Some Implementation Aspects of Iterative Learning Control

Johanna Wallén, Svante Gunnarsson, Mikael Norrlöf

Division of Automatic Control
E-mail: johanna@isy.liu.se, svante@isy.liu.se, mino@isy.liu.se

17th September 2010

Report no.: LiTH-ISY-R-2967
Submitted to 18th IFAC World Congress 2011, Milano, Italy

Address:
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

WWW: http://www.control.isy.liu.se
Abstract
Some implementation aspects of Iterative Learning Control (ILC) are considered. Since the ILC algorithm involves filtering of various signals over finite time intervals, often using non-causal filters, it is important that the boundary effects of the filtering operations are handled in an appropriate way when implementing the ILC algorithm. The paper illustrates in both theoretical analysis using the matrix description and in simulations of a two-mass system that the method of implementation for handling the boundary effects can have large influence over stability and convergence properties of the ILC algorithm.

Keywords: Iterative methods, Learning control, Convergence, Filtering, Implementation, Boundary conditions.
Some Implementation Aspects of Iterative Learning Control∗

Johanna Wallén∗Svante Gunnarsson∗Mikael Norrlöf∗

∗ Division of Automatic Control, Department of Electrical Engineering,
Linköping University, SE-581 83 Linköping, Sweden,
e-mail: {johanna, mino, svante}@isy.liu.se

Abstract: Some implementation aspects of Iterative Learning Control (ILC) are considered. Since the ILC algorithm involves filtering of various signals over finite time intervals, often using non-causal filters, it is important that the boundary effects of the filtering operations are handled in an appropriate way when implementing the ILC algorithm. The paper illustrates in both theoretical analysis using the matrix description and in simulations of a two-mass system that the method of implementation for handling the boundary effects can have large influence over stability and convergence properties of the ILC algorithm.

Keywords: Iterative methods, Learning control, Convergence, Filtering, Implementation, Boundary conditions.

1. INTRODUCTION

The basic assumption in Iterative Learning Control (ILC) is that a system carries out an operation repeatedly, and that it is possible to compensate for errors and disturbances that occur in a repetitive manner. The first publications in the field appeared in the mid eighties, see for example Arimoto et al. [1984], Casalino and Bartolini [1984] and Craig [1984]. ILC is now an intense research area and recent overviews of the area can be found in the surveys Bristow et al. [2006] and Ahn et al. [2007].

The ILC field covers both linear and nonlinear problems, and ILC algorithms are studied in both continuous and discrete time. This paper deals with linear systems operating in discrete time. The purpose of the paper is to study some implementation alternatives for handling the boundary effects that occur in the filtering operations in the ILC algorithm. This topic appears to have received fairly limited attention in the literature, even though it can be of large practical importance. Some comments on implementation aspects and boundary effects are given in for example Moore [1998a], Longman [2000], van de Wijdeven [2008] and Wang and Zhang [2009]. In this paper it will be illustrated using both theoretical analysis and simulations that it can have large influence over the performance of the ILC algorithm.

The paper is organised as follows. Section 2 presents the type of systems and ILC algorithms that are considered in the paper, together with some convergence results that will be used for the analysis. In Section 3 a brief motivating example is given, and Section 4 presents a way to handle boundary effects in a systematic way. The results are illustrated using a numerical example in Section 5, and finally Section 6 gives some conclusions.

2. SYSTEM AND ILC ALGORITHM DESCRIPTION

2.1 System description

The paper considers linear discrete-time systems described by

\[ y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) \]  (1)

with reference signal \( r(t) \), ILC input signal \( u_k(t) \) and output signal \( y_k(t) \) at iteration \( k \) and \( q \) denoting the shift operator. For simplicity load and measurement disturbances are omitted in this paper. The system representation (1) covers systems operating in open loop, where \( T_u(q) = 0 \), as well as systems operating in closed loop, where \( T_u(q) \) and \( T_r(q) \) include the feedback controller. All signals are defined on a finite time interval \( t = 1:T_x, t \in \{0, \ldots, N-1\} \) with \( N \) number of samples. The sampling interval \( T_x = 1 \) is used in this paper, if nothing else is stated.

Parallel to the system description in (1), a matrix description of the system and the ILC algorithm will be used. This description is closely related to the descriptions of systems using ILC in Phan and Longman [1988], Moore [1998b] and Tousain et al. [2001] among others. First, define the vector \( r \) containing the \( N \)-sample sequence of the reference signal \( r(t) \) as

\[ r = (r(0), \ldots, r(N-1))^T \]  (2)

Next, define the vectors \( u_k \) and \( y_k \) similarly. The matrix \( T_r \) is formed by the pulse response coefficients \( g_r(l) \) of the transfer operator \( T_r(q) \) in (1). This results in the \( N \times N \) Toeplitz matrix

\[
T_r = \begin{pmatrix}
g_r(0) & 0 & \cdots & 0 
g_r(1) & g_r(0) & \cdots & 0 
\vdots & \vdots & \ddots & \vdots 
g_r(N-1) & g_r(N-2) & \cdots & g_r(0)
\end{pmatrix}
\]  (3)

and \( T_u \) is defined similarly. It can be noted that the matrices are lower-triangular, since the transfer operators
are causal. Using the matrix representation, the system description (1) can be written as
\[ y_k = T_t r + T_u u_k \]  (4)

2.2 ILC algorithm

Considering linear, discrete-time ILC algorithms there are two main alternative ways to implement a particular algorithm. The first alternative is the matrix formulation in which the new ILC input signal vector is computed according to the expression
\[ u_{k+1} = Q(u_k + L e_k) \]  (5)

where
\[ e_k = r - y_k \]
and where \( Q \) and \( L \) are \( N \times N \) matrices. One way to determine the matrices \( Q \) and \( L \) is to use an optimisation approach to design ILC algorithms, as is described in for example Gunnarsson and Norrlöf [2001].

Another implementation alternative is a filter formulation where the new ILC input is computed as
\[ u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \]  (6)

where
\[ e_k(t) = r(t) - y_k(t) \]
and \( Q(q) \) and \( L(q) \) are possibly non-causal filters. Each alternative has advantages and disadvantages. The matrix formulation (5) is in general more computational demanding, but covers a larger class of algorithms. On the other hand, the filter form (6) is less general and requires less computations.

For both alternatives it is necessary to take care of the boundary effects in the implementation of the ILC algorithm. The main purpose of this paper is to illustrate that the matrix formulation (5) offers a systematic framework for analysing how this affects the algorithm properties. In order to carry out the analysis of an ILC algorithm in filter form (6) it is necessary to interpret it using the matrix formulation (5), and this will be discussed in more detail in Sections 3 and 4.

2.3 Stability and convergence properties

The stability and convergence properties of the ILC algorithm can be analysed by using the results in for instance Norrlöf and Gunnarsson [2002]. First, Corollary 3 in Norrlöf and Gunnarsson [2002] states the well-known result that the system (4) controlled by the ILC algorithm (5) is stable if and only if
\[ \rho(Q(I - LT_u)) < 1 \]  (7)
where \( \rho(\cdot) \) denotes the spectral radius. Combining (4) and (5) also gives
\[ u_{k+1} = Q(I - LT_u)u_k + QL(I - T_r)r \]

A useful result concerning the properties of the ILC input signal vector is that, see for example Theorem 9 in Norrlöf and Gunnarsson [2002], if the system (4) is controlled by the ILC algorithm (5) and
\[ \sigma(Q(I - LT_u)) \leq \lambda < 1 \]  (8)
that is, the largest singular value is less than one, then the system is stable and
\[ \|u_\infty - u_k\| \leq \lambda^k \|u_\infty - u_0\| \]  (9)

with \( u_\infty \) defined as
\[ u_\infty = (I - Q(I - LT_u))^{-1}QL(I - T_r)r \]  (10)
That is, the condition (9) results in \( u_k \) converging to the limit value \( u_\infty \) exponentially and without overshoot (monotone convergence). Using the asymptotic control signal \( u_\infty \) from (10) it is then straightforward to derive the limit of the error signal vector, and it is given by
\[ e_\infty = (I - T_u(I - Q(I - LT_u)))^{-1}QL(I - T_r)r \]  (11)

3. MOTIVATING EXAMPLE

In order to illustrate the issue concerning implementation alternatives for handling the boundary effects, now assume that the ILC algorithm is given by
\[ u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \]  (12)
where
\[ L(q) = \gamma q^\delta \]  (13)
with the integer \( \delta > 0 \) and the scalar \( \gamma > 0 \). This means that in the filtering of the error signal \( e_k(t) \) using the filter \( L(q) \), the error signal is shifted \( \delta \) steps and scaled by \( \gamma \). Since \( e_k(t) \) is only defined in the time interval \( t = l, \quad l \in \{0, \ldots, N-1\} \) an assumption has to be made concerning the values of \( e_k(t) \) outside this time interval, that is, “future” values of the error signal. As an illustration, two alternatives will be studied where the first alternative is to put
\[ e_k(t) = 0, \quad t > N - 1 \]  (14)
This implies that filtering \( e_k(t) \) using \( L(q) \), with condition (14), corresponds to multiplying the error signal vector \( e_k \), see (5), by the matrix
\[ L = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \]  (15)
The second alternative to handle the boundary effect is to put
\[ e_k(t) = e_k(N - 1), \quad t > N - 1 \]  (16)
This can be motivated by situations where the error has not reached zero by the end of the movement. It can also be noted that the alternative (16) is used in for example Moore [1998a], Longman [2000] and Wang and Zhang [2009]. This alternative (16) corresponds to multiplication of \( e_k \) by using the matrix
\[ L = \begin{bmatrix} 0 & 0 & \gamma & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \]  (17)

Since the two alternatives (14) and (16) for handling the boundary effects correspond to different choices of \( L \), it is clear that this will have influence on the criteria (7) and (8), respectively. This will be further illustrated by a simulation example in Section 5.
4. HANDLING OF BOUNDARY EFFECTS

In this section the emphasis will be on handling the boundary effects when filtering using the filter \( Q(q) \) or the matrix \( \bar{Q} \), depending on the chosen implementation.

As mentioned above the matrix form (5) is more general and better suited for analysis, but the filter form (6) is less computational demanding. It is therefore of interest to be able to interpret the filtering using \( Q(q) \) in the matrix framework. An important special case is when this is done using non-causal filtering in order to obtain zero-phase shift. A standard way to carry out such filtering is to use a conventional causal filter of for example Butterworth type and carry out forward-backward filtering in order to obtain a zero-phase filter. Consider therefore a causal filter \( Q(q) \) given by

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

and the corresponding matrix

\[
Q = \begin{bmatrix}
g_Q(0) & 0 & \ldots & 0 \\
g_Q(1) & g_Q(0) & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
g_Q(N-1) & \ldots & g_Q(1) & g_Q(0)
\end{bmatrix}
\]  \hspace{1cm} (18)

Now consider a signal vector

\[
x = (x(0), \ldots, x(N-1))^T
\]

Filtering \( x \) through \( Q(q) \) can then be interpreted as the matrix-vector multiplication

\[
x_f = Qx
\]

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

\[
x_{ff} = Qx
\]

with the matrix \( Q \), see (5), given by

\[
Q = Q^TQ
\]  \hspace{1cm} (20)

The aim is now to present a systematic approach to analyse how different ways to handle the boundary effects will influence the properties of the ILC algorithm. The key idea is to extend the signal vector to be filtered and incorporate some assumptions concerning the properties of the signal outside the given time interval. Consider therefore the vector \( x \) defined by (19). Then introduce a corresponding extended vector where the original vector \( x \) is extended by \( n \) samples at the beginning and the end, that is,

\[
x_e = (x(-n), \ldots, x(0), \ldots, x(N-1), \ldots, x(N-1+n))^T
\]  \hspace{1cm} (21)

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

\[
x_e = Q_e x
\]  \hspace{1cm} (22)

where the \( N \times (N+2n) \) matrix \( Q_e \) can be used to impose assumptions about the properties of the signal outside the given time interval. One alternative is to assume that the signal has the same value outside the interval as at the end points, similar to the assumption in (16). This corresponds to the matrix

\[
Q_e = \begin{bmatrix}
1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & & \vdots & \vdots & & \vdots \\
1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 1 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 0 & \ldots & 1 \\
0 & 0 & \ldots & 0 & 0 & \ldots & 0
\end{bmatrix}
\]  \hspace{1cm} (23)

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

\[
x(-m) = x(0) + (x(0) - x(m))
\]

at the beginning of the signal and

\[
x(N - 1 + m) = x(N - 1) + (x(N - 1) - x(N - 1 - m))
\]

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

\[
Q_e = \begin{bmatrix}
2 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
2 & -1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 & 1 & \ldots & 0 \\
0 & 0 & \ldots & 0 & -1 & 2 & \ldots & 0 \\
\vdots & \vdots & & \vdots & \vdots & & \vdots 
\end{bmatrix}
\]  \hspace{1cm} (24)

It can be noted that this method is used in the Matlab command \texttt{filtfilt}, where the signal is extended by \( n \) samples at the beginning and at the end, see (21). The extension length is given by

\[
n = \max\{\text{length}(a), \text{length}(b)\} - 1
\]  \hspace{1cm} (25)

for a filter with denominator coefficients \( a \) and numerator coefficients \( b \). \texttt{MATLAB}, 2010

In the next step the extended vector \( x_e \), given by (22), is filtered via the matrix-vector multiplication

\[
y_e = Q_e x_e
\]

where \( Q_e \) is the \( (N+2n) \times (N+2n) \) matrix generated from the pulse response coefficients \( \{g_Q(0), \ldots, g_Q(N-1+2n)\} \) of the corresponding filter \( Q(q) \), similarly to how \( Q \) is generated in (20). In the last step the signal \( y_e \) is truncated by removing the first and last \( n \) samples of the filtered vector. This is carried out via the multiplication

\[
y = Q y_e
\]

where

\[
Q_t = \begin{bmatrix}
I_{N \times N} & 0_{N \times n} & 0_{N \times n}
\end{bmatrix}
\]

The entire filtering can hence be summarised as

\[
y = Q_t Q Q_e x = Q x
\]

with the matrix \( Q \), see (5). A similar procedure can of course be used to represent the filtering using \( L(q) \) in (6), but for simplicity the study will be limited to the cases represented by (14) and (16).
5. NUMERICAL ILLUSTRATION

5.1 System description

Consider a flexible two-mass mechanical system, illustrated in Fig. 1. The torque $\tau(t)$ applied to the first mass is the input to the system, and the angular position $q_m(t)$ of the first mass, also referred to as motor angle, is the output. The parameters $k$ and $d$ denote stiffness and damping of the spring respectively, and $f_m$ is the viscous friction coefficient. $J_m$ and $J_a$ denote the moments of inertia and $r_g$ is the gear ratio. The system is described by the equations

$$J_m \ddot{\theta}_m(t) = -f_m \dot{\theta}_m(t) - r_g k (r_g \theta_m(t) - \theta_a(t))$$
$$-r_g d (r_g \dot{\theta}_m(t) - \dot{\theta}_m(t)) + k_r \tau(t)$$
$$J_a \ddot{\theta}_a(t) = k (r_g \theta_m(t) - \theta_a(t)) + d (r_g \dot{\theta}_m(t) - \dot{\theta}_a(t))$$

In the simulations the parameter values in Table 1 will be used. The parameter values correspond with some minor modifications to the ones obtained for the flexible robot used in Gunnarsson et al. [2007] for studying ILC applied to flexible mechanical systems.

![Fig. 1. Flexible two-mass model of a mechanical system.](image)

The system is controlled by using a discrete-time PD-controller, obtained by converting

$$F_C(s) = K_p + \frac{K_d s}{1 + T_f s}$$

(26)

to discrete time using a sampling interval of $T_s = 0.01$ s. The parameter values of the controller are given in Table 2. The structure of the control system is depicted in Fig. 2, where it is shown that the ILC input signal is added to the reference signal of the conventional control system. In the figure the variable $\tau_c(t)$ denotes the control signal (torque) generated by the feedback controller $F(q)$ at iteration $k$ and output is the motor angle. In the simulation example it means that, referring to the relation (1),

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

(27)

![Fig. 2. Control system, where the ILC input signal $u_k(t)$ is added to the reference signal $r(t)$ of the conventional control system at iteration $k$.](image)

The numerical illustration will be based on the ILC algorithm (12) with the filter $L(q)$ given as in (13), that is,

$$u_{k+1}(t) = Q(q)(u_k(t) + \gamma e_k(t))$$

(28)

The filter $Q(q)$ is chosen as a zero-phase low-pass filter with the zero-phase filtering carried out via forward-backward filtering using a second-order low-pass Butterworth filter with cut-off frequency $f_c = 10$ Hz. The ILC design parameter values are given in Table 3, together with the length $n$ of the signal extension, see (21). The choice $n = 6$ corresponds to the signal extension given by (25) when using the MATLAB function `filtfilt` when having a second-order filter.

The purpose of the paper is to provide a qualitative illustration that the implementation of the ILC algorithm can play an important role for the algorithm behaviour. Therefore only one set of design variables of the ILC algorithm will be evaluated.

### Table 1. Model parameter values.

<table>
<thead>
<tr>
<th>$r_g$</th>
<th>$J_m$</th>
<th>$J_a$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.0021</td>
<td>0.0991</td>
<td>8</td>
</tr>
<tr>
<td>$d$</td>
<td>$f_m$</td>
<td>$K_r$</td>
<td>$K_d$</td>
</tr>
<tr>
<td>0.0924</td>
<td>0.0713</td>
<td>0.122</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Controller parameter values.

<table>
<thead>
<tr>
<th>$K_p$</th>
<th>$K_d$</th>
<th>$T_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.2</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Table 3. ILC design parameter values.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$\delta$</th>
<th>$f_c$</th>
<th>$n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>10</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

5.2 Analysis

The analysis will be carried out using the matrix representation of the system and the ILC algorithm. Therefore the matrix $T_u$, see (3), containing the pulse response coefficients of the transfer operator $T_u(q)$ from the ILC input to the output is generated. In a similar way the ILC algorithm (28) is converted to matrix form (5) represented by the matrices $Q$ and $L$, respectively.

The cases to be studied are the following. When filtering the error signal $e_k(t)$ using $L(q)$, the error signal is assumed to be:

A zero outside the time interval, that is, $L$ is interpreted according to (15).

B extended with the last value, that is, $L$ is interpreted according to (17).

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

I not extended.

II extended using the first and last value, that is, $Q_e$ is interpreted according to (23).

III extended according to extrapolation, that is, $Q_e$ is interpreted according to (24).

First the convergence criterion given by (7) is investigated. Inserting the matrices $T_u, Q$, and $L$ into (7) give the results shown in Table 4 for the cases A-B, I-III when using $N = 400$ number of samples. The results show that the implementation plays a crucial role for the convergence properties. Alternatives A-II, A-III and B-III give a divergent algorithm with $\rho(Q(I - LT_u)) > 1$, while the other

<table>
<thead>
<tr>
<th>$\rho(Q(I - LT_u))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
</tr>
</tbody>
</table>

In the simulation example it means that, referring to the relation (1),

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

(27)

Fig. 2. Control system, where the ILC input signal $u_k(t)$ is added to the reference signal $r(t)$ of the conventional control system at iteration $k$. 
alternatives imply convergence. It can also be seen that for alternative II it plays an important role how the filtering via $L(q)$ is handled.

Table 5 shows the results when evaluating the singular value condition (8), and the pattern is similar to the one in Table 4, with monotone convergence according to (9) for case I. Results from case I also show that it can be expected that the choice of $L$ will have influence on the convergence rate of the ILC algorithm. This will be further investigated in the simulations below.

### Table 4. Convergence criterion (7).

<table>
<thead>
<tr>
<th></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>

### Table 5. Singular value condition (8).

<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>

### 5.3 Simulations

The ILC algorithm (28), using the design parameters given in Table 3, with the motor-angle error $e_k(t) = r(t) - q_{m,k}(t)$ (29) and the different implementation alternatives A-B and I-III is now evaluated using simulations. The motor-angle reference signal $r(t)$ for the control system is shown in Fig. 3 with a duration of 4 s at the sampling interval $T_s = 0.01$ s. For the simulations the ILC algorithm (28) is converted to matrix form (5) according to the description in Sections 3 and 4.

Case I

Fig. 4 shows the 2-norm of the error (29) as a function of iteration number for the cases A-I and B-I. The curves can be compared with the first columns of Tables 4 and 5, respectively. The eigenvalues are about the same for the two cases, but the largest singular value is smaller for case B-I, which results in a faster convergence rate for case B-I as can be seen in Fig. 4. In the figure it can also be noted that the magnitude of the final error is affected by how the boundary conditions are handled.

Case II

Fig. 5 shows the corresponding curves for cases A-II and B-II, respectively. In case B-II the largest eigenvalue is slightly larger than for cases A-I and B-I, while the eigenvalue 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

Finally Fig. 6 illustrates the behaviour for cases A-III and B-III, respectively. Here the eigenvalues shown in the third column of Table 4 indicate that the ILC algorithm is unstable for both cases, and this is confirmed by the simulation results in Fig. 6.

The function `filtfilt` [MATLAB, 2010] handles boundary conditions similar to the implementation alternative III above but the function also involves a way of handling initial conditions that depends on the signal itself, which implies that the behaviour is not predictable without knowledge of the filtered signal. In order to illustrate some properties, this case is also investigated. Fig. 7 shows the 2-norm of the error when the filter form (28) of the ILC algorithm is used together with the `filtfilt` command, combined with the two alternative ways A-B of handling...
the filtering with $L(q)$. Also here it is beneficial to use alternative B. Running the algorithm for more iterations gives that the error signal settles at a constant value after approximately 1000 iterations.

**Fig. 6.** 2-norm of the error as function of iteration number for the cases A-III and B-III.

**Fig. 7.** 2-norm of the error as function of iteration number when the filter form (28) of the ILC algorithm is used for the cases A and B together with the `filtfilt` command.

6. CONCLUSIONS

Some implementation aspects have been discussed that occur in the non-causal filtering operations of the ILC algorithms. Different ways to treat the boundary effects have been analysed both theoretically, by interpreting the filtering in the matrix description, and by using simulations of a two-mass mechanical system. The results indicate that the implementation and the method to handle boundary effects can play an important role for the behaviour of the ILC algorithm in terms of stability, convergence and final error level.

REFERENCES


Some Implementation Aspects of Iterative Learning Control

Some implementation aspects of Iterative Learning Control (ILC) are considered. Since the ILC algorithm involves filtering of various signals over finite time intervals, often using non-causal filters, it is important that the boundary effects of the filtering operations are handled in an appropriate way when implementing the ILC algorithm. The paper illustrates in both theoretical analysis using the matrix description and in simulations of a two-mass system that the method of implementation for handling the boundary effects can have large influence over stability and convergence properties of the ILC algorithm.

Keywords: Iterative methods, Learning control, Convergence, Filtering, Implementation, Boundary conditions.