Online Learning of Effective Turbine Wind Speed in Wind Farms

Aoife Henry[1], Michael Sinner[2], Jennifer King[2], Lucy Y. Pao[1]

***Abstract*— To develop better wind farm controllers that can meet more complex objectives, methods of modeling the wind turbine wakes at low computational expense are needed. Gaussian process (GP) regression offers a computationally inexpensive framework for learning complex functions from noisy measurements with very few datapoints. In this work, an online learning approach is presented to learn the rotor-averaged wind velocity at downstream wind turbines with GPs, using the available datastream of wind field measurements and wind turbine control set-points. This framework can readily be integrated into model-based controls methods because the model a) is updated online at low computational expense, b) assumes a mathematically favorable Gaussian form, and c) explicitly quantifies the stochastic nature of the wake field so that the trade-off between exploration and exploitation, and the uncertainty in the prediction, can be utilized. We show that a GP-learned model can match true values with errors within 0.5% on average, with as few as 5 training data points.**

I. INTRODUCTION

Wind energy offers enormous potential for renewable energy generation as part of the clean energy transition necessary to reduce the production of damaging carbon emissions, while fulfilling demand for increasing quantities of affordable and reliable energy. Typically, the purpose of *wind farm control* is to maximize the total power output of a collection of wind turbines, while minimizing their peak mechanical loads. As *wakes* from upstream turbines propagate downstream, they cause lower wind speeds and higher turbulence at downstream turbines, reducing power production and increasing mechanical loads. Given a sufficiently accurate model of the wakes within a wind farm, we can employ yaw misalignment or axial-induction factor control to affect the wake position and shape, and improve farm-level power production [1], [2].

Model-based methods for wind farm control rely on predicting wakes with sufficient accuracy. To date, most wind farm controllers have utilized low-fidelity steady-state flow models, such as FLORIS [3], in an open-loop fashion [2]. Wind farm wake effects are time-varying and stochastic, and improved performance may be achievable using models that account for dynamics and uncertainty in closed-loop control architectures. However, in the context of adaptive model-based controllers, high-fidelity models that can resolve dynamics and turbulence, such as large-eddy simulations (LESs), are too computationally expensive for online use.

An alternative to analytical wake models is to use learning-based surrogate models. Surrogate models are quick to evaluate and can be trained to capture the important dynamic effects. Ashwin et al. train a machine learning (ML) pipeline consisting of deep autoencoders, a neural network (NN) and variational Gaussian processes (GPs) to reconstruct the wake field velocity from measurements [4]. Ti et al. present a NN that learns the wake velocity deficit and added turbulence kinetic energy [5]. The functions learned describe characteristics of the full wake-field structure behind a turbine. Zhang and Zhao develop a controls-oriented, NN-based reduced-order modeling method to predict the future state of unsteady wake fields in wind farms with time-varying turbine yaw angles [6]. Purohit et al. compare the accuracy of three ML algorithms, namely Support Vector Regression, NNs and Extreme Gradient Boosting, in predicting wake characteristics behind a wind turbine relative to analytical models [7].

The large datasets and long training time required, as well as the high fidelity of the full wake-field structure generated by the NN approaches described in [4]–[7], may not be suitable for online control purposes which can rely on only the *rotor-averaged wind velocity* (henceforth *velocity*) at each downstream turbine. Additionally, given that the values of the control actions, yaw angles, and axial induction factors significantly influence the wake field, it is critical that variation in these values is allowed for in a controls context. In [4], [5], both sets of variables are static, and variation in yaw angles is not considered in [7].

Gaussian process regression, or GPR, [8] is a model-free approach to learning unknown functions that is particularly conducive to online learning because of its low computational complexity and good performance with small training datasets. Given a training dataset, it employs Bayesian inference to predict the mean and variance of the target value for a given vector of test inputs. It relies on very few tuning parameters, which can be optimized using the Maximum Likelihood Estimation (MLE) procedure [8].

GPR has been deployed for wind field prediction for control purposes in several previous studies. Nai-Zhi et al. propose a method to find the optimal parameters of an analytical Gaussian wake model for a given wind farm

[1] University of Colorado Boulder, Boulder, CO

[2] National Renewable Energy Laboratory, Golden, CO

This work was authored in part by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency, Renewable Energy Wind Energy Technologies Office and a Palmer Endowed Chair Professorship at the University of Colorado Boulder. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

and range of turbine yaw angles and power outputs, and subsequently train a Random Forest model to predict the optimal parameters of any analytical wake model in real-time [9]. However, it assumes that the analytical model chosen can represent the wake field for some set of parameters. Van der Hoek et al. use a GP to predict the wind direction at each turbine in a farm to improve yaw control, but suggest that online performance could be improved by a more complex approach to data resampling [10]. Moreover, their method does not model the effects of control actions on turbine wakes. In the study that we believe is most similar to the present work, Andersson and Imsland consider updating a low-fidelity analytical model using a GP and using the combined model to acquire optimal control actions [11]. However, the authors point out that wake propagation is not modeled, and the combined model only represents steady-state behavior.

In this work, we learn the correction to a low-fidelity dynamical model to improve on wind speed estimates. This approach is *robust* to unreliability in the GP model in a controls context because, if we cannot depend on the GP error prediction, we can revert to the analytical model. We develop a controls-oriented framework to learn the rotor-averaged wind velocity and its associated uncertainty at downstream wind turbines in a wind farm. This lightweight, online trained model can then be integrated into a model-based controls method, e.g. model predictive control [12].

Our contributions are as follows:

- Implementing a set of GPs that can predict both the mean and variance values of a simplified set of targets (velocity at downstream wind turbines) online from a minimal set of noisy measurements that are available in real wind farm environments.
- Considering the dynamic nature of wake fields by training the model with an auto-regressive input space that includes current *and* delayed measurements. We dynamically select the time-interval between delayed measurements based on the freestream wind speed so that the input vector size remains constant.
- Employing a probabilistic online data selection procedure that uses a replay buffer to maintain a training dataset of limited size that tends to explore regions of high uncertainty.

The remainder of this paper is organized as follows: Sec. II describes the fundamentals of GPR, the analytical wake models employed as the 'base' model from which the GPs learn the model error, our online data collection method, and our performance evaluation metrics. Sec. III describes the case study used to validate the method and evaluates the performance of the learned models for a 3×3 wind farm. Finally, Sec. IV summarizes the concluding remarks and suggestions for further work.

II. METHODOLOGY

A. Gaussian Process Regression

Let $f(\cdot)$ be an unknown function we wish to learn, $\mathcal{T} \coloneqq (\mathbf{X}, \mathbf{y})$ be a training dataset, $\mathbf{X} \in \mathbb{R}^{N_{\text{tr}} \times n}$ be a matrix of training inputs, and $\mathbf{y} \in \mathbb{R}^{N_{\text{tr}}}$ be a vector of noisy function evaluations, or *measurements*, corresponding to each of those inputs. Let $\mathbf{X}_* \in \mathbb{R}^{N_{\text{ts}} \times n}$ be the matrix of test inputs for which we desire function predictions, $\mathbf{f}_*$. A GP [8] is a nonparametric model that is completely specified by its *mean function* μ and *covariance function* or *kernel* $K(\cdot, \cdot)$. It assumes a joint Gaussian distribution between the set of noisy function measurements, $\mathbf{y}$, and predictions, $\mathbf{f}_*$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I & K(\mathbf{X}, \mathbf{X}_*) \\ K(\mathbf{X}_*, \mathbf{X}) & K(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) \tag{1}$$

where we assume that the mean value of the modeled function distribution is $\mathbf{0}$, and that the function evaluations $\mathbf{y}$ have been corrupted by a Gaussian noise with mean 0 and variance σ_n^2.

The covariance kernel determines the correlation between training outputs $\mathbf{y}$ and test predictions $\mathbf{f}_*$ as a function of the input matrices, $\mathbf{X}$ and $\mathbf{X}_*$, and can be tuned with a set of hyperparameters $\boldsymbol{\theta}$. We implement the Squared-Exponential kernel (Eqn. (2a)) and the Matérn kernel (Eqn. (2b)):

$$K(\mathbf{x_i}, \mathbf{x_j}) = \sigma_f^2 \exp\left(-\frac{d_{i,j}^2}{2l^2}\right) + \sigma_n^2 \delta_{i,j} \tag{2a}$$

$$K(\mathbf{x_i}, \mathbf{x_j}) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d_{i,j}}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} d_{i,j}}{l}\right) + \sigma_n^2 \delta_{i,j} \tag{2b}$$

where $d_{i,j} = ||\mathbf{x_i} - \mathbf{x_j}||$ is the Euclidean distance between the ith and jth input points, σ_f^2 is the signal variance, l is the length-scale parameter, ν is the smoothness parameter, $\delta_{i,j}$ is the dirac-delta function, K_ν is a modified Bessel function, and $\Gamma(\nu)$ is the gamma function. The hyperparameters of each covariance kernel, $\boldsymbol{\theta}$, are $[\sigma_f^2, \sigma_n^2, l]$ in the case of (2a) and $[\sigma_f^2, \sigma_n^2, l, \nu]$ in the case of (2b). These hyperparameters must be tuned so that they do not overfit the training inputs and do generalize well on the test inputs. To this end, we employ the commonly used MLE procedure to optimize $\boldsymbol{\theta}$ for a given training dataset $\mathcal{T}$ [8].

Let the notation $\mathbf{f}_* \sim \mathcal{GP}(\mathbf{X}_*)$ be a distribution of function predictions, $\mathbf{f}_*$, at test inputs $\mathbf{X}_*$, determined by a GP. We can generate the *posterior* mean (3a) and variance (3b) of this GP-learned distribution, $\mathbf{f}_*$:

$$\mu[\mathbf{f}_*] = K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} \mathbf{y} \tag{3a}$$

$$\text{var}[\mathbf{f}_*] = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}_*) \tag{3b}$$

In this work, our objective is to train a GP regressor, $\mathcal{GP}_d$, for each of D downstream wind turbines of indices $d \in \mathcal{D}$, to predict the error distribution in the velocity, $\delta U_{d*}(k)$, between the value generated by a *base model*, $U_d^b(k)$, and the true value at time-step k for test inputs $\mathbf{X}_{d*}(k)$. Each error distribution is characterized by the predicted mean, $\mu[\delta U_{d*}(k))]$, and variance, $\text{var}[\delta U_{d*}(k)]$, as computed by Eqns. (3). The resulting $\mathcal{GP}_d$-learned distribution of the

velocity is then given by:

$$\hat{U}_d(k) = U_d^b(k) + \delta U_{d*}(k) \tag{4}$$

with predicted mean, $\mu[\hat{U}_d(k)] = U_d^b(k) + \mu[\delta U_{d*}(k)]$, and variance, $\text{var}[\hat{U}_d(k)] = \text{var}[\delta U_{d*}(k)]$.

Let $\mathcal{U}_d \subseteq D \setminus d$ be the set of indices corresponding to the upstream turbines relative to turbine d, $\mathbf{x}_{u_d}(k_j) = [a_{u_d}(k_j)\, \gamma_{u_d}(k_j)\, U_{u_d}(k_j)]^T$ be the vector of axial induction factor, yaw angle, and velocity at turbine $u_d \in \mathcal{U}_d$, and $\phi_\infty(k_j)$ be the freestream wind direction at time-step k_j, at which a set of measurements is received $\forall u_d \in \mathcal{U}_d$. To allow for the wake propagation delay, the corresponding jth training input for $\mathcal{GP}_d$, $\mathbf{X}_d(k_j)$, consists of those current *and* delayed inputs relative to the time-step of measurement k_j:

$$\mathbf{X}_d(k_j) = \begin{bmatrix} \bigcup\limits_{u_d \in \mathcal{U}_d} \begin{pmatrix} \mathbf{x}_{u_d}(k_j - k_{\text{delay}}) \\ \mathbf{x}_{u_d}(k_j - k_{\text{delay}} + 1) \\ \vdots \\ \mathbf{x}_{u_d}(k_j) \end{pmatrix} \\ \phi_\infty(k_j - k_{\text{delay}}) \\ \phi_\infty(k_j - k_{\text{delay}} + 1) \\ \vdots \\ \phi_\infty(k_j) \end{bmatrix} \tag{5}$$

$$\mathbf{X}_d = \begin{bmatrix} \mathbf{X}_d(k_1)^T \\ \vdots \\ \mathbf{X}_d(k_{N_{\text{tr}}})^T \end{bmatrix}, \quad \tilde{\mathbf{X}}_d := \frac{\mathbf{X}_d - \underline{\mathbf{x}}}{\overline{\mathbf{x}} - \underline{\mathbf{x}}}$$

where $\mathbf{X}_d$ is the full training input dataset for $\mathcal{GP}_d$, including N_{tr} datapoints corresponding to (not necessarily sequential) time-steps $k_1, k_2, \ldots, k_{N_{\text{tr}}}$; $\tilde{\mathbf{X}}_d$ is the normalized equivalent of $\mathbf{X}_d$ (which helps with conditioning); and $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are the expected minimum and maximum values of each dimension in the input space, respectively. Test inputs used for predictions, $\tilde{\mathbf{X}}_{d*}$, are similarly generated. Note that our approach bypasses the assumption of independence between inputs in the GPR, because the rotor-averaged velocity measurements of multiple turbines at different streamwise locations are correlated.

To account for the wake propagation time-delay, we include upstream measurements at previous time-steps in the training dataset. Taylor's frozen hypothesis [13], [14] implies that the wake generated by an upstream wind turbine will propagate downstream at the freestream wind velocity. We estimate the *maximum influential delay time* as double this propagation time from the most upstream turbine in $\mathcal{U}_d$ to downstream turbine d as:

$$\Delta k_d = \frac{2}{\Delta t_l} \frac{\max_{u_d \in \mathcal{U}_d} |x_d - x_{u_d}|}{U_\infty(k)} \tag{6}$$

where Δt_l [s] is the learning sampling time interval, $|x_d - x_{u_d}|$ [m] is the absolute downstream distance between downstream turbine d and upstream turbine u_d, and $U_\infty(k)$ [m/s] is the current freestream wind speed. We then divide the *delayed time-horizon*, $[k - \Delta k_d, k]$, into k_{delay} equal segments and include the delayed measurements at their boundaries. In this way, the input vector has a constant size while including the most influential previous measurements, regardless of the freestream wind speed.

The jth training output for $\mathcal{GP}_d$ is the error between that turbine's velocity at the time-step k_j and the value generated by the base model:

$$\mathbf{y}_d(k_j) = U_d^m(k_j) - U_d^b(k_j), \quad \mathbf{y}_d = \begin{bmatrix} \mathbf{y}_d(k_1) \\ \vdots \\ \mathbf{y}_d(k_{N_{\text{tr}}}) \end{bmatrix} \tag{7}$$

where $U_d^m(k_j)$ is the measured velocity at downstream turbine d at time-step k_j, $U_d^b(k_j)$ is the corresponding value generated by the base model, and $\mathbf{y}_d$ is the full training output dataset for $\mathcal{GP}_d$. See Fig. 1 for a visual representation of this modeling structure.

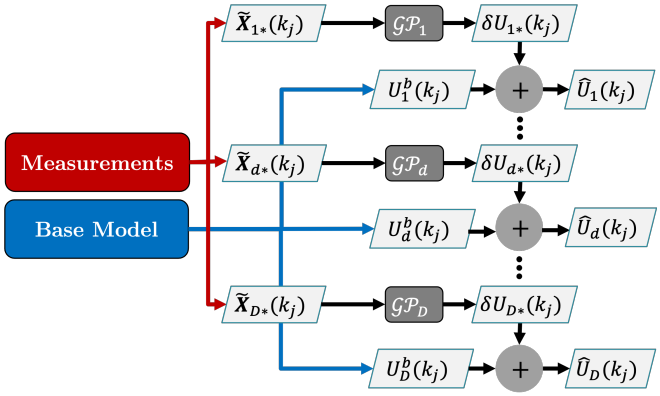

Fig. 1: Computation of GP-predicted rotor-averaged velocity at downstream wind turbines $d \in \mathcal{D}$

B. Base Wake Model

As outlined above, we train GPs to learn the error in the predictions of a 'base' analytical model for the velocity at each downstream turbine $d \in \mathcal{D}$. Typically, analytical models are a function, g, of the current values of the freestream wind speed and direction, and the yaw angles and axial induction factors of all upstream wind turbines relative to turbine d:

$$U_d^b(k + \tau) = g(U_\infty(k), \phi_\infty(k), \{\gamma_{u_d}(k), a_{u_d}(k)\}_{u_d \in \mathcal{U}_d}) \tag{8}$$

where τ is the propagation time as per Taylor's hypothesis.

In this work, we employ a base model that assumes the Jensen velocity deficit model [15], the Gauss wake deflection model [16], [17], the sum-of-squares wake combination model [18], and a propagation time modeled by Taylor's hypothesis [13], [14]. This model is implemented in FLORIS [3]. For brevity, we do not provide the full equations and refer the reader to the relevant citations.

C. Online Data Collection and Learning

At each learning sampling time (separated by Δt_l) we select data generated from measurements collected during operation of the wind farm to add to, and existing ones to remove from, the training dataset. The goal is to reduce the variance in the model predictions over the input space.

Let q be the batch size of datapoints added to each GP training dataset for each *refitting* (tuning of the covariance

kernel hyperparameters and subsequent inversion of the covariance matrix for the purposes of generating predictions as per Eqn. (3)). A batch size of $q > 1$ is chosen to reduce how often the covariance matrix is inverted. We maintain two collections of training inputs: the potential inputs are contained in the *replay buffer*, $\mathcal{R}_d$, and the actual inputs are contained in the training dataset, $\mathcal{T}_d$. Every Δt_l seconds, we check for new data measurements. From these new measurements, we generate potential new training inputs from measurements corresponding to time-steps for which sufficient historic data exists to formulate auto-regressive vectors. We then discard any inputs that are approximate duplicates of points already contained in $\mathcal{R}_d$ or $\mathcal{T}_d$. If q such unique and novel training inputs are available, they are added to $\mathcal{R}_d$. We randomly select q inputs to transfer from $\mathcal{T}_d$ to $\mathcal{R}_d$, to make space for q potentially different inputs. $\mathcal{GP}_d$ is then refitted with the reduced training dataset $\mathcal{T}_d^-$ to generate a temporary $\mathcal{GP}_d^-$ for the purposes of computing the predicted variance, $\text{var}[\hat{U}_d(k)](\mathbf{x}) \, \forall \mathbf{x} \in \mathcal{R}_d$, for each of the training inputs in $\mathcal{R}_d$. Finally, we select q new inputs from $\mathcal{R}_d$ from regions of high prediction variance (as approximated by $\mathcal{GP}_d^-$), where the probability of the ith point being selected is based on the *softmax function* of its variance, $\frac{\exp \text{var}[\hat{U}_d(k)](\mathbf{x}_i)}{\sum_{\mathbf{x} \in \mathcal{R}_d} \exp \text{var}[\hat{U}_d(k)](\mathbf{x})}$, to add to $\mathcal{T}_d^-$ and make the updated training dataset $\mathcal{T}_d$ of size N_{tr}. $\mathcal{R}_d$ is then shuffled and truncated to a size of $10N_{\text{tr}}$. The desired effect of choosing new training inputs based on the predicted variance like this is to promote *exploration* of regions of high uncertainty in a probabilistic sense.

D. Evaluating Prediction Quality

We test our algorithm for several parameters: the kernel function, $K(\cdot, \cdot)$, the training dataset size, N_{tr}, the variance of the Gaussian noise added to the true rotor-averaged wind velocities, σ_n^2, the value of k_{delay}, and the batch size, q.

We quantify the performance of the GPs for different sets of these parameters by computing the Root Mean Squared Error, $RMSE_d$, and the coefficient of determination, R_d^2, between the true, $\{U_d(k)\}_{k=1}^T$, and predicted mean, $\{\mu[\hat{U}_d(k)]\}_{k=1}^T$, values over a given time horizon:

$$RMSE_d \coloneqq \sqrt{\frac{1}{T} \sum_{k'=1}^{T} \left(U_d^t(k') - \mu[\hat{U}_d(k')] \right)^2} \tag{9a}$$

$$R_d^2 \coloneqq 1 - \frac{\sum_{k'=1}^{T} \left(U_d^t(k') - \mu[\hat{U}_d(k')] \right)^2}{\sum_{k'=1}^{T} \left(U_d^t(k') - \bar{U}_d^t \right)^2} \tag{9b}$$

where $\bar{U}_d^t$ is the mean of the true values over time.

We define the relative error at time-step k, $\epsilon_d(k)$, as the relative difference between the GP-predicted mean and true values:

$$\epsilon_d(k) \coloneqq \frac{\mu[\hat{U}_d(k)] - U_d^t(k)}{U_d^t(k)} \times 100\% \tag{10}$$

III. Case Study

To evaluate the method presented, we consider a 3×3 wind farm of NREL 5MW 3-bladed turbines with rotor radii of $R = 63$ m, hub-height of 90 m, and rated rotor speed of 12.1 rpm [19], where the rotor hubs are separated by $14R$ in the streamwise direction and $12R$ in the cross-stream direction (see Fig. 2). A GP is trained for each downstream turbine $d \in \mathcal{D} \coloneqq \{T3, \ldots, T8\}$, where we consider the set of upstream turbines, $\mathcal{U}_d$, to be those closest in the cross-stream direction(s) in the immediate upstream row.

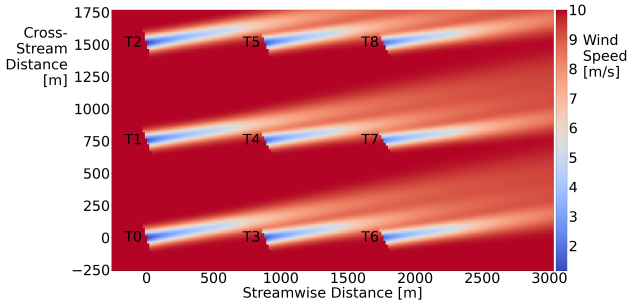

Fig. 2: A 3×3 wind farm layout with wake field for steady freestream wind speed $U_\infty = 10$ m/s and yawed turbines

We generate 500 1-hour wind farm simulations using FLORIS [3], with wake propagation according to Taylor's hypothesis to serve as the "true" data. For the wake, we consider a Gaussian velocity deficit [16], Gauss wake deflection model [16], [17] with the same parameterization as the base model, and the sum-of-squares wake combination model [18]. Note that we are using the GPs to learn the error resulting from the difference in velocity deficit models, because the velocity deflection and wake combination models are equivalent. Gaussian noise of zero mean and standard deviation σ_n (varied for different parameterized cases, see Table I) is added to the velocity measurements. The yaw angles and axial induction factors of the upstream turbines are varied in steps of 5° (between -20° and 20°) and steps of 0.05 (between 0.1 and 0.3), respectively. The freestream wind speed and direction are held constant over each simulation at 8, 10, or 12 m/s and 250°, 260°, or 270°, respectively.

Ten parameterization cases are tested and the resulting median $RMSE$ and R^2 scores over all downstream turbines and all 500 simulations computed (see Table I). The parameterization that results in the least median $RMSE$ value, Case 8, is then selected for the ensuing analysis.

Table I depicts the effect of different learning parameters, where Case 1 corresponds to the baseline set of learning parameters. Lower values of $RMSE$ and values of R^2 closer to 1 correspond to better model fits. We see that increasing the batch size q dramatically reduced the quality of the model fit. It is not clear how higher or lower values of k_{delay} influence the model fit, because increasing it to 8 results in a higher $RMSE$ but also a higher R^2. The choice of σ_n doesn't have a significant effect on the model fit. Reducing N_{tr} improves the model fit, whereas increasing N_{tr} has the opposite effect. Finally, the Matérn kernel results in a better

TABLE I: Scores of the GP-predicted vs. true rotor-averaged wind velocities for different learning parameters

Case	N_{tr}	$K(\cdot,\cdot)$	σ_n	k_{delay}	q	$RMSE$	R^2
1	10	Matérn	0.010	4	1	0.062	0.890
2	10	Matérn	0.010	4	2	0.173	-0.533
3	10	Matérn	0.010	4	4	0.229	-0.934
4	10	Matérn	0.010	2	1	0.061	0.864
5	10	Matérn	0.010	8	1	0.068	0.922
6	10	Matérn	0.001	4	1	0.061	0.895
7	10	Matérn	0.100	4	1	0.063	0.903
8	5	Matérn	0.010	4	1	0.052	0.913
9	20	Matérn	0.010	4	1	0.065	0.884
10	10	SE	0.010	4	1	0.066	0.883

fit than the SE kernel. Overall, the choice of batch size, q, and training dataset size, N_{tr}, are the most influential parameters.

Fig. 3 shows the velocity measurements and GP-predicted mean and variance values for two simulations, A and B, and for the two turbines that experience the greatest wake effect for a southwesterly wind direction, $T7$ and $T8$. The predicted standard deviation is significant in some regions. However, since the prior mean is assumed to be zero, in these areas of high uncertainty the GP models revert to the base model, which is unlikely to deviate drastically from the true values. We can see some abrupt deviations between the predicted and measured values where the green line fluctuates from the red, but relative to the measured values these deviations are small. It is also worth noting that the predicted mean values, which tend to deviate immediately following a step change in the true velocity, seem to converge to the true value while the target value remains constant. It appears that the model generates overestimates following a step increase in the velocity, and generates underestimates following a step decrease. This suggests that the learned error is positive for smaller target values and vice-versa.

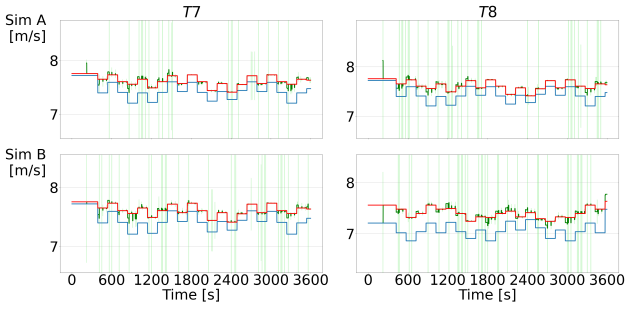

Fig. 3: Rotor-averaged wind velocity time series for simulations A (top row) and B (bottom row) for turbines $T7$ (left column) and $T8$ (right column): True (red line), base model (blue line), GP-predicted mean (dark green line), GP-predicted standard deviation (light green fill)

Fig. 4 illustrates the distribution of the $RMSE_d$ (top) and ϵ_d (bottom) values over all 500 simulations for each downstream turbine in the form of box plots. As can be expected, turbines $T3$ and $T6$ have low $RMSE_d$ and ϵ_d median and inter-quartile range (IQR) values. This is because these turbines are at the lowest cross-stream locations and the further cross-stream the turbine is, the more it will experience the wake effect in southwesterly wind conditions. For the same reason, the $RMSE_d$ median value tends to increase with cross-stream distance. Considering that the units of $RMSE_d$ are in m/s, the median and even the whiskers (which extend from the box by $1.5IQR$) are exceptionally low relative to the mean freestream wind speeds under consideration ($8-12$ m/s). In the worst case, the $T8$ predictions have a maximum $RMSE_d$ of 0.09, but with a median of only 0.03 over all simulations. Looking at the IQR of the $RMSE_d$, we see that it is within 0.05 m/s for all turbines, which is only 0.5% of the average of the wind speeds under investigation (10 m/s).

The bottom row of Fig. 4 shows the box-plot distribution of the relative error, $\epsilon_d(k)$, between the velocity mean values predicted by the GPs and the true values over all simulations for each turbine. Note that the outliers extend above and below the limits of the plot, and we limit the range so that the boxes are visible. The maximum absolute value of ϵ_d over all simulations and turbines is 12% for $T7$ (not visible in Fig. 4), although the median relative error for the same turbine is negligible.

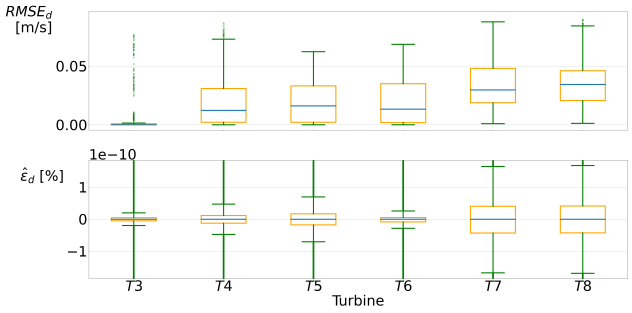

Fig. 4: Median (blue line), $IQR = Q_3 - Q_1$ (orange box), $Q_1 - 1.5IQR$ and $Q_3 + 1.5IQR$ (green whiskers), outliers (green dots above/below whiskers) of the $RMSE_d$ and ϵ_d score over all simulations for each turbine

IV. CONCLUSIONS

In conclusion, we have succeeded in modeling the error between a base wake model and real-time measurements using very few datapoints and computing an associated uncertainty, which can be readily integrated into a stochastic model-based controls algorithm. Of those tested, the best-performing learning parameterization includes a minimal batch size, the Matérn kernel function, and only 5 training inputs. Considering the relative errors over all simulations for this parameterization, the maximum error is greatest for those turbines that experience the greatest wake effect, although the median error is very low. This suggests that if the peak deviations are filtered out of the GP predictions, the model can reliably match the true values for different locations in the wind farm. Refitting the GP for each turbine took approximately 0.04 seconds for this parameterization, which is much less than the typical sampling time of minutes required by a model-based wind farm controller.

Improvements on this study can thus be achieved by

post-processing the GP predictions with smoothing or peak-removal algorithms to attenuate the "noise" in our predictions. The GP predictions tend to deteriorate immediately following a change in the target value, suggesting that the replay buffer is not retaining data that span the full input space. Analysis and comparison with alternative methods for choosing data to increase exploration may improve performance during abrupt changes in effective wind speed. Such changes are, however, unlikely to occur in real wind farms. Other covariance kernel functions, such as those which exploit periodicity in data, may also improve results.

To reduce the size of the input vector and thereby more efficiently refit the GPs, it is desirable to seek a) the minimum number of delayed measurements, k_{delay}, b) the optimal delayed time-horizon, $[k - \Delta k_d, k]$, from which to take these measurements, and c) the minimum set of upstream turbine control actions to consider. It is likely that less frequent delayed yaw angle measurements are necessary as compared with axial induction factor measurements, since the former changes at a significantly slower rate.

In this work we employ GPR as the statistical learning framework, but there are other methods for modeling dynamics and prediction uncertainty in an online fashion worth investigating, e.g. Sparse Identification of Nonlinear Dynamics with Control [20], which is a form of regularized linear regression with nonlinear terms.

In order to better validate our approach, more realistic training data is needed. These tests could consider turbulent wind fields, variation in freestream wind speeds/directions and learn from higher-fidelity data, e.g. LESs.

Extensions to this work include a) integration of the GP-learned wake dynamics with a model-based control algorithm, such as stochastic model-predictive control, and b) learning different characteristics of wake fields relevant to controls, e.g. turbulence intensity, wake propagation direction, downstream propagation time, and wake meandering.

REFERENCES

[1] J. Annoni, P. M. O. Gebraad, A. K. Scholbrock, P. A. Fleming, and J.-W. v. Wingerden, "Analysis of axial-induction-based wind plant control using an engineering and a high-order wind plant model," *Wind Energy*, 19 (6):1135–1150, 2016.

[2] P. Fleming, J. King, K. Dykes, *et al.*, "Initial results from a field campaign of wake steering applied at a commercial wind farm – part 1," *Wind Energy Science*, 4 (2):273–285, 2019.

[3] *FLORIS*, 2019. [Online]. Available: `https://github.com/NREL/floris`.

[4] S. Ashwin Renganathan, R. Maulik, S. Letizia, and G. V. Iungo, "Data-driven wind turbine wake modeling via probabilistic machine learning," *Neural Computing and Applications*, 34 (8):6171–6186, 2022.

[5] Z. Ti, X. W. Deng, and H. Yang, "Wake modeling of wind turbines using machine learning," *Applied Energy*, 257:114025, 2020.

[6] J. Zhang and X. Zhao, "A novel dynamic wind farm wake model based on deep learning," *Applied Energy*, 277:115552, 2020.

[7] S. Purohit, E. Ng, and I. F. S. A. Kabir, "Evaluation of three potential machine learning algorithms for predicting the velocity and turbulence intensity of a wind turbine wake," *Renewable Energy*, 184:405–420, 2022.

[8] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[9] G. Nai-Zhi, Z. Ming-Ming, and L. Bo, "A data-driven analytical model for wind turbine wakes using machine learning method," *Energy Conversion and Management*, 252:115130, 2022.

[10] D. Van Der Hoek, M. Sinner, E. Simley, L. Pao, and J.-W. van Wingerden, "Estimation of the ambient wind field from wind turbine measurements using gaussian process regression," in *2021 American Control Conference (ACC)*, IEEE, 2021, p. 558–563.

[11] L. E. Andersson and L. Imsland, "Real-time optimization of wind farms using modifier adaptation and machine learning," *Wind Energy Science*, 5 (3):885–896, 2020.

[12] C. R. Shapiro, P. Bauweraerts, J. Meyers, C. Meneveau, and D. F. Gayme, "Model-based receding horizon control of wind farms for secondary frequency regulation," *Wind Energy*, 20 (7):1261–1275, 2017.

[13] G. I. Taylor, "The spectrum of turbulence," *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, 164 (919):476–490, 1938.

[14] C. J. Bay, J. Annoni, T. Taylor, L. Pao, and K. Johnson, "Active power control for wind farms using distributed model predictive control and nearest neighbor communication," in *American Control Conference (ACC)*, IEEE, 2018, p. 682–687.

[15] N. O. Jensen, *A note on wind generator interaction*. Citeseer, 1983, vol. 2411.

[16] J. King, P. Fleming, R. King, *et al.*, "Control-oriented model for secondary effects of wake steering," *Wind Energy Science*, 6 (3):701–714, 2021.

[17] M. Bastankhah and F. Porté-Agel, "Experimental and theoretical study of wind turbine wakes in yawed conditions," *J. Fluid Mechanics*, 806:506–541, 2016.

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

[19] J. Jonkman, S. Butterfield, W. Musial, and G. Scott, "Definition of a 5-MW reference wind turbine for offshore system development," NREL, Golden, CO, Tech. Rep., 2009.

[20] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (SINDyC)," *IFAC-PapersOnLine*, 49 (18):710–715, 2016.