

Chapter 2

Iterative Learning Control—An Overview

This chapter gives the required background on iterative learning control. After introducing the defining characteristic of this form of control, attention is restricted to the laws used in the stroke rehabilitation research.

2.1 Introduction

The development of iterative learning control (ILC) emerged from industrial applications where the system involved executes the same operation many times over a fixed finite time interval. When each operation is complete, resetting to the starting location takes place and the next operation can commence immediately, or after a stoppage time. A common example is a gantry robot undertaking a pick and place operation in synchronization with a moving conveyor or assembly line. The sequence of operations is: (a) the robot collects a payload from a fixed location, (b) transfers it over a finite duration, (c) places it on the moving conveyor, (d) returns to the original location for the next payload and then (e) repeats the previous four steps for as many payloads as is required or can be transferred before it is required to stop.

To operate in pick and place mode it is necessary to supply the robot with a trajectory to follow and the task for a control law is to ensure that the robot follows the prescribed trajectory exactly or, more realistically, to within a specified tolerance. In addition to controlling its own movement and that of the payload, the control law must prevent other effects, such as disturbances and signal noise, from degrading tracking and thereby forcing it outside of the tolerance bound. If the robot begins to operate outside permissible limits, the control task is to bring it back within the specified limits as quickly as required or is physically possible. This task must be achieved without causing damage to, e.g., the sensing and actuating technologies used.

In the ILC literature, each completion or execution of the task is described as a pass, iteration or trial, but in this monograph the latter term is exclusively used. Similarly, the finite time each trial takes to complete will be referred to as the trial length. Once a trial is finished, all data used and generated during its completion is available for use in computing the control action to be applied on the next trial. The use of such data is a form of learning and is the essence of ILC, embedding the mechanism through which performance may be improved by past experience.

The ILC mode of operation outlined above is the most common, i.e., complete a trial, reset and then repeat. This is different from repetitive control where the system continuously executes over the period of the reference signal, i.e., with no stoppage time between trials.

This chapter gives an overview of ILC, where the focus is on the algorithms that have been used to date in the technology transfer to next generation healthcare, with pointers to the literature for other design algorithms and applications. The particular area of next generation healthcare addressed is robotic-assisted upper limb stroke rehabilitation. In this context ILC is used to adjust the level of assistive stimulation applied during a treatment session where the patient attempts to re-learn a daily living task, such as reaching out to an object with the affected limb, by repeated attempts guided by a robot.

2.2 The Origins of ILC

The widely recognized starting point for ILC is Arimoto et al. (1984), which considered a simple first order linear servomechanism system for a voltage-controlled dc-servomotor. As in other areas, there is debate on the origins of ILC, for which the survey papers (Ahn et al. 2007; Bristow et al. 2006) and, in particular, Ahn et al. (2007) give coverage and relevant references. In the opening paragraphs of Arimoto et al. (1984) the analogy between ILC and human learning is drawn in the text: ‘It is human to make mistakes, but it also human to learn from such experience. Is it possible to think of a way to implement such a learning ability in the automatic operation of dynamic systems?’.

The analysis in Arimoto et al. (1984) developed, using the servomotor example as a particular example, a control law applicable to systems required to track a desired reference trajectory **of a fixed trial length T and specified a priori**. On completion of each trial, **the system states reset** and during time taken to complete this task the **measured output** is used in the construction the next control output. The system dynamics were assumed to be **trial-invariant** and **invertible**. These distinguishing features led to the establishment of ILC as a major and ongoing area of control systems research and applications. Several of these assumptions, e.g., trial-invariant dynamics, have been relaxed in recent years but the concept of learning from experience gained over repeated trials of a task is retained.

Since it was first introduced ILC has broadened in breadth and depth, including links with established fields such as robust, adaptive and optimal control. Application areas have also expanded beyond industrial robotics and process control. In the

latter area, one starting point for the literature is the survey paper Wang et al. (2009), which also considers the connections with repetitive control and run-to-run control. This chapter now proceeds to consider the ILC theory and algorithms that have found novel application in stroke rehabilitation. For consistency, discrete descriptions of the dynamics are used.

2.3 ILC for Linear Systems

When ILC is applied to discrete dynamics the notation used for a scalar or vector valued variable in this monograph is $y_k(p)$, $p = 0, 1, \dots, T$. Here the nonnegative integer k is the trial number and $T \in \mathbb{N}$ denotes the number of samples on each trial, with the assumption of a constant sampling period. Suppose also that the dynamics of the system or process considered can be adequately modeled as linear and time-invariant. Then the state-space model of such a system in the ILC setting is

$$\begin{aligned} x_k(p+1) &= Ax_k(p) + Bu_k(p) \\ y_k(p) &= Cx_k(p), \quad x_k(0) = x_0 \end{aligned} \quad (2.1)$$

where on trial k , $x_k(p) \in \mathbb{R}^n$ is the state vector, $y_k(p) \in \mathbb{R}^m$ is the output vector and $u_k(p) \in \mathbb{R}^l$ is the control input vector.

In this model it is assumed that the initial state vector does not change from trial-to-trial. The case when this assumption is not valid has also been considered in the literature. The dynamics are assumed to be disturbance-free but again this assumption can be relaxed. It is also possible to write the dynamics in input-output form involving the convolution operator or take the one-sided z transform and hence analysis and design in the frequency domain is possible. To apply the z transform it is necessary to assume $T = \infty$ but in most cases the consequences of this requirement have no detrimental effects. For a more detailed analysis of cases where there are unwanted effects arising from this assumption, see the relevant references in Ahn et al. (2007), Bristow et al. (2006) and more recent work in Wallen et al. (2013).

Let $r(p) \in \mathbb{R}^m$ denote the supplied reference vector. Then the error on trial k is $e_k(p) = r(p) - y_k(p)$ and the core requirement in ILC is to construct a sequence of input functions $u_{k+1}(p)$, $k \geq 0$, such that the performance achieved is gradually improved with each successive trial and after a ‘sufficient’ number of these the current trial error is zero or within an acceptable tolerance. Mathematically this can be stated as a convergence condition on the input and error of the form

$$\lim_{k \rightarrow \infty} \|e_k\| = 0, \quad \lim_{k \rightarrow \infty} \|u_k - u_\infty\| = 0 \quad (2.2)$$

where u_∞ is termed the learned control and $\|\cdot\|$ denotes an appropriate norm on the underlying function space. As one possibility, let $\|\cdot\|_2$ denote the Euclidean norm of its argument and set $\|e\| = \max_{p \in [0, T]} \|e(p)\|_2$. The reason for including the requirement on the control vector is to ensure that strong emphasis on reducing

the trial-to-trial error does not come at the expense of unacceptable control signal demands. In application, only a finite number of trials will ever be completed but mathematically letting $k \rightarrow \infty$ is required in analysis of, e.g., trial-to-trial error convergence.

The standard form of ILC algorithm or law computes the current trial input as the sum of the input used on the previous trial and a corrective term, i.e.,

$$u_{k+1} = u_k + \Delta(u_k, e_k) \quad (2.3)$$

where $\Delta(u_k, e_k)$ is the correction term and is a function of the error and input recorded over the previous trial. A large number of variations exist for computing the correction term, including laws that make use of information generated on a finite number (greater than unity) of previous trials. For the stroke rehabilitation application it is the repeated performance of a finite duration task (with the input on the current trial computed by adding a corrective term that is directly influenced by the previous trial error) that makes ILC particularly suitable.

An extensively used analysis and design setting for discrete systems is based on lifting in the ILC setting. Suppose that (2.1) is asymptotically stable and hence all eigenvalues of the state matrix A have modulus strictly less than unity. If this is not the case then a stabilizing feedback control loop must be first applied. For simplicity, consider single-input single-output (SISO) systems with an assumed relative degree of one, and hence in (2.1) the first Markov parameter $CB \neq 0$. For the cases of multiple-input multiple-output (MIMO) systems and/or the assumption on the Markov parameter does not hold, refer to the relevant references in Ahn et al. (2007), Bristow et al. (2006).

Introduce

$$y_k = \begin{bmatrix} y_k(1) \\ y_k(2) \\ \vdots \\ y_k(T) \end{bmatrix}, \quad u_k = \begin{bmatrix} u_k(0) \\ u_k(1) \\ \vdots \\ u_k(T-1) \end{bmatrix}, \quad d = \begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(T) \end{bmatrix}. \quad (2.4)$$

Then under the assumption that $r(0) = Cx_0$, (2.1) can be written in the form

$$y_k = Gu_k + d \quad (2.5)$$

with

$$G = \begin{bmatrix} p_1 & 0 & \dots & 0 \\ p_2 & p_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_T & p_{T-1} & \dots & p_1 \end{bmatrix} \quad (2.6)$$

where $p_j = CA^{j-1}B$ and $d(j) = CA^j x_0$, $j = 1, \dots, T$.

2.3.1 Control Laws and Structural/Performance Issues

Consider the SISO version of the state-space model (2.1) and suppose that both the system dynamics and the measured output are deterministic, i.e., noise-free. Then a derivative, or D-type, ILC law constructs the current trial input as

$$u_{k+1}(p) = u_k(p) + K_d[e_k(p+1) - e_k(p)] \quad (2.7)$$

where K_d is a scalar to be designed such that $\lim_{k \rightarrow \infty} \|e_k\| = 0$. Also routine analysis shows that this condition holds if and only if $|1 - CBK_d| < 1$. Somewhat surprisingly, this condition is independent of the system dynamics embodied in state matrix A and can only be satisfied if $CB \neq 0$.

The reason why trial-to-trial error convergence (k) is independent of the system state matrix is the finite trial length, over which duration even an unstable linear system can only produce a bounded output. In design based on a lifted model, the solution is to first design a stabilizing feedback control law for the unstable system and then apply ILC to the lifted version of the resulting controlled system. This step may also be required for stable systems to ensure acceptable transient dynamics along the trials. This results in a two stage design whereas the repetitive process, a class of 2D linear systems, setting allows simultaneous design for trial-to-trial error convergence and along the trial dynamics, see, e.g., Hladowski et al. (2010, 2012) where experimental verification on a gantry robot that replicates many industrial processes to which ILC is applicable is also given.

If the system model has relative degree greater than one it follows immediately that trial-to-trial error convergence cannot be achieved. This problem arises for many ILC laws and has received considerable attention in the literature, where one starting point is again the relevant references in the survey papers (Ahn et al. 2007; Bristow et al. 2006). This feature is also present in the 2D systems/repetitive process designs. The most that can be done for a system of relative degree h is to lose control over the first $h - 1$ samples along the trial and design a control law that gives convergence over the remaining samples.

In ILC, once trial k is complete the following information is available for the computation of the control u_{k+1} : (1) Information from the entire time duration of any previous trial and (2) Information up to the current sample on trial $k + 1$. The following is one definition of causality in ILC.

Definition 2.1 An ILC law is causal if and only if the value of the input $u_{k+1}(p)$ at time p on trial $k + 1$ is computed only using data in the time interval $[0, p]$ from the current and previous trials.

For standard linear systems at sample instant p the use of information at future samples $p + 1, p + 2, \dots$ is non-causal and therefore any resulting control law cannot be implemented. The use of non-causal along the trial information in ILC laws is arguably the most important feature.

Consider the ILC control laws

$$u_{k+1}(p) = u_k(p) + K_p e_k(p+1) \quad (2.8)$$

and

$$u_{k+1}(p) = u_k(p) + K_p e_k(p) \quad (2.9)$$

where the first is ILC non-causal and the second is causal. Also let q denote the forward time shift operator acting on, e.g., $x(p)$ as $qx(p) = x(p+1)$. Then the dynamics of (2.1) can be written as

$$y_k(p) = G(q)u_k(p) + d(p) \quad (2.10)$$

where $d(p) = CA^p x_0$ and this term can be extended to represent exogenous system disturbances that enter on trial k . Moreover, this disturbance term influences the error on trial k as

$$e_k(p) = r(p) - G(q)u_k(p) - d(p) \quad (2.11)$$

Hence the non-causal ILC law (2.8) anticipates the disturbance d_{k+1} and uses the input $u_{k+1}(p)$ to preemptively compensate for its effects. This feature is not present in the causal ILC law (2.9).

Causal ILC laws can be shown to be equivalent to a feedback control, i.e., an equivalent control action can be obtained directly from the ILC law and it has been asserted that causal ILC algorithms have little merit. See the discussion, with supporting references, in Bristow et al. (2006) that counters this argument but in any case the vast majority of implemented ILC laws are non-causal.

The finite trial length in ILC allows non-causal signal processing to be used. For many implementations, this is exploited in the form of zero-phase filtering of the previous trial error prior to the computation of the next trial input. An experimental example where zero-phase filtering is used is the gantry robot based results reported in Hladowski et al. (2010, 2012). Essentially, zero-phase filtering between trials can be used to remove unwanted effects, e.g., noise from the measured signals.

A commonly used ILC law is given by

$$u_{k+1}(p) = Q(q) [u_k(p) + L(q)e_k(p+1)] \quad (2.12)$$

where $Q(q)$ is termed the Q -filter and $L(q)$ is the learning function, but these designations are not universally used in the literature. The Q -filter and learning function L can be non-causal, in the ILC sense, with impulse responses

$$\begin{aligned} Q(q) &= \dots + q_{-2}q^2 + q_{-1}q + q_0 + q_1q^{-1} + q_2q^{-2} + \dots \\ L(q) &= \dots + l_{-2}q^2 + l_{-1}q + l_0 + l_1q^{-1} + l_2q^{-2} + \dots \end{aligned} \quad (2.13)$$

This algorithm has many variations, including phase-lead

$$u_{k+1}(p) = u_k(p) + le_k(p + h) \quad (2.14)$$

where the designation ‘phase-lead’ arises from the shifted term $le_k(p + h)$, $h > 0$.

An ILC law of the form (2.12) can also be written in lifted form as

$$u_{k+1} = Q(u_k + Le_k) \quad (2.15)$$

The matrices G of (2.6), Q and L are Toeplitz and when the ILC law is causal Q and L are lower triangular. Other forms of Q and L , such as the fully populated case, correspond to non-causal ILC. Possibilities considered in the literature include time-varying functions, nonlinear functions and trial-to-trial (in k) functions. Imposing a band-diagonal structure results in Finite-Impulse Response (FIR) $Q(q)$ and $L(q)$ operators that can be causal or non-causal. The lifted model description is not applicable to differential dynamics and hence to applications where design by emulation is the only or preferred option. The repetitive process/2D system approach extends to this case.

2.3.2 Control Law Design

As in other control system design areas, the objectives must be specified, starting with stability. Consider applying the ILC law (2.12) to the system (2.10). Asymptotic stability in the SISO case then requires the existence of a real number $\hat{u} > 0$ such that $|u_k(p)| \leq \hat{u}$ for all $p \in [0, T]$ and $k \geq 0$, and for all $p \in [0, T]$, $\lim_{k \rightarrow \infty} u_k(p)$ exists and the learned control is $u_\infty(p) = \lim_{k \rightarrow \infty} u_k(p)$.

Using the lifted form, the controlled dynamics resulting from applying (2.12) to (2.10) can be written as

$$u_{k+1} = Q(I - LG)u_k + QL(r - d) \quad (2.16)$$

and stability holds if and only if all eigenvalues of the matrix $Q(I - LG)$ have modulus strictly less than unity, where I denotes the identity matrix of compatible dimensions. Matrix $Q(I - LG)$ is lower triangular and Toeplitz when the Q filter and learning function L are causal and all eigenvalues are equal and of value $q_0(1 - l_0 p_1)$. Hence stability requires $|q_0(1 - l_0 p_1)| < 1$ and this property cannot hold if the first Markov parameter $p_1 = 0$ as discussed previously in this chapter. Consult the references in Ahn et al. (2007), Bristow et al. (2006) for alternative settings to analyze the stability properties of this form of ILC.

Performance of an ILC system is different from the standard linear systems case as it is necessary to consider trial-to-trial and along the trial dynamics. In the former case, if the system considered above is asymptotically stable, the converged error in k is

$$e_\infty(p) = \lim_{k \rightarrow \infty} e_k(p) = r(p) - G(q)u_\infty(p) - d(p) \quad (2.17)$$

and again there is a z transform version of this result. Performance from trial-to-trial can, of course, be compared in many ways, where one measure is the difference between the final and initial trial errors, i.e., $e_\infty(p)$ and $e_0(p)$. Theorem 3 in Bristow et al. (2006) gives the conditions for convergence to zero error in the case when G and L are not identically zero. These conditions comprise asymptotic stability plus the requirement that $Q(q) = 1$. As discussed previously, in many cases it will also be necessary to design for acceptable transient dynamics along the trials.

Robustness is also an issue in ILC. Early research on the use of an \mathcal{H}_∞ setting is given in Amann et al. (1996a), with other work referenced in the survey papers (Ahn et al. 2007; Bristow et al. 2006), and is largely based on assuming an uncertainty model to represent the unmodeled dynamics, such as norm-bounded. More recent work, such as Hladowski et al. (2010, 2012), uses Linear Matrix Inequalities (LMIs) to compute the robust control law with experimental verification on a gantry robot.

In applications terms, the core task in ILC design is to construct an open-loop signal that approximately inverts the plant's dynamics, tracks the reference and rejects repeating disturbances. In the ideal scenario ILC would only learn repeating disturbances and ignore noise and non-repeating disturbances. Four general control law design methods are now discussed in turn, starting with Proportional plus Derivative (PD)-type designs with tuning that can be applied to a system without extensive modeling and analysis.

2.3.3 Proportional Plus Derivative-Type ILC

Arimoto's original algorithm (2.7) can be expanded for SISO systems to form a PD-type ILC law, which can be written as

$$u_{k+1}(p) = u_k(p) + k_p e_k(p+1) + k_d [e_k(p+1) - e_k(p)]. \quad (2.18)$$

In the lifted setting, (2.18) corresponds to the choice $Q = I$ and

$$L = \begin{bmatrix} k_p + k_d & 0 & \dots & 0 \\ -k_d & k_p + k_d & \dots & 0 \\ 0 & -k_d & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & k_p + k_d \end{bmatrix} \quad (2.19)$$

An alternative is to use $e_k(p)$ instead of $e_k(p+1)$ in the second term on the right-hand side of (2.18). Also the generalization to MIMO systems is immediate.

Unlike Proportional plus Integral plus Derivative (PID) (or three term) control for standard systems, auto-tuning rules are not available for ILC design. Also monotonic trial-to-trial error convergence is not always possible with ILC PD-type laws and an

often used approach to approximately achieve this property is to include a low-pass Q filter in the control law, i.e., as in (2.12), pre-multiply the right-hand side of the control law by $Q(p)$. This filter can be used to block learning at high frequencies and also has other benefits, such as increased robustness and filtering of high-frequency noise.

In tuning-based design, one approach is to first select the Q filter type, such as Butterworth or Chebyshev, and order and then use the filter bandwidth as the tuning variable. This approach is extensively covered in the literature and, e.g., Bristow et al. (2006) gives intuitive guidelines for tuning to achieve good learning transients and low error. Again, the survey papers (Ahn et al. 2007; Bristow et al. 2006) are a starting point for the many methods available for ILC PD design.

2.3.4 Inverse ILC

Plant inversion, or inverse, ILC designs use models of the inverse plant dynamics as the learning function. For discrete systems the control law has the form

$$u_{k+1}(p) = u_k(p) + \hat{G}^{-1}(q)e_k(p) \quad (2.20)$$

or

$$u_{k+1}(p) = u_k(p) + q^{-1}\hat{G}^{-1}(q)e_k(p+1) \quad (2.21)$$

where, since the exact inverse will not often be computable, $\hat{G}^{-1}(q)$ denotes the approximate inverse of $G(q)$. The learning function is

$$L(q) = q^{-1}\hat{G}^{-1}(q) \quad (2.22)$$

which is causal and of zero relative degree, i.e., has the same number of zeros as poles.

2.3.5 Gradient Descent ILC

As in other areas, a natural approach to model based ILC is to minimize a suitable cost function. The gradient descent algorithm for ILC (Furuta and Yamakita 1987) considers the following cost-function for the discrete lifted model

$$J(u_{k+1}) = \|e_{k+1}\|^2, \quad e_{k+1} = r - Gu_{k+1} \quad (2.23)$$

during each trial. Suppose also that the ILC law used is

$$u_{k+1} = u_k + \epsilon_{k+1}\delta_{k+1} \quad (2.24)$$

where ϵ_{k+1} is a scaling factor and δ_{k+1} is the vector that determines the direction of the update vector. Then the tracking error on trial $k + 1$ is

$$\begin{aligned} J(u_{k+1}) &= J(u_k + \epsilon_{k+1}\delta_{k+1}) = \|e_{k+1}\|^2 \\ &= \|e_k\|^2 - 2\epsilon_{k+1}\delta_{k+1}^T G^T e_k + \epsilon_{k+1}^2 \delta_{k+1}^T G^T G \delta_{k+1} \end{aligned} \quad (2.25)$$

and hence

$$\|e_{k+1}\|^2 - \|e_k\|^2 = -2\epsilon_{k+1}\delta_{k+1}^T G^T e_k + \epsilon_{k+1}^2 \delta_{k+1}^T G^T G \delta_{k+1}. \quad (2.26)$$

Monotonic trial-to-trial error convergence occurs when the right-hand side in (2.26) is negative. One option is choosing $\delta_{k+1} = G^T e_k$, resulting in the control law

$$u_{k+1} = u_k + \epsilon_{k+1} G^T e_k \quad (2.27)$$

which corresponds to the choice $Q = I$ and $L = \epsilon_{k+1} G^T$ in the lifted setting.

2.3.6 Norm Optimal ILC

Norm Optimal ILC (NOILC) is a gradient-based update law that includes: (a) automatic choice of step size, and (b) potential for improved robustness through use of causal feedback (current trial error data) and feedforward of data from previous trials. The results below are from Amann et al. (1996b), see also papers cited in Ahn et al. (2007), Bristow et al. (2006) for other versions of this law.

The current trial input is chosen to minimize a cost function involving norms of the trial error and the difference between successive trial control inputs. A general treatment of the cost function and the problem solution in a Hilbert space setting can be found in Amann et al. (1996b). The cost function used for discrete dynamics in the ILC setting described by (2.1) is

$$\begin{aligned} J(u_{k+1}) &= \sum_{i=0}^{T-1} (e_{k+1}(i) - e_k(i))^T Q (e_{k+1}(i) - e_k(i)) \\ &\quad + \sum_{i=0}^{T-1} (u_{k+1}(i) - u_k(i))^T R (u_{k+1}(i) - u_k(i)) \end{aligned} \quad (2.28)$$

where Q and R are symmetric positive definite weighting matrices to be selected. Use of this cost function optimally reduces the trial-to-trial error and ensures that the control input on the next trial does not deviate too much from that used on the previous trial.

Following Amann et al. (1996b) the control input on trial $k + 1$ is given by

$$u_{k+1}(p) = u_k(p) - \left[\left\{ B^T K(p) B + R \right\}^{-1} B^T K(p) \right. \\ \left. \times A \{x_{k+1}(p) - x_k(p)\} \right] + R^{-1} B^T \xi_{k+1}(p) \quad (2.29)$$

where $K(p)$ is the solution of the algebraic Riccati equation

$$K(p) = A^T K(p+1)A + C^T Q C - \left[A^T K(p+1)B \right. \\ \left. \times \left\{ B^T K(p+1)B + R \right\}^{-1} B^T K(p+1)A \right] \quad (2.30)$$

with terminal boundary condition $K(T) = 0$. The feedforward predictive term $\xi_{k+1}(p)$ is generated after each trial as

$$\xi_{k+1}(p) = \left\{ I + K(p) B R^{-1} B^T \right\}^{-1} \left\{ A^T \xi_{k+1}(p+1) + C^T Q e_k(p+1) \right\} \quad (2.31)$$

with terminal boundary condition $\xi_{k+1}(T) = 0$. Moreover, NOILC can also be applied to the lifted model representation of the dynamics.

2.4 Nonlinear Model ILC

Nonlinear ILC has received substantial attention in the literature, especially trial-to-trial error convergence proofs. In this section the background on one method, Newton ILC, which has been used in the stroke rehabilitation research reported in this monograph, is given.

Nonlinear systems can, in general terms, be split into two groups; those that are affine in the control and those that are not. The former are assumed to be of the form

$$\dot{x}(t) = f(x(t)) + B(x(t))u(t) \\ y(t) = h(x(t)) \quad (2.32)$$

where x is the state vector, u is the input and y is the output. A special case is the following model generally used to express the dynamics of robotic systems

$$M_r(x(t))\ddot{x}(t) - C_r(x(t), \dot{x}(t))\dot{x}(t) - g_r(x(t)) - d_r(x(t), \dot{x}(t)) = \tau(u(t)) \quad (2.33)$$

where the vectors $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ are the joint positions, velocities and accelerations, $\tau(u(t))$ is the actuator torque generated using a control input $u(t)$, $M_r(x)$ is the symmetric positive-definite inertial matrix, $C_r(x, \dot{x})$ is the Coriolis and centripetal acceleration matrix, $g_r(x)$ is the gravitational force vector and $d_r(x, \dot{x})$ is the friction torque vector.

The application of ILC to affine nonlinear systems uses a wide variety of laws but a critical common assumption is that the nonlinear system is smooth. This requirement

is often expressed as a global Lipschitz assumption on each of the functions in (2.32) of the form

$$\begin{aligned} |f(x_1) - f(x_2)| &\leq f_0|x_1 - x_2| \\ |B(x_1) - B(x_2)| &\leq b_0|x_1 - x_2| \\ |h(x_1) - h(x_2)| &\leq h_0|x_1 - x_2| \end{aligned} \quad (2.34)$$

The constants f_0 , b_0 and h_0 are used in a contraction mapping setting to obtain (sufficient) conditions for trial-to-trial error convergence and ILC law design. Non-affine systems have the form

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + B(x(t), u(t)) \\ y(t) &= h(x(t)) \end{aligned} \quad (2.35)$$

In the rehabilitation setting, control design is primarily undertaken using a discrete-time system representation. Obtaining discrete-time models for nonlinear systems is sometimes non-trivial but, e.g., trial-to-trial error convergence proofs are simpler and the final design is directly compatible with digital implementation. One other way to study and design ILC for nonlinear systems is to treat the nonlinearities as perturbations to a linearized system model.

2.4.1 Newton ILC

Newton ILC was proposed by Lin et al. (2006) and uses the full model in the computation of the next trial input. It is based on a general discrete-time state-space model of the form

$$\begin{aligned} x_k(p+1) &= f(x_k(p), u_k(p)) \\ y_k(p) &= h(x_k(p)) \end{aligned} \quad (2.36)$$

which can be obtained via discretization of its continuous-time counterpart. As in the linear case $p = 0, 1, \dots, T$ is the sample number, $x_k(p)$ is the state vector, and in lifted form the output and input vectors are given by

$$\begin{aligned} y_k &= [y_k^T(0) \ y_k^T(1) \ \dots \ y_k^T(T)]^T \\ u_k &= [u_k^T(0) \ u_k^T(1) \ \dots \ u_k^T(T)]^T \end{aligned} \quad (2.37)$$

and the reference vector by

$$y_d = [y_d^T(0) \ y_d^T(1) \ \dots \ y_d^T(T)]^T \quad (2.38)$$

The Newton ILC law takes the form

$$u_{k+1} = u_k + g'(u_k)^{-1} e_k \quad (2.39)$$

where $e_k = r - y_k$ is the tracking error. The term $g'(u_k)$ is equivalent to linearizing the system dynamics around u_k , with the system $\tilde{y} = g'(u_k)\tilde{u}$ corresponding to the following linear time-varying state-space model

$$\begin{aligned} \tilde{x}(p+1) &= A(p)\tilde{x}(p) + B(p)\tilde{u}(p) \\ \tilde{y}(p) &= C(p)\tilde{x}(p) \end{aligned} \quad (2.40)$$

over $p = 0, 1, \dots, T$, with

$$\begin{aligned} A(p) &= \left(\frac{\partial f}{\partial x} \right)_{u_k(p), x_k(p)}, & B(p) &= \left(\frac{\partial f}{\partial u_k} \right)_{u_k(p), x_k(p)} \\ C(p) &= \left(\frac{\partial h}{\partial x} \right)_{u_k(p), x_k(p)} \end{aligned} \quad (2.41)$$

The term $g'(u_k)^{-1}$ in (2.39) is computationally expensive and may be singular or contain excessive amplitudes and high frequencies. To overcome this difficulty, introduce

$$e_k = g'(u_k) \Delta u_{k+1} \quad (2.42)$$

and then $\Delta u_{k+1} = u_{k+1} - u_k$ equals the input that forces the system (2.40) to track the error e_k . This is also an ILC problem and can be solved in between experimental trials using any ILC law that converges globally. In this monograph NOILC is used.

References

- Ahn H-S, Chen YQ, Moore KL (2007) Iterative learning control: brief survey and characterization. *IEEE Trans Syst Man Cybern* 37(6):1099–1121
- Amann N, Owens DH, Rogers E, Wahl A (1996a) An \mathcal{H}_∞ approach to iterative learning control design. *Int J Adapt Control Signal Process* 10(6):767–781
- Amann N, Owens DH, Rogers E (1996b) Iterative learning control for discrete-time systems with exponential rate of convergence. *Proc Inst Elect Eng, Part D, Control Theory Appl* 143(2):217–224
- Arimoto S, Kawamura S, Miyazaki F (1984) Bettering operation of robots by learning. *J Robot Syst* 2(1):123–140
- Bristow DA, Tharayil M, Alleyne AG (2006) A survey of iterative learning control: a learning based method for high performance tracking control. *IEEE Control Syst Mag* 26(3):96–114
- Furuta K, Yamakita M (1987) The design of a learning control system for multivariable systems. In: *Proceedings of the IEEE international symposium on intelligent control*, pp 371–376
- Hladowski L, Galkowski K, Cai Z, Rogers E, Freeman CT, Lewin PL (2010) Experimentally supported 2D systems based iterative learning control law design for error convergence and performance. *Control Eng Pract* 18(4):339–348

- Hladowski L, Galkowski K, Cai Z, Rogers E, Freeman CT, Lewin PL (2012) Output based iterative learning control design with experimental verification. *J Dyn Syst Meas Control* 134:021012-1–021012-10
- Lin T, Owens DH, Hatonen J (2006) Newton method based iterative learning control for discrete non-linear systems. *Int J Control* 79(10):1263–1276
- Wallen J, Gunnarsson S, Norrloff M (2013) Analysis of boundary effects in iterative learning control. *Int J Control* 86(3):410–415
- Wang Y, Gao F, Doyle III FJ (2009) Survey on iterative learning control, repetitive control, and run-to-run control. *J Process Control* 19:1589–1600

Iterative Learning Control for Electrical Stimulation and
Stroke Rehabilitation

Freeman, C.; Rogers, E.; Burridge, J.H.; Hughes, A.-M.;
Meadmore, K.L.

2015, VII, 124 p. 69 illus., 34 illus. in color., Softcover

ISBN: 978-1-4471-6725-9