A Flexible-Time Iterative Learning Control Framework for Linear, Time-Based Performance Objectives

Maxwell J. Wu[1], Mitchell Cobb[2], Chris Vermillion[3], and Kira Barton[4]

Abstract— **This paper describes a method for optimizing a user-defined, time-based, linear system performance index through the use of a flexible-time iterative learning control (ILC) framework. This method utilizes a two stage design wherein a point-to-point ILC procedure is conducted to improve sparse reference tracking performance, followed by a linear programming optimization that updates the system timing to minimize the time-based performance cost. A guarantee of strictly monotonic improvement of system performance cost is presented. The technique is applied to a simulated servo positioning system subject to input saturation constraints in order to minimize the time required to track a sequence of waypoints. This framework relaxes restrictions of traditional ILC techniques that require a fixed trial length to allow ILC to be applied effectively to a broader range of system objectives.**

I. INTRODUCTION

While optimal control strategies may be employed to improve a system's performance, these methods rely on the availability of an accurate system model. In the presence of model uncertainty, control signals developed through these techniques will often lead the system to behave in a sub-optimal or undesirable manner. To counteract these effects, Iterative Learning Control (ILC) can be used to improve the performance of systems that operate repetitively. This is accomplished by learning from a system's previous executions of a task to update a feedforward control sequence.

Often, a system's performance is closely linked to its temporal behavior. Thus, the ability to update the system timing requirements from one iteration of the task to the next can be necessary for an ILC framework to optimize system performance. Importantly, enabling system timing to be used as a design parameter would allow for the improved performance of many systems that already utilize ILC techniques. For instance, in manufacturing applications where ILC is used to reduce tracking error during device construction [1], updates to the timing requirements of the system could be used to increase throughput. Meanwhile, such a 'flexible-time' framework would also allow ILC to be applied to new fields. For example, ILC has not traditionally been used in airborne wind energy applications where performance is measured by the amount of energy generated by a system as it traverses along a repetitive flight path. These systems, whose energy harvesting abilities are correlated to their flight velocity [2], would also benefit from an optimization of the system timing along the flight trajectory.

Traditionally, ILC applications have been limited to systems that execute a task identically at each iteration [3]. In particular, ILC is typically used to reduce tracking error over a predefined reference trajectory with a fixed trial duration. However, the restrictions of a reference tracking objective and fixed trial duration greatly limit how the control sequence may be constructed, and therefore how much system performance can be improved.

Advancements in ILC have relaxed reference tracking restrictions. In [4], a point-to-point framework grants flexibility to the shape of the trajectory by enforcing reference tracking at only a few select locations. This idea is extended in [5] by leveraging the increased freedom in the control sequence design to optimize broader performance objectives over the trajectory. In both cases, however, the trial duration and timing requirements of the system are iteration invariant.

Other developments in ILC have relaxed the requirement for a fixed trial duration. In [6], trial duration is treated as a stochastic variable to accommodate learning in systems with randomly varying trial duration. A time-scale transformation is used in [7] to allow ILC to be applied to systems whose trial duration is uncertain before execution. Yet, neither of these techniques allow for the system timing to be used as a design variable for performance optimization.

Time is treated as a design parameter in [8] and [9] wherein the travel duration between points in a point-to-point scheme is modified to optimize performance. However, trial duration remains iteration-invariant in [8], and neither consider objective functions that are explicit functions of time. Alternatively, [10] presents a framework in which trial duration and system timing requirements are treated as a design parameter. Here, a path-coordinate based ILC update is used to minimize both path following error as well as trial duration. This method allows for the optimization of a broader class of objectives, but also requires complete knowledge of the desired path shape before every iteration.

While these developments have removed traditional restrictions on ILC, there is still a need for methods that leverage the flexibility afforded by both sparse tracking

This work is supported by National Science Foundation award numbers 1727371 and 1727779, entitled "Collaborative Research: An Economic Iterative Learning Control Framework with Application to Airborne Wind Energy Harvesting."

[1]Maxwell Wu is a PhD candidate in the Department of Mechanical Engineering at the Unversity of Michigan, Ann Arbor, MI, 48109, USA maxwu@umich.edu

[2]Mitchell Cobb is a PhD candidate in the Department of Mechanical and Aerospace Engineering at North Carolina State University, Raleigh, NC, 27695, USA mcobb@ncsu.edu

[3]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

[4]Kira Barton is an Associate Professor in the Department of Mechanical Engineering at the Unversity of Michigan, Ann Arbor, MI, 48109, USA bartonkl@umich.edu

enforcement and non-strict timing requirements. The main contribution of this paper is the development of an ILC framework that relaxes strict timing requirements to improve linear, time-based performance objectives subject to tracking and input saturation constraints, with provided guarantees on the convergence of system performance. In this manner the applicability and efficacy of ILC is broadened.

The paper is organized as follows. Section II details the lifted system representation of the considered classes of systems. Section III describes existing ILC techniques and their limitations. Section IV outlines the flexible-time ILC algorithm with a corresponding convergence assessment. A simulation implementation is presented in Section V with conclusions given in Section VI.

II. LIFTED SYSTEM MODELS

The class of systems addressed in this work are causal, single-input, single-output, linear, time-invariant, stable systems whose dynamics are expressed in continuous time as

$$\begin{aligned} \dot{\boldsymbol{x}}[t] &= \boldsymbol{A}\boldsymbol{x}[t] + \boldsymbol{B}u[t] \\ y[t] &= \boldsymbol{C}\boldsymbol{x}[t] \end{aligned} \tag{1}$$

where t denotes time, $\boldsymbol{x}[t] \in \mathbb{R}^{g}$ are the system states, $u[t] \in \mathbb{R}$ is an input signal, $y \in \mathbb{R}$ is the output, and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are continuous-time state space matrices.

If $u[t]$ is a zero-order held signal, system (1) then has a discrete time representation

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A_d}(k)\boldsymbol{x}(k) + \boldsymbol{B_d}(k)u(k) \\ y(k) &= \boldsymbol{C_d}(k)\boldsymbol{x}(k) \end{aligned} \tag{2}$$

with

$$\boldsymbol{A_d}(k) = e^{\boldsymbol{A}T(k)}, \quad \boldsymbol{B_d}(k) = \int_0^{T(k)} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau, \quad \boldsymbol{C_d}(k) = \boldsymbol{C} \tag{3}$$

where $k \in \mathbb{N}$ is a timestep index and $T(k)$ is the time period between samples k and $k+1$.

Suppose that there are N timestep intervals, $T(k)$, $k \in \{0, ..., N-1\}$. Additionally, assume that at each iteration, $\boldsymbol{x}(0) = 0$. Define the following sequences

$$\boldsymbol{u} = \begin{bmatrix} u(0) \\ \vdots \\ u(N-1) \end{bmatrix}, \boldsymbol{T} = \begin{bmatrix} T(0) \\ \vdots \\ T(N-1) \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}. \tag{4}$$

Then, $\boldsymbol{y}$ can be described using the lifted system [3]

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{u}. \tag{5}$$

Here $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ is a matrix with elements given by

$$H_{m,n}(\boldsymbol{T}) = \begin{cases} 0 & m < n \\ \boldsymbol{C_d}\boldsymbol{A_d}(m)\boldsymbol{A_d}(m-1)...\boldsymbol{A_d}(n+1)\boldsymbol{B_d}(n) & \\ \quad = \boldsymbol{C}e^{\boldsymbol{A}T_{sum}} \int_0^{T(n)} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau & m \geq n \end{cases} \tag{6}$$

where indexing starts at $m = n = 0$ and $T_{sum} = T(m) + T(m-1) + ... + T(n+1)$.

III. PRELIMINARIES

A. Standard norm-optimal ILC

Norm-optimal ILC (NOILC) is a commonly used ILC strategy that aims to optimize a quadratic function by minimizing various weighted costs to system performance [11]. Consider the case where tracking of reference sequence $\boldsymbol{y_r} = \left[y_r(1), \ \ldots, \ y_r(N)\right]^T$ is desired. Let j denote an iteration index. In NOILC, the timestepping sequence is held fixed such that $\boldsymbol{T}_j = \boldsymbol{T}$ for all j. From (6), this implies that $\boldsymbol{H}_j = \boldsymbol{H}(\boldsymbol{T})$ for all j. Define tracking error, $\boldsymbol{e} \in \mathbb{R}^{N}$, as

$$\boldsymbol{e}_j \triangleq \boldsymbol{y_r} - \boldsymbol{y}_j = \boldsymbol{y_r} - \boldsymbol{H}\boldsymbol{u}_j. \tag{7}$$

A quadratic cost function is then constructed as

$$J = \boldsymbol{e}_{j+1}^T \boldsymbol{Q} \boldsymbol{e}_{j+1} + \boldsymbol{u}_{j+1}^T \boldsymbol{S} \boldsymbol{u}_{j+1} + (\boldsymbol{u}_{j+1} - \boldsymbol{u}_j)^T \boldsymbol{R} (\boldsymbol{u}_{j+1} - \boldsymbol{u}_j) \tag{8}$$

where $\boldsymbol{Q}, \boldsymbol{S}, \boldsymbol{R} \in \mathbb{R}^{N \times N}$ are positive definite matrices that define the relative costs of tracking error, control effort, and iteration-to-iteration changes in input respectively [11], [12].

Setting the gradient of (8) with respect to $\boldsymbol{u}_{j+1}$ to zero yields the update law given in [13] as

$$\boldsymbol{u}_{j+1} = \boldsymbol{L_u}\boldsymbol{u}_j + \boldsymbol{L_e}\boldsymbol{e}_j \tag{9}$$

where

$$\begin{aligned} \boldsymbol{L_u} &= (\boldsymbol{H}^T\boldsymbol{Q}\boldsymbol{H} + \boldsymbol{S} + \boldsymbol{R})^{-1}(\boldsymbol{H}^T\boldsymbol{Q}\boldsymbol{H} + \boldsymbol{R}) \\ \boldsymbol{L_e} &= (\boldsymbol{H}^T\boldsymbol{Q}\boldsymbol{H} + \boldsymbol{S} + \boldsymbol{R})^{-1}\boldsymbol{H}^T\boldsymbol{Q}. \end{aligned} \tag{10}$$

B. Point-to-point ILC with input saturation constraints

While NOILC penalizes tracking error over an entire trajectory, this enforcement can be overly restrictive. For instance, in robotic pick and place applications, accurate tracking is required at only a few critical locations to ensure proper operation. Point-to-point ILC then relaxes trajectory following objectives by only penalizing error at a few points of interest, henceforth referred to as waypoints, within $\boldsymbol{y_r}$.

The waypoints, $W(\eta_\mu) \subset \boldsymbol{y_r}$, define a subsequence of the reference profile where η_μ is a timestep index with $\mu \in \{0, ..., N_w - 1\}$ and $N_w < N$ denotes the number of waypoints.

Define the elements of matrix $\boldsymbol{\Psi} \in \mathbb{R}^{N \times N}$ as

$$\Psi_{m,n} = \begin{cases} 1, & m = n = \eta_\mu - 1, \ \mu \in \{0, ..., N_w - 1\} \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Here, $\boldsymbol{\Psi}$ acts as a selection matrix that identifies the waypoint indices [4]. Suppose that the input signal is bounded by actuator limits which take the form of saturation constraints. Then, the constrained point-to-point ILC algorithm solves

$$\begin{aligned} \min_{\boldsymbol{u}_{j+1}} \quad & (\boldsymbol{\Psi}\boldsymbol{e}_{j+1})^T \boldsymbol{Q} (\boldsymbol{\Psi}\boldsymbol{e}_{j+1}) + (\boldsymbol{u}_{j+1} - \boldsymbol{u}_j)^T \boldsymbol{R} (\boldsymbol{u}_{j+1} - \boldsymbol{u}_j) \\ \text{s.t.} \quad & u_{min} \leq \boldsymbol{u}_{j+1} \leq u_{max}. \end{aligned} \tag{12}$$

In (12), tracking error is only penalized at the waypoints.

The constrained ILC and constrained point-to-point ILC problems are studied in [14], [15]. These techniques allow for sparse tracking enforcement, however, restrictions on an iteration-invariant $\boldsymbol{H}$ forces the waypoints to have a non-modifiable location in time. This paper relaxes this restriction to enable improvement of new performance objectives.

IV. Flexible-Time ILC

A. The Flexible-Time ILC Problem

Due to the restriction on an iteration-invariant timestepping scheme, time-based performance objectives cannot be improved by using point-to-point ILC. Instead, to address systems whose performance is given by a linear function of its timesteps, the flexible-time ILC problem is introduced as

$$\underset{\boldsymbol{u},\boldsymbol{T}}{\text{minimize}} \quad J_{LP} = \mathbf{c}^T \boldsymbol{T} \tag{13}$$

$$\text{subject to} \quad \boldsymbol{T} \geq 0 \tag{14}$$

$$u_{min} \leq \boldsymbol{u} \leq u_{max} \tag{15}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{u} \tag{16}$$

$$\boldsymbol{\Psi}\boldsymbol{y_r} - \Delta \leq \boldsymbol{\Psi}\boldsymbol{y} \leq \boldsymbol{\Psi}\boldsymbol{y_r} + \Delta. \tag{17}$$

Here, $\mathbf{c} \in \mathbb{R}^{\mathrm{N}}$ defines the performance cost. Constraint (14) maintains that all timestep intervals must have a nonnegative duration, (15) enforces input saturation limits, while (16) and (17) require that the system output tracks the waypoints within tolerance $\Delta > 0$ subject to the system dynamics.

To solve the problem given by (13)-(17), a flexible-time ILC algorithm is proposed consisting of two iterative stages:

- **Stage 1** - *Inner-loop constrained point-to-point ILC:* Given a timestepping sequence $\boldsymbol{T}_j$, a point-to-point ILC algorithm is performed to update the input sequence as the solution to the problem

$$\begin{aligned}
\underset{\boldsymbol{u}_{j,i+1}}{\text{minimize}} \quad & J_{PTP} = q\left\|\boldsymbol{\Psi}\boldsymbol{e}_{j,i+1}\right\|_\infty + r\left\|\boldsymbol{u}_{j,i+1} - \boldsymbol{u}_{j,i}\right\|_\infty \\
\text{subject to} \quad & u_{min} \leq \boldsymbol{u}_{j,i+1} \leq u_{max} \\
& \boldsymbol{y}_{j,i+1} = \boldsymbol{H}_j\boldsymbol{u}_{j,i+1} \\
& \boldsymbol{e}_{j,i+1} = \boldsymbol{y_r} - \boldsymbol{y}_{j,i+1} = \boldsymbol{e}_{j,i} + \boldsymbol{H}_j\boldsymbol{u}_{j,i} - \boldsymbol{y}_{j,i+1}
\end{aligned} \tag{18}$$

  where $q, r > 0$ are weighting parameters and i is an inner-loop iteration index denoting the number of times (18) has been solved within outer-loop iterate j.
- **Stage 2** - *Timestep update:* The timestepping scheme is updated as the solution to the linear program

$$\begin{aligned}
\underset{\boldsymbol{T}_{j,h+1}}{\text{minimize}} \quad & J_{LP} = \mathbf{c}^T \boldsymbol{T}_{j,h+1} \\
\text{subject to} \quad & \lambda^h \boldsymbol{\delta T}_{min} \leq \boldsymbol{\delta T}_{j,h+1} \leq \lambda^h \boldsymbol{\delta T}_{max} \\
& \boldsymbol{\delta T}_{j,h+1} = \boldsymbol{T}_{j,h+1} - \bar{\boldsymbol{T}}_j
\end{aligned} \tag{19}$$

  where $\boldsymbol{\delta T}_{min}, \boldsymbol{\delta T}_{max} \in \mathbb{R}^{\mathrm{N}}$, $\lambda \in (0,1)$, $\bar{\boldsymbol{T}}_j$ is a nominal timestepping scheme, and h is an inner-loop iteration index corresponding to the number of times (19) has been solved within outer-loop iterate j.

Here, Stage 1 updates the input sequence to improve waypoint tracking while Stage 2 minimizes the cost, J_{LP}. The algorithm is depicted in block diagram form in Fig. 1.

B. Stage 0 - Initial warm-start solution

Before running the iterative algorithm, a nominal timestepping scheme, $\bar{\boldsymbol{T}}_0$, and input sequence, $\bar{\boldsymbol{u}}_0$, are needed where $\bar{\boldsymbol{T}}_0$ and $\bar{\boldsymbol{u}}_0$ satisfy (14) and (15) respectively. Additionally, $(\bar{\boldsymbol{u}}_0, \bar{\boldsymbol{T}}_0)$ must strictly satisfy the waypoint tracking constraint.

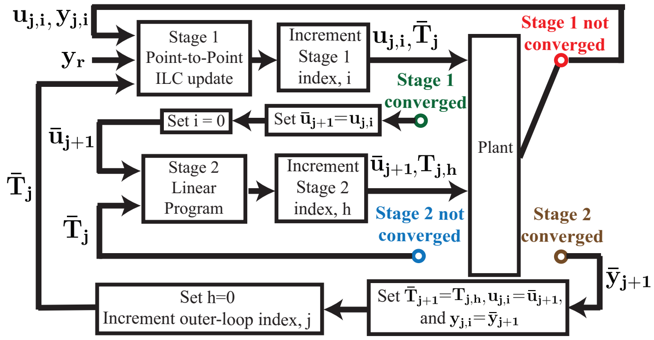

Fig. 1. To optimize performance, the flexible-time ILC algorithm iteratively performs 2 stages. The first stage utilizes a point-to-point ILC algorithm that updates the input sequence to improve waypoint tracking performance. The second stage uses a linear program that updates the timestepping sequence to improve the time-based system performance metric.

In other words, the output produced by applying $\bar{\boldsymbol{u}}_0$ over $\bar{\boldsymbol{T}}_0$ must satisfy (17) with strict inequalities.

C. Stage 1 - Constrained point-to-point ILC

The Stage 1 procedure reduces waypoint tracking error by updating the input sequence. First, set $\boldsymbol{u}_{j,0} \triangleq \bar{\boldsymbol{u}}_j$ and define $\boldsymbol{H}_j(\bar{\boldsymbol{T}}_j)$ according to (6). The input sequence update is performed by iteratively solving (18) with solution $\boldsymbol{u}_{j,i+1}$. $\boldsymbol{u}_{j,i+1}$ is then applied to the plant and reupdated until waypoint tracking converges according to

$$\left\|\boldsymbol{\Psi}(\boldsymbol{y_r} - \boldsymbol{y}_{j,i})\right\|_\infty = \left\|\boldsymbol{\Psi}\boldsymbol{e}_{j,i}\right\|_\infty \leq \varepsilon_i \tag{20}$$

for some sequence of convergence parameters, ε_i, satisfying

$$0 < \varepsilon_i < \varepsilon_{i+1} < \Delta, \forall i \text{ and } \lim_{i\to\infty} \varepsilon_i = \Delta. \tag{21}$$

After (20) is met, define $\bar{\boldsymbol{u}}_{j+1} \triangleq \boldsymbol{u}_{j,i}$ before commencing to Stage 2. Note that by setting $\varepsilon_i < \Delta$, an output sequence $\bar{\boldsymbol{y}}_j = \boldsymbol{y}_{j,i+1}$ that satisfies (20) also strictly satisfies (17).

D. Stage 2 - Timestep update

The Stage 2 procedure aims to reduce the cost, J_{LP}, by generating updated timestepping scheme, $\bar{\boldsymbol{T}}_{j+1}$. Additionally, $\bar{\boldsymbol{T}}_{j+1}$ is designed such that (14) is satisfied and (17) is strictly satisfied. Thus, careful selection of the bounds on the timestep updates in (19), $\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$, is required.

To develop $\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$, first suppose that a $\bar{\boldsymbol{T}}$ and $\bar{\boldsymbol{u}}$ are given such that constraints (14) and (15) are satisfied by $\boldsymbol{T} = \bar{\boldsymbol{T}}$ and $\boldsymbol{u} = \bar{\boldsymbol{u}}$, and that $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{T}})$ strictly satisfies (17). The output after k timesteps, $\bar{y}(k)$, is then given by (5) as

$$\begin{aligned}
\bar{y}(k) =& \boldsymbol{C}e^{\boldsymbol{A}(\bar{T}(1)+\ldots+\bar{T}(k-1))}\int_0^{\bar{T}(0)} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau\bar{u}(0) + \\
&\ldots + \boldsymbol{C}e^{\boldsymbol{A}\bar{T}(k-1)}\int_0^{\bar{T}(k-2)} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau\bar{u}(k-2) \\
&+ \boldsymbol{C}\int_0^{\bar{T}(k-1)} e^{\boldsymbol{A}\tau}\boldsymbol{B}d\tau\bar{u}(k-1).
\end{aligned} \tag{22}$$

The Taylor expansion of $\bar{y}(k)$ about $\bar{\boldsymbol{T}}$ is then given as

$$y(k) = \bar{y}(k) + \sum_{i=0}^{k-1} \frac{\partial \bar{y}(k)}{\partial T(i)}(T(i) - \bar{T}(i)) + R_1 \tag{23}$$

where R_1 is the Lagrange remainder and $\frac{\partial \bar{y}}{\partial T(i)}$ is given by

$$\begin{aligned}\frac{\partial \bar{y}(k)}{\partial T(i)} =& \boldsymbol{C} e^{\boldsymbol{A}(\bar{T}(i)+\ldots+\bar{T}(k-1))} \boldsymbol{B} \bar{u}(i) \\ &+ \sum_{l=0}^{i-1} \boldsymbol{C} e^{\boldsymbol{A}(\bar{T}(l+1)+\ldots+\bar{T}(k-1))} (e^{\boldsymbol{A}\bar{T}(l)} - \boldsymbol{I}) \boldsymbol{B} \bar{u}(l)\end{aligned} \tag{24}$$

where $\boldsymbol{I}$ is the identity matrix. Denote the change in timestep from the nominal value as $\delta T(\cdot) \triangleq T(\cdot) - \bar{T}(\cdot)$. If $\delta T(\cdot) \in [\delta T^-, \delta T^+]$ where $-\bar{T}(i) \leq \delta T^- \leq 0$ and $\delta T^+ \geq 0$, then

$$|R_1| \leq \frac{1}{2} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} p^{m,n} |\delta T(m) \delta T(n)| \tag{25}$$

over the set $\Theta = \{\boldsymbol{\delta T}^k : \delta T^- \leq \delta T(i) \leq \delta T^+, \forall i = 0, \ldots, k-1\}$, where

$$\begin{aligned} p^{m,n} = \max_{\boldsymbol{T}^k} \quad & \left| \frac{\partial^2 \bar{y}(k)}{\partial T(m) \partial T(n)} \right| \\ \text{subject to} \quad & \delta T^- \leq \boldsymbol{\delta T}^k \leq \delta T^+. \end{aligned} \tag{26}$$

Here $\boldsymbol{T}^k \in \mathrm{IR}^{\mathrm{k}}$ and $\boldsymbol{\delta T}^k \in \mathrm{IR}^{\mathrm{k}}$ represent sequences of the first k values of $T(\cdot)$ and $\delta T(\cdot)$ respectively. For $\rho = \min\{m, n\}$,

$$\begin{aligned}\frac{\partial^2 \bar{y}(k)}{\partial T(m) \partial T(n)} =& \boldsymbol{CA} e^{\boldsymbol{A}(\bar{T}(\rho)+\ldots+T(k-1))} \boldsymbol{B} \bar{u}(\rho) \\ &+ \sum_{l=0}^{\rho-1} \boldsymbol{CA} e^{\boldsymbol{A}(\bar{T}(l+1)+\ldots+\bar{T}(k-1))} (e^{\boldsymbol{A}\bar{T}(l)} - \boldsymbol{I}) \boldsymbol{B} \bar{u}(l).\end{aligned} \tag{27}$$

Consider the case when $\delta T = \delta T(m) = \delta T(n), \forall m, n \in \{0, \ldots, k-1\}$. Then (23) can be rewritten as

$$y(k) = \bar{y}(k) + L\delta T + R_1 \text{ where } L = \sum_{m=0}^{k-1} \frac{\partial \bar{y}(k)}{\partial T(m)}. \tag{28}$$

Additionally, (25) is rewritten as

$$|R_1| \leq \frac{1}{2} P \delta T^2 \text{ where } P = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} p^{m,n}. \tag{29}$$

Given this bound on the Lagrange remainder, observe that if

$$\begin{aligned} \bar{y}(k) + L\delta T - \frac{1}{2} P \delta T^2 \geq r(k) - \Delta \\ \bar{y}(k) + L\delta T + \frac{1}{2} P \delta T^2 \leq r(k) + \Delta \end{aligned} \tag{30}$$

holds for all k corresponding to waypoints, then (17) holds. Since it is given that $(\bar{\boldsymbol{u}}, \bar{\boldsymbol{T}})$ strictly satisfies (17), then $r(k) - \Delta < \bar{y}(k) < r(k) + \Delta$. It follows that since $P \geq 0$, solving (30) will give real values $\delta T^{lo} < 0$ and $\delta T^{hi} > 0$ such that for all $\delta T \in [\delta T^{lo}, \delta T^{hi}]$, (30) is satisfied.

However, because P is only evaluated for $\boldsymbol{\delta T}^k \in \Theta$, satisfying (30) at the μ^{th} waypoint, w_μ, with corresponding timestep k does not imply that the tracking requirement at w_μ is satisfied. This may occur if $\delta T^{lo} < \delta T^-$ or $\delta T^{hi} > \delta T^+$. Thus, to satisfy the tracking requirement at w_μ, $\boldsymbol{\delta T} \in \mathrm{IR}^{\mathrm{N}}$ must lie in the set $\Xi_\mu = \{\boldsymbol{\delta T} : \max(\delta T^{lo}, \delta T^-) \leq \delta T(i) \leq \min(\delta T^{hi}, \delta T^+), \forall i = 0, \ldots, k-1\}$ where $\boldsymbol{\delta T}$ represents the change from nominal timing scheme $\bar{\boldsymbol{T}}$.

Ξ_μ is then calculated for each waypoint, w_μ, with $\mu \in \{0, \ldots, N_w - 1\}$. To satisfy constraint (17), tracking is required at each waypoint. Thus, tracking is satisfied if $\boldsymbol{\delta T} \in \Xi = \bigcap_{\mu=0,\ldots N_w-1} \Xi_\mu$. Specifically, Ξ is given by

$$\Xi = \{\boldsymbol{\delta T} : \delta T_{min}(i) \leq \delta T(i) \leq \delta T_{max}(i), \forall i = 0, \ldots, N-1\}. \tag{31}$$

$\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$ are then given by their elements $\delta T_{min}(i)$ and $\delta T_{max}(i)$.

$\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$ serve as a model-based guess on bounds for $\boldsymbol{\delta T}$. In other words, if $\boldsymbol{\delta T}_{min} \leq \boldsymbol{\delta T} \leq \boldsymbol{\delta T}_{max}$, the system model predicts that waypoint tracking constraint (17) will be satisfied by applying $\bar{\boldsymbol{u}}$ over timesteps $\bar{\boldsymbol{T}} + \boldsymbol{\delta T}$.

Given $\bar{\boldsymbol{T}}_j$ and $\bar{\boldsymbol{u}}_{j+1}$, bounds $\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$ are calculated. However, the existence of model uncertainty may cause $\boldsymbol{\delta T}_{min}$ and $\boldsymbol{\delta T}_{max}$ to be too large, thus preventing (17) from being satisfied. Hence, (19) is solved iteratively with increasingly tightened constraints dictated by λ^h until

$$\left\| \boldsymbol{\Psi}(\boldsymbol{y_r} - \boldsymbol{y}_{j,h}) \right\|_\infty < \Delta. \tag{32}$$

The parameter λ is used in (19) to control the rate at which the bounds on changes in timesteps are tightened. Once (32) is met, define $\bar{\boldsymbol{T}}_{j+1} \triangleq \boldsymbol{T}_{j,h}$ before returning to Stage 1.

E. The flexible-time ILC algorithm

The flexible-time ILC algorithm is given in Algorithm 1.

Algorithm 1 Flexible-time ILC for linear timing objectives

Input: State space model $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, initial system timing $\bar{\boldsymbol{T}}_0$, reference $\boldsymbol{y_r}$, selection matrix $\boldsymbol{\Psi}$, waypoint tracking tolerance Δ, initial convergence parameter ε_0, input saturation bounds (u_{min}, u_{max}), ILC weighting parameters (q, r), timestep constraint tightening parameter λ, function $f(\boldsymbol{T})$ to determine nominal bounds on changes in timesteps, $(\delta T^-, \delta T^+)$.

Initialisation : Set iteration indices $j = i = h = 0$. Define $\bar{\boldsymbol{u}}_0$ such that applying $\bar{\boldsymbol{u}}_0$ over $\bar{\boldsymbol{T}}_0$ strictly satisfies (17).

1: **while** true **do**
2: Set $\boldsymbol{u}_{j,0}$ to $\bar{\boldsymbol{u}}_j$
3: **repeat**
4: Apply the solution, $\boldsymbol{u}_{j,i+1}$, of (18) over $\bar{\boldsymbol{T}}_j$. Set ε_{i+1} according to (21). Increment i.
5: **until** Condition (20) is met.
6: Set $\bar{\boldsymbol{u}}_{j+1} = \boldsymbol{u}_{j,i}$. Set $i = 0$. Set $\boldsymbol{T}_{j,h} = \bar{\boldsymbol{T}}_j$.
7: Calculate $\boldsymbol{\delta T}_{min}, \boldsymbol{\delta T}_{max}$ from (31).
8: **repeat**
9: Apply $\bar{\boldsymbol{u}}_{j+1}$ over the solution to (19), $\boldsymbol{T}_{j,h+1}$. Increment h
10: **until** Condition (32) is met.
11: Set $\bar{\boldsymbol{T}}_{j+1}$=$\boldsymbol{T}_{j,h}$. Set h = 0. Increment j.
12: **end while**

F. Convergence properties

Theorem 1: If an initial input sequence, $\boldsymbol{u}_{j,0}$, that is feasible for (18) produces a $\boldsymbol{y}_{j,0}$ that strictly satisfies (17) such that

$$\left\| \boldsymbol{\Psi} \boldsymbol{e}_{j,0} \right\|_\infty < \Delta, \tag{33}$$

then the Stage 1 algorithm will terminate after a finite number of iterations by satisfying convergence criterion (20).

Proof: At inner-loop iterate, i, denote the solution to (18) as $\boldsymbol{u}_{j,i+1}$. Since $\boldsymbol{u}_{j,0}$ is feasible for (18), this implies that a solution to (18), $\boldsymbol{u}_{j,i}$, exists for all $i \in \mathbb{N}_0$. Note that the cost of $\boldsymbol{u}_{j,i}$ is

$$J_{PTP}(\boldsymbol{u}_{j,i}) = q\left\|\boldsymbol{\Psi e}_{j,i}\right\|_\infty + r\left\|\boldsymbol{u}_{j,i} - \boldsymbol{u}_{j,i}\right\|_\infty \tag{34}$$

$$= q\left\|\boldsymbol{\Psi e}_{j,i}\right\|_\infty + r\|\boldsymbol{0}\|_\infty = q\left\|\boldsymbol{\Psi e}_{j,i}\right\|_\infty. \tag{35}$$

Since $\boldsymbol{u}_{j,i+1}$ minimizes (18), this implies

$$J_{PTP}(\boldsymbol{u}_{j,i}) \geq q\left\|\boldsymbol{\Psi e}_{j,i+1}\right\|_\infty + r\left\|\boldsymbol{u}_{j,i+1} - \boldsymbol{u}_{j,i}\right\|_\infty \tag{36}$$

$$\geq q\left\|\boldsymbol{\Psi e}_{j,i+1}\right\|_\infty. \tag{37}$$

Combining (33), (35), and (37) then gives

$$\Delta > \left\|\boldsymbol{\Psi e}_{j,0}\right\|_\infty \geq \left\|\boldsymbol{\Psi e}_{j,i}\right\|_\infty, \forall i. \tag{38}$$

In other words, the waypoint tracking constraint is strictly satisfied at every inner-loop iteration. From (21), this implies that there exists $\hat{\imath}$ such that

$$\varepsilon_i > \left\|\boldsymbol{\Psi e}_{j,0}\right\|_\infty \geq \left\|\boldsymbol{\Psi e}_{j,i}\right\|_\infty, \forall i \geq \hat{\imath}. \tag{39}$$

Condition (20) is then satisfied after, at most, $\hat{\imath}$ iterations. ■

Theorem 2: Given that the output of the true system has bounded first and second derivatives with respect to the timesteps such that

$$\left|\frac{\partial y(k)}{\partial T(n)}\right| < l^{max}, \quad \left|\frac{\partial^2 y(k)}{\partial T(m)\partial T(n)}\right| < p^{max}, \forall \boldsymbol{T} \geq 0 \tag{40}$$

and nominal timestepping and input sequences $\bar{\boldsymbol{T}}_j$ and $\bar{\boldsymbol{u}}_{j+1}$ produce an output trajectory, $\bar{\boldsymbol{y}}$, that strictly satisfies (17), then the Stage 2 algorithm will terminate after a finite number of iterations by satisfying criterion (32).

Proof: From (29) and (40), the Lagrange remainder is bounded by

$$|R_1| < \frac{1}{2}P\delta T^2 \text{ with } P = k^2 p^{max} \tag{41}$$

where k is the timestep index corresponding to an arbitrary waypoint. Since $P > 0$ and $\bar{\boldsymbol{y}}$ strictly satisfies waypoint tracking, then there exists $\delta T^{lo}_{true} < 0, \delta T^{hi}_{true} > 0$ such that for all $\delta T \in [\delta T^{lo}_{true}, \delta T^{hi}_{true}]$, (30) holds. Define $\boldsymbol{\delta T}^{lo}_{true}, \boldsymbol{\delta T}^{hi}_{true} \in \mathbb{R}^N$ as vectors whose elements all equal δT^{lo}_{true} and δT^{hi}_{true} respectively. Because $|R_1|$ is strictly bounded in (41), this implies that applying input $\bar{\boldsymbol{u}}_{j+1}$ over $\bar{\boldsymbol{T}}_j + \boldsymbol{\delta T}$ with $\boldsymbol{\delta T} \in \Xi^{true}$ produces $\boldsymbol{y}$ such that (17) is strictly satisfied where

$$\Xi^{true} = \{\boldsymbol{\delta T} : \delta T^{lo}_{true} \leq \delta T(i) \leq \delta T^{hi}_{true}, \forall i = 0, ..., N-1\}. \tag{42}$$

Let $\Xi_{j,h}$ denote the set of $\boldsymbol{\delta T}_{j,h+1}$ such that the first constraint of (19) is satisfied. Observe that because $\lambda \in (0,1)$,

$$\lim_{h\to\infty} \Xi_{j,h} = \boldsymbol{0} \tag{43}$$

where $\boldsymbol{0}$ denotes the zero element. Since $\boldsymbol{0}$ is in the interior of Ξ^{true} and $\boldsymbol{\delta T}_{j,h+1} \in \Xi_{j,h}$, this implies that there exists $\hat{h} < \infty$ such that for $h = \hat{h}$, convergence condition (32) is met. ■

Note that the more accurate $\bar{\boldsymbol{y}}$ tracks the waypoints, the larger the set Ξ^{true} becomes. Thus, more accurate waypoint tracking allows larger reductions in J_{LP}. Therefore, the use of point-to-point ILC in Stage 1 tends to improve the convergence rate of the system cost.

Theorem 3: Suppose $\bar{\boldsymbol{T}}_0$ and $\bar{\boldsymbol{u}}_0$ satisfy (14) and (15) and satisfy (17) strictly. Additionally, suppose the true system obeys (40). Then the flexible-time ILC algorithm does not terminate after a finite number of outer-loop iterations and

$$J_{LP}(\bar{\boldsymbol{T}}_{j+1}) < J_{LP}(\bar{\boldsymbol{T}}_j) \tag{44}$$

Proof: Suppose $\bar{\boldsymbol{T}}_j$ and $\boldsymbol{u}_{j,0} = \bar{\boldsymbol{u}}_j$ satisfy (14) and (15) and strictly satisfy (17). Then the required conditions for Theorem 1 are satisfied at iteration j. Thus, after some number, $\hat{\imath}$, of Stage 1 iterations, (20) is satisfied and $(\bar{\boldsymbol{u}}_{j+1}, \bar{\boldsymbol{T}}_j)$ strictly satisfies (17). Combined with (40), the conditions for Theorem 2 then hold at iteration j. Thus, after some number, $\hat{h}$, of Stage 2 iterations, (32) is satisfied. This implies that $\bar{\boldsymbol{T}}_{j+1}$ and $\boldsymbol{u}_{j+1,0} = \bar{\boldsymbol{u}}_{j+1}$ satisfy (14) and (15) and satisfy (17) strictly. Since $\bar{\boldsymbol{T}}_0$ and $\bar{\boldsymbol{u}}_0$ meet the conditions for Theorem 1, the algorithm is therefore infinitely recursive giving the first claim of the theorem.

Additionally, note that since $(\bar{\boldsymbol{u}}_{j+1}, \bar{\boldsymbol{T}}_j)$ strictly satisfies (17), $\boldsymbol{0}$ is in the interior of $\Xi_{j,\hat{h}}$. Hence, $\bar{\boldsymbol{T}}_j$ is in the interior of the feasible region of (19). However, the fundamental theorem of linear programming gives that the solution to a linear program must lie on a boundary of the feasible set. Therefore $\bar{\boldsymbol{T}}_j$ is not the minimizer of (19). Let $\bar{\boldsymbol{T}}_{j+1}$ denote the solution to (19). Then

$$J_{LP}(\bar{\boldsymbol{T}}_{j+1}) < J_{LP}(\bar{\boldsymbol{T}}_j). \tag{45}$$

which gives the second claim of the theorem. ■

V. SIMULATION EXAMPLE

The flexible-time ILC framework is simulated on a servo system test case described in [13], with application as a positioning system for microscale additive manufacturing (AM). The continuous time dynamics are given by

$$\begin{aligned} \dot{\boldsymbol{x}}_j[t] &= \begin{pmatrix} -4.86 & -21.86 \\ 1 & 0 \end{pmatrix} \boldsymbol{x}_j[t] + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \boldsymbol{u}_j[t] \\ \boldsymbol{y}_j[t] &= \begin{pmatrix} 4.36 & 21.86 \end{pmatrix} \boldsymbol{x}_j[t]. \end{aligned} \tag{46}$$

However, suppose that the system is inaccurately modeled such that the user model of the dynamics is given as

$$\begin{aligned} \dot{\boldsymbol{x}}_j[t] &= \begin{pmatrix} -3.39 & -25.59 \\ 1 & 0 \end{pmatrix} \boldsymbol{x}_j[t] + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \boldsymbol{u}_j[t] \\ \boldsymbol{y}_j[t] &= \begin{pmatrix} 3.85 & 25.59 \end{pmatrix} \boldsymbol{x}_j[t]. \end{aligned} \tag{47}$$

For this case study, the time-optimal waypoint tracking problem is investigated wherein performance objective J_{LP}, as given in (13), is defined by $\mathbf{c} = [1, ..., 1]^T$ which aims to minimize the total trial duration. Such an objective is relevant in AM, as faster tracking allows for improved throughput. The tolerance on waypoint tracking error is $\Delta = 0.2$ and the input signal is bounded as $\boldsymbol{u} \in [u_{min}, u_{max}] = [-1, 1]$. The ILC weighting parameters are $(q,r) = (1, 10^{-12})$ and the initial Stage 1 convergence parameter is $\varepsilon_0 = 0.01$ which is updated within a given execution of Stage 1 according to

$$\varepsilon_{i+1} = 0.9\varepsilon_i + 0.1\Delta. \tag{48}$$

The timestep constraint tightening parameter is $\lambda = 0.95$. At outer-loop iterate j, the update to the timestepping scheme set in Stage 2 is bounded by $\delta T^- = -0.05\bar{\boldsymbol{T}}_j$ and $\delta T^+ = 1.01\bar{\boldsymbol{T}}_j$. $N_w = 2$ waypoints are defined at timesteps 10 and 20 for which $y_r(10) = 1$ and $y_r(20) = 0$. These waypoints positions mimic raster printing patterns commonly used in AM. The initial trial duration is $0.50s$ over which $N = 20$ equal duration timestep intervals are defined to give $\bar{\boldsymbol{T}}_0$. $\bar{\boldsymbol{u}}_0$ is then given as the solution to (18) for $\boldsymbol{e}_{j,i} = \boldsymbol{y_r}$, $u_{j,i} = \boldsymbol{0}$.

After running the algorithm for 100 outer-loop iterations, the input sequences shown in Fig. 2a drive the system to the trajectories in Fig. 2b. Note that the input never exceeds the saturation bounds, thereby satisfying constraint (15). Additionally, the output stays within the tracking tolerance at the waypoints, thereby satisfying constraint (17). Then, since all of the timestep intervals are positive, the solution is feasible for problem (13)-(17). As shown in Fig. 3, the trial duration undergoes a strictly monotonic decrease from $0.50s$ to $0.30s$, as predicted by Theorem 3. Therefore, whereas optimal control techniques would rely on an accurate system model to develop the optimal timestepping scheme, the flexible-time ILC framework is able to counteract model uncertainty to improve the system's temporal performance while satisfying input saturation and waypoint tracking constraints.

VI. CONCLUSION

This work proposes an ILC method for improving a time-based linear performance objective subject to input saturation and sparse tracking constraints. A constrained point-to-point ILC input update is used to reduce waypoint tracking error, followed by an iterative linear programming procedure that

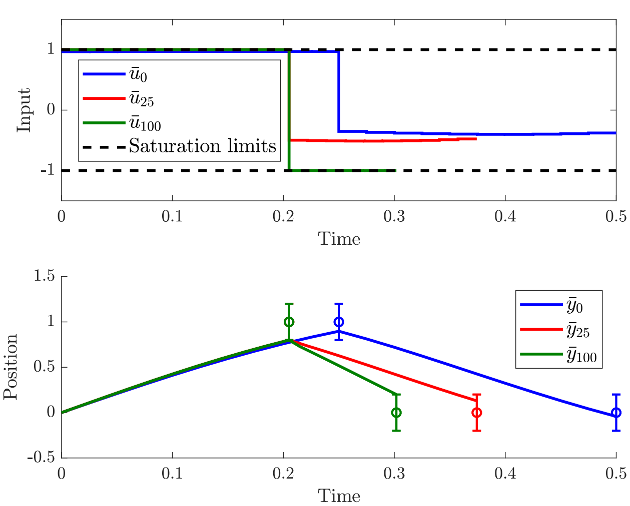

Fig. 2. a) Input sequences and b) output trajectories at iterations 1, 25, and 100 of the outer-loop. The circular markers indicate the waypoints at a given iteration with associated error tolerance, $\Delta = 0.2$.

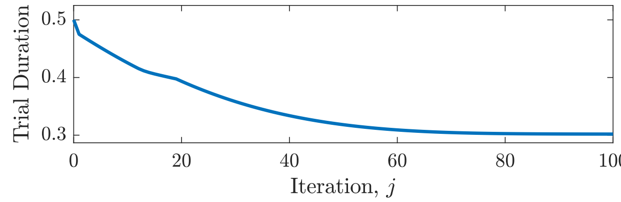

Fig. 3. Trial duration monotonically decreases with each iteration of the outer-loop procedure

is used to update the timestepping sequence to minimize a system performance cost. This methodology relaxes the conventional ILC restriction on an iteration-invariant trial duration. A simulated case study of the framework demonstrates its ability to use time as a flexible optimization parameter wherein system performance is improved monotonically.

Future work includes extensions to linear time-varying and linear parameter-varying systems, as well as an exploration to a broader class of system performance objectives.

REFERENCES

[1] K. L. Barton and A. G. Alleyne, "A norm optimal approach to time-varying ilc with application to a multi-axis robotic testbed," *IEEE Trans. Contr. Syst. Technol.*, vol. 19, no. 1, pp. 166–180, Jan 2011.

[2] M. L. Loyd, "Crosswind kite power (for large-scale wind power production)," *Journal of Energy*, vol. 4, no. 3, pp. 106–111, May 1980.

[3] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 96–114, June 2006.

[4] C. T. Freeman, Z. Cai, E. Rogers, and P. L. Lewin, "Iterative learning control for multiple point-to-point tracking application," *IEEE Trans. Contr. Syst. Technol.*, vol. 19, no. 3, pp. 590–600, May 2011.

[5] I. Lim and K. L. Barton, "Pareto iterative learning control: Optimized control for multiple performance objectives," *Control Engineering Practice*, vol. 26, pp. 125 – 135, 2014.

[6] X. Li, J. Xu, and D. Huang, "An iterative learning control approach for linear systems with randomly varying trial lengths," *IEEE Trans. Automat. Contr.*, vol. 59, no. 7, pp. 1954–1960, July 2014.

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

[8] Y. Chen, B. Chu, and C. T. Freeman, "Point-to-point iterative learning control with optimal tracking time allocation," *IEEE Trans. Contr. Syst. Technol.*, vol. 26, no. 5, pp. 1685–1698, Sep. 2018.

[9] Y. Chen, B. Chu, and C. T. Freeman, "A coordinate descent approach to optimal tracking time allocation in point-to-point ilc," *Mechatronics*, vol. 59, pp. 25 – 34, 2019.

[10] P. Janssens, W. Van Loock, G. Pipeleers, F. Debrouwere, and J. Swevers, "Iterative learning control for optimal path following problems," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 6670–6675.

[11] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," *IEE Proceedings - Control Theory and Applications*, vol. 143, no. 2, pp. 217–224, March 1996.

[12] S. Gunnarsson and M. Norrlöf, "On the design of ilc algorithms using optimization," *Automatica*, vol. 37, no. 12, pp. 2011 – 2016, 2001.

[13] D. A. Bristow, K. L. Barton, A. G. Alleyne, and W. S. Levine, "Iterative learning control," *The Control Systems Handbook: Control System Advanced Methods*, pp. 1–19, 2010.

[14] B. Chu and D. H. Owens, "Iterative learning control for constrained linear systems," *International Journal of Control*, vol. 83, no. 7, pp. 1397–1413, 2010.

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