A Gaussian Process Based Approach to Estimate Wind Speed Using SCADA Measurements from a Wind Turbine

Eduardo B. R. F. Paiva [*,**] **Hoai-Nam Nguyen** [*]
Olivier Lepreux [*] **Delphine Bresch-Pietri** [**]

[*] *IFP Energies nouvelles, Solaize, France (e-mail: {eduardo.bezerra-rufino-ferreira-paiva, hoai-nam.nguyen, olivier.lepreux}@ifpen.fr).*
[**] *MINES ParisTech, Paris, France (e-mail: delphine.bresch-pietri@mines-paristech.fr)*

Abstract: In this paper, we propose a method for estimating in real-time the speed of the wind to which a turbine is subjected using its SCADA (Supervisory Control And Data Acquisition) measurements. The approach is fully data-driven. It is based on Gaussian Process Regression. We use real experimental SCADA data from an operating commercial 3-bladed horizontal axis wind turbine. The reference values for the wind speed are obtained from a nacelle LiDAR (Light Distancing and Ranging) sensor. The comparison of the obtained estimation results with the measurements provided by the LiDAR sensor emphasizes the performance of the proposed method and underlines its interests for control purposes. Assessing performance on a day of operation, we obtain median errors of less than 1%. A numerical comparison with a more traditional model-based approach is also provided.

Keywords: Wind Speed Estimation; Wind Energy; Machine Learning; Gaussian Process Regression; Real-time

1. INTRODUCTION

In recent years, the pursuit of clean energy has led to massive investments in renewable sources. In particular, wind energy has received a lot of attention and is now a substantial part of the electrical supply in many countries. According to Lee and Zhao (2020), the worldwide installed capacity has surpassed 650 GW in 2019, when the two biggest players, China and the USA, have reached an installed capacity of about 236 GW and 105 GW, respectively.

In this context, control and estimation techniques for wind turbines have witnessed a significant expansion. The main goals are usually power production maximization and load-mitigation (to increase the life-span of the turbines), as well as diagnostics, both at the turbine scale and at the farm scale. Many of these techniques require knowledge of the wind speed, which can be obtained by sensors or estimation methods. For a review on wind turbine control based on wind speed estimation, we refer the reader to Jena and Rajendran (2015).

A common approach in a model-based setting is to use a Kalman Filter to estimate the aerodynamic torque on the rotor and then solve the nonlinear equation relating this torque and the wind speed. This is the case, e.g., in (Boukhezzar and Siguerdidjane, 2011; Nam et al., 2011). Some other model-based approaches are presented by Østergaard et al. (2007); Sung et al. (2011); Song et al. (2017a,b). These estimation strategies have the drawback of requiring knowledge of the torque coefficient map of the turbine, which might not be available. Another trend of research uses machine Learning solutions, often combined with model-based techniques, to estimate the wind speed, e.g., (Yang et al., 2006; Jaramillo-Lopez et al., 2016; Deng et al., 2019; Deng et al., 2020; Sierra-García and Santos, 2021). These works exhibit interesting capabilities. Hence, we propose to follow this trend.

In this paper, we present an estimation method based on Gaussian process regression to obtain the wind speed value using only SCADA (Supervisory Control and Data Acquisition) measurements that are commonly available in wind turbines. Namely, we use the active power, the rotor speed, and the pitch angle of the blades. The algorithm works by learning a map from the recent history of these SCADA measurements to the wind speed and then using this map to compute the wind speed for the query inputs.

For training, i.e., for learning this map, we need the wind speed reference values which we obtain from a LiDAR (Light Detection and Ranging) sensor mounted on the turbine nacelle. Our goal is thus to have an algorithm that achieves wind speed estimates as close as possible to what would be measured using the LiDAR sensor. The point is that, after training, one could consider removing the LiDAR and using the algorithm instead. This would reduce the maintenance cost associated with the wind turbine. Also, one could use the same training for several turbines of the same model under similar conditions (e.g., all the front row turbines in the same wind farm, provided the terrain

conditions are the same for all of them). The method we propose is fully data-driven, requiring no physical model of the turbine. In particular, the torque coefficient map is not needed. This map is not always available or may be known only approximately. Furthermore, we obtain our results using real turbine data from the field instead of a simulator, which is not the case for most studies available in the literature.

The rest of this paper is organized as follows. In Section 2, we make the formal problem statement. In Section 3, we present a short review of the basic theory behind Gaussian process regression. In Section 4, we explain how we apply Gaussian process regression to our problem of wind speed estimation. In Section 5, we present some results to show the performance of the method. Finally, in Section 6, we present some concluding remarks.

Notation

Let $\boldsymbol{M}$ be a matrix, then $\boldsymbol{M}^{-1}$ is its inverse and $\boldsymbol{M}^T$ is its transpose. We write $||\boldsymbol{v}||_2$ to denote the Euclidean norm of the vector $\boldsymbol{v} \in \mathbb{R}^n$, $n \in \mathbb{N}$. $\boldsymbol{v}|\boldsymbol{M} \sim \mathcal{N}(\boldsymbol{a}, \boldsymbol{A})$ denotes that $\boldsymbol{v}$ conditioned on $\boldsymbol{M}$ has a Gaussian distribution with mean $\boldsymbol{a}$ and covariance $\boldsymbol{A}$.

2. PROBLEM STATEMENT

In this section, we make a formal presentation of the research problem faced in this paper.

Consider a wind turbine for which measurements of the rotor angular speed, the active power, and the pitch angle are available at each sampling time and let $r \in \mathbb{N}$ be a fixed number chosen a priori. Our goal is to design an algorithm that, given a time sequence of these measurements from time $t - r\Delta t$ to time t, produces an estimate of the wind speed at time t, where Δt is the sampling period. This estimate should be as close as possible to what would be obtained with a LiDAR sensor mounted on the turbine nacelle. Notice that the time sequence under consideration is discrete.

Let ω_t, p_t, β_t, and v_t represent, respectively, the rotor angular speed, the active power, the pitch angle (assumed to be the same for all blades), and the wind speed at time t for a given wind turbine. As already mentioned, the values of ω_t, p_t, and β_t come from the SCADA system of the turbine and the values of v_t needed for training are obtained using a LiDAR sensor.

Our problem is to estimate v_t given

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{\omega}_{t-r:t} \\ \boldsymbol{\beta}_{t-r:t} \\ \boldsymbol{p}_{t-r:t} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{\omega}_{t-r:t} = [\omega_{t-r}, \ldots, \omega_{t-1}, \omega_t]^T$, and similar definitions hold for $\boldsymbol{p}_{t-r:t}$ and $\boldsymbol{\beta}_{t-r:t}$. $\boldsymbol{\omega}_{t-r:t}$, $\boldsymbol{\beta}_{t-r:t}$, and $\boldsymbol{p}_{t-r:t}$ are time-series of SCADA measurements. The dimension of the input $\boldsymbol{x}_t$ is $D = 3(r+1)$. For simplicity, the notation assumes $\Delta t = 1$.

3. PRELIMINARIES ON GAUSSIAN PROCESS REGRESSION

Definition 1. A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution (Williams and Rasmussen, 2006, Definition 2.1).

Let f be a process from $\mathbb{R}^D$ to $\mathbb{R}$, with $D \in \mathbb{N}$. Define its mean function $m : \mathbb{R}^D \mapsto \mathbb{R}$ and its covariance function (or kernel) $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$, such that

$$\begin{aligned} m(\boldsymbol{x}) &= \mathcal{E}[f(\boldsymbol{x})], \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathcal{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))], \end{aligned} \tag{2}$$

where $\mathcal{E}$ denotes mathematical expectation and $\boldsymbol{x}$ and $\boldsymbol{x}'$ are points in $\mathbb{R}^D$. We write $f \sim \mathcal{GP}(m, k)$ to denote f is a Gaussian process with mean m and covariance k. The random variables are the values outputted by f, hence, given an input location $\boldsymbol{x} \in \mathbb{R}^D$, we have $f(\boldsymbol{x})|\boldsymbol{x} \sim \mathcal{N}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))$. Similarly, for N input locations $\boldsymbol{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$, the values outputted by f for these inputs have a joint Gaussian distribution,

$$\begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix} |\boldsymbol{X} \sim \mathcal{N}\left(m(\boldsymbol{X}), k(\boldsymbol{X}, \boldsymbol{X})\right), \tag{3}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times D}$, $m(\boldsymbol{X})$ is a vector in $\mathbb{R}^N$ such that its ith element is $m(\boldsymbol{x}_i)$, and $k(\boldsymbol{X}, \boldsymbol{X})$ is a matrix in $\mathbb{R}^{N \times N}$ such that its element at position (i, j) is $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

To use Gaussian Process Regression (GPR), we first assume a certain measured variable y_i is related to the input $\boldsymbol{x}_i$ by

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \tag{4}$$

with $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ and $f \sim \mathcal{GP}(m, k)$ for some suitable choice of m and k. Thus, for N training pairs $(\boldsymbol{x}_1, y_1)$, …, $(\boldsymbol{x}_N, y_N)$, we have

$$\boldsymbol{y}|\boldsymbol{X} \sim \mathcal{N}(m(\boldsymbol{X}), \boldsymbol{K}_y), \tag{5}$$

where $\boldsymbol{y} = [y_1, \ldots, y_N]^T$, $\boldsymbol{K}_y = k(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \boldsymbol{I}$, and $\boldsymbol{I}$ is the identity matrix of adequate dimensions.

Next, consider test input locations $\boldsymbol{x}_{*i}$, $i = 1, \ldots, N_*$, for which one wants to estimate the corresponding values of f. Let $\boldsymbol{f}_* = [f(\boldsymbol{x}_{*1}), \ldots, f(\boldsymbol{x}_{*N_*})]^T$ and $\boldsymbol{X}_* = [\boldsymbol{x}_{*1}, \ldots, \boldsymbol{x}_{*N_*}]^T$. Then, from the joint Gaussian distribution property of the Gaussian process,

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} |\boldsymbol{X}, \boldsymbol{X_*} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_y & k(\boldsymbol{X}, \boldsymbol{X}_*) \\ k(\boldsymbol{X}_*, \boldsymbol{X}) & k(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix} \right), \tag{6}$$

where $m(\boldsymbol{X}_*) = [m(\boldsymbol{x}_{*1}), \ldots, m(\boldsymbol{x}_{*N_*})]^T$, $k(\boldsymbol{X}, \boldsymbol{X}_*) = [k(\boldsymbol{X}_*, \boldsymbol{X})]^T$ is a matrix in $\mathbb{R}^{N \times N_*}$ such that its element at position (i, j) is $k(\boldsymbol{x}_i, \boldsymbol{x}_{*j})$, and $k(\boldsymbol{X}_*, \boldsymbol{X}_*)$ is a matrix in $\mathbb{R}^{N_* \times N_*}$ such that its element at position (i, j) is $k(\boldsymbol{x}_{*i}, \boldsymbol{x}_{*j})$.

Using a property of joint Gaussian distributions (see, e.g., (Williams and Rasmussen, 2006, Eq. (A.6))), we are able to compute the distribution conditioned on $\boldsymbol{y}$:

$$\boldsymbol{f}_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_* \sim \mathcal{N}(\mu(\boldsymbol{X}_*), \Sigma(\boldsymbol{X}_*, \boldsymbol{X}_*)), \tag{7}$$

where

$$\begin{aligned} \mu(\boldsymbol{X}_*) &= m(\boldsymbol{X}_*) + k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}_y^{-1}\left(\boldsymbol{y} - m(\boldsymbol{X})\right), \\ \Sigma(\boldsymbol{X}_*, \boldsymbol{X}_*) &= k(\boldsymbol{X}_*, \boldsymbol{X}_*) - k(\boldsymbol{X}_*, \boldsymbol{X})\boldsymbol{K}_y^{-1}k(\boldsymbol{X}, \boldsymbol{X}_*). \end{aligned} \tag{8}$$

We can use $\mu(\boldsymbol{X}_*)$ as an estimate for $\boldsymbol{f}_*$ and $\Sigma(\boldsymbol{X}_*, \boldsymbol{X}_*)$ to measure the uncertainty in these estimations. We can understand GPR as a Bayesian method in which we start with a prior $\mathcal{GP}(m, k)$ for f and, after observing the

training data, we find a posterior distribution: $f|\boldsymbol{X}, \boldsymbol{y} \sim \mathcal{GP}(\mu, \Sigma)$.

The kernel k is chosen so to encode prior beliefs on the latent function f (such as smoothness, periodicity, etc) and is often parameterized by the so called hyper-parameters $\boldsymbol{\theta}$. The requirements on k are that it is symmetric ($k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$) and that it is positive semidefinite in the sense that all the eigenvalues of $k(\boldsymbol{X}, \boldsymbol{X})$ are non-negative for all $\boldsymbol{X}$ for any choice of N (see (Williams and Rasmussen, 2006, Chapter 4)).

For instance, a common choice of kernel is the squared exponential kernel with automatic relevance determination parameters, which is given by

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{x} - \boldsymbol{x}')\right), \quad (9)$$

with $\boldsymbol{\Lambda} = \text{diag}\{l_1^2, \ldots, l_D^2\}$ (recall D is the dimension of the inputs). In this case, the hyper-parameters are $l_1, \ldots, l_D$, and σ_f. The standard deviation of the noise, σ_ϵ, is usually not known in advance and can be itself considered as a hyper-parameter.

A popular way to set the hyper-parameters is to maximize the log marginal likelihood

$$\ln(p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\ln(\det(\boldsymbol{K}_y)) - \frac{N}{2}\ln(2\pi). \quad (10)$$

The first term on the right-hand side of (10) is related to the fitting of the training data, the second is a model complexity penalty, and the third is a normalization constant.

For more on Gaussian process, refer to Williams and Rasmussen (2006); Schulz et al. (2018) and the references therein.

4. GAUSSIAN PROCESS REGRESSION BASED WIND SPEED ESTIMATION

In this section, we explain the procedure we propose to obtain an estimation for the wind speed at time t using a sequence of SCADA measurements from time $t - r\Delta t$ to time t, $r \in \mathbb{N}$. Recall that time is discrete in the setup considered here.

We start with the data collected from the turbine SCADA system and the LiDAR and we define the pairs $(\boldsymbol{x}_t, y_t)$, with $y_t = v_t$ and $\boldsymbol{x}_t$ a sequence of SCADA measurements as defined in (1).

In the following subsections, we describe the algorithm to perform the estimation.

4.1 Clustering

The first step of the training process is to cluster the data. Indeed, since we expect the turbine to have different behaviors under different operating conditions, it seems reasonable to look for different estimations depending on the operating range.

Remember each input point $\boldsymbol{x}_i$ in our training set is a D-dimensional vector (with $D = 3(r+1)$) containing measurements of rotational speed, pitch angle, and active power. Since these measurements have different scales, we first standardize the training data. In other words, we compute the mean and the standard deviation of each component of the input points across the training set, and then from each component we subtract the corresponding mean and then divide the result by the corresponding standard deviation.

The second step is to use Principal Component Analysis (PCA) (Wold et al., 1987) to make a change of coordinates. Note that PCA is not used for dimensional reduction here, but only to design the clustering.

Finally, we use the k-means++ algorithm to split the data into clusters (Arthur and Vassilvitskii, 2006).

4.2 Finding an Estimate per Cluster

In this subsection, we treat each cluster separately. Here, we show how to compute an estimate for the wind speed of a given test point using data from one cluster only.

Kernel and Mean Function To use GPR we need to define a kernel k and a mean function m that encode our prior beliefs on the latent function that relates the inputs to the outputs. We use a kernel of the form

$$\begin{aligned} k(\boldsymbol{x}_a, \boldsymbol{x}_b) = {} & k_\omega(\boldsymbol{\omega}_{a-r:a}, \boldsymbol{\omega}_{b-r:b}) \\ & + k_\beta(\boldsymbol{\beta}_{a-r:a}, \boldsymbol{\beta}_{b-r:b}) \\ & + k_p(\boldsymbol{p}_{a-r:a}, \boldsymbol{p}_{b-r:b}), \end{aligned} \quad (11)$$

where $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are (possibly different) inputs and

$$\begin{aligned} k_\omega(\boldsymbol{z}, \boldsymbol{z}') &= \boldsymbol{z}^T \boldsymbol{L}_1^{-1} \boldsymbol{z}' \\ &\quad + \sigma_1^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}')^T \boldsymbol{L}_2^{-1}(\boldsymbol{z} - \boldsymbol{z}')\right), \\ k_\beta(\boldsymbol{z}, \boldsymbol{z}') &= \sigma_2^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}')^T \boldsymbol{L}_3^{-1}(\boldsymbol{z} - \boldsymbol{z}')\right), \\ k_p(\boldsymbol{z}, \boldsymbol{z}') &= \boldsymbol{z}^T \boldsymbol{L}_4^{-1} \boldsymbol{z}' \\ &\quad + \sigma_3^2 \exp\left(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}')^T \boldsymbol{L}_5^{-1}(\boldsymbol{z} - \boldsymbol{z}')\right), \\ \boldsymbol{L}_i &= \text{diag}\{((r+1)l_i)^2, \ldots, (2l_i)^2, l_i^2\}, \end{aligned} \quad (12)$$

with $\boldsymbol{z}$ and $\boldsymbol{z}'$ replacing the corresponding arguments for k_ω, k_β, and k_p used in (11) (due to lack of space). Hence, the hyper-parameters for the kernel in (11) are l_1, ..., l_5, σ_1, σ_2, and σ_3. Notice this is a valid kernel since it is a sum of valid kernels (Williams and Rasmussen, 2006). The ideas behind this kernel are:

(1) the linear contributions (i.e., those of the form $\boldsymbol{z}^T \boldsymbol{L}_i \boldsymbol{z}'$) account for the fact that we expect the wind speed to be higher when the turbine is producing more power and the rotor speed is faster;
(2) the nonlinear contributions are in the form of the kernel (9), which is a more generic form and should account for the more intricate dynamics of the turbine;
(3) the matrices $\boldsymbol{L}_i$ are as presented in (12) to yield smaller weights for the input components containing information further in the past while maintaining a small number of hyper-parameters to be selected.

For the mean function we use a constant function equal to the mean of the wind speed for the training points in the cluster. Also, since we consider the reference wind speed to be the ground truth, we use $\sigma_\epsilon = 0$, so $\boldsymbol{K}_y = k(\boldsymbol{X}, \boldsymbol{X})$.

Training and Computing Estimates We select N pairs $(\boldsymbol{x}_i, y_i)$ from the cluster to use in the following computations. Here we use the index i instead of t because these training pairs are randomly chosen from the available data, so that, e.g., y_2 does not necessarily correspond to wind speed measurement obtained in the sample time right after the one corresponding to y_1. However, the measurements that constitute a particular $\boldsymbol{x}_i$ are ordered. Thus, here, and in the sequel, one should understand that the pair $(\boldsymbol{x}_i, y_i)$ corresponds not to time $t = i$, but to some time t_i.

After selecting the training pairs, the next step is to tune the hyper-parameters by maximizing the marginal likelihood (10). Notice that, for different values of r, the training pairs are different and so might be the hyper-parameters. Hence, we can effectively have several different models for the relation between the SCADA measurements and the wind speed. However, once the training data is prepared and the hyper-parameters are set, the procedure to obtain the estimates is the same regardless of the choice of r.

Recall that, for a test point $\boldsymbol{x}_*$, we would normally find the estimate

$$\hat{y}_* = m(\boldsymbol{x}_*) + k(\boldsymbol{x}_*, \boldsymbol{X})\boldsymbol{K}_y^{-1}\left(\boldsymbol{y} - m(\boldsymbol{X})\right). \tag{13}$$

However, real-time computation of $\hat{y}_*$ using (13) leads to a numerical burden since the time to compute $k(\boldsymbol{x}_*, \boldsymbol{X})$ scales with the size N of the training set Thus, it is desired to have a large training set to learn the map well from SCADA measurements to wind speed, but a small one to perform estimations in real-time. In order to solve this dilemma, we follow the method proposed by Mayer et al. (2020), which consists in using the following approximation

$$\hat{y}_* \approx m(\boldsymbol{x}_*) + k(\boldsymbol{x}_*, \bar{\boldsymbol{X}})\bar{\boldsymbol{\alpha}}, \tag{14}$$

where $\bar{\boldsymbol{\alpha}}$ is found as the solution of a regularized least squares (LS) problem,

$$\begin{aligned}\bar{\boldsymbol{\alpha}} &= \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^M} ||\left(\boldsymbol{y} - m(\boldsymbol{X})\right) - \bar{\boldsymbol{K}}_y\boldsymbol{\alpha}||_2^2 + \mu||\boldsymbol{\alpha}||_2^2 \\ &= \left(\bar{\boldsymbol{K}}_y^T\bar{\boldsymbol{K}}_y + \mu\boldsymbol{I}\right)^{-1}\bar{\boldsymbol{K}}_y^T\left(\boldsymbol{y} - m(\boldsymbol{X})\right),\end{aligned} \tag{15}$$

$\bar{\boldsymbol{X}}$ is obtained by deleting a number of rows from $\boldsymbol{X}$, $\bar{\boldsymbol{K}}_y$ is obtained by deleting the corresponding columns of $\boldsymbol{K}_y$, $\mu \geq 0$ is a regularization coefficient, and M is the number of rows of $\bar{\boldsymbol{X}}$. We refer to this method as GPR+LS in the following.

Notice $\bar{\boldsymbol{\alpha}}$ depends only on the training set, hence, we can store its value and then compute $k(\boldsymbol{x}_*, \bar{\boldsymbol{X}})$ online for each new test point. Therefore, once $\bar{\boldsymbol{\alpha}}$ is known, the time to compute $\hat{y}_*$ using (14) is $\mathcal{O}(M)$. Furthermore, we found in our numerical experiments that the matrix $\boldsymbol{K}_y$ is often poorly conditioned, which leads to numerical problems in computing the estimates using the exact GPR method.

Fig. 1 depicts a flowchart of the algorithm for one cluster. Notice once $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{\alpha}}$ are known, we can input each test point $\boldsymbol{x}_*$ to the model displayed on the right-hand side of Fig. 1 and obtain an output $\hat{y}_*$. We included a computation of $m(\boldsymbol{x}_*)$ for the figure to be more general, but in our case $m(\boldsymbol{x}_*)$ is the mean wind speed for the cluster, so it is fixed.

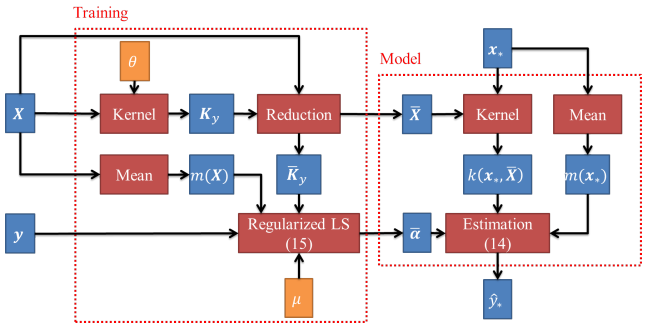

Fig. 1. Schematic of the algorithm for one cluster.

4.3 Computing the Final Estimate

In the last subsection, we performed the regression using each cluster separately. Thus, we can find C output estimates $(\hat{y}_{1,*}, \ldots, \hat{y}_{C,*})$ for the same test point $\boldsymbol{x}_*$, where C is the number of clusters. In this subsection, we discuss how to compute a final value for the estimate.

First, we apply to $\boldsymbol{x}_*$ the same transformations we applied to the training set in Subsection 4.1 (standardization with the training set statistics and PCA) to obtain $\boldsymbol{x}'_*$. Then, we propose two approaches to find the final estimate:

(1) to use only the estimate associated with the cluster whose centroid is closest to $\boldsymbol{x}'_*$;
(2) to compute a weighted average of all the estimates

$$\hat{y}_{\text{final},*} = \frac{\sum_{i=1}^{C} w_i\hat{y}_{i,*}}{\sum_{i=1}^{C} w_i}, \tag{16}$$

where the weights $w_1, \ldots, w_C$ are computed as

$$w_i = \frac{1}{||\boldsymbol{x}'_* - \boldsymbol{c}_i||_2^n}, \tag{17}$$

with $n > 0$ and $\boldsymbol{c}_i$ the centroid of the ith cluster.

Notice that, when using only the nearest cluster, i.e., the first option, one can reduce the computational cost of computing the estimate by first verifying which is the nearest cluster and then computing the estimation only with the model which is associated to it (since the remaining $\hat{y}_{i,*}$'s are not necessary for the final computation in this case). Fig. 2 depicts a diagram for the final estimation computation.

5. NUMERICAL RESULTS FROM EXPERIMENTAL DATA

5.1 Data Description

In this section, we use data corresponding to several days of operation of a 3-bladed horizontal axis wind turbine. This turbine is located at a wind farm in Ablaincourt-Pressoir, France, operated by Engie Green. The SCADA data were provided by Engie Green, and the Lidar data were provided by Leosphere, within the framework of the ANR (French National Research Agency) project "SmartEole". We use measurements of the rotor speed, the pitch angle, the produced power which are obtained from the SCADA system of the turbine. On the other hand, the wind speed is obtained with the LiDAR sensor.

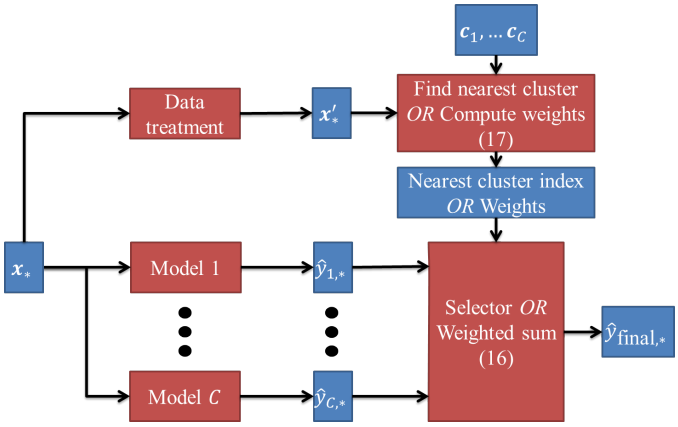

Fig. 2. Diagram for final estimation computation. Notice that, when using only the nearest cluster estimate, we need to compute only the one relevant $y_{i,*}$ for each test point.

The sampling period for the SCADA measurements is 1 second, while the one for the sampling period for the wind speed measurements is 0.25 second. Due to missynchronization, we use a simple linear interpolation to get wind speed at time instants corresponding to the SCADA measurements and use these values as reference.

The LiDAR is used in conjunction with the IFP Energies nouvelles WiSE WindField commercial software to compute the reference wind speed. The main features of the algorithm are i) this is a real time three dimensional wind field reconstruction; ii) the knowledge of past wind field information, obtained from a temporal history of LiDAR measurements, can be incorporated into current and future estimates of the wind field; iii) the blade blocking effect is taken into account. The WiSE WindField algorithm is data-driven and requires no access to the turbine model, in line with the requirements of the current paper. The interested reader may refer to Guillemin et al. (2018); Nguyen and Guillemin (2020) to get more information about the algorithm.

5.2 Performance Measurement

To assess the performance of the proposed methods, we compute the following index (relative error):

$$e_* = \left| \frac{\hat{y} - y_*}{y_*} \right| \times 100\%, \tag{18}$$

where y_* is a reference value and $\hat{y}$ is the corresponding estimate. For each set of tests, we use boxplots of this index to depict its distribution.

5.3 Results

We start with a large set $\mathcal{S}_{\text{train}}$ of training points, split it into C clusters, and compute their centroids $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_C$. Then, for each cluster, we randomly choose N points to build the matrix $\boldsymbol{X}$ and a subset of these with $M < N$ points to build the matrix $\bar{\boldsymbol{X}}$ to perform the regression as discussed in Subsection 4.2. Finally, we select test inputs (SCADA measurements) from another set, $\mathcal{S}_{\text{test}}$, and estimate the corresponding outputs (i.e., the wind speed values) as discussed in Subsection 4.3. The sets $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test}}$ are disjoint. In this section, we present the results for different choices of C, M, and N. We keep $r = 20$ fixed.

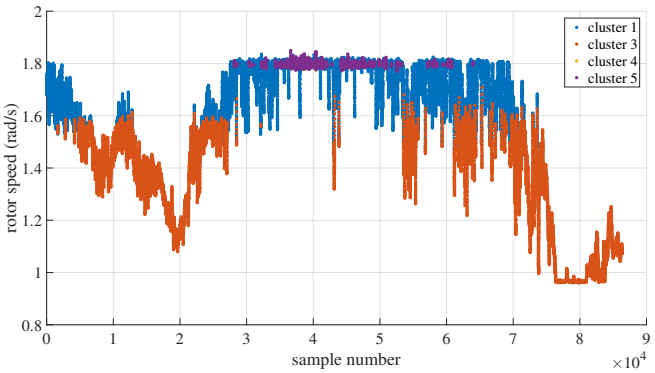

Fig. 3. Rotor speed time-series corresponding to Fig. 6. The points are color coded according to the cluster used for the estimation of the wind speed at the corresponding time.

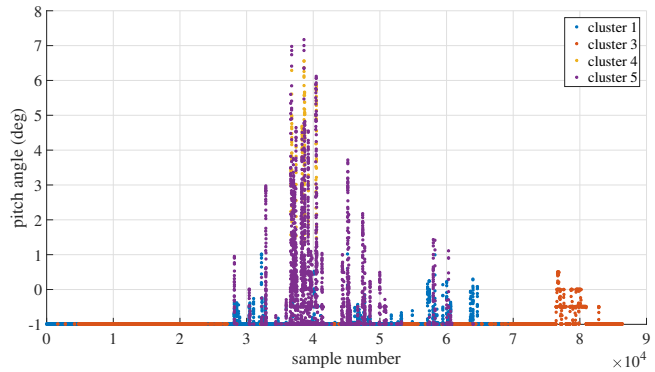

Fig. 4. Pitch angle time-series corresponding to Fig. 6. The points are color coded according to the cluster used for the estimation of the wind speed at the corresponding time.

Simulation of a Real-Time Application To simulate a real-time application, we took consecutive points from a randomly chosen day and estimated the corresponding wind speed values.

Figs. 3, 4, and 5 show, respectively, the rotor speed, the pitch angle, and the produced power time-series corresponding to the wind speed time-series in Fig. 6. The points at time t are color coded according to the cluster used for the estimation of the wind speed at the corresponding time, i.e., the one whose centroid is nearest to the input point at that time. We used 5 clusters in this case, but this particular day happens not to have any input point associated with cluster 2. Note that the selected day covers a relatively large range of wind speeds, from cut-in to above-rated, but very high speeds are absent. It has been verified on other days that the estimation behaves similarly well in this range. Clustering is required to obtain a good behaviour both in the cut-in speed and the above-rated regions.

The estimation results are depicted in Figs. 6 and 7. In this case, the number of clusters is $C = 5$, the time window size is $r = 20$, the number of points per cluster (used offline) is $N = 10^4$, and number of points per cluster used in the

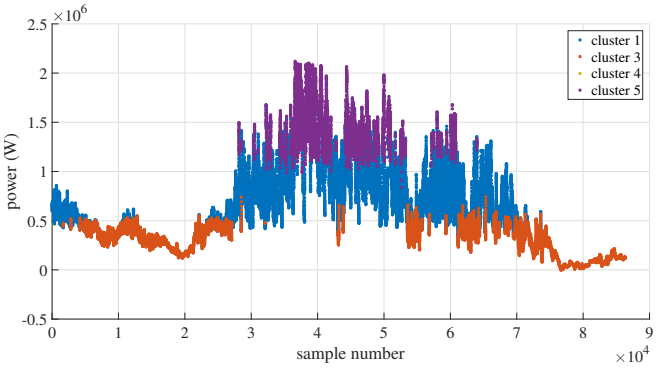

Fig. 5. Produced power time-series corresponding to Fig. 6. The points are color coded according to the cluster used for the estimation of the wind speed at the corresponding time.

online computations is $M = 3 \times 10^3$. We use only the nearest cluster estimate. We observe the result in this case is highly satisfactory for control applications.

Also, in Fig 7, we included the results obtained with a more traditional model based approach. It consists in a two-step strategy. First, the aerodynamic torque on the rotor is estimated using a linear Kalman Filter with the following state-space formulation of the turbine model:

$$\begin{aligned} \dot{\omega} &= \frac{1}{J}T_a - \frac{1}{J}T_g + \xi_1 \\ \dot{T}_a &= \xi_2, \end{aligned} \tag{19}$$

where ω is the rotor speed, T_a is the aerodynamic torque, T_g is the generator torque, ξ_1 and ξ_2 are noise, and J is the inertia of the turbine. Then, this estimation is used to calculate the wind speed by numerically solving the following nonlinear equation:

$$T_a = \frac{1}{2}\rho\pi R^3 C_q\left(\beta, \frac{\omega R}{v}\right)v^2, \tag{20}$$

where ρ is the air density, R is the rotor radius, C_q is the torque coefficient map, β is the blades pitch angle, and v is the wind speed. This is similar to, e.g., what Nam et al. (2011) used to estimate the wind speed.

The error comparison is presented formally in Fig. 8. Notice our proposed approach has comparable performance to that of the Kalman Filter approach and, most importantly, requires no access to the turbine model. Furthermore, the Kalman Filter requires manual tuning of the parameters. In this case, we carefully tuned the Kalman Filter using part of our data before testing it with the data of the day presented in Figs. 6 and 7.

Performance Assessment on Data from Several Days Here, we show the effects of varying the parameters C (number of clusters), M (number of points per cluster used in the online computation), and N (numbers of points per cluster) on the performance of our method. We do it by inputting test points collected from several different days of operation of the turbine to estimate the corresponding wind speed and then plotting the corresponding boxplots to show how the relative error distribution changes according to the changes in these parameters.

First, we choose $N = 10^4$, $M = 3000$, and $C = 5$ to present the effect of combining the estimates from each cluster

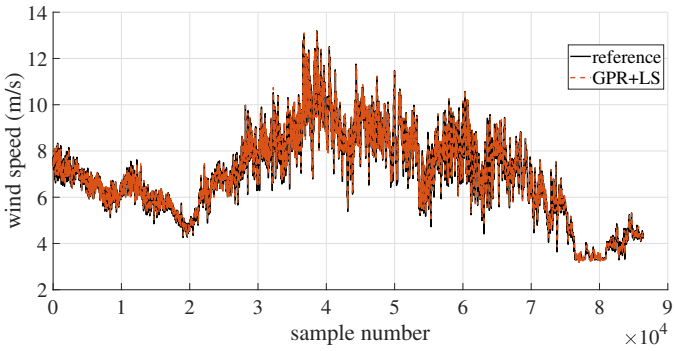

Fig. 6. Estimation for consecutive points to simulate a real-time application. The parameters are $C = 5$, $r = 20$, $N = 10^4$, and $M = 3 \times 10^3$. We use only the nearest cluster estimate.

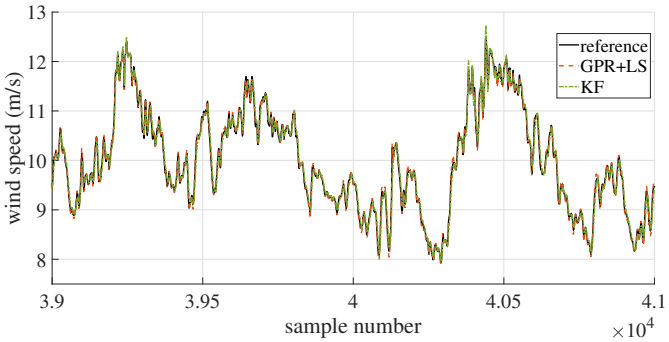

Fig. 7. Zoomed-in view of Fig. 6, with comparison to a Kalman Filter (KF) approach.

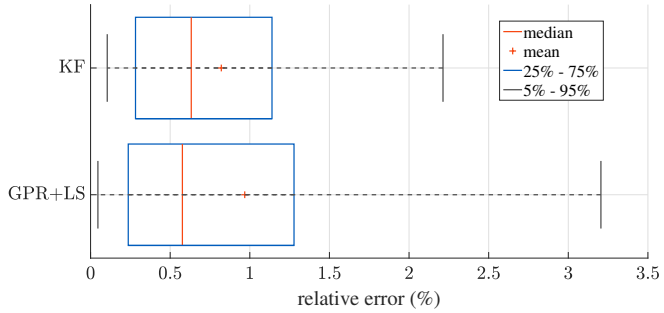

Fig. 8. Boxplots of the estimation error for the day shown in Figs. 6 and 7.

using different weight functions. Boxplots of four different functions are shown in Fig. 9. Notice the best mean error is obtained when only the nearest cluster is used. This is advantageous since, as commented on Subsection 4.3, the estimate can be computed much faster. Considering this result, we use only the nearest cluster estimate for the other tests presented in the paper.

In Fig. 10, we show the effect of varying the number of clusters C. The median error steadily decreases with the number of clusters. From $C = 5$, the mean error and the 25%-quantile error are low and steadily decreasing. Only the 5%-quantile error is still slightly irregular. Note that we need not to select a low value of C, since it does not impact the cost of online computations (when using only the nearest cluster estimate). Finally, for the results presented in this paper, we pick up $C = 5$.

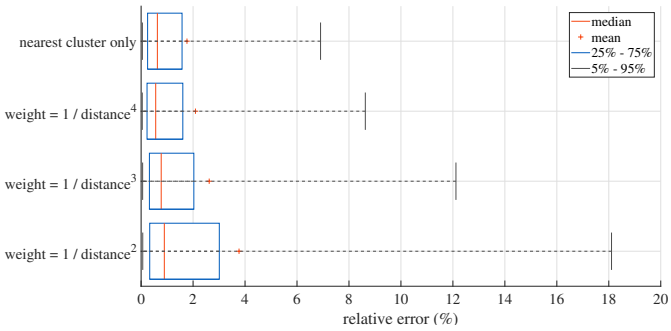

Fig. 9. Relative error for different choices of weight function. In this case $r = 20$, $C = 5$, $N = 10^4$, and $M = 3000$. From the top: in the first box plot we use only the nearest cluster estimate, in the others, we combine the estimations of each cluster using (16)-(17) with $n = 4$, $n = 3$, and $n = 2$, respectively.

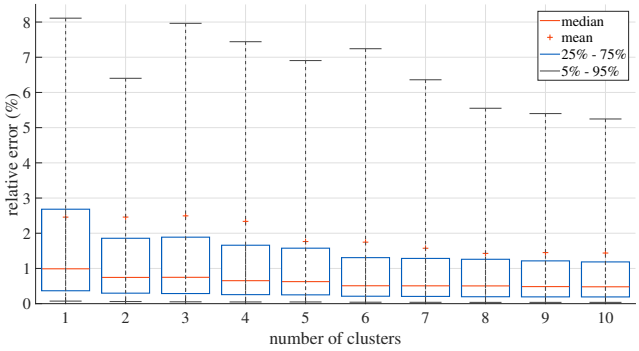

Fig. 10. Relative error for different choices of number of clusters. In this case $r = 20$, $N = 10^4$, $M = 3000$, and we only use the nearest cluster estimate.

Also note that, using a very low number of clusters such as $C = 1$ would lead to unsatisfactory results in the vicinity of the turbine cut-in speed and/or above-rated speeds. The results are not shown here due o lack of space, but this is particularly visible when plotting time-series of the estimates.

Fig. 11 depicts a comparison among the results using different values of M. We see the trade-off related to using an approximation instead of the exact GPR (in which case we would have $M = N$ and would use (13) to perform the estimation with each kernel). We see a downward trend in the median of the error as we increase the number of points, as expected.

Finally, we show the effect of varying N in Fig. 12. N is related to the offline computational cost, so it needs not to be selected very carefully, and can be much larger than M. In this paper, we work with $N = 10^4$. Again, we see an overall downward trend in the median of the error as the number of points increases.

6. CONCLUSION

In this paper, we present a method based on Gaussian Process Regression (GPR) to estimate in real-time the wind speed using only SCADA measurements that are commonly available on wind turbines. We use GPR together with a regularized least squares optimization to

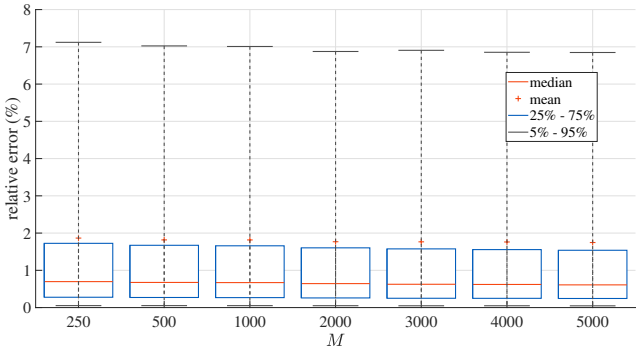

Fig. 11. Relative error for different values of M. The remaining parameters are $C = 5$, $r = 20$, and $N = 10^4$. We use only the nearest cluster estimate.

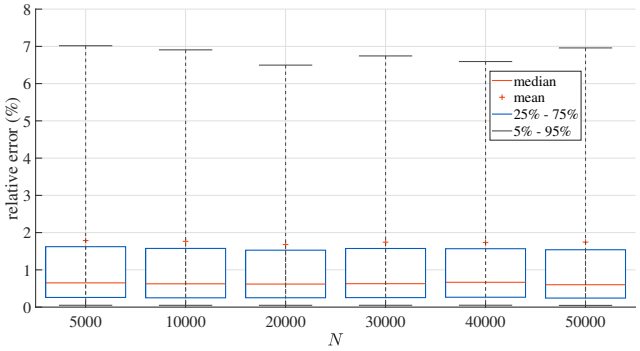

Fig. 12. Relative error for different values of N. The remaining parameters are $C = 5$, $r = 20$, and $M = 3000$. We use only the nearest cluster estimate.

obtain a map from the recent history of the SCADA measurements to the current wind speed. We train and test the method using data gathered from an operating commercial wind turbine equipped with a nacelle LiDAR sensor that gives the reference wind speed. The method presented requires no physical modeling as it is fully data-driven.

The obtained numerical results feature very interesting estimation capabilities, so that this algorithm would be accurate enough to be considered for controls and/or diagnostics purposes. It constitutes an interesting alternative to more traditional methods, as it requires no access to turbine manufacturing data or torque coefficient maps, uses only commonly available measurements, has a light and simple tuning procedure, and shows very good performance. Also, we depict how the performance is affected by the number of training points (offline computations) and how it is affected by the number of points chosen for the least squares approximation used for the online computations.

In a practical application, one would have to find the optimal trade-off between accuracy and computational cost in order to set the parameters of the algorithm.

Promising directions of work include the use of this algorithm to control and diagnostics applications or as a basis to estimate other wind properties at the wind farm scale, extend its capabilities to perform prediction (i.e., estimate the future wind speed rather than the current

one), and investigate the use of different mean functions for the GPR.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The Advantages of Careful Seeding. Technical Report 2006-13, Stanford InfoLab. URL `http://ilpubs.stanford.edu:8090/778/`.

Boukhezzar, B. and Siguerdidjane, H. (2011). Nonlinear Control of a Variable-Speed Wind Turbine Using a Two-Mass Model. *IEEE Transactions on Energy Conversion*, 26(1), 149–162.

Deng, X., Yang, J., Sun, Y., Song, D., Yang, Y., and Joo, Y.H. (2020). An effective wind speed estimation based extended optimal torque control for maximum wind energy capture. *IEEE Access*, 8, 65959–65969. doi: 10.1109/ACCESS.2020.2984654.

Deng, X., Yang, J., Sun, Y., Song, D., Xiang, X., Ge, X., and Joo, Y.H. (2019). Sensorless effective wind speed estimation method based on unknown input disturbance observer and extreme learning machine. *Energy*, 186, 115790.

Guillemin, F., Nguyen, H.N., Sabiron, G., Di Domenico, D., and Boquet, M. (2018). Real-time three dimensional wind field reconstruction from nacelle LiDAR measurements. *Journal of Physics: Conference Series*, 1037, 032037. doi:10.1088/1742-6596/1037/3/032037.

Jaramillo-Lopez, F., Kenne, G., and Lamnabhi-Lagarrigue, F. (2016). A novel online training neural network-based algorithm for wind speed estimation and adaptive control of pmsg wind turbine system for maximum power extraction. *Renewable Energy*, 86, 38–48.

Jena, D. and Rajendran, S. (2015). A review of estimation of effective wind speed based control of wind turbines. *Renewable and Sustainable Energy Reviews*, 43, 1046–1062.

Lee, J. and Zhao, F. (2020). Global Wind Report 2019. Technical Report, GWEC.

Mayer, J., Basarur, A., Petrova, M., Sordon, F., Zea, A., and Hanebeck, U.D. (2020). Position and Speed Estimation of PMSMs Using Gaussian Processes. In *21st IFAC World Congress*.

Nam, Y.S., Kim, J.G., Paek, I.S., Moon, Y.H., Kim, S.J., and Kim, D.J. (2011). Feedforward pitch control using wind speed estimation. *Journal of Power Electronics*, 11(2), 211–217.

Nguyen, H.N. and Guillemin, F. (2020). Method for acquiring and modelling an incident wind field by means of a lidar sensor. US2020/0124026 A1.

Østergaard, K.Z., Brath, P., and Stoustrup, J. (2007). Estimation of effective wind speed. In *Journal of Physics: Conference Series*, volume 75, 012082. IOP Publishing.

Schulz, E., Speekenbrink, M., and Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85, 1–16.

Sierra-García, J.E. and Santos, M. (2021). Improving wind turbine pitch control by effective wind neuro-estimators. *IEEE Access*, 9, 10413–10425. doi: 10.1109/ACCESS.2021.3051063.

Song, D., Yang, J., Cai, Z., Dong, M., Su, M., and Wang, Y. (2017a). Wind estimation with a non-standard extended Kalman filter and its application on maximum power extraction for variable speed wind turbines. *Applied energy*, 190, 670–685.

Song, D., Yang, J., Dong, M., and Joo, Y.H. (2017b). Kalman filter-based wind speed estimation for wind turbine control. *International Journal of Control, Automation and Systems*, 15(3), 1089–1096.

Sung, H.C., Park, J.B., and Joo, Y.H. (2011). Robust observer-based fuzzy control for variable speed wind power system: Lmi approach. *International Journal of Control, Automation and Systems*, 9(6), 1103–1110.

Williams, C.K.I. and Rasmussen, C.E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3), 37–52.

Yang, X., Han, X., Xu, L., and Liu, Y. (2006). Soft sensor based on support vector machine for effective wind speed in large variable wind. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, 1–4. doi:10.1109/ICARCV.2006.345278.