A Gaussian Process Iterative Learning Control for Aircraft Trajectory Tracking

Almudena Buelta, Alberto Olivares, Ernesto Staffetti, Waqas Aftab, and Lyudmila Mihaylova, *Senior Member, IEEE*

***Abstract*—This paper proposes a recursive Gaussian process regression with a joint optimization-based iterative learning control algorithm to estimate and predict disturbances and model uncertainties affecting a flight. The algorithm proactively compensates for the predicted disturbances, improving precision in aircraft trajectory tracking. Higher precision in trajectory tracking implies an improvement of the aircraft trajectory predictability and therefore of the air traffic management system efficiency. Airlines can also benefit from this higher predictability by reducing the number of alterations when following their designed trajectories, which entails a reduction of costs and emissions. The iterative learning control algorithm is divided into two steps: first, a recursive Gaussian process regression estimates and predicts perturbations and model errors with no need for prior knowledge about their dynamics and with low computational cost, and second, this information is used to update the control inputs so that the subsequent aircraft intending to fly the same planned trajectory will follow it with greater precision than the previous ones. This method is tested on a simulated commercial aircraft performing a continuous climb operation and compared to an iterative learning algorithm using a Kalman filter estimator in a similar scenario. The results show that the proposed approach provides 62% and 42% precision improvement in tracking the desired trajectory, as compared to the Kalman filter approach, in two experiments where no prior knowledge of the unmodeled dynamics was available, also achieving it in less iterations.**

***Index Terms*—Aircraft Trajectory Tracking, Gaussian Process Regression, Iterative Learning Control.**

I. Introduction

THE increasing importance that precise trajectory tracking will have in the implementation of Trajectory-Based Operations (TBO), which are one of the key aspects in the modernization of Air Traffic Management (ATM), makes it necessary to explore new approaches in aircraft control to achieve the precision required. The TBO concept is based on Four-Dimensional (4D) aircraft trajectories, which consist in a precise description of an aircraft path in space and time, in which delays are considered deviations as much as horizontal or vertical position errors. In TBO, aircraft trajectories are planned according to the preferences of airlines, and after conflict detection and resolution, they have to be followed with precision. Trajectory predictability, that is to say the correspondence between the planned and flown trajectories, is key to the implementation of TBO. Higher trajectory predictability implies better traffic synchronisation, which results in an improvement in the safety, efficiency, and capacity of the ATM system. Moreover, higher trajectory predictability, by reducing the number of alterations when following the planned trajectories, entails a reduction of costs and emissions. However, random factors, such as weather conditions, affect the precision, which result in deviations from the reference trajectory that can neither be predicted nor corrected by standard trajectory tracking controllers, which react to disturbances only after they occur, usually employing feedback control techniques. Another control strategy is needed which is able to compensate for disturbances before they occur. The method presented in this paper meets this demand by combining Iterative Learning Control (ILC) and Gaussian Process Regression (GPR) to use data gathered during operation to estimate model uncertainties and external disturbances affecting a flight and predict their values. The control inputs are corrected so that these predicted values are compensated in the following execution of a similar operation. This method has been tested on simulated flights of commercial aircraft which intend to follow a planned trajectory precisely under the assumption that both the aircraft dynamic model and the trajectory to follow are the same at each iteration.

A. Buelta, A. Olivares, and E. Staffetti are with the Department of Signal Theory and Communications and Telematic Systems and Computing, Universidad Rey Juan Carlos, Fuenlabrada, Spain (e-mail: almudena-jose.buelta@urjc.es; alberto.olivares@urjc.es; ernesto.staffetti@urjc.es). (Corresponding author: Almudena Buelta.)

W. Aftab and L. Mihaylova are with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, United Kingdom (e-mail: waqasaftabmalik@gmail.com; l.s.mihaylova@sheffield.ac.uk).

A. Previous approaches

ILC is a widely used control method which has proven its effectiveness in improving a system's performance of a repetitive process. It is based on the idea that the system can learn from data measured in past executions to update the control input iteratively, leading the output to converge to a given desired trajectory.

The ILC methodology is introduced in [1] as a prominent research field in control systems with applications in robotics, including connections to other control approaches such as ILC based on neural networks, nonlinear ILC, adaptive schemes in ILC, and robustness (see Section 2 of [2] and references therein). The surveys [3] and [4], which discuss the key results in ILC design and analysis, as well as various textbooks such as [5], which deals with ILC for both linear and nonlinear systems, and [6], which is mostly concerned with real-time applications, are among the main literature related to ILC.

The scope of applications has also widened beyond industrial robotics. Variants of the ILC method have been applied to trajectory tracking for Unmanned Aerial Vehicles (UAV). See

for example [7] and references therein. In [8], an ILC approach is applied to precise quadrocopter trajectory tracking under the presence of model errors and other recurrent disturbances. This approach, applied to aircraft trajectory tracking in [9] and [10], divides the ILC algorithm into two steps: estimation and control update, where the estimation of the modeling errors is calculated from past measurements and serves as the input for the update step to determine a more adequate control input in the following iteration.

Most ILC approaches rely on the system model as a basis of the learning algorithm, making its performance very sensitive to model uncertainties. In [8], [9], and [10], the estimation step is performed using a Kalman filter [11]. The Kalman filter approach to recursive estimation has seen extensive use in aerospace applications since its introduction in this area [12] and continues to form the basis for many new filtering approaches to the state estimation problem [13], [14]. However, the underlying model dynamics must be known for it to be possible to tune the Kalman filter, and it is suitable only for slight linear changes in the unmodeled dynamics. Enhanced variants of the Kalman filter, such as the extended Kalman filter, would allow nonlinear changing dynamics of the disturbances to be estimated, but would also require some knowledge of its behavior. In the new approach presented in this paper, the Kalman filter is replaced with a recursive GPR method, aiming to provide an alternative ILC approach in which no prior knowledge of the disturbances is needed, meaning that no prior tuning is required, and which is capable of estimating the disturbances even if they vary between iterations, despite not knowing the dynamics of their variations.

Gaussian processes provide a nonparametric statistical learning of nonlinear dynamic systems from noisy data, which are characterized by covariance functions that usually have a set of hyperparameters [15]. Regression methods based on Gaussian processes are used in many areas of application, such as machine learning [16], [17], signal processing [18], and control engineering [19], [20].

The main drawback of Gaussian processes is the considerable computational cost when working with large data sets. Multiple approaches have been proposed in order to reduce the computational burden, for example in [21], [22], and [23], where the training data is reduced to a number of so-called inducing points. Another difficulty is the determination of the hyperparameters, which are usually learned by optimizing the marginal likelihood [24].

In [25], an online procedure for updating Gaussian process parameters is proposed, in which the regression is performed on a set of basis vectors with low computational and memory demands, simultaneously learning the hyperparameters. This approach has been adapted in this paper to be included in the estimation and prediction steps of the ILC algorithm. To do so, the sequential nature of ILC has been taken into account in the time series prediction of the disturbances affecting the flight, assuming that the behavior of these disturbances is more correlated to the recent past observations than to the distant past ones. Therefore, the location of the basis vectors is updated at each iteration replacing the oldest one with the new data estimation.

As already mentioned, the method described in this paper has been applied to commercial aircraft trajectory tracking in the context of TBO, which allow for optimized trajectories improving efficiency, predictability, and airspace capacity. As in [9], the trajectory chosen to test the method is a Continuous Climb Operation (CCO), an aircraft operating technique in which the airspace and the departing procedures are designed in such a way that aircraft, after take-off, are allowed to continuously climb following trajectories optimized based on their performance, until the cruise level is reached. This technique, in which level-off segments of flight are eliminated, leads to significant reductions of fuel consumption and emissions [26]. The feasible CCO aircraft trajectory has been generated here using a pseudospectral optimal control approach [27].

The experiments were carried out in a realistic, simulated scenario built in MATLAB/Simulink, a proprietary programming platform developed by MathWorks[1] for numerical computation, in which dynamical systems can be modeled, simulated and analyzed. In this scenario, a flight simulator has been included, which has been built using commercial aircraft data retrieved from the Base of Aircraft Data (BADA) developed by EUROCONTROL[2]. BADA is an aircraft performance model intended for aircraft trajectory simulation and prediction in ATM research and development. It provides both model specifications, in which the theoretical aircraft dynamic models are described, and data sets, in which the aircraft-specific parameters are given [28].

B. Contributions of the paper.

The main contributions of this work are threefold:

1) A novel framework for precise aircraft trajectory tracking is introduced. It includes a GPR within an ILC scheme.
2) This learning GPR estimates its hyperparameters online. The disturbances and model uncertainties affecting a flight are also estimated without the need for any prior knowledge about them.
3) The developed framework is validated and tested over a range of examples with different uncertainties.

The proposed framework is flexible and has significant advantages compared to the Kalman filters used as an estimator in earlier ILC works [8], since it learns from the data. In contrast, Kalman filters need to be correctly tuned using a priori knowledge of the dynamics to be estimated, and it is assumed that they remain unchanged between iterations in the prediction step. The computational costs required by the GPR are significantly reduced by using a recursive approach, which is updated at each iteration dismissing the oldest data and incorporating the newest observations. This makes it appealing for real-time control.

C. Organization of the paper.

The paper is organized as follows: the ILC paradigm is summarized in Section II. The problem formulation of the

[1] https://es.mathworks.com/

[2] https://www-test.eurocontrol.int/services/bada

GPR is described in Section III. The implementation of the recursive GPR with learning of the hyperparameters into the ILC algorithm is presented in Section IV, and the experiments and numerical results are described in Section V. Finally, Section VI contains the conclusions.

II. Iterative Learning Control

In this paper, GPR is implemented for estimation and prediction of the disturbances in an ILC algorithm. The ILC approach used is the one presented in [8], which is an effective and computationally efficient learning strategy that has proven to improve precision in trajectory tracking by pure feedforward adaptation of the control input. A detailed explanation of the ILC approach considered in this paper can be found in [8] and its application and adaptation to aircraft trajectory tracking can be found in [9] and [10]. In all three references, the estimation is performed using a Kalman filter. This ILC scheme explicitly takes into consideration input and state constraints of a nominal model and determines an updated control by using optimal filtering for disturbance estimation and convex optimization for trajectory update.

The system dynamics are captured by a time-varying nonlinear model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \\ \mathbf{y}(t) &= \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t), \end{aligned} \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and $\dot{\mathbf{x}}(t) \in \mathbb{R}^{n_x}$ are the state and the state's derivative, respectiveley, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the control input, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the output, and $\mathbf{f}$ and $\mathbf{h}$ are assumed to be continuously differentiable in $\mathbf{x}$ and $\mathbf{u}$.

Constraints on the state $\mathbf{x}(t)$, the input $\mathbf{u}(t)$, and their time derivatives are represented by

$$\mathbf{Z}\mathbf{q}(t) \preceq \mathbf{q}_{\max}, \tag{2}$$

where

$$\begin{aligned} \mathbf{q}(t) = \Big[& \mathbf{x}^T(t), \mathbf{u}^T(t), \dot{\mathbf{x}}^T(t), \dot{\mathbf{u}}^T(t), \\ & \ldots, \frac{d^m}{dt^m}\mathbf{x}^T(t), \frac{d^m}{dt^m}\mathbf{u}^T(t) \Big]^T, \end{aligned} \tag{3}$$

$\mathbf{Z}$ is a constant matrix of appropriate dimensions, and $\mathbf{q}_{\max} \in \mathbb{R}^{n_q}$, being n_q the total number of constraints. The inequality denoted by the symbol $\preceq$ is defined component-wise.

The purpose of the learning algorithm is to track a feasible predefined output trajectory, the desired trajectory $\hat{\mathbf{y}}(t)$, precisely over a finite time interval.

A. Lifted system representation

A linearized and time-discretized version of this model is represented in the lifted domain, in which the state, input, and output vectors are the stacked vectors of all discretization time-steps of the trajectory (see [29], [30]), that is,

$$\begin{aligned} \boldsymbol{u}_j &= \left[\tilde{\boldsymbol{u}}_j^T(0), \tilde{\boldsymbol{u}}_j^T(1), \ldots, \tilde{\boldsymbol{u}}_j^T(N-1)\right]^T \in \mathbb{R}^{Nn_u}, \text{[3]} \\ \boldsymbol{x}_j &= \left[\tilde{\boldsymbol{x}}_j^T(1), \tilde{\boldsymbol{x}}_j^T(2), \ldots, \tilde{\boldsymbol{x}}_j^T(N)\right]^T \in \mathbb{R}^{Nn_x}, \\ \boldsymbol{y}_j &= \left[\tilde{\boldsymbol{y}}_j^T(1), \tilde{\boldsymbol{y}}_j^T(2), \ldots, \tilde{\boldsymbol{y}}_j^T(N)\right]^T \in \mathbb{R}^{Nn_y}, \end{aligned} \tag{4}$$

with

$$\begin{aligned} \tilde{\boldsymbol{u}}_j(k) &= \mathbf{u}_j(k) - \hat{\mathbf{u}}(k), \\ \tilde{\boldsymbol{x}}_j(k) &= \mathbf{x}_j(k) - \hat{\mathbf{x}}(k), \\ \tilde{\boldsymbol{y}}_j(k) &= \mathbf{y}_j(k) - \hat{\mathbf{y}}(k), \end{aligned}$$

where the subscript j represents the $j-$th execution of the desired task, and $k \in \mathcal{K} = \{0, 1, ..., N\}$, with $N < \infty$, represents the discrete-time index. For each iteration j, the triplet $(\boldsymbol{x}_j, \boldsymbol{u}_j, \boldsymbol{y}_j)$ indicates the lifted vectors describing small deviations from the desired trajectory and its corresponding state, control input, and output, denoted by $(\hat{\mathbf{x}}, \hat{\mathbf{u}}, \hat{\mathbf{y}})$. The lifted representation characterizes the actual system dynamics by a static map in which a given discrete-time input signal is mapped to the corresponding lifted states through a constant matrix. The output is similarly represented in the lifted domain. Using this notation, the system (1) can be described as

$$\begin{aligned} \boldsymbol{x}_j &= \boldsymbol{F}\boldsymbol{u}_j + \boldsymbol{d}_j + \boldsymbol{\xi}_j, \\ \boldsymbol{y}_j &= \boldsymbol{G}\boldsymbol{x}_j + \boldsymbol{H}\boldsymbol{u}_j + \boldsymbol{\epsilon}_j, \end{aligned} \tag{5}$$

where the lifted matrices $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ account for the model nominal dynamics. Vector $\boldsymbol{d}_j$ captures the repetitive disturbances along the reference trajectory, including model errors and the free response of the system to the initial deviation. $\boldsymbol{\xi}_j$ and $\boldsymbol{\epsilon}_j$ account for the trial-uncorrelated process and measurement noise, respectively.

Based on this notation, the ILC algorithm is divided into two steps: estimation and control.

B. Disturbance estimation

As previously mentioned, in [8], [9], and [10], the estimation step is performed by a time-varying Kalman filter [11] in which only slight changes of the disturbance $\boldsymbol{d}_j$ between iterations are considered and captured in a zero-mean Gaussian white noise variable. The Kalman filter provides an estimation of the current error $\widehat{\boldsymbol{d}}_j$ taking into account the output signals from previous trials. This estimated error is used as the predictive disturbance for the following iteration, that is, $\boldsymbol{d}_{j+1}^p = \widehat{\boldsymbol{d}}_j$.

In this paper, the Kalman filter is substituted by a GPR estimation and prediction of the disturbance affecting the system. The incorporation of this method into the ILC algorithm is explained in Section IV.

C. Input update

The update step consists in deriving a model-based update rule to compute a new control input $\boldsymbol{u}_{j+1}$, which, in response to the predicted disturbance $\boldsymbol{d}_{j+1}^p$, minimizes the deviation from the nominal trajectory in the following iteration.

Figures 1 and 2 show the control inputs generated in five iterations of the ILC using a Kalman estimator and the evolution of the path described by the aircraft, which intends to fly a CCO (dashed black line in Fig. 2). The path followed

[3]Note that the notation $\mathbb{R}^{Nn_u}$ refers to the real vector space which dimension is the scalar product $N \cdot n_u$. It is defined analogously in all the vectors where this notation appears throughout this article.

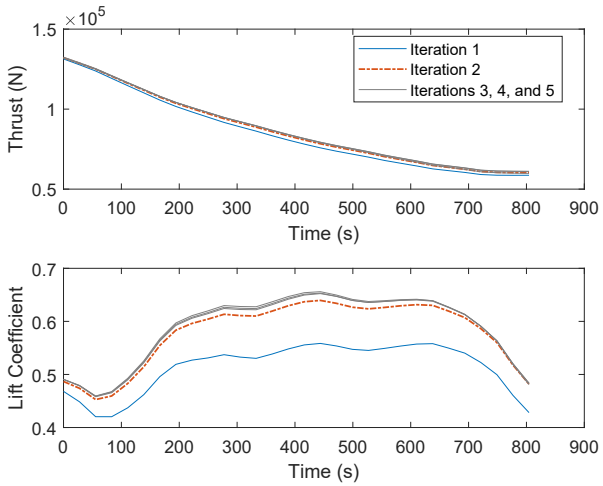

Fig. 1. Evolution of the thrust and lift coefficient over iterations.

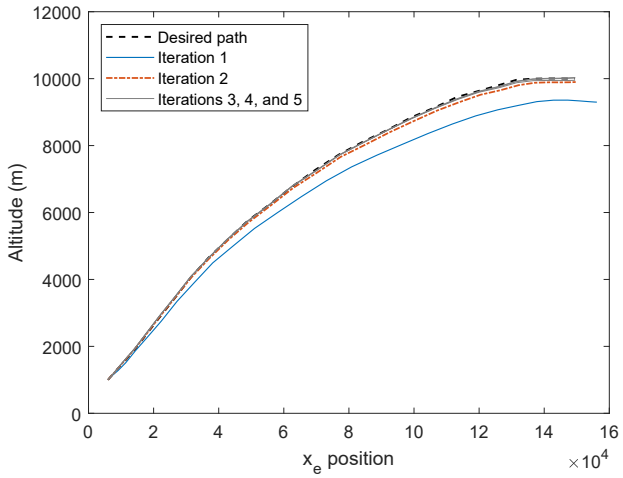

Fig. 2. Evolution of the path $x_e - h$ described by the aircraft over iterations.

by the aircraft in the first iteration remains below the desired one because of modeling and disturbance errors. These errors are estimated and the inputs are updated in accordance with the estimations, achieving more precision at each iteration.

III. Gaussian Process Regression

A. Problem formulation

For GPR, it is assumed that a set of data $\mathcal{D} = \{(\boldsymbol{v}_1, y_1), \ldots, (\boldsymbol{v}_n, y_n)\}$ is drawn from the noisy process

$$y_i = g(\boldsymbol{v}_i) + \epsilon,$$

where the column vectors $\boldsymbol{v}_i \in \mathbb{R}^{n_v}$ are the inputs, $y_i \in \mathbb{R}$ are the observations or outputs, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a zero-mean Gaussian noise with variance σ^2. A Gaussian Process (GP) is used to infer the latent function g from the set of n past observations $\mathcal{D}$, also called training data. For $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ being either the training or the testing input data vectors, the GP is completely defined by a mean function $\mu(\boldsymbol{v}_i) \equiv E\left[g(\boldsymbol{v}_i)\right]$, which specifies the expected output value, with $E[\cdot]$ being the mathematical expectation operation, and a positive semi-definite covariance function $k(\boldsymbol{v}_i, \boldsymbol{v}_j) \equiv \text{cov}\left\{g(\boldsymbol{v}_i), g(\boldsymbol{v}_j)\right\}$, which specifies the covariance between pairs of inputs $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ and is often called a kernel. Typical examples are the zero-mean function $\mu(\boldsymbol{v}_i) = 0$ and the Squared Exponential (SE) kernel

$$k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \alpha^2 exp\left(-\frac{1}{2}(\boldsymbol{v}_i - \boldsymbol{v}_j)^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{v}_i - \boldsymbol{v}_j)\right),$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix of the characteristic length-scales for each input dimension, which determine the length of the "wiggles" in the function, and α^2 is the variance of the latent function g. Such parameters of the mean and covariance functions together with the noise variance σ^2 are called the hyperparameters of the GP and are here grouped in the vector $\boldsymbol{\eta}$.

B. Prediction

The GP framework described above can be used to predict the function values $\boldsymbol{g}_* = [g(\boldsymbol{v}_{n+1}), \ldots, g(\boldsymbol{v}_{n+m})]^T$ at the arbitrary inputs $\boldsymbol{v}_* = [\boldsymbol{v}_{n+1}, \ldots, \boldsymbol{v}_{n+m}]^T$, based on the training data $\mathcal{D} = \{(\boldsymbol{v}_1, y_1), \ldots, (\boldsymbol{v}_n, y_n)\}$. Given this observations, the collection of functions $\boldsymbol{g} = [g(\boldsymbol{v}_1), \ldots, g(\boldsymbol{v}_n)]^T$ at the training inputs $\boldsymbol{v} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]^T$ follows a multivariate Gaussian distribution with mean function $\boldsymbol{\mu}(\boldsymbol{v}) = [\mu(\boldsymbol{v}_1), \ldots, \mu(\boldsymbol{v}_n)]^T$ and covariance matrix $\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) \in \mathbb{R}^{n \times n}$, the $(i,j)-$th element of which is $\boldsymbol{K}_{ij} = k(\boldsymbol{v}_i, \boldsymbol{v}_j)$.

The joint distribution of the observed data $\boldsymbol{y}$ and the predicted function values $\boldsymbol{g}_*$ is given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{g}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{v}) \\ \boldsymbol{\mu}(\boldsymbol{v}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T \\ \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}) & \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) \end{bmatrix}\right), \tag{6}$$

where $\boldsymbol{\mu}(\boldsymbol{v}_*) = [\mu(\boldsymbol{v}_{n+1}), \ldots, \mu(\boldsymbol{v}_{n+m})]^T$, $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}) \in \mathbb{R}^{m \times n}$ has the $(i,j)-$th element defined as $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})_{ij} = k(\boldsymbol{v}_{n+i}, \boldsymbol{v}_j)$, and $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) \in \mathbb{R}^{m \times m}$ has the $(i,j)-$th element defined as $\boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*)_{ij} = k(\boldsymbol{v}_{n+i}, \boldsymbol{v}_{n+j})$.

Thus, the predictive distribution conditioned by the data set $\mathcal{D}$ is given by a Gaussian distribution with mean and variance

$$\begin{aligned} \boldsymbol{\mu}_g(\boldsymbol{v}_*) &= \boldsymbol{\mu}(\boldsymbol{v}_*) + \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T (\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu}(\boldsymbol{v})), \\ \boldsymbol{\sigma}_g^2(\boldsymbol{v}_*) &= \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}_*) - \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v})^T (\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v}) + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{K}(\boldsymbol{v}_*, \boldsymbol{v}). \end{aligned} \tag{7}$$

C. Recursive Gaussian Process with learning

The GP prediction depends on the inverse of $\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v})$, which scales with $\mathcal{O}(n^3)$. For large data sets this is computationally unfeasible. The recursive GPR proposed in [25] aims to perform all calculations on a sparse representation of the GP formed by a set of $\ell \ll n$ so-called basis vectors, relying on the basis vectors for estimating the latent function and learning the hyperparameters on-line from data.

1) On-line regression: The basis vectors are located at $\boldsymbol{v} \equiv [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_\ell]^T$, and store local estimates $\boldsymbol{g} \equiv g(\boldsymbol{v})$ of the latent function, and are updated on-line with n_t new observations, $\boldsymbol{y}_t$, at inputs $\bar{\boldsymbol{v}}_t \equiv [\boldsymbol{v}_{t,1}, \ldots, \boldsymbol{v}_{t,n_t}]^T$ and time steps $t = 0, 1, \ldots$. The basis vectors are fixed in number and location for each time step. Assuming known hyperparameters, the goal is to calculate recursively the posterior distribution $p(\boldsymbol{g}|\boldsymbol{y}_{1:t})$, with $\boldsymbol{y}_{1:t} \equiv [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t]^T$, by updating the prior distribution of $\boldsymbol{g}$ from the distribution on the previous step $t-1$, $p(\boldsymbol{g}|\boldsymbol{y}_{1:t-1})$.

For deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p(\boldsymbol{g}|\boldsymbol{y}_{1:t}) = \int c_t \cdot p(\boldsymbol{y}_t|\boldsymbol{g}, \bar{\boldsymbol{g}}_t) \cdot p(\bar{\boldsymbol{g}}_t|\boldsymbol{g}) \cdot p(\boldsymbol{g}|\boldsymbol{y}_{1:t-1}) d\bar{\boldsymbol{g}}_t, \quad (8)$$

where $\bar{\boldsymbol{g}}_t = [g(\boldsymbol{v}_{t,1}), \ldots, g(\boldsymbol{v}_{t,n_t})]^T$, and c_t is a normalization constant.

2) Learning: If the hyperparameters, $\boldsymbol{\eta}$, are unknown, they are estimated simultaneously with the values of the latent function at the basis vectors. This is done calculating a joint posterior distribution $p(\boldsymbol{z}_t|\boldsymbol{y}_{1:t})$, where $\boldsymbol{z}_t^T \equiv [\boldsymbol{g}^T, \boldsymbol{\eta}_t^T]$ is the joint hidden state.

A detailed explanation of the recursive GP with the learning algorithm can be found in [25].

IV. Recursive GPR estimation and prediction in ILC

The goal is to estimate the disturbances, $\boldsymbol{d}_j$, affecting the aircraft at each iteration of the task, that is to say at each intent of flying the planned trajectory, and, ultimately, to predict the disturbances in the following iteration, $\boldsymbol{d}_{j+1}$.

The Kalman filter estimation proposed in [8] assumes that the disturbances are almost constant, that is to say

$$\boldsymbol{d}_j = \boldsymbol{d}_{j-1} + \boldsymbol{\omega}_{j-1}, \quad (9)$$

where $\boldsymbol{\omega}_j$ is a stochastic zero-mean Gaussian white noise variable.

GP regression allows for non-linear changes in the disturbances between iterations. The system dynamics is separated into two components, a known function of the state and control input, $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j)$, and an unknown function, $\boldsymbol{g}$, which represents an unknown, deterministic dynamics that is not captured by the a priori model $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j)$. The function $\boldsymbol{g}$ depends on τ_j, which is the time elapsed between the first and the $j-$th iteration, to capture the evolution of the unknown dynamics along time. The state and output at each iteration are therefore modeled as

$$\begin{aligned} \boldsymbol{x}_j &= \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{g}(\tau_j), \\ \boldsymbol{y}_j &= \boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{u}_j) + \boldsymbol{\epsilon}_j, \end{aligned} \quad (10)$$

where, using the lifted representation described in Section II, $\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j) = \boldsymbol{F}\boldsymbol{u}_j$, the disturbances are captured in the unknown function $\boldsymbol{g}(\tau_j)$, and $\boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{u}_j) = \boldsymbol{G}\boldsymbol{x}_j + \boldsymbol{H}\boldsymbol{u}_j$. In this paper, it is assumed that full-state information is available and, without loss of generality, $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{\epsilon}_j$.

We finally get the disturbance in the format

$$\boldsymbol{d}_j = \boldsymbol{y}_j - \boldsymbol{F}\boldsymbol{u}_j = \boldsymbol{g}(\tau_j) + \boldsymbol{\epsilon}_j. \quad (11)$$

The recursive GPR is applied separately to each single element $d_j^i = g^i(\tau_j) + \epsilon_j^i$ of the lifted vector $\boldsymbol{d}_j$, with $i = 1, 2, \ldots, N \cdot n_x$, where N is the number of time-steps in which the system dynamics is discretized and n_x is the number of state variables, i.e., each state variable at each time-step of the discretization of the trajectory is treated as an independent output and assumed uncorrelated to any other state variable, or the same one at any other time-step. On the other hand, for each output, the cross-covariance between the basis vectors and the new observations is taken into account.

For the sake of clarity, the superindex i indicating the $i-$th element of vectors $\boldsymbol{d}_j$ and $\boldsymbol{y}_j$ will be omitted in the remainder of this section. The unknown function value, g^i, corresponding to d_j^i, will be denoted by g. Analogously, the standard deviation of ϵ_j^i will be denoted by σ.

The approach presented in [25] is adapted here to the ILC algorithm and modified to incorporate the new observations to the basis vectors. Therefore, the basis vectors are fixed in number, but their location changes at each iteration, replacing the first one with the new data estimation. Since the function g depends solely on the time between the first and the current iterations, the basis vectors at the $j-$th iteration are the values of τ corresponding to the ℓ prior iterations, that is to say adding the subindex j to the notation in Section III-C, $\boldsymbol{v}_j = [\tau_{j-\ell}, \ldots, \tau_{j-1}]^T$. Similarly, $\boldsymbol{g}_j = [g(\tau_{j-\ell}), \ldots, g(\tau_{j-1})]^T$ is the vector of local estimates of the latent function at the ℓ basis vectors $\boldsymbol{v}_j$.

Although the observations are made on the output data, $\boldsymbol{y}_j$, the estimated and predicted values are the unmodeled dynamics and disturbances $\boldsymbol{d}_j$ affecting the system. The indirect observation of each element d_j becomes explicit in (11), as the corresponding element of $\boldsymbol{d}_j = \boldsymbol{y}_j - \boldsymbol{F}\boldsymbol{u}_j$.

1) Estimation: As explained in Section III-C, the algorithm recursively estimates the values of the latent function and simultaneously learns the hyperparamenters. As such, no prior knowledge of the disturbances is needed. This is effected by calculating a joint posterior Gaussian distribution $p(\boldsymbol{z}_j|\boldsymbol{y}_{1:j})$, where $\boldsymbol{z}_j^T \equiv [\boldsymbol{g}_j^T, \boldsymbol{\eta}_j^T]$ is the joint hidden state with mean and covariance

$$\boldsymbol{\mu}_j^z \equiv \begin{bmatrix} \boldsymbol{\mu}_j^g \\ \boldsymbol{\mu}_j^\eta \end{bmatrix}, \quad \boldsymbol{C}_j^z \equiv \begin{bmatrix} \boldsymbol{C}_j^g & \boldsymbol{C}_j^{g\eta} \\ \boldsymbol{C}_j^{\eta g} & \boldsymbol{C}_j^\eta \end{bmatrix}, \quad (12)$$

$\boldsymbol{\mu}_j^g$ and $\boldsymbol{C}_j^g$ being the mean and covariance of $\boldsymbol{g}_j$, $\boldsymbol{\mu}_j^\eta$ and $\boldsymbol{C}_j^\eta$ the mean and covariance of the hyperparameters, and $\boldsymbol{C}_j^{g\eta}$ and $\boldsymbol{C}_j^{\eta g}$ the cross-covariance matrices at iteration j.

The starting point for this calculation is a joint prior distribution $p(\boldsymbol{z}_{j-1}|\boldsymbol{d}_{1:j-1})$ at iteration $j-1$, which is updated with the new observation d_j. To incorporate the new input at iteration j, it is necessary to infer the latent function $\bar{g}_j = g(\tau_j)$. The state-space model incorporating the hyperparameters is given by

$$\begin{bmatrix} \boldsymbol{z}_{j-1} \\ \bar{g}_j \end{bmatrix} = \boldsymbol{A}_j(\boldsymbol{\eta}_j - 1)\boldsymbol{z}_{j-1} + \boldsymbol{w}_j, \quad (13)$$

where

$$\boldsymbol{A}_j(\boldsymbol{\eta}_{j-1}) = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{J}_j(\boldsymbol{\eta}_{j-1}) & \boldsymbol{0} \end{bmatrix}$$

and the noise $\boldsymbol{w}_j$ is Gaussian with mean and covariance

$$\boldsymbol{\mu}_j^w \equiv \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ b(\boldsymbol{\eta}_{j-1}) \end{bmatrix}, \quad \boldsymbol{C}_j^w \equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & B(\boldsymbol{\eta}_{j-1}) \end{bmatrix}. \tag{14}$$

The kernel functions in $\boldsymbol{J}_j(\boldsymbol{\eta}_{j-1}) \equiv \boldsymbol{K}(\tau_j, \boldsymbol{v})\boldsymbol{K}(\boldsymbol{v}, \boldsymbol{v})^{-1}$, $B(\boldsymbol{\eta}_{j-1}) \equiv \boldsymbol{K}(\tau_j, \tau_j) - \boldsymbol{J}_j k(\boldsymbol{v}, \tau_j)$, and in the vector $b(\boldsymbol{\eta}_{j-1}) \equiv \mu(\tau_j) - \boldsymbol{J}_j \mu(\boldsymbol{v})$ are calculated using the hyperparameters $\boldsymbol{\eta}_{j-1}$.

The model (13) is nonlinear with respect to the hyperparameters $\boldsymbol{\eta}_{j-1}$ but, for a given hyperparameter, the model is linear and the prediction inducing the desired joint distribution $p(\boldsymbol{z}_{j-1}, \bar{g}_j | \boldsymbol{d}_{1:j-1})$ can be performed exactly using a Kalman predictor. Here, a collection of s sigma points $\hat{\boldsymbol{\eta}}_i$ have been selected and given weights $\boldsymbol{\omega}_i$, $i = 1, \ldots, s$, using the unscented transform [31] constrained so that the hyperparameters are positive [32]. A Kalman predictor is applied to each sigma point obtaining the joint Gaussian distribution $p(\boldsymbol{z}_{j-1}, \bar{g}_j | \boldsymbol{d}_{1:j-1})$ with mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^p &= \sum_{i=1}^{s} \boldsymbol{\omega}_i \boldsymbol{\mu}_i^p, \\ \boldsymbol{C}_j^p &= \sum_{i=1}^{s} ((\boldsymbol{\mu}_i^p - \boldsymbol{\mu}_j^p)(\boldsymbol{\mu}_i^p - \boldsymbol{\mu}_j^p)^T + \boldsymbol{C}_i^p), \end{aligned} \tag{15}$$

where

$$\begin{aligned} \boldsymbol{\mu}_i^p &\equiv \boldsymbol{A}_j(\hat{\boldsymbol{\eta}}_i) \begin{bmatrix} \boldsymbol{\mu}_{j-1}^g + \boldsymbol{C}_{j-1}^{g\eta}(\boldsymbol{C}_{j-1}^{\eta})^{-1}(\hat{\boldsymbol{\eta}}_i - \boldsymbol{\mu}_{j-1}^{\eta}) \\ \hat{\boldsymbol{\eta}}_i \end{bmatrix} \\ &\quad + \boldsymbol{\mu}_j^w(\hat{\boldsymbol{\eta}}_i), \\ \boldsymbol{C}_i^p &\equiv \boldsymbol{A}_j(\hat{\eta}_i) \begin{bmatrix} \boldsymbol{C}_{j-1}^g - \boldsymbol{C}_{j-1}^{g\eta}(\boldsymbol{C}_{j-1}^{\eta})^{-1}\boldsymbol{C}_{j-1}^{\eta g} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{A}_j(\hat{\boldsymbol{\eta}}_i)^T \\ &\quad + \boldsymbol{C}_t^w(\hat{\boldsymbol{\eta}}_i). \end{aligned} \tag{16}$$

The incorporation of the new observation, d_j, is performed in two steps, in which the joint distribution is decomposed into an observed and an unobserved part

$$p(\boldsymbol{z}_j | \boldsymbol{d}_{1:j}) = \int p(\boldsymbol{g}_j, \boldsymbol{\eta}_j^- | \sigma, \bar{g}_j) \cdot p(\sigma, \bar{g}_j | \boldsymbol{d}_{1:j}) d\bar{g}_j,$$

where $\boldsymbol{\eta}_j^-$ is the vector of all the hyperparameters except σ. As such, $\boldsymbol{\mu}_j^p$ and $\boldsymbol{C}_j^p$ can be expressed as

$$\boldsymbol{\mu}_j^p \equiv \begin{bmatrix} \boldsymbol{\mu}_{j-1}^u \\ \boldsymbol{\mu}_j^o \end{bmatrix}, \quad \boldsymbol{C}_j^z \equiv \begin{bmatrix} \boldsymbol{C}_{j-1}^u & \boldsymbol{C}_j^{uo} \\ \boldsymbol{C}_j^{ou} & \boldsymbol{C}_j^o \end{bmatrix}, \tag{17}$$

where $\boldsymbol{o}^T = [\sigma, \bar{g}_j^T]$ is the observable state with mean $\boldsymbol{\mu}_g^o$ and covariance $\boldsymbol{C}_j^o$, $\boldsymbol{u}_{j-1}^T = [\boldsymbol{g}_j^T, [\boldsymbol{\eta}_j^-]^T]$ is the unobservable state with mean $\boldsymbol{\mu}_{j-1}^u$ and covariance $\boldsymbol{C}_{j-1}^u$, and $\boldsymbol{C}_j^{uo}$ and $\boldsymbol{C}_j^{ou}$ are the cross-covariance matrices between the observable and unobservable states.

The observable state can be directly updated and, assuming that the observed state and the observations are jointly Gaussian, the conditional distribution $p(\sigma, \bar{g}_j | \boldsymbol{d}_{1:j})$ will have mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^e &= \boldsymbol{\mu}_j^o + \boldsymbol{C}_j^{od}(\boldsymbol{C}_j^d)^{-1}(d_j - \mu_j^d), \\ \boldsymbol{C}_j^e &= \boldsymbol{C}_j^o - \boldsymbol{C}_j^{od}\left[\boldsymbol{C}_j^{od}(C_j^d)^{-1}\right]^T, \end{aligned} \tag{18}$$

where $\mu_j^d = E[\bar{g}_j]$, $C_j^d = cov[\bar{g}_j] + var[\sigma] + E[\sigma]^2$, and $C_j^{od} = cov[\boldsymbol{o}_j, \bar{g}_j]$ are elements of $\boldsymbol{\mu}_j^o$ and $\boldsymbol{C}_j^o$. The unobservable part is then updated as a Gaussian distribution $p(\boldsymbol{g}_j, \boldsymbol{\eta}_j^- | \sigma, \bar{g}_j)$ with mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_j^u &= \boldsymbol{\mu}_{j-1}^u + \boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}(\boldsymbol{\mu}_j^e - \boldsymbol{\mu}_j^o), \\ \boldsymbol{C}_j^u &= \boldsymbol{C}_{j-1}^u + \boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}(\boldsymbol{C}_j^e - \boldsymbol{C}_j^o)\left[\boldsymbol{C}_j^{uo}(\boldsymbol{C}_j^o)^{-1}\right]^T. \end{aligned} \tag{19}$$

Finally, the mean and covariance of the the joint posterior distribution $p(\boldsymbol{z}_j | \boldsymbol{d}_{1:j})$ with updated basis vectors and hyperparameters is obtained combining and rearranging (18) and (19) as $\boldsymbol{z}_j^T = [[\boldsymbol{g}_j^-]^T, \bar{g}_j^T, \boldsymbol{\eta}_j^T]$, where $\boldsymbol{g}_j^-$ indicates the latent function at the basis vectors except the basis vector corresponding to the iteration $j - \ell$. This is done to incorporate the latent function estimation of the new observations, $\bar{g}_j$, into the basis vectors while keeping a fixed number of them, since the most recent data will become more significant than the oldest in the following iterations. The updated basis vectors will be then $\boldsymbol{v}_{j+1} = [\tau_{j-\ell+1}, \ldots, \tau_j]$.

2) Prediction: To predict the latent function at iteration $j+1$, the process is repeated to calculate $\boldsymbol{\mu}_{j+1}^p$ as in (15), using the joint posterior distribution, $p(\boldsymbol{z}_j | \boldsymbol{d}_{1:j})$, obtained in the previous steps. The predicted disturbance will be then

$$d_{j+1}^p = E[\bar{g}_{j+1}], \tag{20}$$

where $E[\bar{g}_{j+1}]$ can be extracted from $\boldsymbol{\mu}_{j+1}^p$.

3) Input update: Repeating the previous process for each element of $\boldsymbol{d}_j$, the vector $\boldsymbol{d}_{j+1}^p$ is built, which contains the predicted values of the disturbances affecting all the state variables at every time-step in which the trajectory has been discretized. Following [8], a new control input $\boldsymbol{u}_{j+1}$ is calculated optimally compensating for the predicted disturbance by solving the constrained optimization problem

$$\begin{aligned} \min_{\boldsymbol{u}_{j+1}} \quad & \|\boldsymbol{F}\boldsymbol{u}_{j+1} + \boldsymbol{d}_{j+1}^p\| + \alpha\|\boldsymbol{D}\boldsymbol{u}_{j+1}\| \\ \text{subject to} \quad & \boldsymbol{L}\boldsymbol{u}_{j+1} \leq \boldsymbol{q}_{\max}, \end{aligned} \tag{21}$$

where the system's constraints (2) are explicitly taken into account. The additional term $\alpha \geq 0$ and the matrix $\boldsymbol{D}$ are introduced to penalize the input or approximations of its derivatives pursuing smoothness of the optimal solution. The update law in (21) can be expressed as a standard convex optimization problem.

V. Numerical results

To show the effectiveness of the GPR for estimation in ILC, it has been tested in a simulated experiment consisting of the precise trajectory tracking of an aircraft in a CCO.

A. Aircraft dynamics and trajectory generation

A common three-degrees-of-freedom dynamic model has been used, considering that trajectories of commercial aircraft only involve small aircraft rotations. This model describes the point variable-mass motion of the aircraft over a non-rotating flat Earth model with International Standard Atmosphere (ISA) [33]. In particular, a symmetric, leveled-wing flight has been considered on the vertical plane. It has therefore been assumed that there is a constant heading angle, no sideslip and bank angle, and all forces lie in the plane of symmetry of the aircraft. The effects of wake vortices have not been included in the model because it is assumed that the time and distance separations between aircraft executing successive CCO fulfill the separation minima specified in the wake turbulence regulation, which ensures that the wake turbulence generated by the previous aircraft has no effect on the following aircraft. The following equations of motion describe the aircraft's dynamics:

$$\begin{aligned}
\dot{V}(t) &= \frac{T(t) - D(h_e(t), V(t), C_L(t)) - m(t) \cdot g \cdot \sin\gamma(t)}{m(t)}, \\
\dot{\gamma}(t) &= \frac{L(h_e(t), V(t), C_L(t)) - m(t) \cdot g \cdot \cos\gamma(t)}{m(t) \cdot V(t)}, \\
\dot{x}_e(t) &= V(t) \cdot \cos\gamma(t), \\
\dot{h}_e(t) &= V(t) \cdot \sin\gamma(t), \\
\dot{m}(t) &= -T(t) \cdot \eta(V(t)).
\end{aligned} \tag{22}$$

The kinematic equations in (22) are expressed in a ground-based reference frame, whereas the dynamic equations are expressed in an aircraft-attached reference frame. The state variables are captured in the state vector $\mathrm{x}(t) = (V(t), \gamma(t), x_e(t), h_e(t), m(t))$, where V is the true airspeed, γ is the flight path angle, x_e is the horizontal position, h_e is the altitude, and m is the mass of the aircraft. The control vector is $\mathrm{u}(t) = (T(t), C_L(t))$, where T is the engine thrust, and C_L is the lift coefficient. Drag, $D = C_D S_w \hat{q}$, and lift, $L = C_L S_w \hat{q}$, are the components of the aerodynamic force, with $\hat{q} = \frac{1}{2}\rho V^2$ being the dynamic pressure, and S_w the reference wing surface area. A parabolic drag polar $C_D = C_{D0} + K C_L^2$ is assumed. The lift coefficient, C_L, is a known function of the angle of attack and the Mach number. Parameter η is the fuel efficiency.

The flight envelope constraints are those derived from the aerodynamic and structural characteristics of the aircraft and the engine power.

$$\begin{aligned}
&0 \le h_e(t) \le \min[h_{M_0}, h_u(t)], && \gamma_{min} \le \gamma(t) \le \gamma_{max}, \\
&M(t) \le M_{M_0}, && m_{min} \le m(t) \le m_{max}, \\
&\dot{V}(t) \le a_l, && C_v V_s(t) \le V(t) \le V_{M_0}, \\
&\dot{\gamma}(t) V(t) \le a_n, && 0 \le C_L(t) \le C_{L_{max}}, \\
&T_{min}(t) \le T(t) \le T_{max}(t), && \mu(t) \le \bar{\mu}.
\end{aligned} \tag{23}$$

The parameters in (23), as well as the performance limitations model, have been obtained from BADA. These parameters are the maximum operational altitude, h_{M_0}, the maximum operative altitude at a given mass, $h_u(t)$, the Mach number, $M(t)$, the maximum operating Mach number, M_{M_0}, the maximum normal and longitudinal accelerations for civilian aircraft, a_n and a_l, respectively, the minimum speed coefficient, C_v, the stall speed, $V_s(t)$, and the maximum operating Calibrated Airspeed (CAS), V_{M_0}. Finally, $T_{min}(t)$ and $T_{max}(t)$ are the minimum and maximum available thrust, respectively, and $\bar{\mu}$ is the maximum bank angle due to structural limitations. In the experiments, the chosen lower and upper bounds for the flight path angle are $\gamma_{min} = -0.05°$ and $\gamma_{max} = 11.5°$, respectively.

The optimal trajectory to be followed by the simulated aircraft has been generated using a pseudospectral optimal control method minimizing fuel consumption [27]. It is a CCO associated with the Standard Instrument Departure (SID) Madrid Barajas PINAR1U. The definition of the SID can be found in the Spanish Aeronautical Information Publication (AIP) service[4], managed by ENAIRE. The path to be followed starts at the altitude of 1000 m after takeoff and ends at 10000 m, when the aircraft reaches the cruise level. The initial velocity and path angle are 130 m/s and 7.3°, respectively. The initial mass of the aircraft has been set to 60000 kg.

The experiments have been carried out in a simulated environment structured in the following main blocks:

- a realistic flight simulator, which includes the aircraft model and perturbations affecting the flight such as weather disturbances, model uncertainties, and measurement errors, and
- an ILC controller, composed of the GPR estimator and predictor of the disturbances affecting the aircraft and a nonlinear programming solver that calculates the updated control input for the following iteration.

The weather disturbances considered in the trajectory tracking experiments are the wind and the non-standard atmosphere. The wind model includes the combined effect of the mean horizontal wind speed and Dryden turbulence, which have been generated using built-in MATLAB functions[5]. The mean horizontal wind model has been simulated using the U.S. Naval Research Laboratory Horizontal Wind Model routine [34]. The mathematical representation of the Dryden wind turbulence model described in the Military Specification MIL-F-8785C [35] has been used. The vertical component of the wind has been neglected. Fig. 3 shows a realization of the wind speed observed by the aircraft when following the trajectory described above. The atmosphere model described in the military climatic standard MIL-STD-210C [36] has been employed in the trajectory tracking experiments, whereas the ISA model has been used for the generation of the trajectory to be followed.

The flight simulator has been developed in Simulink, a widely used software in aircraft simulation [37]. A 3-DOF longitudinal model of an Airbus A320 aircraft has been used, where the state, output, and control variables are those described in the previous sections.

[4]https://ais.enaire.es/AIP

[5]https://es.mathworks.com/help/aeroblks/wind.html

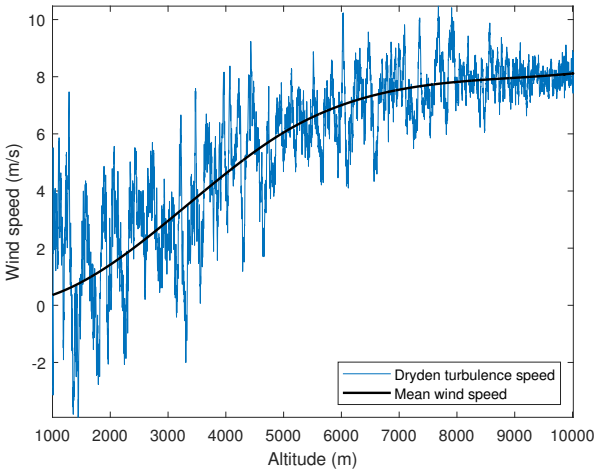

Fig. 3. Horizontal wind speed model.

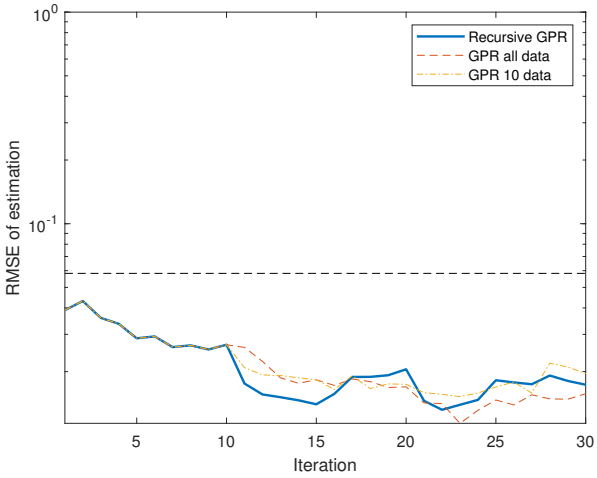

Fig. 4. Comparison of RMSE of estimation with different GPR approaches.

B. Results

The experiments have been conducted using the GPR and the Kalman filter methods for the estimation and prediction steps of the ILC in similar conditions and with different parameters to compare their performance.

Ten experiments have been performed consisting of 30 iterations each. At each iteration, the same aircraft intends to fly the CCO trajectory described above. The time between iterations is one hour. The figures show the median of the ten experiments of:

- the Root Mean Square Error (RMSE) of the scaled disturbance estimation at each iteration,
- the RMSE of the scaled disturbance prediction at each iteration for the following one, and
- the weighted state error, which is calculated as $e_{w,j} = ||\boldsymbol{S}\boldsymbol{y}_j||_2$, where $\boldsymbol{S}$ is the weighted scaling matrix of the output variables and $\boldsymbol{y}_j$ is the measured output vector at each iteration.

The black dashed horizontal line in Figures 4 to 9 is the output error standard deviation, which characterizes the system noise level. It is obtained from the variations in the trajectory when applying the same input to the aircraft several times.

1) Comparison between GPR methods: A comparison between the GPR method using all the available data, GPR using only data from last 10 iterations, and recursive GPR using 10 basis vectors is shown in Figures 4, 5, and 6. The training data used to define the basis vectors is obtained from the first 10 iterations as well as the initial guess of the hyperparameters. The non-recursive GPR methods have been implemented using MATLAB's embedded function for Gaussian process regression. As shown in the figures, after some iterations, the recursive GPR achieves, both in estimation and prediction, an RMSE similar to, and in some iterations even lower than the non-recursive GPR approach. All three methods exhibit a similar behavior when implemented into the ILC algorithm in terms of the weighted state error. The peaks observed at iterations 12 and 16 are caused by a change in the trend of the mean horizontal wind speed. Moreover, the

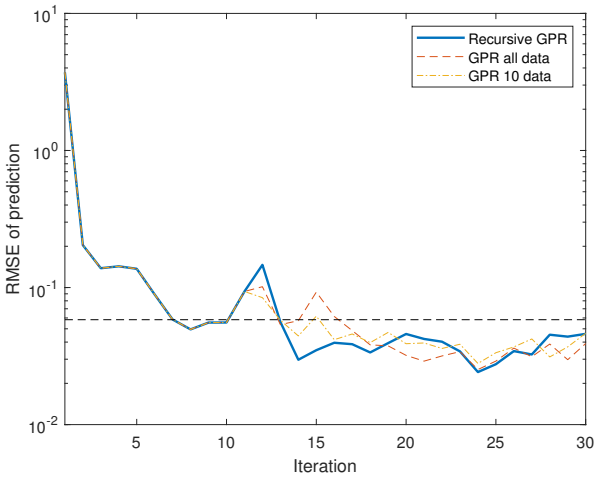

Fig. 5. Comparison of RMSE of prediction with different GPR approaches.

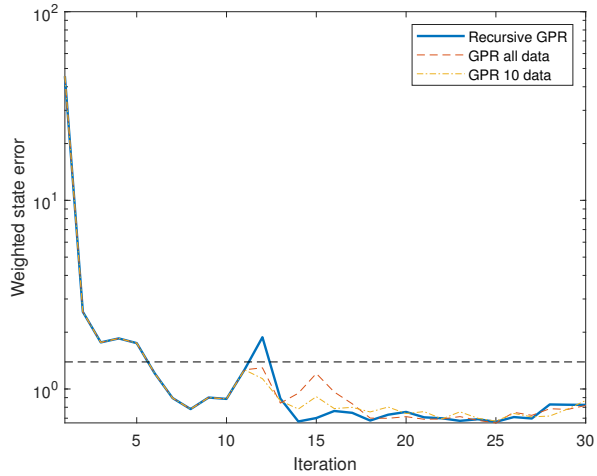

Fig. 6. Comparison of weighted state error with different GPR approaches.

difference between the wind speed values at these iterations and the previous ones is greater than in any other pair of consecutive iterations.

Table I shows the mean execution times of the Kalman filter and the non-recursive and recursive GPR approaches, observed on a standard laptop computer with a 1.6 GHz Intel Core i5 processor and 16 GB RAM. In the same table, the corresponding mean weighted state error after three iterations is also reported, showing that the non-recursive GPR approach is faster than the recursive GPR in reducing the tracking error when the training data is obtained from the previous 5 or 10 iterations. In both approaches, the weighted state error converges to its minimum value when using training data from more than 20 past iterations. The execution time of the recursive GPR, although higher than that of the Kalman filter, is significantly lower than the execution time of the non-recursive GPR method. Note that the computational effort increases as the iterations take place and more data is available.

	Training data	Mean computation time (s)	Mean e_w after 3 iterations
Non-recursive GPR	5 iterations	1.5316	0.8840
	10 iterations	2.2788	0.9615
	20 iterations	3.0928	0.7063
	50 iterations	3.3923	0.7005
Recursive GPR	5 iterations	0.1186	1.0891
	10 iterations	0.1284	1.0162
	20 iterations	0.1633	0.7551
	50 iterations	0.7150	0.7697
Kalman filter		0.0003	1.3606

TABLE I
MEAN COMPUTATION TIMES PER ITERATION AND WEIGHTED STATE ERROR OBSERVED USING A STANDARD LAPTOP COMPUTER.

2) Comparison between GPR and Kalman filter: As explained above, one of the advantages of using GPR is that no previous knowledge about the system dynamics is needed, in contrast to the Kalman filter, which requires the noise and measurement covariance matrices to be set. The recursive GPR for estimation and prediction has been compared here to the Kalman filter's performance in the ILC method in three different cases: with $\mathbf{\Omega} = 0.1\boldsymbol{M}$, with $\mathbf{\Omega} = \boldsymbol{M}$, and with $\mathbf{\Omega} = 10\boldsymbol{M}$, where $\mathbf{\Omega}$ is the covariance matrix of the process noise and $\boldsymbol{M}$ is the covariance matrix of the measurement noise.

Figures 7 and 8 show that the recursive GPR approach achieves better estimations and prediction of the disturbances than any of the Kalman filters. In the first iterations, recursive GPR and the Kalman filter with $\mathbf{\Omega} = 10\boldsymbol{M}$ are comparable and much faster than the Kalman filter with $\mathbf{\Omega} = \boldsymbol{M}$ and $\mathbf{\Omega} = 0.1\boldsymbol{M}$. In terms of the weighted state error, Fig. 9 shows that recursive GPR and the Kalman filter with $\mathbf{\Omega} = 10\boldsymbol{M}$ are comparable and that only after 20 iterations do all three methods return similar results.

Fig. 10 shows an enlarged view of the last part of the path described by the aircraft in the first iterations using GPR and a Kalman filter with $\mathbf{\Omega} = \boldsymbol{M}$. In the first iteration, the inputs fed to the aircraft are the nominal ones obtained in the trajectory generation where disturbances are not taken into account, and the resulting path is clearly deviated from the desired one.

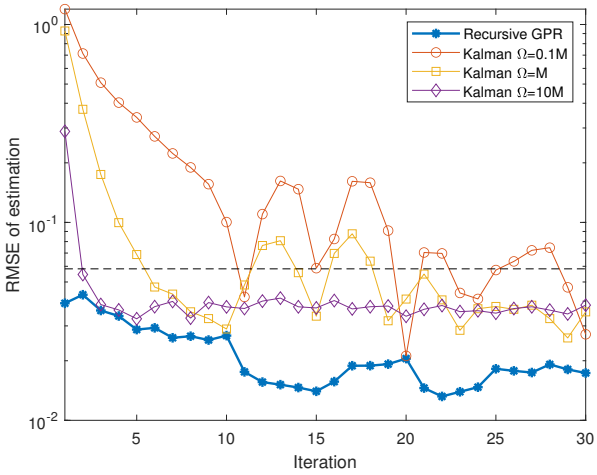

Fig. 7. RMSE of estimation with recursive GPR and Kalman filter.

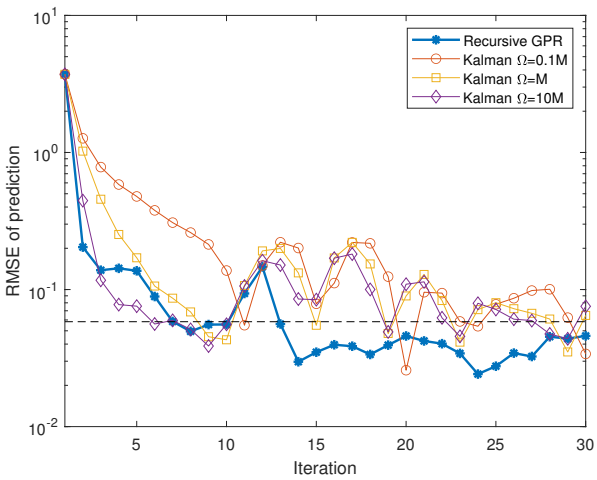

Fig. 8. RMSE of prediction with recursive GPR and Kalman filter.

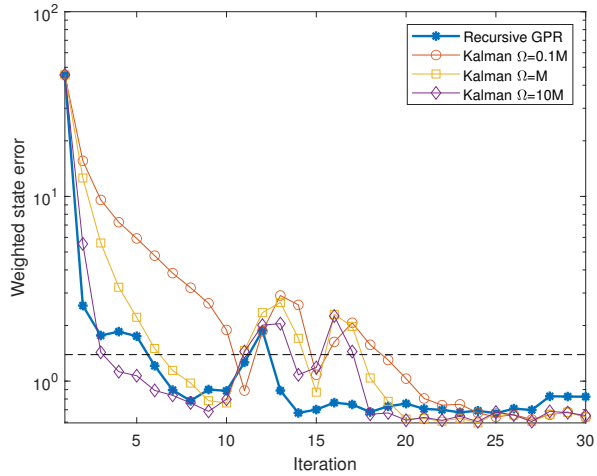

Fig. 9. Weighted state error with recursive GPR and Kalman filter.

In the following iterations, the ILC method amends these deviations. Fig. 10 shows that with GPR estimation and prediction, the convergence to the desired path is faster than with the Kalman estimator until reaching the best achievable accuracy, which is represented by the path that corresponds to iteration 25. It can be seen that the error between the actual and the desired paths is significantly reduced, but is not zero due to the aircraft's operational limits and the system's noise, which cannot be accurately predicted.

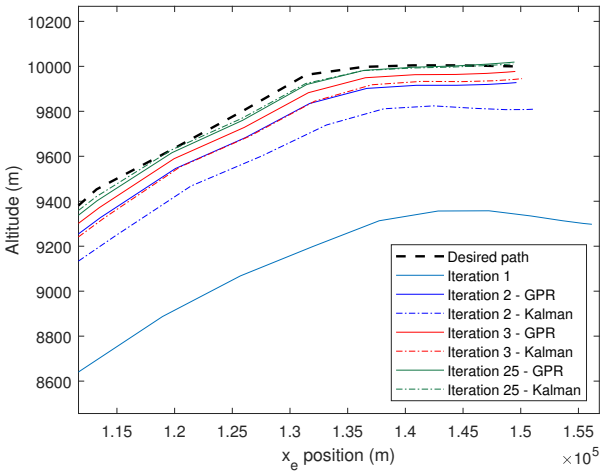

Fig. 10. Enlarged view of the path $x_e - h$ described by the aircraft using GPR and Kalman predictions of the disturbances.

Finally, Fig. 11 shows the control inputs generated by the ILC method when the estimation and prediction steps are performed using GPR and the Kalman filter. It can be observed that, as in Fig. 10, with GPR, the control inputs converge faster to the ones that drive the aircraft to the most precise achievable tracking of the trajectory (iteration 25) than those generated when using the Kalman filter.

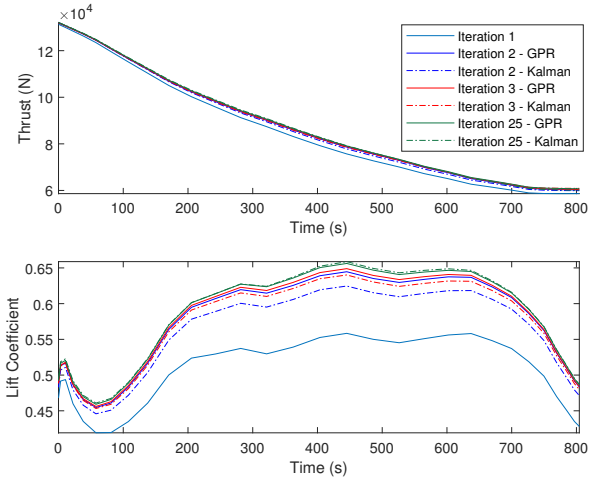

Fig. 11. Evolution of the thrust and lift coefficient over iterations using GPR and Kalman predictions of the disturbances.

VI. Conclusion

Estimation and prediction are key for precision and fast convergence of ILC, since the update of the inputs given to the aircraft is based on the predicted perturbations and the estimated model errors that cause deviations from the planned trajectory. In this paper, GPR has been implemented as the estimation and prediction step of an optimization-based ILC method to compensate for the disturbances affecting a sequence of identical aircraft performing the same trajectory. In order to reduce the high computational effort required by GPR, a recursive approach has been used, which performs the regression on a set of points that are updated at each iteration to dismiss old data in favor of the most recent measurements, simultaneously learning the hyperparameters.

With no prior knowledge of the behavior of the disturbances, the experiments show that recursive GPR provides much better estimations and predictions in the first iterations compared to the Kalman filter with $\Omega = 0.1M$ and $\Omega = M$, reaching a mean improvement of 80% and 67%, respectively, in the weighted state error along the first 5 iterations, therefore achieving faster and more precise tracking of the aircraft trajectory. The mean improvement across all 30 iterations is 62% and 42%, respectively. The results obtained by both methods are comparable only when the Kalman filter is correctly tuned, with $\boldsymbol{\Omega} = 10\boldsymbol{M}$, which means collecting some prior knowledge about the dynamics of the disturbances. Additionally, recursive GPR allows for estimation and prediction of nonlinear dynamics.

Acknowledgment

This work has been partially supported by the grants number TRA2017-91203-EXP and RTI2018-098471-B-C33 of the Spanish Government. We are also grateful to the UK EPSRC for the support through EP/T013265/1 project NSF-EPSRC:ShiRAS: Towards Safe and Reliable Autonomy in Sensor Driven Systems which is a joint project with the USA National Science Foundation under Grant NSF ECCS 1903466.

References

[1] Arimoto, S., Kawamura, S., and Miyazaki, F., "Bettering operation of robots by learning," Journal of Robotic Systems, vol. 1, no. 2, pp. 123–140, 1984.

[2] Moore, K. L., Dahleh, M., and Bhattacharyya, S. P., "Iterative learning control: A survey and new results," Journal of Robotic Systems, vol. 9, no. 5, pp. 563–594, 1992.

[3] Bristow, D. A., Tharayil, M., and Alleyne, A., "A survey of iterative learning control," IEEE Control Systems Magazine, vol. 26, pp. 2039–2114, 2006.

[4] Ahn, H., Chen, Y., and Moore, K. L., "Iterative learning control: Brief survey and categorization," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 6, pp. 1099–1121, 2007.

[5] Xu, J.-X. and Tan, Y., Linear and Nonlinear Iterative Learning Control. Springer, 2003.

[6] Xu, J.-X., Panda, S. K., and Lee, T. H., Real-time Iterative Learning Control: Design and Applications. Springer, 2009.

[7] Liang, X., Zheng, M., and Zhang, F., "A scalable model-based learning algorithm with application to UAVs," IEEE Control Systems Letters, vol. 2, no. 4, pp. 839 – 844, 2018.

[8] Schoellig, A. P., Mueller, F. L., and D'Andrea, R., "Optimization-based iterative learning for precise quadrocopter trajectory tracking," Autonomous Robots, vol. 33, no. 1-2, pp. 103–127, 2012.

[9] Buelta, A., Olivares, A., and Staffetti, E., "Iterative learning control for precise aircraft trajectory tracking in continuous climb operations," in Proceedings of the Thirteenth USA/Europe Air Traffic Management Research and Development Seminar, Vienna, Austria, June 2019.
[10] ——, "Iterative learning control for precise aircraft trajectory tracking in continuous descent approaches," in Proceedings of the 8th European Conference for Aeronautics and Aerospace Sciences, Madrid, Spain, July 2019.
[11] Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., Estimation with Applications to Tracking and Navigation. John Wiley & Sons, 2002.
[12] Hutchinson, C. E., "The Kalman filter applied to aerospace and electronic systems," IEEE Transactions on Aerospace and Electronic Systems, vol. AES-20, no. 4, pp. 500–504, 1984.
[13] Dunik, J., Straka, O., Simandl, M., and Blasch, E., "Random-point-based filters: Analysis and comparison in target tracking," IEEE Transactions on Aerospace and Electronic Systems, vol. 51, no. 2, pp. 1403–1421, 2015.
[14] Bonyan Khamseh, H., Ghorbani, S., and Janabi-Sharifi, F., "Unscented Kalman filter state estimation for manipulating unmanned aerial vehicles," Aerospace Science and Technology, vol. 92, pp. 446 – 463, 2019.
[15] Rasmussen, C. E. and Williams, C. K., Gaussian Processes for Machine Learning. MIT Press, 2006.
[16] Ko, J. and Fox, D., "GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models," Autonomous Robots, vol. 27, pp. 75–90, 2008.
[17] Van Gestel, T., Suykens, J., Lanckriet, G., Lambrechts, A., De Moor, B., and Vandewalle, J., "Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel Fisher discriminant analysis," Neural Computation, vol. 14, pp. 1115–1147, 2002.
[18] Perez-Cruz, F., Van Vaerenbergh, S., Murillo-Fuentes, J., Lázaro-Gredilla, M., and Santamaria, I., "Gaussian processes for nonlinear signal processing: An overview of recent advances," IEEE Signal Processing Magazine, vol. 30, pp. 40–50, 2013.
[19] Pillonetto, G., "A new kernel-based approach to hybrid system identification," Automatica, vol. 70, pp. 21 – 31, 2016.
[20] Kocijan, J., Girard, A., Banko, B., and Murray-Smith, R., "Dynamic systems identification with Gaussian processes," Mathematical and Computer Modelling of Dynamical Systems, vol. 11, 2005.
[21] Lawrence, N., Seeger, M., and Herbrich, R., "Fast sparse Gaussian process methods: The informative vector machine," in Proceeding of the 2002 Neural Information Processing Systems Conference, Vancouver, BC, Canada, December 2002.
[22] Snelson, E. and Ghahramani, Z., "Sparse Gaussian processes using pseudo-inputs," in Proceedings of the 18th International Conference on Neural Information Processing Systems, Cambridge, MA, USA, 2005.
[23] Williams, C. and Seeger, M., "Using the Nystrom method to speed up kernel machines," Advances in Neural Information Processing Systems, vol. 13, 2001.
[24] Brahim-Belhouari, S. and Bermak, A., "Gaussian process for nonstationary time series prediction," Computational Statistics and Data Analysis, vol. 47, pp. 705–712, 2004.
[25] Huber, M., "Recursive Gaussian process: On-line regression and learning," Pattern Recognition Letters, vol. 45, 2014.
[26] International Civil Aviation Organization,, "Continuous Climb Operations (CCO) Manual," ICAO, Tech. Rep. Doc. 9993 AN/495, 2012.
[27] Gong, Q., Fahroo, F., and Ross, I. M., "Spectral algorithm for pseudospectral methods in optimal control," Journal of Guidance, Control, and Dynamics, vol. 31, no. 3, pp. 460–471, 2008.
[28] Mouillet, V., "User Manual for the Base of Aircraft Data (BADA) Revision 3.14," EUROCONTROL Experimental Centre, Brétigny, France, Tech. Rep., 2017.
[29] Tousain, R., van der Meche, E., and Bosgra, O., "Design strategy for iterative learning control based on optimal control," in Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228), Orlando, FL, USA, December 2001.
[30] Bamieh, B., Pearson, J. B., Francis, B. A., and Tannenbaum, A., "A lifting technique for linear periodic systems with applications to sampled-data control," Systems & Control Letters, vol. 17, no. 2, pp. 79 – 88, 1991.
[31] Julier, S. J., "The scaled unscented transformation," in Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301), Anchorage, AK, USA, May 2002.
[32] Kandepu, R., Imsland, L., and Foss, B., "Constrained state estimation using the unscented Kalman filter," in Proceedings of the 2008 Mediterranean Conference on Control and Automation, Ajaccio, France, July 2008.
[33] Hull, D. G., Fundamentals of Airplane Flight Mechanics. Springer-Verlag, 2007.
[34] Drob, D. P., Emmert, J. T., Meriwether, J. W., Makela, J. J., Doornbos, E., Conde, M., Hernandez, G., Noto, J., Zawdie, K. A., McDonald, S. E., Huba, J. D., and Klenzing, J. H., "An update to the Horizontal Wind Model (HWM): The quiet time thermosphere," Earth and Space Science, vol. 2, no. 7, pp. 301–319, 2015.
[35] U.S. Department of Defense,, "The Flying Qualities of Piloted Airplanes," U.S. Federal Government, Tech. Rep. U.S. Military Specification MIL-F-8785C, 1980.
[36] ——, "Global Climatic Data for Developing Military Products," U.S. Federal Government, Tech. Rep. U.S. Military Specification MIL-STD-210C, 1987.
[37] Moin, L., Baig, A., and Uddin, V., "State space model of an aircraft using Simulink," International Journal of System Modeling and Simulation, vol. 2, no. 4, pp. 1–6, 2017.

Almudena Buelta is a Teaching Assistant of Statistics and a PhD student at the Universidad Rey Juan Carlos in Madrid, Spain. She received her MSc degree in Aeronautical Engineering from the Universidad Politécnica de Madrid. She previously worked for Airbus Defence and Space, developing training and operational documentation and providing operational support for in-service aircraft. Her research focuses on iterative learning control applied to commercial aircraft trajectory tracking.

Alberto Olivares is a Professor of Statistics and Vector Calculus at the Universidad Rey Juan Carlos in Madrid, Spain. He received his MSc degree in Mathematics and his BSc degree in Statistics from the Universidad de Salamanca, Spain, and his PhD degree in Mathematical Engineering from the Universidad Rey Juan Carlos. He has worked with the Athens University of Economics and Business. His research interests include statistical learning, stochastic hybrid optimal control, and model predictive control with applications to biomedicine, robotics, aeronautics, and astronautics.

Ernesto Staffetti is a Professor of Statistics and Control Systems at the Universidad Rey Juan Carlos in Madrid, Spain. He received his MSc degree in Automation Engineering from the Università degli Studi di Roma "La Sapienza," and his PhD degree in Advanced Automation Engineering from the Universitat Politècnica de Catalunya. He has worked with the Universitat Politècnica de Catalunya, the Katholieke Universiteit Leuven, the Spanish Consejo Superior de Investigaciones Científicas, and with the University of North Carolina at Charlotte. His research interests include stochastic hybrid optimal control, iterative learning control, and model predictive control with applications to robotics, aeronautics, and astronautics.

Waqas Aftab works as Deputy Director Command and Control Systems for the Pakistan Air Force. He received his B.Eng. (Avionics) degree from the College of Aeronautical Engineering (NUST), Pakistan, and MS (Control Systems Engineering) degree from Air University, Pakistan, in 2005 and 2014, respectively. His doctoral research was focused on using Gaussian processes for multiple target tracking and he completed his PhD at the University of Sheffield, UK, in 2020. His main interests are multisensor multitarget tracking and fusion.

Lyudmila Mihaylova is a Professor of Signal Processing and Control in the Department of Automatic Control and Systems Engineering at the University of Sheffield, Sheffield, United Kingdom. Her research interests are in the areas of autonomous systems with applications to cities, autonomous and assisted living systems. She has expertise in the areas of machine learning, intelligent sensing and sensor data fusion. She won the Tammy Blair best award from the International Conference of Information Fusion 2017, best paper awards from the IEEE DESSERT'2019, 17th IEEE SPA'2013 Conference and IEEE Sensor Data Fusion Workshop, 2013 and others. Prof. Mihaylova is on the Board of Directors of the International Society of Information Fusion (ISIF) and was the ISIF President in the period 2016–2018. She has given a number of talks and tutorials, including an invited talk in Harbin (Intelligent Navigation and Advanced Information Technology Workshop'2020), plenary talk in Cairo (JIC Smart Cities'2019), NATO SET-262 AI 2018 (Hungary), Fusion 2017 (Xi'an, China), plenary talks for the IEEE Sensor Data Fusion 2015 (Germany), invited talks at IPAMI Traffic Workshop 2016 (USA) and many others. She was the general vice-chair for the International Conference on Information Fusion 2018 (Cambridge, UK), of the IET Data Fusion & Target Tracking 2014 and 2012 Conferences, publications chair for ICASSP 2019 (Brighton, UK), program chair of Fusion 2020, on the organising committee of Fusion'2021, IEEE MFI'2021 and others.