Analysis of boundary effects in iterative learning control

Johanna Wallen, Svante Gunnarsson and Mikael Norrlöf

Linköping University Post Print

N.B.: When citing this work, cite the original article.

This is an electronic version of an article published in:

International Journal of Control is available online at informaworldTM:
http://dx.doi.org/10.1080/00207179.2012.733889

Copyright: Taylor & Francis: STM, Behavioural Science and Public Health Titles
http://www.tandf.co.uk/journals/default.asp

Postprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-96999
Boundary effects in Iterative Learning Control (ILC) algorithms are considered. ILC algorithms involve filtering of input and error signals over finite-time intervals, often using non-causal filters, and it is important that the boundary effects of the filtering operations are handled in an appropriate way. The topic is studied using both a proposed theoretical framework and simulations, and it is shown that the method for handling the boundary effects has impact on the stability and convergence properties of the ILC algorithm.

Keywords: Iterative learning control; boundary effects; implementation; convergence

1 Introduction

This paper considers ILC in a linear discrete time framework, and the aim is to study the boundary effects that occur in the filtering operations included in an ILC algorithm. This topic has received fairly limited attention, although it can be of large practical importance. Some comments are given in Moore (1998), Wang and Zhang (2009) where the error signal is extended with the last value at the end, and in Elci et al. (2002) where the signal is extended at the beginning and the end before filtering. Other relevant references are Plotnik and Longman (1999) and Longman and Songchon (1999), where Plotnik and Longman (1999) discuss which type of filter to use, which signal to filter, and how to perform the filtering when implementing the ILC algorithm. The performance when using different filtering and extension alternatives are compared in simulations of a robot joint. In Longman and Songchon (1999) the subject of implementation is discussed, assuming that the boundary effects can be ignored so that the ILC algorithm can be analysed in the frequency domain. In Plotnik and Longman (1999), Longman and Songchon (1999), the effects of the filtering operations are discussed with focus on zero-phase filtering of the signal to cut off the learning above a given frequency. The effects of the filtering operations are made as small as possible so that frequency-domain analysis of the algorithm properties can be applied.

The main objective in this paper is to investigate how the chosen method to handle the boundary effects influences stability and performance of the system. The analysis is performed in the time domain by studying the implementation aspects of the filtering in the matrix formulation of the ILC algorithm. The paper is to a large extent based on Wallén (2011).
2 System and ILC algorithm description

Consider linear discrete-time SISO systems described by
\[
y_k(t) = T_r(q)r(t) + T_u(q)u_k(t)
\]  
(1)

with reference \( r(t) \), ILC input \( u_k(t) \) and output \( y_k(t) \) at iteration \( k \). The shift operator is denoted \( q \). Load and measurement disturbances are omitted for simplicity. All signals are defined on a finite time interval \( t = nT_s, n \in [0, N - 1] \) with \( N \) number of samples and sampling interval \( T_s = 1 \) if nothing else is stated. The system can have internal feedback, so \( T_u(q) \) and \( T_r(q) \) contain the system to be controlled and the controller in operation. Parallel to the description (1), a matrix description of the system will be used. First, define the vector \( r \) of the \( N \)-sample sequence of \( r(t) \). Next, define \( u_k, y_k \) and \( z_k \) similarly. The Toeplitz matrix \( T_r \) is formed by the pulse-response coefficients \( g_{T_r}(l) \) of the transfer operator \( T_r(q) \) in (1),

\[
T_r = \begin{pmatrix}
g_{T_r}(0) & 0 & \cdots & 0 \\
g_{T_r}(1) & g_{T_r}(0) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
g_{T_r}(N-1) & g_{T_r}(N-2) & \cdots & g_{T_r}(0)
\end{pmatrix}
\]  
(2)

and \( T_u \) is defined similarly. The system in matrix form is
\[
y_k = T_r r + T_u u_k
\]  
(3)

Two alternative ways to implement a particular ILC algorithm will be discussed. The first alternative is the matrix formulation, where the ILC input is updated as in

\[
u_{k+1} = Q(u_k + L e_k)
\]  
(4)

\[
e_k = r - y_k
\]  
(5)

and \( Q \) and \( L \) are \( N \times N \) matrices. The second implementation alternative is the filter formulation,

\[
u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t))
\]  
(6)

\[
e_k(t) = r(t) - y_k(t)
\]  
(7)

where \( Q(q) \) and \( L(q) \) may be non-causal filters. The matrix formulation is generally more computational demanding, but covers a larger class of algorithms. For both alternatives it is necessary to take care of the boundary effects in the implementation. The goal here is to illustrate that the matrix formulation offers a systematic framework for analysing how this will affect the algorithm properties. Therefore, the implementation of the ILC algorithm in filter form (6) is interpreted using the matrix formulation (4), see Sections 3 and 4.

Stability and convergence properties of ILC algorithms can be found in for example Norrlöf and Gunnarsson (2002), and the main results can be stated as follows. The system (3) controlled by the ILC algorithm (4) is stable if and only if

\[
\rho(Q(I - LT_u)) < 1
\]  
(8)

where \( \rho(\cdot) \) denotes the spectral radius of the matrix. Furthermore, if the system (3) is controlled by the ILC algorithm (4) and the largest singular value is less than one, i.e.

\[
\sigma(Q(I - LT_u)) \leq \lambda < 1
\]  
(9)
then the system is stable and
\[ \| u_\infty - u_k \| \leq \lambda^k \| u_\infty - u_0 \| \] (10)
which means that the input vector will converge monotonically to its limit value \( u_\infty \).

3 Motivating example

Assume that the ILC algorithm is given by (6), with
\[ L(q) = \gamma q^\delta \] (11)
where the integer \( \delta > 0 \) and the scalar \( \gamma > 0 \), i.e. the error \( e_k(t) \) is shifted \( \delta \) steps and scaled by \( \gamma \) when filtered using \( L(q) \). Since \( e_k(t) \) is defined on a finite time interval \( t \in [0, N-1] \), an assumption has to be made concerning the values of \( e_k(t) \) outside this interval. As an illustration, two alternatives will be studied. First, let
\[ e_k(t) = 0, \quad t > N - 1 \] (12)
where it is implicitly assumed that the error has reached zero at the end of the time interval. Filtering \( e_k(t) \) using the filter \( L(q) = \gamma q^\delta \) with the condition (12) corresponds to multiplying the error signal vector \( e_k \) by the matrix
\[ L = \begin{pmatrix} 0 & \ldots & \gamma & \ldots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma & \ldots \\ 0 & \ldots & 0 & \ldots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \ldots & 0 \end{pmatrix} \] (13)
A second alternative, used in Moore (1998), Wang and Zhang (2009), is to put
\[ e_k(t) = e_k(N-1), \quad t > N - 1 \] (14)
which is motivated by situations where the error has not reached zero by the end of the movement. The alternative (14) corresponds to the matrix \( L \) given by
\[ L = \begin{pmatrix} 0 & \ldots & \gamma & \ldots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma & \ldots \\ 0 & \ldots & 0 & \gamma & \ldots \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \gamma & \ldots \end{pmatrix} \] (15)
Since the two alternatives (12) and (14) correspond to different matrices \( L \), it is clear that this will influence the criteria (8) and (9), respectively. This is illustrated numerically in Section 6.
4 A method for analysing boundary effects

The issue of boundary effects is caused by the finite time horizon character of ILC algorithms. It is therefore natural that the matrix framework for describing and analysing ILC algorithms is suitable for studying also the boundary effects. The basic idea for the analysis is to consider the problem in three steps. The first step is to extend the signal to be filtered and include assumptions concerning the properties of the signal outside the time interval \( t \in [0, N - 1] \). In the second step the signal is filtered by multiplying the extended signal vector by a suitably chosen matrix, and in the third step the signal is truncated back to its original length.

The emphasis here is on handling the boundary effects when filtering using the filter \( Q(q) \) or the matrix \( Q \), depending on the chosen implementation. For analysis of the effects of the chosen implementation, filtering using \( Q(q) \) is interpreted in the matrix formulation. One important special case is when implementing non-causal filters in order to obtain zero-phase shift. A standard way is to use a causal filter, and carry out forward-backward filtering. Consider a causal filter \( \bar{Q}(q) \),

\[
\bar{Q}(q) = \sum_{n=0}^{\infty} g_Q(n)q^{-n}
\]

and the corresponding \( N \times N \) matrix \( \bar{Q} \), defined similarly as in (2). Consider also a signal \( x(t) \), defined on the time interval \( t \in [0, N - 1] \), with the elements stacked in a vector \( x \). Filtering the signal \( x(t) \) through the filter \( \bar{Q}(q) \) can then be interpreted as the multiplication

\[
x_f = \bar{Q}x
\]

Zero-phase filtering is obtained by reversing the order of the data points in \( x_f \), filtering the data once more through \( \bar{Q}(q) \), and reversing the order of the data points. Using the matrix representation this corresponds to

\[
x_{ff} = \bar{Q}^T \bar{Q}x = Qx
\]

where \( Q \) can be used in the ILC update equation (4).

In order to carry out the signal extension step consider again the vector \( x \). Introduce an extended vector \( x_e \), where \( x \) is extended by \( m \) samples at the beginning and at the end of the time interval,

\[
x_e = (x(-m) \ldots x(0) \ldots x(N-1) \ldots x(N-1+m))^T
\]

where \( m \) is a design variable. The extension of the signal vector can be generated by the multiplication

\[
x_e = Q_e x
\]

where the \( N \times (N + 2m) \) matrix \( Q_e \) can be used to impose assumptions about the properties of \( x \) outside the time interval. One alternative is to assume that the signal has the same value
outside the time interval as at the end points, similar to (14). This corresponds to the matrix

$$Q_e = \begin{pmatrix}
1 & 0 & \ldots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
1 & 0 & \ldots & 0 & 0 \\
0 & 0 & \ldots & 1 & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & 0 & \ldots & 1 & 0 \\
0 & 0 & \ldots & 0 & 1
\end{pmatrix}$$

(21)

A second alternative is to do a linear extrapolation of the signal outside the time interval, and for example let

$$x(-l) = x(0) + (x(0) - x(l))$$

(22)

at the beginning of the signal and

$$x(N - 1 + l) = x(N - 1) + (x(N - 1) - x(N - 1 - l))$$

(23)

at the end of the signal. This corresponds to the matrix

$$Q_e = \begin{pmatrix}
\vdots & \ddots & \ddots & \vdots & \vdots \\
2 & -1 & 0 & \ldots & 0 & 0 & 0 \\
1 & 0 & 0 & \ldots & 0 & 0 & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \ldots & 0 & 1 & 0 \\
0 & 0 & 0 & \ldots & 0 & 0 & 1 \\
0 & 0 & 0 & \ldots & 0 & -1 & 2 \\
\vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots
\end{pmatrix}$$

(24)

The extension method (22) – (23) is used in the `filtfilt` command of MATLAB Signal Processing Toolbox (2010), where the extension length $m$ at the beginning and the end of the time interval is three times the filter order. This way of reflecting the signal at the end points is discussed in Elci et al. (2002), Plotnik and Longman (1999).

In the next step, i.e. the filtering step, the extended vector $x_e$, given by (20), is filtered via the multiplication

$$y_e = \tilde{Q}x_e$$

(25)

where $\tilde{Q}$ is the $(N + 2m) \times (N + 2m)$ matrix generated from the pulse-response coefficients $g_Q(l)$ of the corresponding filter $Q(q)$, similarly to how $Q$ is generated in (18). In the last step the signal $y_e$ is truncated by removing the $m$ first and last samples of the filtered vector, via

$$y = Q_I y_e = (0_{N \times m} \ 1_{N \times N} \ 0_{N \times m}) \ y_e$$

(26)

The entire filtering can hence be written as

$$y = Qx$$

(27)
where
\[ Q = Q_t \tilde{Q} Q_e \] (28)
is used in the ILC algorithm (4). A similar procedure can be used to represent the filtering using the filter \( L(q) \), but for simplicity this study will be limited to the cases (12) and (14).

The analysis of the algorithm properties for different choices of extension methods is hence carried out using the criteria (9) and (10) respectively, with \( Q \) defined by (28) and \( L \) defined in a similar way.

5 Simulation example

Consider a model consisting of two masses connected by a spring-damper pair, where the torque \( \bar{u}(t) \) to the first mass is the input signal. The angular position of the first mass (motor angular position) is denoted \( q_m(t) \), while the angular position of the second mass is \( q_a(t) \). \( M_m \) and \( M_a \) are the moments of inertia, \( k \) and \( d \) denote stiffness and damping of the spring, \( f_m \) is the viscous friction coefficient and \( \eta \) is the gear ratio. The system is described by the equations
\[
M_m \ddot{q}_m(t) = -f_m \dot{q}_m(t) - \eta k (\eta q_m(t) - q_a(t)) - \eta d (\eta \dot{q}_m(t) - \dot{q}_a(t)) + k \tau \bar{u}(t)
\]
\[
M_a \ddot{q}_a(t) = k (\eta q_m(t) - q_a(t)) + d (\eta \dot{q}_m(t) - \dot{q}_a(t))
\]
(29)
The parameter values in this example are \( \eta = 0.2, M_m = 0.0021, M_a = 0.0991, k = 8, d = 0.0924, f_m = 0.0713, \) and \( k_\tau = 0.122 \) respectively. The values correspond, with minor modifications, to the parameters obtained for the flexible robot arm in Gunnarsson et al. (2007). The system (29) is controlled using a discrete time controller with sampling interval \( T_s = 0.01 \) s. The motor angular position \( q_m(t) \) is the measured output of the system, denoted \( G(q) \) in Figure 1. The controller is a discrete-time PD-controller \( F(q) \) including a low-pass filter,
\[
F(q) = K_1 + \frac{K_2 q - K_3}{q - K_4}
\]
(30)
where \( K_1 = 5, K_2 = K_3 = 2, \) and \( K_4 = 0.905 \). The numerical illustration is based on the ILC algorithm (6) with the filter \( L(q) \) given as in (11) and the motor angular position error at iteration \( k \),
\[
u_{k+1}(t) = Q(q)(u_k(t) + \gamma q^\delta e_k(t))
\]
\[
e_k(t) = r_m(t) - q_{m,k}(t)
\]
(31)
(32)
Figure 1. Simulation setup; the two-mass system, represented by \( G(q) \), the feedback controller \( F(q) \), and ILC input \( u_k(t) \).
The ILC input signal \( u_k(t) \) is added to the reference signal \( r_m(t) \), see Figure 1. It means that, referring to (1),

\[
T_r(q) = T_u(q) = \frac{F(q)G(q)}{1 + F(q)G(q)}
\]  

(33)

The filter \( Q(q) \) is chosen as a zero-phase low-pass filter. The zero-phase filtering is carried out via forward-backward filtering using a second-order low-pass Butterworth filter with cutoff frequency \( f_c = 10 \) Hz. The ILC design parameter are chosen as \( \gamma = 0.9 \) and \( \delta = 10 \) and \( m = 6 \) in the signal extension step, (19). The choice \( m = 6 \) corresponds to the signal extension when using \texttt{filtfilt} and having a second-order filter.

6 Analysis and simulations

The analysis is carried out using the matrix representation of the discrete-time system and ILC algorithm. The matrix \( T_u \) is generated from the pulse-response coefficients of \( T_u(q) \) in (33) of the two-mass system described in Section 5. Similarly, the ILC algorithm (31) is converted to matrix form (4) represented by \( Q \) and \( L \), respectively. \( N = 400 \) number of samples is used in the calculations, motivated by simulation of the system during 4 s with \( T_s = 0.01 \) s in the simulations.

When filtering the error signal \( e_k(t) \) using the filter \( L(q) \), the error signal is assumed to be:

A: zero outside the time interval, i.e. \( L \) is interpreted according to (13).
B: extended with the last value, i.e. \( L \) is interpreted according to (15).

When filtering through \( Q(q) \), the signal is assumed to be:

I: not extended.
II: extended using the first and last value, i.e. \( Q_e \) is interpreted according to (21).
III: extended according to extrapolation, i.e. \( Q_e \) is interpreted according to (24).

Inserting \( T_u, Q, \) and \( L \) into the time-domain stability criterion (8) gives the results shown in Table 1 for the combinations of the cases A-B and I-III.

Table 1. Stability criterion (8) for the alternatives A-B and I-III. Alternatives A-II, A-III and B-III give a divergent algorithm \( \rho(\cdot) > 1 \), while the other alternatives imply convergence.

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.895</td>
<td>1.007</td>
<td>1.101</td>
</tr>
<tr>
<td>B</td>
<td>0.891</td>
<td>0.931</td>
<td>1.080</td>
</tr>
</tbody>
</table>

It can be seen that the way to treat the boundary effects plays a crucial role for the ILC algorithm properties. Alternatives A-II, A-III and B-III give a divergent algorithm \( \rho(\cdot) > 1 \), while the other alternatives imply convergence. For alternative II it plays an important role how the filtering via \( L(q) \) is handled.

Table 2 shows the singular value condition (9) for the different alternatives. The pattern is similar to the one seen in Table 1, with monotone convergence according to (10) for case I only. Results for case I also show that it can be expected that the choice of \( L \) will influence the convergence rate of the ILC algorithm. This will be further investigated in simulations.

The properties of the ILC algorithm (31) applied to the discrete-time two-mass system are evaluated using simulations in MATLAB for the implementation alternatives A-B and I-III. The reference \( r_m(t) \) for the two-mass system is a smooth unit step with a duration of 4 s and sampling interval \( T_s = 0.01 \) s. For the simulations, the ILC algorithm (31) is converted to the
Table 2. Singular value condition (9) for the alternatives A-B and I-III. Alternatives A-I and B-I gives monotone convergence \( \sigma(\cdot) < 1. \)

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.944</td>
<td>1.048</td>
<td>1.371</td>
</tr>
<tr>
<td>B</td>
<td>0.904</td>
<td>1.018</td>
<td>1.397</td>
</tr>
</tbody>
</table>

matrix form (4) according to the descriptions in Sections 3 and 4. The system performance is evaluated using the 2-norm of the vector of the error (32), that is, \( \| e_k \|_2. \)

Case I: Figure 2 shows \( \| e_k \|_2 \) as a function of iteration number for the cases A-I and B-I. The curves can be compared to the first column of Tables 1 and 2, respectively. The spectral radius is about the same for the two cases. The largest singular value is smaller for case B-I, which results in a better guaranteed convergence rate than for case A-I, see (9) and (10). A better convergence rate for case B-I compared to case A-I can be noticed in Figure 2. It is also seen that the magnitude of the final error is affected by how the boundary conditions are handled.

Case II: Figure 3 shows the corresponding curves for cases A-II and B-II. In case B-II the spectral radius is slightly larger than for cases A-I and B-I, while the spectral radius for case A-II is just outside the stability boundary. This property is seen in the simulation results, where the norm of the error for case A-II decreases during the first 50 iterations and then starts to increase.

Case III: Figure 4 illustrates the behaviour for cases A-III and B-III. The values for the spectral radius shown in Table 1 indicate an unstable ILC system for both cases. This is confirmed by the simulation results in Figure 4.

![Figure 2. 2-norm of the error as function of iteration number for the cases A-I and B-I.](image-url)

7 Conclusions

The paper has presented two main messages. First, it has been illustrated that the way to handle the boundary effects of the filtering operations in ILC algorithms can have substantial influence on the properties of the ILC algorithm in terms of stability, convergence and final error level. Second, a theoretical framework for analysis of the boundary effects has been proposed. The influence of the choice of method to handle the boundary effects is illustrated using a simulation example, and the observed properties of the ILC algorithm are supported by the theoretical analysis. One issue for future work is to study other ways of extending the signal vector outside the time interval than the ones considered in the paper. Another issue is to investigate if there
is one method that gives overall better performance than other. It should be noted that the observations in this paper are based on one particular example.

Acknowledgements

This work was supported by Vinnova’s Industry Excellence Centre LINK-SIC and the strategic research centre ELLIIT at Linköping University, which is gratefully acknowledged.

References


