., \alpha x_6)$, where x_i is

the distance from the hub to the midpoint of the i-th region, divided by the total blade length. The (i,j)-entry of the covariance matrix is

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

This choice is justified by the observation that nearby regions have similar erosion levels, the variability of erosion accumulation increases towards the tip of the blade, and greater erosion severity is correlated with a longer exposure to erosion and hence a wider variability in erosion damage.

Region Aerodyn Nodes Distance from Hub Region Length Airfoils Included 1 5-6 10.25 m 8.2 m Du40, Du35 2 7-8 18.45 m 12.3 m Du35, Du30 3 9-10 30.75 m 8.2 m Du25 classifier. In order to eliminate potential bias, we perform feature ranking and selection on an independent dataset in Section 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 Section 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 data sets are shown in row 4 11-12 38.95 m 8.2 m Du21 5 13-14 47.15 m 7.5 m NACA64 6 15-19 (18) 54.67 m 6.8 m NACA64

We use five outputs to detect erosion: generator power , the lift and drag coefficient pressure sensors of Table 5. In Section 4.4, flap-wise acceleration of the tip of the blade , and the edge-wise moment of the root of the blade . Note that we only track the sensor in the tip of the blade. Each output from our OpenFAST simulation is a vector with 160 entries per second of simulation time. Following a similar approach to (), we use statistical moments of each of the five output time-series, specifically the mean, standard deviation, skew, and kurtosis , rather than the time-series itself.

Output OpenFAST Unit Description Blade Tip Acceleration TipALxB1 m s^{-2} Blade local flapwise (absolute) tip acceleration. Drag Sensor B1N6Cd (-) Drag coefficient detected by Region 6 sensor. Lift Sensor B1N6Cl (-) Lift coefficient detected by Region 6 sensor. Generator Power GenPwr MW Generator Power. Blade Root Moment RootMxb1 kW-m Blade Edgewise Moment (i.e., the momentcaused by edgewise forces) at the blade root.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.

Our goal in this work is to compare two random forest classifiers, one trained using OpenFAST-simulated data (500 points). The second trained using 5000 predictions from the PPzGP emulator. In Figure 2 we visualize the workflow for classifying leading edge erosion. After selecting an appropriate proxy for erosion, we identify influential input variables, and fix those that are non-influential. A primary dataset is simulated by OpenFAST and analyzed to determine the most important inputs for erosion damage classification. These inputs inform the quantities of interest that will be predicted by our PPzGP surrogate model 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 hyper-parameter tuning. The emulator-trained classification models do not train on simulation data. The two random forest classifers are compared to evaluate the effectiveness of the surrogate-based approach in Section 4.

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 / 1,000 / 5,000 / 10,000 / 50,000	GP emulator	training emulator-trained classifier
5	Classifier testing dataset	600	OpenFAST	hyperparameter tuning & final classifier evaluation

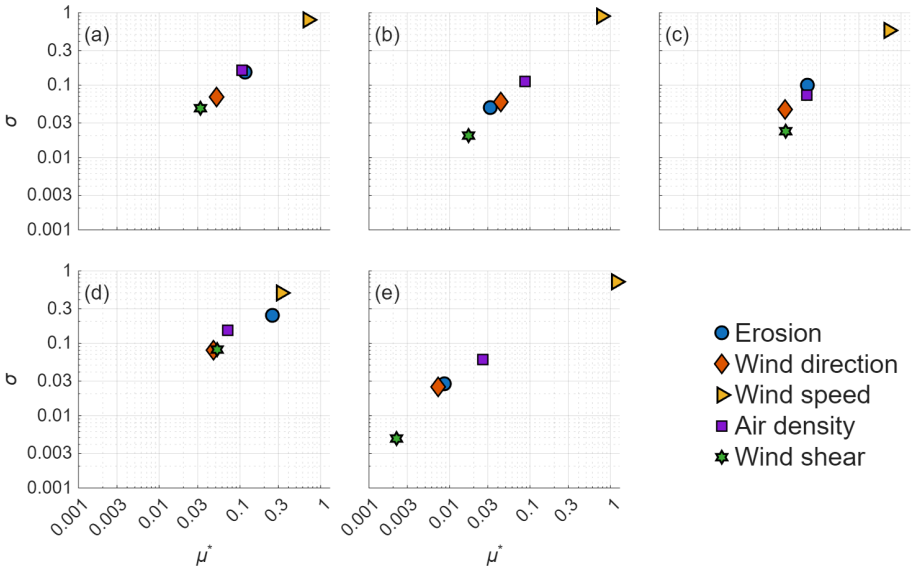

Figure 3. Elementary effects plots normalized for variable magnitude on a log vs. log scale, for the mean of the QoIs; (a) blade root moment (M_{root}), (b) blade tip acceleration (a_{tip}), (c) lift coefficient (C_L), (d) drag coefficient (C_D), and (e) generator power (P_{gen}). The same symbols are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

4 Results

4.1 Global sensitivity analysis

~~The elementary~~ The elementary effects study was carried out on the means of the five sensor output time series described in Section 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 ~~QoI's~~QoIs. In ~~Figure~~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 ~~QoI's~~ 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, $P = \frac{1}{2}\rho A v^3 C_p$, 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. ~~This~~ In summary, the GSA study reveals that lift and drag channels are more sensitive to erosion ~~inputs than any of~~ than the other outputs we tracked ~~, supporting the use of aerodynamic pressure data for erosion detection .~~(Abdallah et al., 2022; Carmona-Troyo et al., 2025; 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 ~~number of inputs~~ dimension of the input space is reduced. ~~We~~ In this subsection, we describe how we selected the inputs to the LEE classification problem from the sensor outputs. We considered twenty-three potential damage predictors, including wind speed, wind direction, 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 ~~initial random forest to perform this selection.~~ 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 ~~train-test~~ 5-fold cross-validation splits of the simulated ~~training data. The training data for the random forest classification algorithm included~~ data. This training data includes 100 samples from each erosion class ~~(drawn from~~ obtained using a Latin hyper-cube design (Helton and Davis, 2003)~~). Wind speed, wind direction, air density along with the mean, standard deviation, skew, and kurtosis of the QoI's listed in Table 4 are the potential damage classification inputs. .~~.

~~In Figure 4 we show a~~ We show the ranking of the damage ~~predictors.~~ predictors in Fig. 4. The variability of the score assigned to each predictor is ~~small~~relatively small, which indicates that the rank of each damage predictor is stable ~~. These damage predictors are the expected value and standard deviation of the aerodynamic drag sensor , the expected value~~ 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-effects sensitivity study. The most important sensor outputs include the mean and standard deviation of the ~~aerodynamic lift sensor, the standard deviation and skew of the~~ drag coefficient and the mean lift coefficient. Additional selected predictors include the skewness and standard deviation of blade

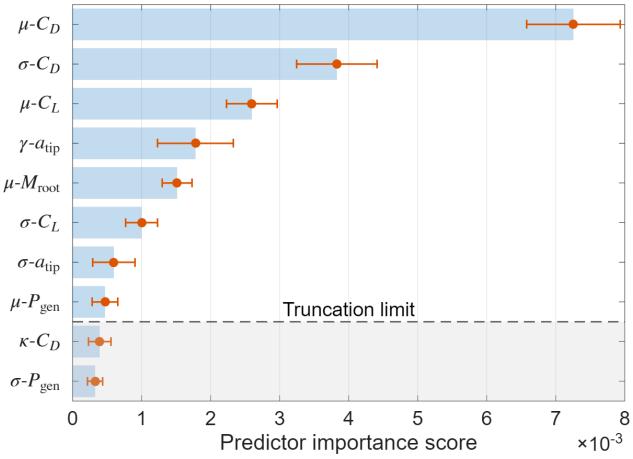

Figure 4. Average predictor importance score via 10 5-fold repeated cross-validation studies. Error bars represent the standard deviation of the score over the 50 train/test splits. The variables denote the mean, μ, standard deviation, σ, skew, γ, and/or kurtosis, κ, of the drag coefficient, C_D, the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

tip acceleration, the ~~expected value of the~~ mean blade root moment~~and of the generator power. We use these quantities as the QoI's we predict using the PPzGP surrogate.~~

~~The elementary effects study and the classifier input ranking support the use of pressure coefficient sensor data for erosion detection and underscore the importance of multi-modal data for accurate damage classification. The three most influential damage classification inputs were the mean aerodynamic drag coefficient sensor~~, the standard deviation of ~~aerodynamic drag sensor~~the lift coefficient, and the mean ~~aerodynamic lift sensor.~~ generator power.

4.3 Emulator evaluation

4.3.1 Comparing scalar-GP emulation and PPzGP emulation

~~The PPzGP emulator is~~ We trained and evaluated ~~on 210 total OpenFAST simulations (a separate data set from the one we use for classification). This dataset includes~~ the GP emulators on the third dataset in Table 5. These simulations were generated using OpenFAST with 50 samples for each of the ~~4~~ four eroded blade classes and 10 for the clean blade class. ~~The environmental inputs were selected~~ The training dataset varies the three environmental inputs identified by the GSA using Latin hypercube sampling over the ranges in Table ~~2~~6. The erosion level ~~ranges are~~ in blade sections 1-6 are varied according values given in Table ~~??.~~ 6, which we selected to ensure coverage of the stochastic erosion classes from Section 4.4.

~~We test the accuracy of~~ First, we compare the training efficiency and prediction accuracy of scalar-GP emulation to that of PPGP emulation, with and without the additional pre-processing step of zero-censored emulation. We test the ~~PPzGP predictions using five distinct train/test splits of the simulated dataset.~~ emulators, and compute 95% confidence levels, using 5 repeated 5-fold cross-validation sets of the OpenFAST dataset. Each split allocates ~~173~~ 168 training points proportionally

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

across erosion severity classes, with the remaining ~~37~~ 42 data points reserved for testing. ~~In Figure 5, we compare the PPzGP emulator predictions (blue) with~~

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

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

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 8 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 squared 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 ~~simulated outputs (red), and we display the associated 95% credible intervals for four QoI's. For each output ,~~ lift coefficient, σ-C_L, and the mean of the ~~majority of true values fall within the credible intervals, indicating good surrogate model calibration. We note that the drag sensor mean and generator power predictions exhibit narrow credible intervals. Predictions for large root moments show wider intervals, reflecting increased uncertainty in this QoI. The flat plateau in the~~ generator power~~predictions corresponds to the model respecting the rated power limit, while the physical constraint that the blade tip acceleration standard deviation is non-negative is similarly enforced. These model attributes are possible due to the zGP formulation and ensure that the PPzGP's outputs are physically meaningful.~~ , μ-P_{gen}.

~~In the left column of Table ??~~ 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 outputs across five training/test splits. While the expected coverage is around~~ 8 sensor quantities. The 95% ~~based on the GP's estimated variance, observed coverage typically ranges between 80~~GP credible interval contains the true function value with rate of 0.95. If the GP model is calibrated well, under repeated

Table 8. Normalized root mean square error (NRMSE) for each emulator and sensor quantity selected in Fig. 4. Values are reported as mean ± 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	$(6.5 \pm 1.3)\%$	$(12.1 \pm 1.0)\%$	$(2.6 \pm 0.3)\%$	$(13.4 \pm 1.5)\%$	$(9.2 \pm 0.4)\%$	$(6.9 \pm 1.2)\%$	$(11.1 \pm 0.9)\%$	$(1.7 \pm 0.2)\%$
zGP	$(6.5 \pm 1.3)\%$	$(11.8 \pm 1.1)\%$	$(2.6 \pm 0.3)\%$	$(13.4 \pm 1.5)\%$	$(9.2 \pm 0.4)\%$	$(6.8 \pm 1.0)\%$	$(9.5 \pm 0.7)\%$	$(1.7 \pm 0.1)\%$
PPGP	$(6.3 \pm 1.1)\%$	$(12.6 \pm 1.4)\%$	$(2.5 \pm 0.2)\%$	$(13.4 \pm 1.6)\%$	$(8.8 \pm 0.5)\%$	$(7.6 \pm 0.9)\%$	$(9.3 \pm 0.7)\%$	$(2.0 \pm 0.2)\%$
PPzGP	$(6.1 \pm 1.1)\%$	$(11.8 \pm 1.2)\%$	$(2.6 \pm 0.2)\%$	$(12.7 \pm 1.5)\%$	$(8.5 \pm 0.5)\%$	$(7.3 \pm 0.9)\%$	$(9.0 \pm 0.7)\%$	$(2.1 \pm 0.2)\%$

sampling, the empirical coverage of the credible interval will also converge to 95%. We observe coverage between 84% and 95% ~~. This discrepancy may be attributed to violations of the shared covariance assumptions inherent to the PPzGPmodel or to insufficient training data. In the right column, we show the standard deviation of the coverage, with the skew of the tip acceleration exhibiting the highest variability across splits.This is consistent with its elevated normalized root mean squared error (NRMSE) . The results in the left column of Table **??**, show that the average NRMSE is less than 10% for all but one output. The corresponding standard deviations shown in the right column are small, indicating consistent model performance across different data splits.~~ 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.

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	$(84.7 \pm 3.5)\%$	$(85.4 \pm 2.4)\%$	$(85.7 \pm 2.5)\%$	$(86.1 \pm 2.0)\%$	$(78.6 \pm 2.5)\%$	$(85.3 \pm 3.0)\%$	$(79.4 \pm 2.7)\%$	$(81.9 \pm 2.6)\%$
zGP	$(84.8 \pm 3.5)\%$	$(86.4 \pm 1.8)\%$	$(85.7 \pm 2.5)\%$	$(86.1 \pm 2.0)\%$	$(78.6 \pm 2.5)\%$	$(86.6 \pm 2.6)\%$	$(83.7 \pm 2.2)\%$	$(84.8 \pm 2.5)\%$
PPGP	$(90.6 \pm 2.2)\%$	$(87.6 \pm 2.6)\%$	$(91.1 \pm 1.7)\%$	$(87.3 \pm 2.1)\%$	$(82.2 \pm 2.6)\%$	$(89.3 \pm 2.6)\%$	$(83.3 \pm 2.7)\%$	$(94.5 \pm 1.7)\%$
PPzGP	$(92.8 \pm 2.2)\%$	$(89.1 \pm 2.6)\%$	$(93.0 \pm 1.7)\%$	$(88.0 \pm 1.8)\%$	$(84.1 \pm 2.8)\%$	$(89.5 \pm 2.5)\%$	$(84.8 \pm 2.6)\%$	$(95.2 \pm 1.5)\%$

In Table 10 we compare the performance of the ~~range-limited outputs (see) is the most expensive part of building the PPzGP, but we can reduce this cost through parallelization. After this step, the actual training of~~ standard-GP and PPzGP emulators across the selected sensor outputs 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 ~~emulator takes a few seconds. Ultimately, taking predictions from the emulator is essentially free. A single simulation is on the order of 10 seconds, while one PPzGP evaluation is on the order of 0.001 of a second. (4 orders of magnitude faster than running the simulator)~~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

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
~~Coefficient of Drag mean~~ μ-C_D
~~Coefficient Lift standard deviation~~ σ-C_D
~~Coefficient of Lift mean~~ μ-C_L
~~Tip Acceleration skew~~ γ-a_{tip}
~~Root Moment mean~~ μ-M_{root}
~~Coefficient of Lift standard deviation~~ σ-C_L
~~Tip Acceleration standard deviation~~ σ-a_{tip}
~~Generator Power mean 95.135 1.209~~ μ-P_{gen}

~~Coefficient of Drag mean 0.075 0.028 Coefficient Lift standard deviation 0.103 0.025 Coefficient of Lift mean 0.029 0.007 Tip Acceleration skew 0.13~~
~~In Table 11, we show the computational time for the different stages of training and running the GP emulator . The pre-processing step for~~

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	~~Time~~ Wall-clock time (s)
Single OpenFAST simulation	~~46 s~~ 28.4 ± 0.3
PPzGP training ~~simulations (173 points)~~	~~7900 s~~ 3.67 ± 0.56
~~PPzGP imputation (173 training points)~~ Standard GP training	~~1400 s~~ 52.3 ± 1.0
~~PPzGp fitting (173 training points)~~ PPzGP prediction	~~3.4 s~~ 0.93 ± 0.01
~~Single PPzGP~~ Standard GP prediction	~~0.0093 s~~ 7.6 ± 0.4
Direct OpenFAST generation, 10,000 samples	$\approx 2.84 \times 10^5$
Full PPzGP workflow, 10,000 samples	$\approx 6.2 \times 10^3$

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.

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.

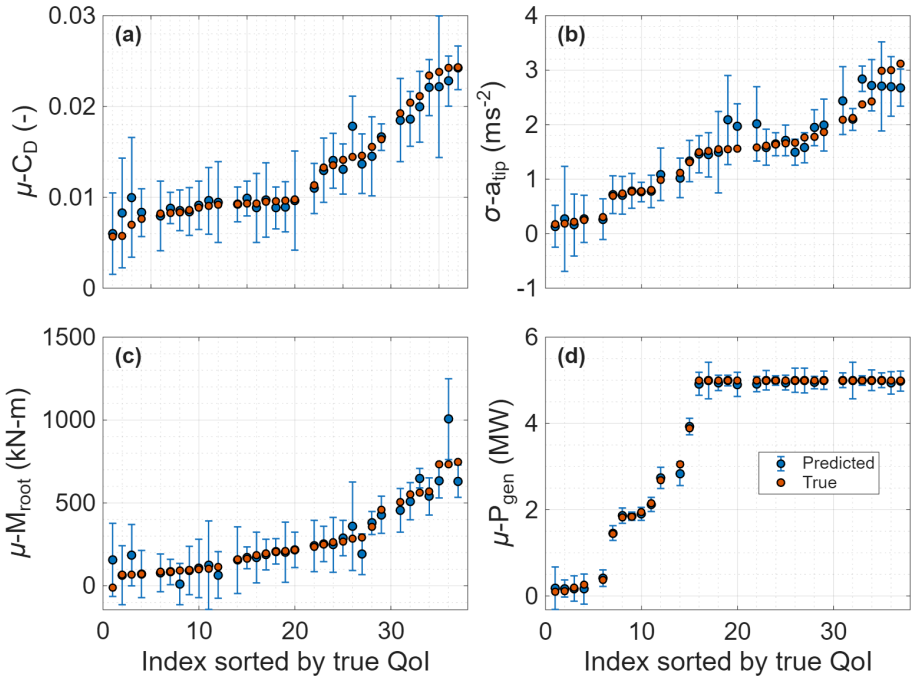

Figure 5. Predictions, ordered by simulated (true) value, and 95% credible intervals of the PPzGP emulator model for four of eight outputs: (a) mean of drag sensor, (b) standard deviation of blade tip acceleration, (c) mean of root moment, and (d) mean of generator power.

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 (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 five-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 $46\times$. 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.

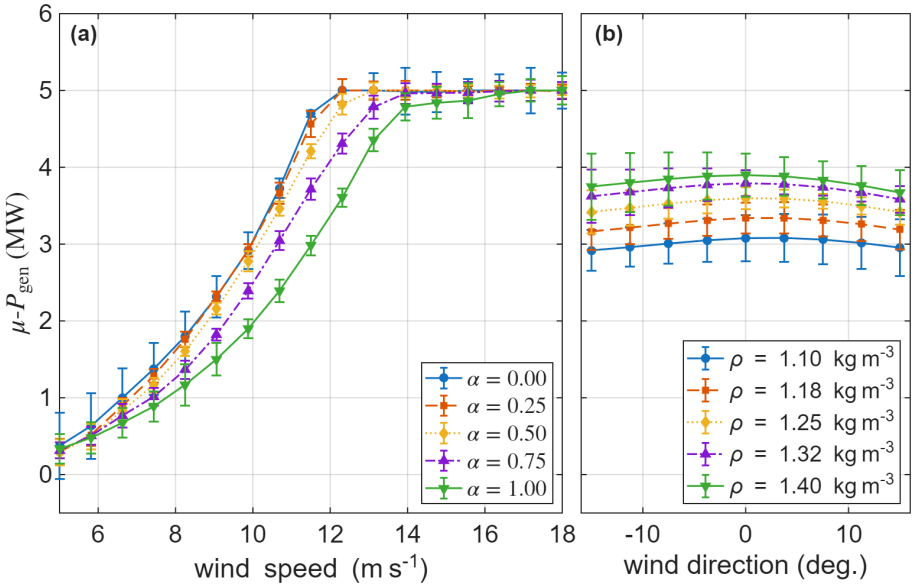

Figure 6. (a) Mean generator power as a function of wind speed for five levels of LEE. (b) Mean generator power as a function of wind direction for different air densities at fixed wind speed with clean blades.

In ~~Figure~~Fig. 6 we show that the PPzGP ~~surrogate~~ 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~~qualitatively~~, 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. ~~Unlike their single-output model, the~~ The PPzGP provides vector-valued predictions across multiple sensor outputs~~, with normalized RMSEs ranging from 2% to 14% (Table ??)~~. 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~~(not at zero degrees)~~. These relationships are consistent with prior studies (Hulsman et al., 2022) and are also evident in ~~Figure~~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 Figure 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 ~~severe~~ extreme turbine operational states.

4.4 Classification comparison

~~The main results of our work are presented in this section in which we compare two random forest leading edge erosion classifiers~~In this section we present our main results comparing two LEE classifiers using the random forest algorithm. The

first~~classifier~~, which we refer to as the simulator-trained classifier, is trained directly on ~~simulated data from OpenFAST. The second is trained using data predicted by the PPzGP emulator. We use two simulated and one emulated dataset for these experiments. For the classifier trained on OpenFASTdata, we use a design of 500 simulations for training and testing the classifiers~~the fifth dataset (OpenFAST) in Table 5. This dataset includes ~~100~~ 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 hyper-cube design ~~(see the~~ with ranges in Table~~2). We train the emulator using a different set of 210 OpenFAST simulations (see the emulation training outlined in Section 2.2). The secondrandom forest classifieris trained on emulator-predicted data which results from varying the environmental inputs in the same ranges, but uses 1000 data points per class instead of 100. Table 11 indicates that emulator predictions are very inexpensive compared to simulations. We compare the classification accuracy of the simulation-trained and emulation-trained damage classifiers using 10-fold cross-validation.~~

~~We quantify the accuracy of each classifier using the~~ 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 data sets 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 Section 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 five-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, 1,000, 5,000, 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) ~~curve~~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.

In Fig. 7 we show that increasing the size of the emulated training dataset improves the downstream performance of the LEE classifier. With 500 samples the performance of the emulator-trained classifier is worse than that of the simulator-trained classifier, with 1,~~which plots false positive rates on the horizontal axis and the true positive rateson the vertical axis. Larger values of the AUC metric, which is the area under the ROC curve, indicate higher accuracyof the classifier.~~ 000 samples the performance is comparable, and with 5,000-, 10,000- and 50,000 samples the emulator-trained classifier outperforms the simulator-trained classifier, as assessed using macro-AUC and macro-F1 metrics.

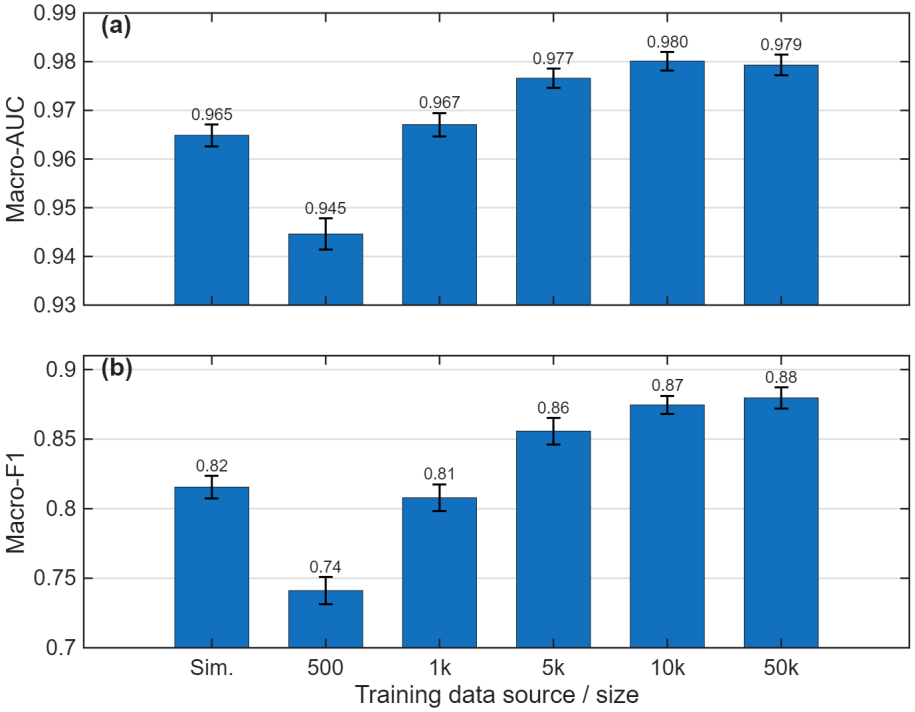

Figure 7. Comparison of classifier performance as a function of training data source and size. Panel (a) shows macro-AUC and panel (b) shows macro-F1 for the simulator-trained classifier and emulator-trained classifiers using 500, 1,000, 5,000, 10,000, and 50,000 emulated training samples. Error bars indicate 95% confidence intervals across repeated cross-validation splits.

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 $100\times$ the raw metric difference. Positive values indicate better performance for the emulator-trained classifier.

Emulated training size	Macro-AUC		Balanced accuracy		Macro-F1	
	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]	$< 10^{-14}$
1,000	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}
5,000	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]	$< 10^{-14}$

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 1,000 samples, the differences are small, indicating comparable performance. With 5,000, 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 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 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%].

~~There are several conclusions we can draw from the simulation-trained results displayed in Figure ??. First, the simulation model is fairly accurate, predicting the correct label for between 68% and 100% of each class (diagonal boxes in blue) . Most notably the class with the highest level of erosion is correctly predicted for 79 of the 100 samples. Second, the~~ A per-class ~~AUC is high, above 0.93 for each~~ comparison indicates that the emulator-trained classifier preserves performance on the clean class while improving accuracy for all eroded classes. 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 ~~clean class has the highest AUC, while the intermediate damage classes(0.25, 0.50) are lower, which implies that the~~ largest improvements occur for the intermediate ~~erosion severity~~

Table 14. Class-wise differences between models, emulator trained minus simulator trained. Differences are reported in points, defined as $100\times$ 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 ($< 10^{-8}$)

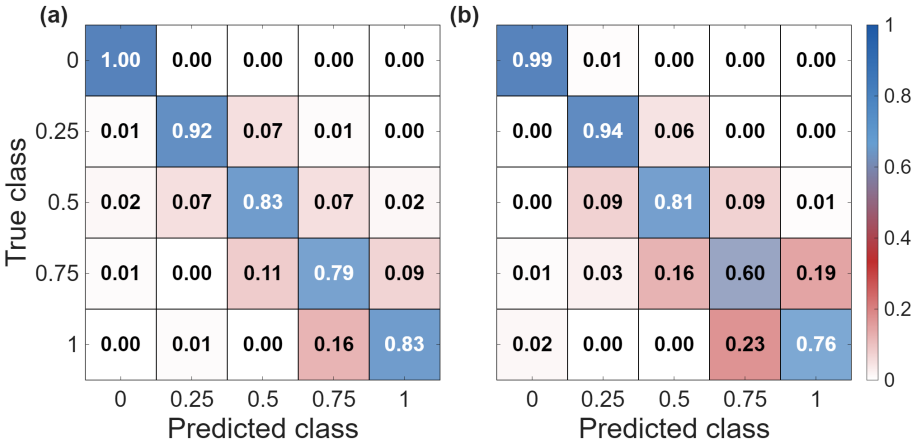

Figure 8. Normalized confusion matrices comparing the best emulator-trained classifier and the simulator-trained classifier. Panel (a) shows the emulator-trained classifier with 10,000 training points, and panel (b) shows the simulator-trained classifier. Rows indicate the true erosion class, columns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

~~levels are harder to distinguish. Third, incorrect classification~~ 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~~to occur between damage classes that are very close (off-diagonals on the confusion matrix). Indeed, only 5 out of 500 cases are misclassified by more than one damage level.~~ .

~~We compare the leading edge erosion classification results shown in Figure ?? to the emulation-trained random forest classifier results in Figure 9. We find that overall there is a slight increase in predictive accuracy observed for each class. Second, the AUC metrics are somewhat higher~~ 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 ~~classes~~damage classes. This improvement is especially clear for classes $\alpha = 0.5$ and $\alpha = 0.75$, where the ~~simulation-trained classifier was the weakest. Overall, 7 out of 500 cases were misclassified by more than one~~

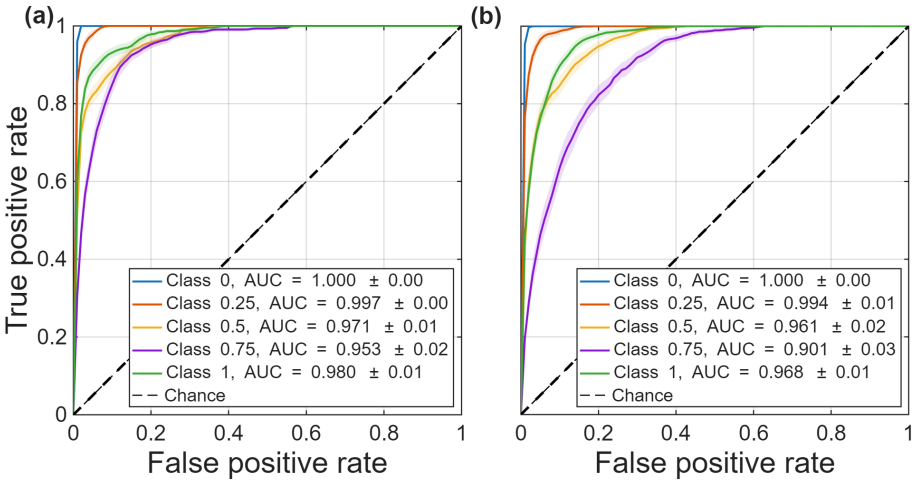

Figure 9. One-versus-rest ROC curves for the best emulator-trained classifier and the simulator-trained classifier. Panel (a) shows the emulator-trained classifier using 10,000 emulated training samples, while panel (b) shows the simulator-trained classifier. Curves are shown for each erosion class,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

~~damage level.We note that the most serious misclassification for either classifier occurs when the blade has an erosion level of 1 but is classified as being a clean blade (no erosion).~~ simulator-trained classifier has the lowest AUC values, indicating that the larger emulator-generated training set helps resolve the most challenging class boundaries.

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.

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	~~**Simulation-Trained**~~ Emulator-trained recall [95% CI]	~~**Emulation-Trained**~~ Simulator-trained recall [95% CI
~~**AUC mean**~~ $\alpha = 0.00$	~~0.9670~~ 1.00 [1.00, 1.00]	~~0.9834~~ 0.99 [0.99, 1.00]
~~**AUC std**~~ $\alpha = 0.25$	~~0.0265~~ 0.92 [0.90, 0.93]	~~0.0150~~ 0.94 [0.92, 0.95]
~~**% Improvement**~~ $\alpha = 0.50$	~~-~~0.83 [0.81, 0.85]	~~1.6970%~~ 0.81 [0.78, 0.83]
~~**Accuracy mean**~~ $\alpha = 0.75$	~~0.8300~~ 0.79 [0.77, 0.81]	~~0.8674~~ 0.60 [0.57, 0.63]
~~**Accuracy std**~~ $\alpha = 1.00$	~~0.0455~~ **0.83 [0.81, 0.85]**	~~0.0669~~ **0.76 [0.73, 0.78]**
~~**% Improvement** - 4.5060%~~ height		

~~In Table **??**, we summarize the results of the classification experiments. First, we highlight the AUC metrics (averaged across the 5 classes). The emulation-trained~~ 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 ~~slightly higher AUC on average. While the classification accuracy improvement is modest,~~ 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 four 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 $\bar{C} = \frac{1}{N}\sum_{i,j} n_{ij} C_{ij}$, 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 11(b) shows a noticeable reduction in recall for the $\alpha = 0.75$ class under the tuned rule compared to Fig.8(b), 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 severe false negatives, with a risk-reduction tradeoff comparable to that of the ~~computational savings are significant. The PPzGP emulator (once trained) generates each random forest classifier training data sample in $\sim 0.01s$ as opposed to a single simulation run which takes $\sim 40s$. Even accounting for the 210 simulations we use to train the emulator,~~ 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%

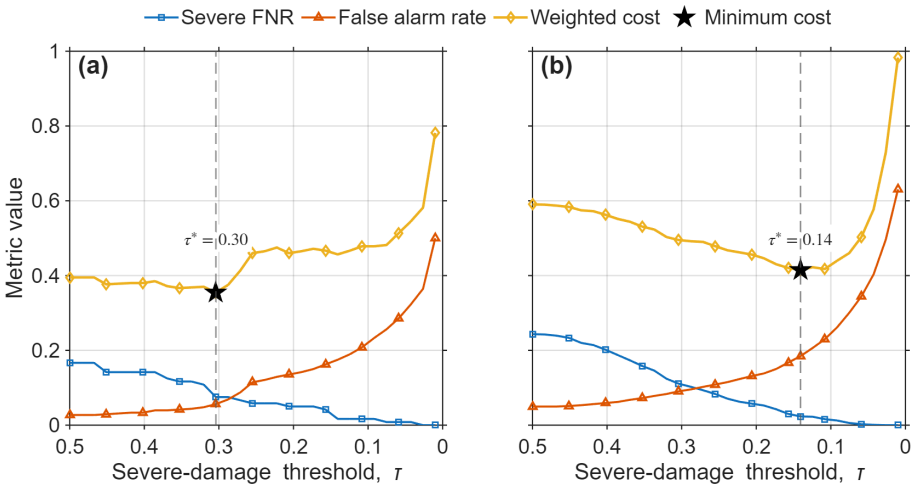

Figure 10. Risk-sensitive threshold tuning for severe-damage detection. The severe-damage alarm threshold τ is varied for the emulator-trained classifier (a) and simulator-trained classifier (b). The dashed vertical line and star marker indicate the threshold τ^* that minimizes the weighted confusion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

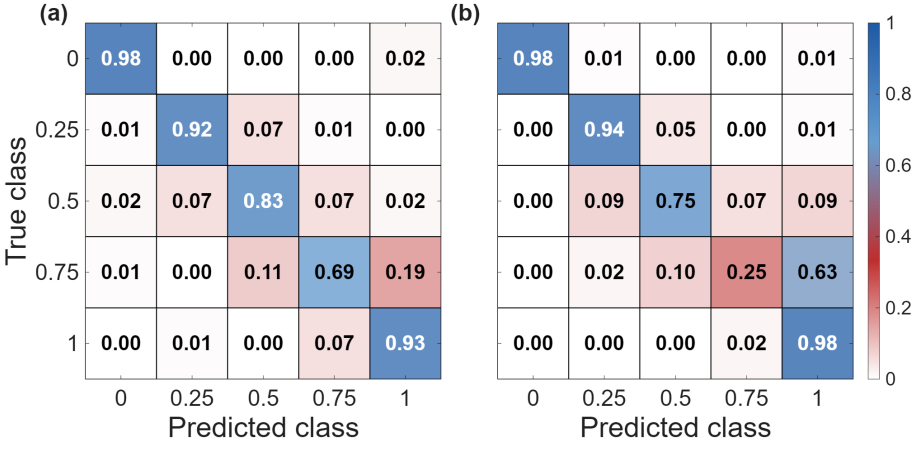

Figure 11. Row-normalized confusion matrices for the emulator-trained classifier (a) and simulator-trained classifier (b) at the cost-minimizing severe-damage threshold τ^* in Figure 10.

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 two from Table 5 and the ~~overall computational effort of~~ selection procedure from Section 4.2 and shown in Fig. 4, retaining μ-C_D, σ-C_D, γ-a_{tip}, σ-C_L, μ-C_L, μ-M_{root}, μ-a_{tip}, κ-C_D, and κ-a_{tip}. The emulator is then trained on a noisy version of its original training dataset. We compare the simulator-trained classifier to the ~~complete PPzGP workflow is drastically reduced.~~ emulator-trained classifier. We fit the emulator using a noisy version of dataset three from Table 5, and generated 10,000 datapoints for training the classifier. We use a noisy version of dataset five from Table 5 for 10 repeated 5-fold cross-validation studies.

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 $100\times$ 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}

~~Surrogate modeling is gaining traction in wind turbine modeling.For instance, ()used an augmented Kalman filter to predict tower-top loads with less than 10% error. employed polynomial response surfaces to explore load uncertainty using only 140 simulations. () combined GPs with Bayesian Quadrature to estimate torque loading, optimizing training point placement to minimize posterior variance and save full-model interrogations. These examples reinforce the effectiveness of computationally efficient surrogate modeling.~~ 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.

~~The advantage of the PPzGP is its computational efficiency. The PPzGP accurately predicts multiple QoIs simultaneously while only requiring a relatively small training dataset. Our PPzGP emulator relies on the correlation between the different sensor outputs (they share the same input wind-field and structural coupling through the blades, hub, and tower) to produce multiple outputs using a single model, which eliminates the need to train and validate multiple independent surrogates. The surrogate also enforces physically meaningful bounds on output QoIs (such as generator power). Finally, we have demonstrated that our surrogate modeling approach can be used effectively for damage classification.~~

4.4.2 Classification without direct aerodynamic sensing

~~Structural health monitoring for leading edge erosion provides operators with the ability to make maintenance decisions that preserve integrity of~~ 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 its 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

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

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 ~~turbine . The studieswe present in this paper indicate that a random forest classifier is particularly effective at this prediction. Our random forest model trained on simulated dataaccurately classifies leading edge erosion in 83% of test cases. Classification accuracy is influenced by the number of erosion categories. Unsurprisingly, a model tasked with distinguishing between 4 classesinstead of 10 showed a 30% increase in accuracy . The machine learning model's ability to discriminate between the 5 damage classes we use allows for better decisions to be made regarding the operation and maintenance of the rotor and blades. Importantly, the emulator enables extremely efficient generation of more training samples which could lead to classifiers that are able to accurately distinguish between more erosion levels.~~ three-bladed turbine, both the once-per-revolution band (1P) and blade-passing band (3P) were considered.

~~Combining data from multiple sensors improves our ability to distinguish erosion effects from broader environmental influences. The use of additional sensor data is particularly valuable near rated wind speeds, where optimizing energy recovery is critical. For example, blade root moment is a promising input for erosion classification. The selected classification features also include multiple statistical moments. Notably, both the mean and standard deviation of the aerodynamic drag coefficient give high importance scores (see Figure 4), suggesting that higher-order moments provide valuable information beyond mean values alone. This finding is consistent with the fact that wind approaching at an angle can activate structural modes, leading to periodic oscillations~~

We repeated the predictor selection process from Section 4.2 four times, ranking and selecting the top 14 quantities from an initial list of 108; γ-$a_{x,tip}$, μ-$M_{x,root}$, μ-$F_{y,root}$, ρ_1-$a_{z,tip}$, BP_{3P}-$M_{z,root}$, NFD-$a_{z,tip}$, ρ_1-$M_{y,tower}$, ρ_1-$M_{z,shaft}$, κ-$M_{z,shaft}$, ρ_1-$M_{y,shaft}$, BP_{3P}-$M_{y,tower}$, ρ_1-$M_{x,tower}$, κ-$M_{z,root}$, and κ-$a_{x,tip}$. The two highest ranked predictors, γ-$a_{x,tip}$ and μ-$M_{x,root}$, 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 validation and inner-loop hyperparameter tuning to compare the simulator-trained and emulator-trained classifiers with dataset five from Table 5. In this ~~study, periodic outputs such as blade tip acceleration show thatchanges in standard deviation, and even skew, offer stronger predictive signals than the mean alone.~~ experiment, we used the emulator to generate a dataset of 50,000 samples.

Table 17. Additional OpenFAST output channels used in the reduced-sensing experiment. Note, we relable `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

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

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

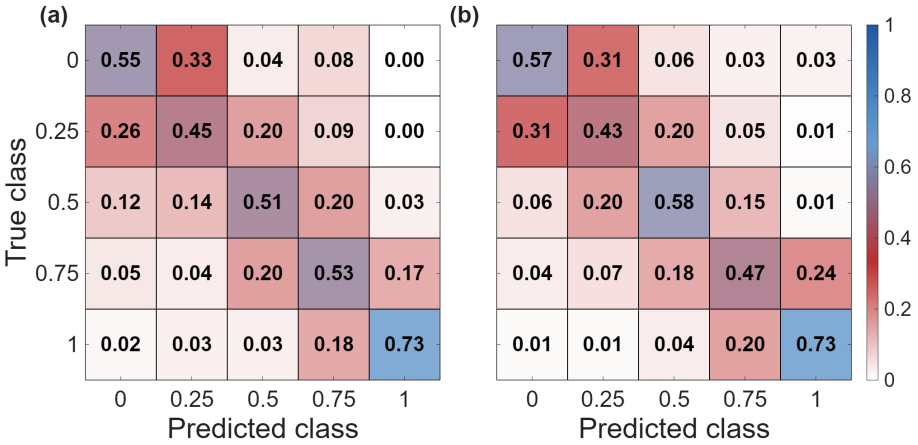

Figure 12. Confusion matrices for classification using non-aerodynamic sensor channels with emulator-trained classifier (a) and simulator-trained classifier (b).

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 $100\times$ the raw metric difference.

~~There are several interesting future research directions for the PPzGP surrogate modeling framework and random forest classifiers. First, we did not con~~

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, 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 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.

~~The PPzGP emulator lays the foundation for~~

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 monitoring LEE. We envisage that the PPzGP emulator in this paper would serve as the kernel of a probabilistic, graph-based digital twin ~~capable of~~ for tracking damage progression over time (Kapteyn et al., 2021). ~~Digital twins inform control decisions that influence turbine operation such as adjusting tip speed during storms to reduce erosion . While simulation demands were manageable in this study, complex applications, such as digital twin development, would benefit greatly from the speed-up~~

~~enabled by surrogate modeling.~~ Areas in which this study could be extended include accounting for how real turbine monitoring data is affected by sensor noise, controller behavior, atmospheric variability, and maintenance history. Further, in this work we do not track erosion progression over time. To close 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 ~~leading edge erosion~~ LEE classification using both a full physical simulator-trained random forest classifier and one trained by sampling a computationally efficient ~~PPzGP~~ emulator. We show that these two damage classifiers ~~are equally accurate~~ provide similar classification results as evaluated through a variety of ML metrics at predicting levels of ~~leading edge erosion damage on wind turbine blades~~ 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 ~~an increase in~~ increasing the overall training cost~~. And the use of the range limited emulator (zGP) ensures that real-world turbine specifications can be incorporated into our model. Once trained, the emulator is several orders of magnitude faster to evaluate than the full simulator.~~ , and the zero-censored emulator incorporates physical turbine constraints. Taken together, the advantages of ~~the PPzGP~~ this emulation-based framework lead to a reduction in computational ~~demands and support~~ cost and could be integrated into real-time ~~decision-making~~SHM systems, such as ~~that required for turbine digital twins~~those that comprise a digital twin. ~~This study establishes the PPzGP-trained classifier as a promising tool with the potential to enhance operational decision-making and optimize turbine performance in dynamic environmental conditions.~~

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 Gettemy (2025).

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 authors declare they have no competing interests.

Acknowledgments

We would like to thank Todd Griffith for the initial suggestion for the project and guidance on wind turbine engineering. We are also grateful to Ipsita Mishra for her help learning how to run OpenFAST as well as useful discussions on wind turbine modeling in general. Aidan Gettemy was supported on this project by National Science Foundation Grant #2401945.

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 Energy Science, 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, Probabilistic Engineering Mechanics, 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, 032 089, 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, https://doi.org/10.5075/epfl-298799_978-2-9701614-0-0, 26-30 June, 2022, 97–104, 2022.

~~Aslmostafa, E., Hamida, M. A~~

Avendano-Valencia, L. D., Chatzi, E. N., and ~~Plestan, F.: Nonlinear control strategies for a floating wind turbine with PMSG in Region 2: A comparative study based on the OpenFAST platform, Ocean Eng., 300, 117~~Tcherniak, D.: Gaussian process models for mitigation of operational variability in the structural health monitoring of wind turbines, Mechanical Systems and Signal Processing, 142, 106 ~~507, , 2024.~~ 686, 2020.

~~Bak, C., Forsting, A.~~

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

Barber, S., Deparday, J., Marykovskiy, Y., Chatzi, E., Abdallah, I., Duthé, G., Magno, M., ~~and Sorensen, N. N.: The influence of leading edge roughness, rotor control and wind climate on the loss in energy production, J.Phys. Conf.Ser.~~Polonelli, T., Fischer, R., and Müller, H.: Development of a wireless, non-intrusive, ~~1618, 052 050, , 2020.~~ MEMS-based pressure and acoustic measurement system for large-scale operating wind turbine blades, Wind Energy Science Discussions, 2022, 1–25, 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, Progress in Aerospace Sciences, 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 Energy Science, 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, Annual review of statistics and its application, 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, Applied Sciences, 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.

~~Branlard, E., Jonkman, J., Dana, S., and Doubrawa, P.: A digital twin based on OpenFAST linearizations for real-time load and fatigue estimation of land-based turbines, J. Phys. Conf. Ser., 1618, 022 030, , 2020.~~

Breiman, L.: Random forests, Machine learning, 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, 071 002, 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, 113 254, 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. Soft., 22, 1509–1518, https://doi.org/10.1016/j.envsoft.2006.10.004, 2007.

~~Cardaun, M., Roscher, B., Schelenz, R., and Jacobs, G.: Analysis of wind-turbine main bearing loads due to constant yaw misalignments over a 20 years timespan, Energies, 12, 1768, , 2019.~~

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 Systems, 42, e70 024, 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, 042 045, https://doi.org/doi: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.

De Kooning, J. D., Stockman, K., De Maeyer, J., Jarquin-Laguna, A., and Vandevelde, L.: Digital twins for wind energy conversion systems: a literature review of potential modelling techniques focused on model fidelity and computational load, Processes, 9, 2224, , 2021.

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, pp. 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, Mechanical Systems and Signal Processing, 141, 106 445, 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, Mathematical and Computational Applications, 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://doi.org/https://pmc.ncbi.nlm.nih.gov/articles/PMC7204421/, 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 Sensors Journal, 21, 6914–6921, https://doi.org/10.1109/JSEN.2020.3037237, 2020.

Fiore, G. and Selig, M. S.: Simulation of damage progression on wind turbine blades subject to particle erosion, in: 54th AIAA Aerospace Sciences Meeting, San Diego, California, USA, , 4-6 January 2016, 0813, 2016.

Fuller, A., Fan, Z., Day, C., and Barlow, C.: Digital twin: enabling technologies, challenges and open research, IEEE access, 8, 108 952–108 971, , 2020.

Gaudern, N.: A practical study of the aerodynamic impact of wind turbine blade leading edge erosion, J. Phys. Conf. Ser., 524, 012 031, https://doi.org/10.1088/1742-6596/524/1/012031, 2014.

Gettemy, A.: Classification-of-Leading-Edge-Erosion-Severity-via-Machine-Learning-Surrogate-Models, https://doi.org/10.5281/zenodo.16729170, 2025.

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, Applied Energy, 299, 117 286, 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., pp. 1317–1347, https://doi.org/doi: 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, The R Journal, 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, 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, 100 545, 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://doi.org/https://hdl.handle.net/10735.1/9016, 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, 109 382, 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, Journal of the American Statistical Association, 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, 032 074, 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, 18 155, 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, 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, https://doi.org/10.2514/6.2013-202, 7–10 January 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., Jonkman, B., Damiani, R., Murray, R., et al.: AeroDyn v15 user's guide and theory manual, NREL Draft Report, 46, 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, 112 723, 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.

~~Keegan, M. H., Nash, D., and Stack, M.: On erosion issues associated with the leading edge of wind turbine blades, J. Phys. D Appl. Phys., 46, 383 001, , 2013.~~

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, 113 371, 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.

~~McGugan, M. and Mishnaevsky Jr, L.: Damage mechanism based approach to the structural health monitoring of wind turbine blades, Coatings, 10, 1223, , 2020.~~

Choudhary]menberg2016new ~~Menberg, K., Heo, Y., Augenbroe, G., and Choudhary, R.: New Extension Of Morris Method For Sensitivity Analysis Of Building Energy Models, , 2016.~~

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 Struct., 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, Renewable energy, 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, 111 005, 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.nrel.gov/docs/libraries/wind-docs/inflowwind_manual-pdf.pdf?sfvrsn=691bab12_1, 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.

~~Pryor, S. C., Coburn, J. J., and Barthelmie, R. J.: Spatiotemporal variability in wind turbine blade leading edge erosion, Energies, 18, 425, , 2025.~~

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, pp. 63–71, Springer, 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, 110 886, 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, 052 052, 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, 032 057, 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, Renewable Energy, 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.

~~Saltelli, A.: Sensitivity analysis: Could better methods be used?, J. Geophys. Res.-Atmos., 104, 3789–3793, , 1999.~~

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, 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, 042 001, 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 Energy Science Discussions, 2024, 1–28, 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.

~~Sobol, I. M.: Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Math. Comput. Simulat., 55, 271–280, , 2001.~~

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, 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, pp. 176–189, Springer, 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, 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, Composite structures, 241, 112 096, 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 Energy Science, 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/10.1080/00401706.1992.10485229, 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, 21 020–21 031, 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, International journal of energy research, 40, 2054–2077, https://doi.org/10.1002/er.3580, 2016.