Flexible-Time Receding Horizon Iterative Learning Control with Application to Marine Hydrokinetic Energy Systems

Mitchell Cobb[1*], James Reed[2*], Maxwell Wu[3], Kirti D. Mishra[4], Kira Barton[5], and Chris Vermillion[6]

***Abstract*—This paper presents an iterative learning control (ILC) framework for a class of repetitive control applications characterized by (i) continuous operation, (ii) flexible iteration time, and (iii) an economic performance metric. Specifically, the effect of iteration-varying initial conditions, resulting from the continuous nature of operation, is accounted for through an iteration domain receding horizon formulation. To address the need for flexible iteration times, the time-domain dynamics are transformed into path-domain dynamics characterized by a non-dimensional parameter spanning an iteration-invariant range. The resulting model is used to derive learning filters that minimize a multi-objective economic cost. The proposed methodology is applied to the control of a kite-based marine hydrokinetic (MHK) system, which executes high-speed, repetitive flight paths with the objective of maximizing its lap-averaged power output. The proposed approach is validated via simulations of a medium-fidelity nonlinear model of a kite-based MHK system, and the results demonstrate robust and fast convergence of the kite to power-optimal flight patterns.**

***Index Terms*—Iterative Learning Control, Receding Horizon Control, Optimal Path Following, Repetitive Control, Renewable Energy Systems, Marine Energy Systems**

I. Introduction

THe past several years have seen the advancement of numerous technologies that exhibit continuous cyclic operation and stand to benefit significantly from cycle-to-cycle learning. For example, in the autonomous racing example of [1], it is desirable to follow the predefined path (track) while minimizing lap time. In the case of an actively controlled prosthesis [2], the objective is to follow a spatially defined reference path for the gait, while minimizing or maximizing an energetic, metabolic, or rehabilitation-related metric. Lastly, a kite-based marine hydrokinetic (MHK) energy system, shown in Fig. 1, can generate more than an order of magnitude more power than an equivalently-sized stationary MHK system by executing high-speed, repetitive figure-eight flight patterns perpendicular to the prevailing flow [3].

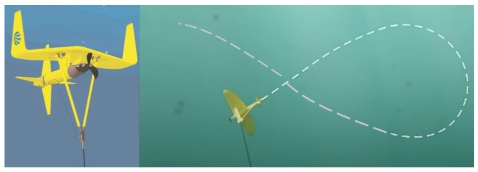
Fig. 1. Example of a kite-based marine hydrokinetic (MHK) energy system from Minesto. Image credit: Minesto, Ltd. Obtained with permission.

The aforementioned applications share three characteristics: (i) an economic performance objective, (ii) flexible cycle time, and (iii) continuous operation. It is not coincidental that the second and third features tend to appear when the first feature appears; restricting the cycle time severely restricts the space of control trajectories/sequences that can be explored, and requiring a pause between iterations can defeat the purpose of the economic optimization if time is critical.

Interestingly, the class of systems under consideration does not fall neatly within existing repetitive control (RC) or iterative learning control (ILC) frameworks. On one hand, the consideration of continuous operation (which causes non-resetting initial conditions) deviates from traditional ILC assumptions and satisfies RC assumptions. On the other hand, the systems being considered do not lend themselves to an internal model or a zero-error manifold, particularly owing to the focus on flexible cycle time and an economic (rather than tracking) performance objective.

Within the RC research community, techniques have been successfully used for control of continuously operating repetitive systems (e.g., in [4] and [5], for example). However, the focus largely has been on using cycle-to-cycle learning for output trajectory tracking, or achieving a balance between trajectory tracking and control input size via a quadratic cost formulation (see [6], [7]), all assuming fixed cycle times.

Turning to the ILC literature, point-to-point and Pareto optimal ILC approaches, such as those in [8]–[14], relax tracking requirements between prescribed waypoints, which can be leveraged to account for additional economic metrics to be minimized or maximized. However, the point-to-point ILC strategies described in the above-referenced papers all require

This brief was submitted for review on March 1st, 2021. This work was funded by National Science Foundation grant number 1727779. Additionally, the * symbol denotes equal contribution.

[1]Mitchell Cobb is a controls engineer at Blue Origin. mcobb@ncsu.edu.
[2]James Reed is an a PhD student at North Carolina State University jcreed2@ncsu.edu.
[3]Maxwell Wu is a PhD candidate at The University of Michigan maxwu@umich.edu.
[4]Kirti D. Mishra is a Post-doctoral Research Associate in the Department of Mechanical and Aerospace Engineering at North Carolina State University kdmishra@ncsu.edu.
[5]Kira Barton is an Associate Professor in the Department of Mechanical Engineering at The University of Michigan bartonkl@umich.edu.
[6]Chris Vermillion is an Associate Professor in the Department of Mechanical and Aerospace Engineering at North Carolina State University, Raleigh, NC 27695, USA cvermil@ncsu.edu. He is also a technical advisor and equity stakeholder for Altaeros Energies, Inc. and Windlift, Inc.

a fixed trial duration (while [15], [16], [17], and [18] allow for iteration-to-iteration variation in intermediate waypoint arrival times, they still fix the trial duration). While several studies within the ILC community, such as [19]–[21], allow for variable trial durations, they are restricted to tracking control. Furthermore, all of the aforementioned formulations assume an initial condition reset between iterations, which implies discontinuous operation (i.e., a pause between iterations). In fact, the existence of the pause and initial condition reset is so ingrained within ILC frameworks that it has often been deemed a defining distinction between RC and ILC.

A few research efforts from the ILC and model predictive control (MPC) communities have also investigated receding horizon approaches for repetitive tasks [22]–[29]. For example, [22]–[24] utilize prediction to derive learning filters, but still make classical assumptions of discontinuous operation, fixed iteration time, and a tracking objective. The assumptions of fixed iteration time and a tracking objective are lifted in [25] and [26], which fuses time-domain MPC with iterative learning, but an initial condition reset is assumed. In contrast, the use of learning MPC for autonomous racing in [27] and [28] does simultaneously address continuous operation, an economic objective (lap time), and flexible iteration time, but the formulation leverages a *time-domain* MPC optimization that requires an online optimization and dispenses with the attractive closed-form mathematical machinery present in traditional ILC formulations.

The present work proposes a flexible-time receding horizon ILC technique that leverages the well-established mathematical machinery and closed-form update formulation of norm-optimal ILC, with two new features that allow for flexible iteration time and continuous operation:

1) The lifted system dynamics are re-parameterized in path domain to allow for flexible cycle time.
2) Predictive control is performed in *iteration domain*, which allows the ILC update to consider iteration-to-iteration variation in initial conditions.

Ultimately, while the application of this algorithm falls closely under the realm of RC, the controller structure and implementation are more similar to strategies commonly found in the field of ILC. Hence, we refer to our approach as an ILC algorithm, despite the fact that we are leveraging ideas from both RC and ILC domains.

As a concrete case study, we focus on a kite-based MHK system. Here, the economic control objective is lap-averaged power output, which for a given path length and rotor power coefficient translates to maximization of lap-averaged speed. Given this correlation between the economic performance metric of interest and speed, flexible cycle time is a necessity. Furthermore, due to its continuous operation, the system's initial conditions vary from lap to lap. Consequently, the kite-based MHK system contains all of the features present in the class of systems targeted in the proposed ILC formulation. It is worth noting that the present case study *complements* but does not replace or duplicate earlier work from the authors, including [30], [31], [32], and [33], which focus on using iterative learning techniques to adjust the parameters/waypoints that define a flight path. In the present work, the focus lies on optimally traversing a set of specified waypoints, taking into account continuous operation.

II. Flexible-Time, Receding Horizon Iterative Learning Control: Mathematical Formulation

As noted earlier, the proposed methodology uses a transformation of the dynamic model from the time domain to spatial domain for control design. Towards this end, the spatial model is discretized and used to derive the lifted input-output model. The resulting lifted model is used to derive predictive and economically optimal learning filters, which are implemented in an iteration-domain receding horizon manner for improved robustness to inconsistent iteration-to-iteration initial conditions. A schematic of the ILC methodology proposed in this paper is shown in Fig. 2, wherein these learning filters are used inside the ILC Update block. These learning filters are spatial in nature and generate a control correction for the next trial in the spatial domain, which is parameterized by the non-dimensional path parameter s that spans an iteration-invariant range during every trial. The control input computed in the spatial domain for the next trial is transformed to the time domain via the look-up table of Fig. 2, which stores the control input values as functions of a set of waypoints. More specifically, this transformation amounts to looking up the control input value using the computed value of the non-dimensional path parameter during an iteration.

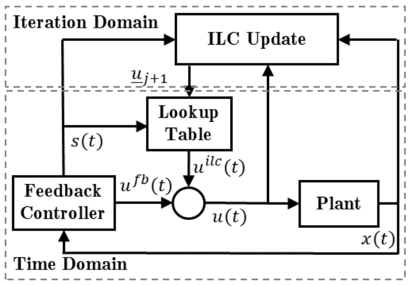

Fig. 2. Block diagram of the control structure used for the path-parameterized ILC update law.

A. Path Domain Lifted Model

Obtaining a path domain lifted model requires spatial re-parameterization of both the input-output dynamics of the plant and the state and control trajectories from the previous iteration. This was shown in [30]. Once the this re-paramaterization is accomplished, a discrete path-paramaterized system model is given by:

$$x_{q+1} = x_q + A_q\left(x_{q+1} - x_q\right) + B_q\left(u_{q+1} - u_q\right), \quad (1)$$

where $q \in \{1, 2, ..., n_s\}$ refers to a discrete index along the path, Δ_s is the user-specified path discretization level and s_q represents the path position at path step q, i.e., $s_q = q\Delta_s$. Note that this relationship is derived assuming linearization around a generic set of two sequences, $x_l(s)$ and $u_l(s)$. In subsequent sections we will always choose to linearize our system around the sequences achieved during the previous iteration, $x^j(s)$ and $u^j(s)$. Thus the discrete-path dynamics of (1) become

$$x_{q+1}^{j+1} = x_q^j + A_q^j \left(x_{q+1}^{j+1} - x_q^j\right) + B_q^j \left(u_{q+1}^{j+1} - u_q^j\right). \tag{2}$$

Given these discrete-path dynamics, and these definitions of the lifted vectors from the previous iteration j,

$$\begin{aligned} \underline{x}_j &\triangleq \left[\left(x_0^j\right)^T \quad \left(x_1^j\right)^T \quad \dots \quad \left(x_{n_s-1}^j\right)^T \quad \left(x_{n_s}^j\right)^T\right]^T, \\ \underline{u}_j &\triangleq \left[\left(u_0^j\right)^T \quad \left(u_1^j\right)^T \quad \dots \quad \left(u_{n_s-1}^j\right)^T \quad \left(u_{n_s}^j\right)^T\right]^T, \end{aligned} \tag{3}$$

it is now straightforward to derive a lifted system representation to relate the state sequence at the next iteration, $\underline{x}_{j+1}$ (defined similarly to (3)), to the state sequence from the previous iteration, $\underline{x}_j$, the control sequence from the previous iteration, $\underline{u}_j$, the control sequence for the next iteration, $\underline{u}_{j+1}$ (defined similarly to (3)), the initial condition for the previous iteration, x_0^j, and the initial condition for the next iteration, x_0^{j+1}:

$$\underline{x}_{j+1} = \underline{x}_j + G_j \left(\underline{u}_{j+1} - \underline{u}_j\right) + F_j \left(E_F - E_I\right) \underline{x}_j, \tag{4}$$

where the $n_x \times n_u$ block element in the mth block row and pth block column of G_j are given by:

$$G_{m,p}^j = \begin{cases} \mathbb{0}^{n_x \times n_u} & m < p \\ B_m^j & m = p \\ A_m^j A_{m-1}^j \dots A_p^j B_m^j & \text{otherwise,} \end{cases} \tag{5}$$

where $\mathbb{0}$ denotes the zero matrix. Similarly, the block element in the mth block row of F^j is given by

$$F_m^j = \begin{cases} \mathbb{I}^{n_x \times n_x} & m = 1 \\ A_m^j A_{m-1}^j \dots A_2^j A_1^j & \text{otherwise,} \end{cases} \tag{6}$$

where $\mathbb{I}$ denotes the identity matrix. Lastly, note that in (4), E_F and E_I are designed to select the first or last state vector from the lifted sequence, $\underline{x}_j$. Thus, they are defined as

$$E_F \triangleq \left[\mathbb{0}^{n_x \times n_x(n_s-1)} \quad \mathbb{I}^{n_x \times n_x}\right] \tag{7}$$

$$E_I \triangleq \left[\mathbb{I}^{n_x \times n_x} \quad \mathbb{0}^{n_x \times n_x(n_s-1)}\right]. \tag{8}$$

Note here that because the initial conditions are not reset between iterations, the final condition from iteration j is the initial condition of iteration $j+1$. Thus, the last term in (4) expresses the deviation in initial conditions between iteration j and iteration $j+1$.

B. Receding Horizon Iterative Learning Control

For many systems, it is suboptimal or infeasible to reset the initial conditions of the system between iterations. Without an initial condition reset, the control sequence from one iteration impacts the initial condition, and therefore the achievable performance, for the next iteration. Therefore, it is advantageous to account for this interdependence. To quantify this interdependence, we first develop a dynamic model that represents the behavior of the system over multiple future iterations, which is subsequently used to estimate the performance of the system over multiple future iterations and derive learning filters for optimizing this estimated performance. The procedure for deriving the update law and learning filters leverages fundamental tools [14], which are extended to account for multiple iterations. First, we write a performance index in terms of the lifted model, then we take the gradient of that performance index and set it equal to the zero vector, rearranging the result to obtain an update law and expressions for learning filters.

To clarify the following math and notation, we define a standard of notation and nomenclature. In previous sections, we referred to $\underline{x}_j$ and $\underline{u}_j$ as "lifted" vectors. That is, they are formed by concatenating a set of state or control input vectors ordered according to their path position (or time stamp) as in (3). We also referred to the matrices G_j and F_j as "lifted" system matrices. In this section, it will be necessary to concatenate multiple instances of vectors and matrices that are already lifted. We refer to these further concatenated vectors and block matrices as "super-lifted" to emphasize that they are different from but dependent upon vectors and matrices from the previous section which themselves are already lifted. These super-lifted vectors and matrices will be notated with a bold-faced symbol.

In general, the lifted model of (4) can be used to predict the state sequence resulting from any control input. Thus, if we wish to predict the state sequence over the $(j+n_i)$th iteration, we could write

$$\underline{x}_{j+n_i} = \underline{x}_j + G_j \left(\underline{u}_{j+n_i} - \underline{u}_j\right) + F_j \left(E_F - E_I\right) \underline{x}_{j+n_i-1}. \tag{9}$$

Noting the recursive form of this relationship, ($\underline{x}_{j+n_i}$ depends on $\underline{x}_{j+n_i-1}$), we can use it to derive a relationship between the super-lifted state and control vectors $\mathbf{x}_{j+1}$ and $\mathbf{u}_{j+1}$, which are defined as

$$\mathbf{x}_{j+1} \triangleq \begin{bmatrix} \underline{x}_{j+1} \\ \underline{x}_{j+2} \\ \vdots \\ \underline{x}_{j+N_i-1} \\ \underline{x}_{j+N_i} \end{bmatrix}, \quad \text{and} \quad \mathbf{u}_{j+1} \triangleq \begin{bmatrix} \underline{u}_{j+1} \\ \underline{u}_{j+2} \\ \vdots \\ \underline{u}_{j+N_i-1} \\ \underline{u}_{j+N_i} \end{bmatrix}, \tag{10}$$

respectively. The relationship between $\mathbf{x}_{j+1}$ and $\mathbf{u}_{j+1}$ is then given as

$$\mathbf{x}_{j+1} = \left(\mathbf{I}_x + \mathbf{F}_j\right) x_j + \mathbf{G}_j \left(\mathbf{u}_{j+1} - \mathbf{I}_u u_j\right) \tag{11}$$

where

$$\mathbf{F}_j \triangleq \begin{bmatrix} \mathbb{I} \\ \mathbb{I} + F_j E_F \\ \vdots \\ \mathbb{I} + \sum\limits_{k=1}^{N_i-1} \prod\limits_{m=1}^{k} F_j E_F \\ \mathbb{I} + \sum\limits_{k=1}^{N_i} \prod\limits_{m=1}^{k} F_j E_F \end{bmatrix}, \tag{12}$$

$$\mathbf{G}_j \triangleq \begin{bmatrix} G_j & \mathbb{0} & \dots & \dots & \mathbb{0} \\ F_j E_F G_j & G_j & \dots & \dots & \mathbb{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \left(\prod\limits_{m=1}^{N_i-1} F_j E_F\right) G_j & \left(\prod\limits_{m=1}^{N_i-2} F_j E_F\right) G_j & \dots & G_j & \mathbb{0} \\ \left(\prod\limits_{m=1}^{N_i} F_j E_F\right) G_j & \left(\prod\limits_{m=1}^{N_i-1} F_j E_F\right) G_j & \dots & F_j E_F G_j & G_j \end{bmatrix} \tag{13}$$

and $\mathbf{I}_{(\cdot)}$ is defined as an appropriately sized identity matrix. Similarly, we can calculate a super-lifted error vector as

$$\mathbf{e}_{j+1} = \mathbf{I}_r\underline{r} - \mathbf{x}_{j+1} \tag{14}$$

where $\underline{r}$ is the lifted reference signal vector constructed from an ordered set of reference states $\{r_0, r_1, \dots, , r_{n_s-1}, r_{n_s}\}$, where $r_q \in \mathbb{R}^{n_x}$ according to the same method as shown in (3). Here, we assume that at every point along the path (or in time), there is a reference state r with the same number of elements as x. In practice, however, one can build a reference state vector that contains the desired values of the states for only the indices considered by the tracking penalty in the subsequent learning filters.

We then select the following general form of performance index:

$$\begin{aligned}J_{j+N}=\sum_{k=1}^{N}\big(&\underline{u}_{j+k}^TQ_u\underline{u}_{j+k}+\left(\underline{u}_{j+k}-\underline{u}_{j+k-1}\right)^TQ_{\delta u}\left(\underline{u}_{j+k}-\underline{u}_{j+k-1}\right)\\&+\underline{x}_{j+k}^TQ_x\underline{x}_{j+k}+\left(\underline{x}_{j+k}-\underline{x}_{j+k-1}\right)^TQ_{\delta x}\left(\underline{x}_{j+k}-\underline{x}_{j+k-1}\right)\\&+\underline{e}_{j+k}^TQ_e\underline{e}_{j+k}+\left(\underline{e}_{j+k}-\underline{e}_{j+k-1}\right)^TQ_{\delta e}\left(\underline{e}_{j+k}-\underline{e}_{j+k-1}\right)\\&-S_xx_{j+k}\big).\end{aligned} \tag{15}$$

The diagonal positive definite weighting matrices of the quadratic terms, Q_u, $Q_{\delta u}$, Q_x, $Q_{\delta x}$, Q_e, and $Q_{\delta e}$ are defined in detail in [34], but they follow the standard norm-optimal point-to-point structure detailed in other work such as [14].

In order to accommodate economic objectives, we introduce the last term of (15), namely S_xx_{j+k}. The quantity $S_x \in \mathbb{R}^{n_xn_s}$ here is taken to be a weighted linear approximation (or exact representation) of the economic objective to be maximized or minimized. If the economic objective is described by $J_e(\underline{x}, \underline{u})$, then

$$S_x \triangleq \begin{bmatrix} -\nabla_x J_e(x,u)|_{\substack{x=x_l(s_1)\\u=u_l(s_1)}} \\ \vdots \\ -\nabla_x J_e(x,u)|_{\substack{x=x_l(s_{n_s})\\u=u_l(s_{n_s})}}\end{bmatrix}^T Q_{sx}, \tag{16}$$

where Q_{sx} is a diagonal weighting matrix designed to weight and also normalize the states in the overall performance.

The performance index of (15) can now be written in an equivalent super-block structure using the super-lifted vectors $\mathbf{u}_{j+1}$, and $\mathbf{u}_{j+1}$:

$$\begin{aligned}J_{j+N} =&\mathbf{u}_{j+1}^T\mathbf{Q_u}\mathbf{u}_{j+1} + \mathbf{u}_{j+1}^T\mathbf{D}_u^T\mathbf{Q}_{\delta\mathbf{u}}\mathbf{D}_u\mathbf{u}_{j+1}\\&+\left(\mathbf{E}_u\mathbf{u}_{j+1}-u_j\right)^TQ_{\delta u}\left(\mathbf{E}_u\mathbf{u}_{j+1}-u_j\right)\\&+\mathbf{x}_{j+1}^T\mathbf{Q_x}\mathbf{x}_{j+1}+\mathbf{x}_{j+1}^T\mathbf{D}_x^T\mathbf{Q}_{\delta\mathbf{x}}\mathbf{D}_x\mathbf{x}_{j+1}\\&+\left(\mathbf{E}_x\mathbf{x}_{j+1}-x_j\right)^TQ_{\delta x}\left(\mathbf{E}_x\mathbf{x}_{j+1}-x_j\right)\\&+\mathbf{e}_{j+1}^T\mathbf{Q_e}\mathbf{e}_{j+1}+\mathbf{e}_{j+1}^T\mathbf{D}_e^T\mathbf{Q}_{\delta\mathbf{e}}\mathbf{D}_e\mathbf{e}_{j+1}\\&+\left(\mathbf{E}_e\mathbf{e}_{j+1}-e_j\right)^TQ_{\delta e}\left(\mathbf{E}_e\mathbf{e}_{j+1}-e_j\right)\\&-\mathbf{S_x}\mathbf{x}_{j+1}\end{aligned} \tag{17}$$

where $\mathbf{D}_u \in \mathbb{R}^{(N_i-1)n_sn_u\times N_in_sn_u}$, $\mathbf{D}_x \in \mathbb{R}^{(N_i-1)n_sn_u\times N_in_sn_u}$, and $\mathbf{D}_e \in \mathbb{R}^{(N_i-1)n_sn_u\times N_in_sn_u}$ are difference operator matrices designed to calculate the difference between sequences of subsequent iterations according to

$$\mathbf{D}_{(\cdot)} = \begin{bmatrix}\mathbb{1} & -\mathbb{1} & \mathbb{0} & \dots\\ \mathbb{0} & \mathbb{1} & -\mathbb{1} & \dots\\ \vdots & \vdots & \vdots & \ddots\end{bmatrix}, \tag{18}$$

where the identity matrix has dimensions of $n_{(\cdot)}n_s \times n_{(\cdot)}n_s$. Note that $\mathbf{D}_{(\cdot)}$ has $N_i - 1$ block rows and N_i block columns. Additionally, the matrices $\mathbf{E}_u$, $\mathbf{E}_u$, and $\mathbf{E}_u$ are defined in such a way as to select the first lifted vector from the super-vector. Specifically,

$$\mathbf{E}_{(\cdot)} \triangleq \begin{bmatrix}\mathbb{1}^{n_{(\cdot)}n_s\times n_{(\cdot)}n_s} & \mathbb{0}^{n_{(\cdot)}n_s\times(N_i-1)n_{(\cdot)}n_s}\end{bmatrix}. \tag{19}$$

By using (11) in (14) and (17), we can obtain an expression for the performance index completely in terms of the known quantities from the last iteration, $\underline{u}_j$ and $\underline{x}_j$, along with the control sequences over the next several iterations $\mathbf{u}_{j+1}$. By then differentiating this expression for performance $\mathbf{J}_{j+1}$ with respect to the elements of $\mathbf{u}_{j+1}$, setting the result equal to the zero vector, and re-arranging the resulting expression, we obtain an update law of the form

$$\mathbf{u}_{j+1} = L_uu_j + L_ee_j + L_xx_j + L_c, \tag{20}$$

where

$$L_0 \triangleq \left(\hat{\mathbf{Q}}_\mathbf{u} + \mathbf{G}_j^T\left(\hat{\mathbf{Q}}_\mathbf{x}+\hat{\mathbf{Q}}_\mathbf{e}\right)\mathbf{G}_j\right)^{-1} \tag{21}$$

$$L_u \triangleq L_0\left(\mathbf{E}_u^TQ_{\delta u} + \mathbf{G}_j^T\left(\hat{\mathbf{Q}}_\mathbf{x}+\hat{\mathbf{Q}}_\mathbf{e}\right)\mathbf{G}_j\mathbf{I}_u\right) \tag{22}$$

$$L_x \triangleq L_0\mathbf{G}_j^T\left(\mathbf{E}_x^TQ_{\delta x} - \hat{\mathbf{Q}}_\mathbf{x}\mathbf{I}_x - \hat{\mathbf{Q}}_\mathbf{x}\mathbf{F}_j - \hat{\mathbf{Q}}_\mathbf{e}\mathbf{F}_j\right) \tag{23}$$

$$L_e \triangleq L_0\mathbf{G}_j^T\left(\hat{\mathbf{Q}}_\mathbf{e}\mathbf{I}_e - \mathbf{E}_e^TQ_{\delta e}\right) \tag{24}$$

$$L_c \triangleq -\frac{1}{2}L_0\mathbf{G}_j^T\mathbf{S}_x^T \tag{25}$$

and

$$\hat{\mathbf{Q}}_\mathbf{u} \triangleq \mathbf{Q_u} + \mathbf{D}_u^T\mathbf{Q}_{\delta\mathbf{u}}\mathbf{D}_u + \mathbf{E}_u^TQ_{\delta u}\mathbf{E}_u \tag{26}$$

$$\hat{\mathbf{Q}}_\mathbf{x} \triangleq \mathbf{Q_x} + \mathbf{D}_x^T\mathbf{Q}_{\delta\mathbf{x}}\mathbf{D}_x + \mathbf{E}_x^TQ_{\delta x}\mathbf{E}_x \tag{27}$$

$$\hat{\mathbf{Q}}_\mathbf{e} \triangleq \mathbf{Q_e} + \mathbf{D}_e^T\mathbf{Q}_{\delta\mathbf{e}}\mathbf{D}_e + \mathbf{E}_e^TQ_{\delta e}\mathbf{E}_e. \tag{28}$$

It should be emphasized here that this update law produces the super-lifted vector $\mathbf{u}_{j+1}$, which contains control sequences for multiple future iterations. Thus, while it would be technically possible to perform multiple iterations based on this single update law, a receding horizon framework is used here, wherein the control input corresponding to the first of multiple sequences computed is implemented for control. Similar to MPC and other predictive control schemes, the receding horizon framework results in increased robustness of the ILC methodology presented here, particularly through its ability to consider the impact on performance of iteration-to-iteration variations in initial conditions.

Convexity of the cost function in (17) with respect to $\mathbf{u}_{j+1}$ (the decision variable) follows from statements: (i) the weighing matrices $Q_\mathbf{u}, Q_{\delta\mathbf{u}}, Q_{\delta u}, Q_\mathbf{x}, Q_{\delta\mathbf{x}}, Q_{\delta x}, Q_\mathbf{e}, Q_{\delta e}, Q_{\delta\mathbf{e}}$ are positive definite, (ii) the selector matrices $\mathbf{E}_x, \mathbf{E}_u, \mathbf{E}_e$ and the difference operator matrix $\mathbf{D}_x, \mathbf{D}_u, \mathbf{D}_e$ have full row ranks, (iii) the linearity of the lifted systems in (11), and

(iv) full rank of the input-output matrix $\mathbf{G}_j$ in (11). Using these, the cost function in (17) can be expressed as a quadratic function of $\mathbf{u}_{j+1}$ with a positive definite Hessian.

A convergence analysis for the proposed approach is detailed in [35] and summarized here for self-containment. From [35], a vector $\underline{z}_j$ can be defined as

$$\underline{z}_j \triangleq \begin{bmatrix} \underline{u}_j \\ \underline{x}_j^0 \end{bmatrix} \tag{29}$$

where $\underline{x}_j^0$ are the initial conditions at iteration j. The dynamic equation governing $\underline{z}_j$ can be written as

$$\underline{z}_{j+1} = \mathbf{A}_{z_j} \underline{z}_j + \underline{\eta}_j(\underline{d}) \tag{30}$$

where $\mathbf{A}_{z_j}$ describes the natural dynamics of the input sequence and initial conditions, and $\underline{\eta}_j(\underline{d})$ captures the effects of the environment on the dynamics. Under this setup, the convergence of $\underline{z}_j$ is established under two assumptions: (i) The spectral radius of $\mathbf{A}_{z_j}$ must be less than 1 for all iterations, and (ii) $\mathbf{A}_{z_j}$ and $\underline{d}_j$ must converge to constants $\mathbf{A}_z$ and $\underline{d}$, respectively. Under those assumptions, the convergence result is stated mathematically as:

$$\lim_{j \to \infty} \underline{z}_j = (\mathbf{I} - \mathbf{A}_z)^{-1} \underline{\eta}_j(\underline{d}). \tag{31}$$

Hence, if conditions (i) and (ii) are satisfied, a closed-form expression for the converged value of the input sequence and initial conditions can be generated. In practice, assumption (i) can be satisfied with a stabilizing lower-level controller, which can be verified analytically or (for complex systems such as the MHK kite) via simulation studies. Condition (ii) is satisfied if the rate of change of the environment is slow relative to the convergence of the controller. In practice, we show in Section IV that convergence is actually quite robust to the rate of change in the environment, at least for the kite application.

III. Case Study: A Kite-Based MHK System

To illustrate the effectiveness of the proposed flexible-time receding horizon ILC framework, we consider a kite-based MHK system, as described in Section I and Fig. 1 and 3. As noted, kite-based MHK systems can generate significantly more power than similarly sized stationary devices by flying high-speed cross-current flight paths [3], perpendicular to the prevailing flow as shown in Fig. 3.

Given the characteristics of the kite flight control problem, the ILC strategy developed in previous sections of this work allows us to address (i) an economic metric (lap-averaged power output), (ii) continuous operation (the kite cannot stop between laps), and (iii) flexible lap time (while the ultimate objective is net power output, higher flight speeds will result in increased power output, but higher flight speeds are associated with decreased lap time).

To validate our proposed approach, we utilize a dynamic model first introduced in [31] and [32]. In formulating the ILC problem (in particular, in selecting the control signals to be updated at each cycle), we note that successful cross-current flight for a kite-based MHK system hinges upon three requirements, namely (i) efficiently flying a cross-current path, (ii) controlling the lifting body's *attitude* (orientation) so that it maximizes the apparent flow speed, and (iii) simultaneously controlling the lifting body's position and attitude so that it tracks waypoints along the path. In order to address each of these goals, we select the heading angle setpoint (ψ_{SP}) as the control variable for the ILC update. This setpoint is provided to a lower level controller. By manipulating this control variable, we are able to influence both power production and path following.

To simplify the lower-level tracking control problem (related to requirements ii and iii above), while preserving fundamental attributes of the upper-level flight optimization problem (which relates directly to requirement i), we utilize a model referred to as the "unifoil" model. This model fully characterizes the longitudinal dynamics of the undersea system, which are critical to power production, while imposing a kinematic constraint (a "unicycle" constraint) whereby the kite-based MHK system is constrained to move only in the direction that it is pointed. This allows us to solve the path following problem relatively easily, which in turn allows us to focus on validating the use of an iterative learning process to optimize the control sequences.

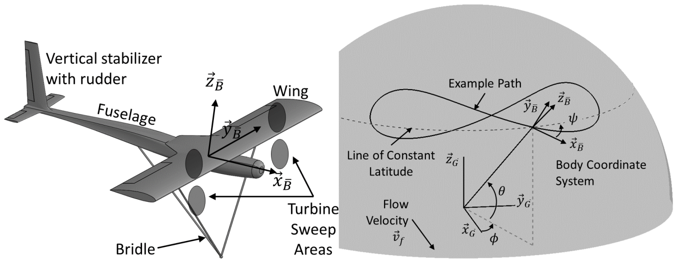

Fig. 3. Schematic of numerical unifoil MHK model. The top subfigure shows a close up of the system. The middle subfigure shows a wide-angle view of the system including the polar coordinate system characterized by ϕ, and θ, an example system path, the tether, the sphere on which the system operates, and the heading angle, ψ.

A. 3D "Unifoil" Model

This model is based on the design of a rigid wing system with on-board turbines, similar to the system shown in Fig. 1 and shown schematically in Fig. 3. The complete mathematical details of this dynamic model have been covered extensively in prior work such as [30], [31], [32], and [34], but a brief summary is provided here for completeness. The lower level model reference controller actuates the rudder to generate a turning or yawing moment. The system's kinematics are simplified through the following constraints, which are reflective of a system with (i) stiff tethers, (ii) limited sideslip, and (iii) a high lift/weight ratio:

1) The kite is constrained to lie on a sphere centered at the origin with radius r. This is also equivalent to assuming that the tether(s) is always perfectly taut.
2) The kite only translates on the sphere in the direction that the fuselage is pointing. This is the familiar no-slip kinematic constraint.

3) The body-fixed x-z plane (i.e., the longitudinal plane) is constrained to be perpendicular to the sphere. This assumption is appropriate when lift forces significantly exceed gravitational forces. In these cases, the kite system requires minimal roll, relative to the sphere, to stay aloft.

Under these constraints, the system can be completely described by ten variables: azimuth angle, ϕ, elevation angle, θ, heading angle, ψ, speed, v, angular rotation rate, ω, and their first derivatives, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, $\dot{v}$, and $\dot{\omega}$. Note that the heading angle, ψ, is the complement of the velocity angle popularized in work such as [36].

B. Path Following Rudder Controller

Because of the kinematic modeling constraints introduced in section III-A, the rotational dynamics of this system are dominated by the effects of the vertical stabilizer, whose control surface is the rudder. The closed-loop controller for the rudder is composed of two blocks, namely the path following block, which is responsible for choosing a heading angle setpoint, u_r^{fb}, and the model reference controller, which is responsible for selecting a rudder angle, u_r to follow the prescribed heading angle setpoint, ψ_{SP}. This control strategy has been used in [31] and [32]. However, a summary is included here for completeness.

1) Definitions: Path Shape, Path Variable, and Path Position: The *path shape* is a parametric curve in three dimensions, with a radius equal to the current radial position of the system. The other two spherical coordinate variables, azimuth, p_ϕ, and elevation, p_θ, are described by the lemniscate of Gerono [37]:

$$\begin{bmatrix} p_\phi(\sigma) & p_\theta(\sigma) \end{bmatrix} = \begin{bmatrix} \frac{\phi_0}{2}\sin(2\pi\sigma)+\phi_1 & -\frac{\theta_0}{2}\sin(4\pi\sigma)+\theta_1 \end{bmatrix}. \tag{32}$$

Here, the constant scalars ϕ_0, ϕ_1, θ_0 and θ_1 are user-defined parameters that define the overall shape of the path.The *path variable* is a generic parametric variable, denoted here by σ, that corresponds to specific points in space along the path. In this work, we have structured (32) so that σ is normalized, but this is not required. The *path position* is the closest point on the path, $\vec{p}(\sigma)$, relative to the current position of the system. Mathematically, the path position is defined as

$$s(t) = \arg\min_{\sigma} \left(\phi(t) - p_\phi(\sigma)\right)^2 + \left(\theta(t) - p_\theta(\sigma)\right)^2. \tag{33}$$

2) Path Following Controller: The path following controller is implemented as a pure-pursuit type controller. At every time step, this controller performs three steps:

1) It calculates the current path position by solving (33) using the numerical techniques described in [30], [34], and [38]. These techniques also apply the constraint that the value of s must be within a finite range of the previous value. This resolves many of the complexities that arise from self-intersecting paths (i.e., the figure-eight).
2) Then, given this path position s, it adds a user-specified constant value to s to obtain a "target" path position.
3) Last, it calculates the great-sphere heading that connects the current position and the target path position. This is the heading angle setpoint, u_r^{fb}.

3) Model Reference Controller: Given a heading angle setpoint, ψ_{SP}, this block computes a rudder deflection angle, u_r, using the same second order reference model detailed in previous work such as [30], [34], and [38].

C. Kite-based MHK-Specific Implementation of Flexible-Time Receding Horizon ILC

This section describes how the general algorithm of section II is tailored to the kite-based MHK application. Because variable iteration duration is imperative for the optimization of the economic metric – the lap-averaged power generation – spatial learning filters derived using the path-parameterized dynamic model of section II-A are used. As the lifted form spatial model is a function of the path used for model transformation, and the path taken by the kite changes from one iteration to the next, the learning filters are iteration-varying. The control input sequence computed in the spatial domain for the next trial is used to update the look-up table of Fig. 3 before the start of the next trial, during which the path position, computed in real time, is use to retrieve the corresponding control input, thereby transforming the spatial domain control input to the time domain.

The analytical, closed-form expressions for key quantities relating to the linearization and reparamaterization of the kite system's dynamics are far too large to be included in this paper. This is due to the combined effect of (i) the highly nonlinear form of the dynamic model and (ii) the complexity of the path shape in (32). However, the full MATLAB, Simulink, and Mathematica code used to generate the results in this section are available online at https://github.com/NCSUCORE/unifoilFTILC.

IV. Results

To test the effectiveness of this method, we simulated an MHK unifoil system with a 10 meter tip-to-tip wingspan in a spatiotemporally constant flow environment. All of the parameters used in these simulations for the plant, environment and controller may be viewed in our online repository at https://github.com/NCSUCORE/unifoilFTILC. In order to maximize power generation, we select the weights of the economic S_x term of (15) to incentivize maximization of the speed state of the model. For this work, we chose to place waypoints at $s = \frac{1}{4}$, $s = \frac{1}{2}$, $s = \frac{3}{4}$, and $s = 1$.

Fig. 4 shows how the shape of the achieved flight path changes from the first iteration to the last iteration. Additionally, the nominal path that is used to calculate path position and waypoint locations is shown as a finely dotted gray line. The waypoints are also indicated on the plot. We can see from this figure that the achieved flight path over the first iteration does not accurately track the waypoints, whereas the final flight path does. We can also see that the final flight path starts and ends at nearly the same point, suggesting that the initial conditions have converged by the final iteration.

Fig. 5 shows how the total performance index changes from iteration to iteration, the mean speed of the kite from iteration to iteration, and illustrates the convergence of the waypoint tracking term in the performance index, namely $e_j^T Q_e e_j$. Note that this performance index is the actual achieved performance, which is not the same as (15) or (17), which capture predicted values of future performance over multiple iterations. Instead, this plot shows the actual value of the performance index as achieved by the system on the last iteration. Finally, Fig. 6 shows the power factor, defined as

$$P_f = \frac{\int_{t_i}^{t_f} \|\vec{v}_a\|^3 dt}{(t_f - t_i) v_f^3}. \quad (34)$$

This scalar valued function is the ratio of the power that was produced during cross-current flight to the power that would have been produced if the system had been sitting still (non-cross-current flight). Thus, this number captures the effectiveness of cross-current flight in augmenting the energy generation of the system. Figures 4 through 6 were created using a horizon length of two iterations.

To show the robustness of our receding horizon iterative learning control (RHILC) algorithm, we simulated the system with modeling uncertainty in a flow speed that was unknown to the ILC algorithm. This uncertainty took the form of a constant flow disturbance, with results shown in Fig. 7, and a turbulent flow disturbance created using the turbulence model in [39], with results shown in Fig. 8. In both cases the RHILC algorithm can be seen to converge. For the constant disturbance case, we compared the RHILC algorithm's performance to the performance of a converged control input, created by running RHILC to convergence at a constant flow of 1 m/s. In Fig. 7, RHILC can be seen to outperform the converged input under a 1 m/s flow speed (which represents the performance of an offline-optimized solution based on the incorrect flow speed).

To see the effect of horizon length on convergence speed, horizon lengths of 2, 3, 4, 5 and 6 iterations are compared in Fig. 9. It can be seen that as the horizon length increases, the convergence speed also increases.

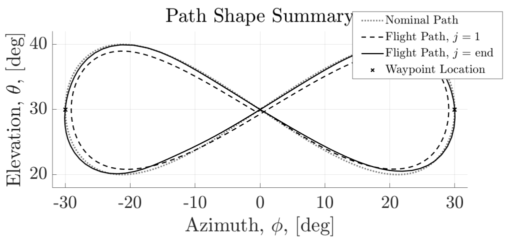

Fig. 4. Path shape over the first and last iteration. Note that the achieved flight path over the last iteration tracks the waypoints much more closely than the first iteration.

V. Conclusions & Future Work

An iterative learning control (ILC) framework was presented for repetitive control applications with continuous operation, flexible iteration times, and economic performance metrics. The framework was validated for kite-based MHK systems. The use of prediction in the presented framework was shown to result in iterative improvement of the economic metric in the presence of iteration-varying initial conditions resulting from the continuous operation of these systems, and the use of model transformation to a spatial domain was shown to enable iterative learning using trials with different iteration durations. Using medium-fidelity simulations, it was shown that the ILC method converged to optimal motion patters in 10-15 iterations, starting from different inaccurate initial control input settings. Further, it was observed that convergence to the optimal control setting was monotonic in nature.

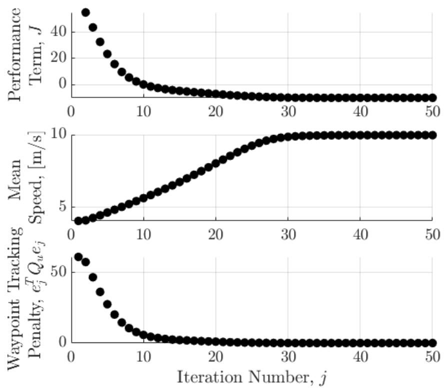

Fig. 5. Performance index, mean kite speed, and waypoint tracking over the course of 50 iterations. Note that lower values indicate better performance for performance and waypoint tracking, and higher values indicate better performance for mean speed.

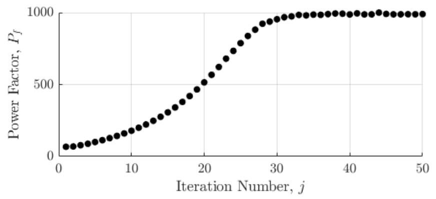

Fig. 6. Power factor, P_f versus iteration number. A larger power factor indicates better system performance.

References

[1] Roborace, 2020. [Online]. Available: https://roborace.com/

[2] Locomotor Control Systems Laboratory at The University of Michigan, 2020. [Online]. Available: https://gregg.engin.umich.edu/

[3] M. Loyd, "Crosswind kite power," *Journal of Energy*, vol. 4, no. 3, pp. 106–111, 1980.

[4] S. Hara, Y. Yamamoto, T. Omata, and M. Nakano, "Repetitive control system: a new type servo system for periodic exogenous signals," *IEEE Transactions on Automatic Control*, vol. 33, no. 7, pp. 659–668, 1988.

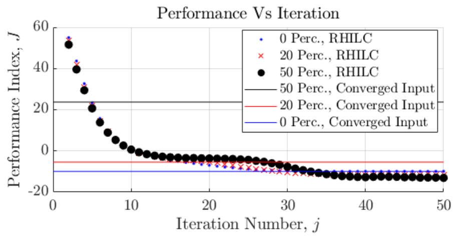

Fig. 7. Results showing the RHILC's performance as compared to the "converged input" case.

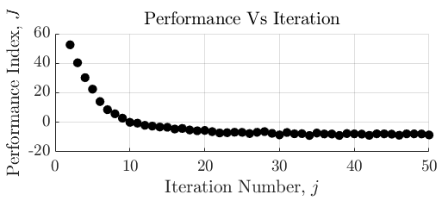

Fig. 8. Performance index when the kite is experiencing a turbulence intensity of 30 percent at frequencies between 0.1 and 1 hz.

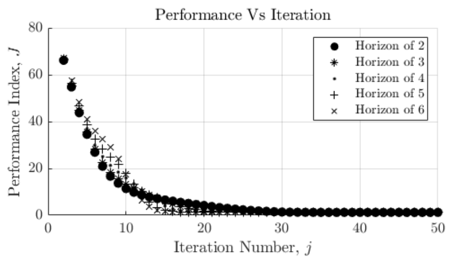

Fig. 9. Performance index over different horizon lengths, J versus iteration number. Note that lower values indicate better performance.

[5] Li Cuiyan, Zhang Dongchun, and Zhuang Xianyi, "A survey of repetitive control," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, 2004, pp. 1160–1166 vol.2.

[6] L. Wang, C. T. Freeman, S. Chai, and E. Rogers, "Multivariable repetitive-predictive control of a robot arm with experimental results," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 7672 – 7677, 2011, 18th IFAC World Congress.

[7] L. Wang, S. Chai, and E. Rogers, "Predictive repetitive control based on frequency decomposition," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 4277–4282.

[8] D. H. Owens, C. T. Freeman, and T. Van Dinh, "Norm-optimal iterative learning control with intermediate point weighting: Theory, algorithms, and experimental evaluation," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 999–1007, 2013.

[9] T. Duy Son, H.-S. Ahn, and K. Moore, "Iterative learning control in optimal tracking problems with specified data points," *Automatica*, vol. 49, p. 1465–1472, 05 2013.

[10] B. Chu, C. T. Freeman, and D. H. Owens, "A novel design framework for point-to-point ILC using successive projection," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1156–1163, 2015.

[11] C. Freeman and Y. Tan, "Iterative learning control with mixed constraints for point-to-point tracking," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 604–616, 2012.

[12] C. Freeman, "Constrained point-to-point iterative learning control with experimental validation," *Control Engineering Practice*, vol. 20, no. 5, pp. 489–498, 2012.

[13] C. Freeman, Z. Cai, E. Rogers, and P. Lewin, "Iterative learning control for multiple point-to-point tracking application," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 590–600, 2010.

[14] I. Lim and K. Barton, "Pareto optimization-based iterative learning control," *Proceedings of the American Control Conference*, 2013, Washington, D.C.

[15] Y. Chen, B. Chu, and C. T. Freeman, "Spatial path tracking using iterative learning control," *IEEE Conference on Decision and Control*, 2016, Las Vegas, NV.

[16] ——, "Point-to-point iterative learning control with optimal tracking time allocation," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1685–1698, 2018.

[17] ——, "Point-to-point iterative learning control with optimal tracking time allocation," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1685–1698, 2018.

[18] M. Wu, M. Cobb, C. Vermillion, and K. Barton, "A flexible-time iterative learning control framework for linear, time-based performance objectives," *Proceedings of the American Control Conference*, pp. 4792–4797, 2020.

[19] R. Longman and K. Mombaur, *Investigating the Use of Iterative Learning Control and Repetitive Control to Implement Periodic Gaits*. Berlin, Heidelberg: Springer, 2006, pp. 189–218.

[20] T. Seel, T. Schauer, and J. Raisch, "Monotonic convergence of iterative learning control systems with variable pass length," *International Journal of Control*, vol. 90, no. 3, pp. 393–406, 2017.

[21] P. M. Sammons, D. Hoelzle, and K. Barton, "Time scale transformed ILC for a class of nonlinear systems with uncertain trial duration," *IEEE Transactions on Control Systems Technology*, vol. PP, pp. 1–8, 2019.

[22] N. Amann, D. H. Owens, and E. Rogers, "Predictive optimal iterative learning control," *International Journal of Control*, vol. 69, no. 2, pp. 203–226, 1998.

[23] L. Wang and E. Rogers, "Predictive iterative learning control using laguerre functions," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5747–5752, 2011.

[24] B. Chu, D. H. Owens, and C. T. Freeman, "Iterative learning control with predictive trial information: convergence, robustness, and experimental verification," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1101–1108, 2015.

[25] P.-C. Lu, J. Chen, and L. Xie, "Iterative learning control (ilc)-based economic optimization for batch processes using helpful disturbance information," *Industrial & Engineering Chemistry Research*, vol. 57, no. 10, pp. 3717–3731, 2018.

[26] Y. Long, L. Xie, and S. Liu, "Nontracking type iterative learning control based on economic model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 18, pp. 8564–8582, 2020.

[27] U. Rosolina, A. Carvhalo, and F. Borrelli, "Autonomous racing using learning model predictive control," *Proceedings of the 2017 IFAC World Congress*, 2017, Toulouse, France.

[28] M. Brunner, R. Ugo, J. Gonzales, and F. Borelli, "Repetitive learning model predictive control: An autonomous racing example," *Proceedings of the 56th Conference on Decision and Control*, 2017, Melbourne, Australia.

[29] N. Scianca, U. Rosolia, and F. Borrelli, "Learning model predictive control for periodic repetitive tasks," in *2020 European Control Conference (ECC)*, 2020, pp. 29–34.

[30] M. Cobb, M. Wu, K. Barton, and C. Vermillion, "Flexible-time economic iterative learning control: A case study in airborne wind energy," in *2019 IEEE 58th Annual Conference on Decision and Control (CDC)*, 12 2019.

[31] M. K. Cobb, K. Barton, H. Fathy, and C. Vermillion, "Iterative learning-based path optimization for repetitive path planning, with application to 3-d crosswind flight of airborne wind energy systems," *IEEE Transactions on Control Systems Technology*, pp. 1–13, 2019.

[32] M. Cobb, K. Barton, H. Fathy, and C. Vermillion, "An iterative learning approach for online flight path optimization for tethered energy systems undergoing cyclic spooling motion," in *2019 American Control Conference (ACC)*, 7 2019, pp. 2164–2170.

[33] M. Cobb, J. Reed, J. Daniels, A. Siddiqui, M. Wu, H. Fathy, K. Barton, and C. Vermillion, "Iterative learning-based path optimization with application to marine hydrokinetic energy systems," *IEEE Transactions on Control Systems Technology*, 2021.

[34] M. Cobb, "Economic iterative learning control with application to tethered energy systems," Ph.D. dissertation, North Carolina State University, 2020.

[35] M. Wu, M. Cobb, J. Reed, K. Mishra, C. Vermillion, and K. Barton, "Receding Horizon Iterative Learning Control for Continuously Operated Systems," *arXiv:2108.06866 [cs, eess]*, Aug. 2021, arXiv: 2108.06866. [Online]. Available: http://arxiv.org/abs/2108.06866

[36] L. Fagiano, A. Zgraggen, M. Morari, and M. Khammash, "Automatic crosswind flight of tethered wings for airborne wing energy: Modeling, control design, and experimental results," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1433–1447, 2014.

[37] J. Delgado and J. M. Pena, "Progressive iterative approximation and bases with the fastest convergence rates," *Computer Aided Geometric Design*, vol. 24, no. 1, pp. 10–18, 2007.

[38] M. Cobb, K. Barton, H. Fathy, and C. Vermillion, "Iterative learning-based waypoint optimization for repetitive path planning, with application to airborne wind energy systems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 2698–2704.

[39] P. Pyakurel, J. H. VanZwieten, M. Dhanak, and N. I. Xiros, "Numerical modeling of turbulence and its effect on ocean current turbines," *International Journal of Marine Energy*, vol. 17, pp. 84–97, 2017.