ISSN 0280–5316
ISRN LUTFD2/TFRT--5672--SE

Iterative Learning Control with Application to Robotics

Domenico Scalamogna

Department of Automatic Control
Lund Institute of Technology
June 2001

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 28, 2001

Document Number
ISRN LUTFD2/TFRT--5672--SE

Author(s)
Domenico Scalamogna

Supervisor
Rolf Johansson, Anders Robertsson,
Mattias Grundelius, and Edoardo Mosca

Sponsoring organisation

Title and subtitle
Iterative Learning Control with Application to Robotics

Abstract

Many machines and robots working today in factories are programmed to perform the same task repeatedly. By observing the tracking error in each iteration of the same task it becomes clear that it is actually repetitive, even though disturbances from noise and possibly sligthly changing friction dynamics affect the response.

The idea of Iterative learning Control (ILC) is to use the knowledge from the previous iterations of the same task to reduce the tracking error the next time the task is performed. ILC utilizes the tracking error knowledge from the previous iteration to change the input signal to the system.

In the thesis ILC is applied to an ABB Irb-2000 industrial robot. Using ILC the tracking error on the motor side has been reduced without changing the internal structure or any parameter in the robot controller.

Three different ILC algorithms are considered in the thesis. Also some important theorems about ILC stability are taken into account.

Two of these three ILC algorithms have been applied to the robot to improve the tracking of desired trajectories.

ILC has also been used in order to improve the robot motion of an open container with liquid. The purpose was to shorten the motion time of the package transfer with control of the slosh inside.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
63

Recipient's notes

Security classification

Contents

Acknowledgments

I would like to thank Professor Rolf Johansson for the welcome, the availability and the precious advice he gave me during the time I spent to accomplish my thesis work.

I would like to thank Dr. Anders Robertsson for all the theoretical and practical support, including that he taught me how to use and program robots. I must say that he spent many hours during the weekends and also some nights in the robot lab to help me for the solution of various problems.

I would like to thank my Italian supervisor Professor Edoardo Mosca for the trust he has in me and the availability he had every time I contacted him for advice.

I also want to thank Tekn. Lic. Mattias Grundelius for the suggestions, the theoretical subjects, the equipment and the software. Without his help and contribution it would not have been possible to develop the most important experiments of my thesis.

I am grateful to Civ. Ing. Leif Andersson for teaching me how to use LaTeX and for the availability he had to solve all the problems I got with computers. Moreover I like very much his sense of humor.

1. Introduction

1.1 A brief history

The idea of using an iterative method to compensate for a repetitive error is not new. When letting a machine do the same task repeatedly it is, at least from an engineering point of view, very sound to use knowledge from previous iterations of the same task to try to reduce the error the next time the task is performed. The first academic contribution to what today is called ILC appears to be a paper by Uchiyama (1978). Since it was published in Japanese only, the ideas did not become widely spread. What is a bit remarkable is however that an application for an US patent on 'Learning control of actuators in control systems' (Garden 1971) was done already in 1967 and it was accepted as a patent in 1971. The idea in the patent is to store a command signal in a computer memory and iteratively update the command signal using the error between the actual response and the desired response of the actuator. This is clearly an implementation of ILC, although the actual ILC updating equation is not explicitly formulated in the patent.
From an academic perspective it was not until 1984 that ILC started to become an active research area. Arimoto (1984), Casalino and Bartolini (1984), and Craig (1984) independently published papers about a method that iteratively could compensate for model errors and disturbances. The name Iterative Learning Control was first introduced by Arimoto.
The development of ILC stems originally from the robotics area where repetitive motions show up naturally in many applications.
The focus for the ILC research in the late 90's and in the beginning of the 00's is not so easy to establish but it seems that it has moved from being very focused on stability towards also considering design and performance.

1.2 ILC in relation to other techniques

The classical formulation of the Iterative Learning Control problem is, given a reference trajectory and a system, find (using an iterative procedure) the input to the system such that the output follows the desired trajectory as well as possible. Clearly, if a description of the system is available the optimal solution is to invert this description (if possible) and use this to calculate the input that produces the desired output. This is a one-step procedure which can be considered as a feed-forward control scheme. This approach has been applied successfully, for example, in robotics control where it is referred to as inverse dynamics.
If the system presentation, describing the mapping from input to output, is not completely known, then it is obvious that the inverse dynamics approach will never achieve a perfect tracking. If instead it is assumed that the structure of the system is known while the exact value of one or more of the parameters are unknown, another well known technique can be applied, namely identification. Normally identification together with control

is referred to as adaptive control. The adaptive control approach is very appealing since it will, theoretically, give a good behavior for all input signals, during all working conditions. It should be noted that this is true as long as the structure of the system is correct and some conditions on excitation are met.
Iterative Learning Control is an alternative to the inverse dynamics and adaptive control approaches in the case when, given a particular reference trajectory and a system, the input signal shall be calculated such that the output follows the desired output as well as possible. Using ILC the control signal is found by an iterative procedure. This can be seen as an iterative search procedure which obviously has to converge to give a successful result. Convergence has been and still is an important research field for ILC.

1.3 Thesis outline

Chapter 2 describes three ILC approaches: Heuristic ILC, Model Based ILC and Optimization Based ILC. Some theorems about ILC stability are also shown. In the last section of the chapter simulation results are presented.

Chapter 3 gives an overview of the robot system used in the experiments. The first section presents the physical robot, the second section the experimental platform and the third section the interaction between the robot and Matlab.

In Chapter 4 are shown some experimental results regarding the application of ILC algorithms to the robot.

Chapter 5 introduces the problem of the motion control of open containers with slosh constraints. An algorithm for the slosh control is described together with simulations. The last section reports experimental results obtained applying ILC to the robot in order to improve the motion performance of open containers with liquid.

Conclusions and some recommendations for further work are given in Chapter 6.

2. ILC Description

2.1 ILC formulation

Suppose to have a closed loop system (process + controller) and a reference trajectory which is sent repeatedly as input. For example our system could be an industrial robot that has to repeat every time the same task, therefore the same input trajectories are sent to robot to be tracked. By observing the tracking error in each iteration of the same task it becomes clear that it is actually highly repetitive,even though disturbances from noise and possibly slightly changing friction dynamics affect the response.

Iterative Learning Control (ILC) allows to iteratively compensate for and, hence to remove this error. The main idea of ILC is that when repetitive tasks must be executed it is possible to improve the control system in the current task by using the information from the tasks previously executed.

At every iteration an external control signal (ILC) will be added to the input. The aim is to have, iteration by iteration, a smaller tracking error, ideally converging to zero value. Figure 2.1 shows the scheme we are taking into account. Let $y_d(t)$ be the reference signal that we want

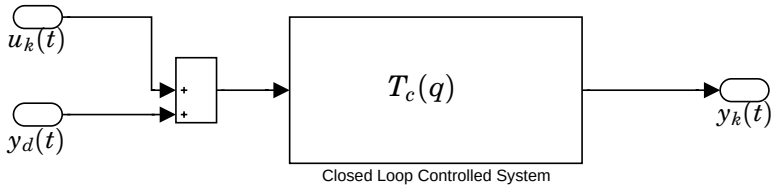

Figure 2.1 ILC applied to the controlled system

to track and $y_k(t)$, $u_k(t)$, $e_k(t) = y_d(t) - y_k(t)$ are the output, the ILC input and the tracking error of the system at iteration k respectively. The following equations describe the problem:

$$y_k(t) = T_c(q)y_d(t) + T_c(q)u_k(t) \tag{2.1}$$
$$e_k(t) = y_d(t) - y_k(t) \tag{2.2}$$
$$u_k(t) = Q(q)[u_{k-1}(t) + L(q)e_{k-1}(t)] \tag{2.3}$$

where $T_c(q)$ is the closed loop transfer function of the system and q is the time shift operator.
Equation (2.3) is the ILC control added to the system and $Q(q)$, $L(q)$ are filters to be designed. In fact our task is to design these filters so that the system has satisfactory tracking performance.

Now if we consider the equations (2.1), (2.2), (2.3), after some calculations, we obtain the following recursive expression for the tracking error:

$$e_k(t) = [(1 - Q)(1 - T_c)]y_d(t) + [Q(1 - L \cdot T_c)]e_{k-1}(t) \tag{2.4}$$

By analyzing Equation (2.4), it can be observed that in order to have asymptotically zero iterative error it is necessary to choose $Q(q) = 1$. Only with this choice the first term on the right hand side of Equation (2.4) can be zero and if the filter $(1 - L \cdot T_c)$ is asymptotically stable it will result in $\lim_{k\to\infty} e_k = 0$. We must observe that in real applications, for robustness considerations, the filter $Q(q)$ is chosen as a low-pass filter and consequently this implies that the iterative tracking error will not converge to zero asymptotically. Anyway, if an opportune choice of $Q(q)$ is done, the asymptotic error will reach values close to zero and that is satisfactory.

It is important when we start to iterate and update the control signal using ILC, that the system is stable along the iterations. If the error starts growing as a function of iterations, nothing is gained from using the method. It is therefore important to introduce the following criterion for the convergence of ILC.

THEOREM 2.1—(NORRLÖF, 1999)
Given a SISO LTI system on the form (2.1) using an ILC given by (2.3), convergence will be achieved if

$$| 1 - L(e^{i\omega t_s}) \cdot T_c(e^{i\omega t_s}) | < | Q^{-1}(e^{i\omega t_s}) | \tag{2.5}$$

where $\omega \in [-\pi, \pi]$ and t_s is the sampling time. □

The problem in ILC control is to design the filters $Q(q)$ and $L(q)$ subject to the stability constraint (2.5). After this brief description we can now introduce three different ILC algorithms [Norrlöf, 2000].

2.2 Heuristic ILC

The first design algorithm does not use much knowledge of a system model in the design of the $Q(q)$ and $L(q)$ filters. Consequently it might not give very good performance.
A heuristic design procedure:

1. Choose the Q filter as a low-pass filter with cut-off frequency such that the band-width of the learning algorithm 'is sufficient'.
2. Let $L(q) = \kappa q^{\delta}$. Choose κ and δ such that the stability criterion (2.5) is fulfilled. Normally it suffices to choose δ as the system time delay and $\kappa : 0 < \kappa \leq 1$ to get a stable ILC system.

It is to be observed that in the algorithm shown the filters $Q(q)$ and $L(q)$ can be chosen anti-causal. The implementation of anti-causal filters is possible in ILC algorithms because, in order to calculate the control sequence $\mathbf{u}_k$ at iteration k, $Q(q)$ and $L(q)$ operate on the whole time sequences $\mathbf{u}_{k-1}$ and $\mathbf{e}_{k-1}$, already known from the previous iteration $k - 1$.

2.3 Model Based ILC

Here we consider an algorithm which uses the knowledge of a system model to design the filters Q and L.

Algorithm:

1. Build a model of the relations between the ILC input and the resulting correction on the output (i.e. find a model $\widehat{T}_c$ of T_c).
2. Choose a filter $H_b(q)$ such that it represents the desired convergence rate for each frequency. Normally this means an high-pass filter.
3. Calculate L by $L(q) = \widehat{T}_c^{-1}(q)(1 - H_b(q))$.
4. Choose the Q filter as a low pass with cut-off frequency such that the band-width of the resulting ILC is high enough and desired robustness is achieved.

The meaning of the filter $H_b(q)$ can be explained by Equation (2.4). In fact if we choose $Q(q) = 1$ we obtain $e_k(t) = (1 - L \cdot T_c)e_{k-1}(t)$ and if $\widehat{T}_c$ is a good model (i.e. $\widehat{T}_c \approxeq T_c$) then $H_b \approxeq (1 - L \cdot T_c)$. So clearly the choice of H_b decides the nominal convergence rate for the error.

In the frequency domain the filter H_b can be adjusted to give a slower but more robust convergence for some frequencies. The choice of H_b must be realizable. It is clearly not possible to choose H_b small for frequencies where the model is very uncertain since this most likely leads to a divergent behavior of the resulting ILC. The choice of H_b has, therefore, also to include robustness considerations, although robustness is also achieved with the Q filter.

The resulting L filter might have an unnecessary high degree, therefore it can be possible to make a model reduction.

2.4 Optimization Based ILC

This algorithm derives from the minimization of a quadratic cost and in this meaning we refer to it as optimal.
In general the system at iteration k is described by :

$$y_k(t) = T_r(q)y_d(t) + T_u(q)u_k(t) \tag{2.6}$$

where $T_r(q)$ is the transfer function from the reference to the output and $T_u(q)$ the transfer function from the ILC control to the output. Let us now introduce the matrix form for the system (2.6):

$$\mathbf{y}_k = \mathbf{T}_r\mathbf{y}_d + \mathbf{T}_u\mathbf{u}_k \tag{2.7}$$

where $\mathbf{y}_k$, $\mathbf{y}_d$ and $\mathbf{u}_k$ are time-indexed vectors. $\mathbf{T}_r$ and $\mathbf{T}_u$ are lower triangular Toeplitz matrices obtained from the impulse response of $T_r(q)$ and $T_u(q)$ respectively. Let the quadratic criterion be formulated according to

$$J_{k+1} = \mathbf{e}_{k+1}^T\mathbf{W}_e\mathbf{e}_{k+1} + \mathbf{u}_{k+1}^T\mathbf{W}_u\mathbf{u}_{k+1}$$

where $\mathbf{e}_{k+1} = \mathbf{y}_d - \mathbf{y}_{k+1}$ is a vector. The idea is to determine $\mathbf{u}_{k+1}$ in such a way that the the error $\mathbf{e}_{k+1}$ becomes as small as possible with respect to the criterion. The weighting matrices $\mathbf{W}_e$ and $\mathbf{W}_u$ decide the trade off

between performance and input energy. The criterion is minimized subject to the constraint:

$$(\mathbf{u}_{k+1} - \mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k) \leq \delta \tag{2.8}$$

Introducing the Lagrange multiplier yields the criterion:

$$\overline{J}_{k+1} = \mathbf{e}_{k+1}^T \mathbf{W}_e \mathbf{e}_{k+1} + \mathbf{u}_{k+1}^T \mathbf{W}_u \mathbf{u}_{k+1} + \lambda((\mathbf{u}_{k+1} - \mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k) - \delta) \tag{2.9}$$

From (2.7) it follows that $\mathbf{e}_{k+1}$ is given by

$$\mathbf{e}_{k+1} = (I - \mathbf{T}_r)\mathbf{y}_d - \mathbf{T}_u \mathbf{u}_{k+1} \tag{2.10}$$

Using this result together with (2.9) makes it possible to do a differentiation of $\overline{J}_k$ with respect to $\mathbf{u}_{k+1}$. This gives:

$$\frac{\partial \overline{J}_{k+1}}{\partial \mathbf{u}_{k+1}} = -\mathbf{T}_u^T \mathbf{W}_e \mathbf{e}_{k+1} + \mathbf{W}_u \mathbf{u}_{k+1} + \lambda(\mathbf{u}_{k+1} - \mathbf{u}_k) = 0 \tag{2.11}$$

where the optimum is achieved when the derivative equals zero. In the last equation the aim is to extract $\mathbf{u}_{k+1}$, but $\mathbf{T}_u$ is unknown and $\mathbf{e}_{k+1}$ at iteration $k+1$ is not still available.
In order to overcome these problems we can use $\widehat{\mathbf{T}}_u$ which is a model of $\mathbf{T}_u$ and we can try to predict $\mathbf{e}_{k+1}$. A prediction $\widehat{\mathbf{e}}_{k+1}$ of $\mathbf{e}_{k+1}$ in Equation (2.10) is given by the next equation

$$\widehat{\mathbf{e}}_{k+1} = (I - \widehat{\mathbf{T}}_r)\mathbf{y}_d - \widehat{\mathbf{T}}_u \mathbf{u}_{k+1} \tag{2.12}$$

Notice that $\widehat{\mathbf{T}}_r$ and $\widehat{\mathbf{T}}_u$ are lower triangular Toeplitz matrix created from the impulse response of $\widehat{T}_r(q)$ and $\widehat{T}_u(q)$. $\widehat{T}_r(q)$ and $\widehat{T}_u(q)$ denote a nominal model of the closed loop system and the transfer function from the ILC input to the output, respectively. This implies that some *a priori* knowledge of the system to be controlled is available. So Equation (2.11) is transformed into :

$$-\widehat{\mathbf{T}}_u^T \mathbf{W}_e (I - \widehat{\mathbf{T}}_r)\mathbf{y}_d + \widehat{\mathbf{T}}_u^T \mathbf{W}_e \widehat{\mathbf{T}}_u \mathbf{u}_{k+1} + \mathbf{W}_u \mathbf{u}_{k+1} + \lambda(\mathbf{u}_{k+1} - \mathbf{u}_k) = 0 \tag{2.13}$$

which solved with respect to $\mathbf{u}_{k+1}$ gives:

$$\mathbf{u}_{k+1} = \mathbf{Q}(\mathbf{u}_k + \mathbf{L}\widehat{\mathbf{e}}_k) \tag{2.14}$$

$$\mathbf{Q} = (\mathbf{W}_u + \lambda \cdot I + \widehat{\mathbf{T}}_u^T \mathbf{W}_e \widehat{\mathbf{T}}_u)^{-1}(\lambda \cdot I + \widehat{\mathbf{T}}_u^T \mathbf{W}_e \widehat{\mathbf{T}}_u) \tag{2.15}$$

$$\mathbf{L} = (\lambda \cdot I + \widehat{\mathbf{T}}_u^T \mathbf{W}_e \widehat{\mathbf{T}}_u)^{-1} \widehat{\mathbf{T}}_u^T \mathbf{W}_e \tag{2.16}$$

The updating matrices $\mathbf{Q}$ and $\mathbf{L}$ hence depend on the nominal model $\widehat{\mathbf{T}}_u$ and the weighting matrices $\mathbf{W}_u$ and $\mathbf{W}_e$. Note however that the Lagrange multiplier λ is not computed explicitly. It is instead used as a design variable which puts a weight on $((\mathbf{u}_{k+1} - \mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k) - \delta)$ but does not necessarily fulfill the constraint (2.8).

In Equation (2.14) the error signal is formed using the nominal model of the system, while in real use the actual error signal from the system is used. The conventional definition of the error $\mathbf{e}_k = \mathbf{y}_d - \mathbf{y}_k$ leads to

$$\mathbf{u}_{k+1} = \mathbf{Q}(\mathbf{u}_k + \mathbf{L}\mathbf{e}_k) \tag{2.17}$$

where $\mathbf{Q}$ and $\mathbf{L}$ are given by (2.15) and (2.16). The next theorem is useful for the filters design strategy.

THEOREM 2.2—(NORRLÖF, 2000)
If $\lambda > 0$ and the nominal model corresponds to the true system then the proposed optimization based ILC algorithm always gives a stable ILC system. □

Moreover in the design of $\mathbf{Q}$ and $\mathbf{L}$, if we choose $\mathbf{W}_e = I$ and $\mathbf{W}_u = \rho \cdot I$, with $\rho > 0$, then we can deal also with non-minimum phase systems and assure that the stability criteria fulfilled. Therefore we have the following algorithm :

1. Build a model of the relations between the ILC input and the resulting correction on the output (i.e., find a model $\widehat{\mathbf{T}}_u$ of $\mathbf{T}_u$). The matrix $\widehat{\mathbf{T}}_u$ is simply the lower triangular Toeplitz matrix created from the impulse response of $\widehat{T}_u(q)$.
2. Choose the weight matrices as $\mathbf{W}_e = I$ and $\mathbf{W}_u = \rho \cdot I$ with $\rho > 0$, choose also $\lambda > 0$.
3. $\mathbf{Q}$ and $\mathbf{L}$ are calculated according to

$$\mathbf{Q} = ((\rho + \lambda) \cdot I + \widehat{\mathbf{T}}_u^T \widehat{\mathbf{T}}_u)^{-1} (\lambda \cdot I + \widehat{\mathbf{T}}_u^T \widehat{\mathbf{T}}_u) \tag{2.18}$$

$$\mathbf{L} = (\lambda \cdot I + \widehat{\mathbf{T}}_u^T \widehat{\mathbf{T}}_u)^{-1} \widehat{\mathbf{T}}_u^T \tag{2.19}$$

4. Use the ILC updating equation 2.17 with $\mathbf{u}_0 = 0$.

It has to be remarked that the computational complexity of this algorithm grows fast with data size.

2.5 Simulations using ILC algorithms

The aim of this section is to show how some algorithms based on Iterative Learning Control work when applied to a simple model of robot joint. All the results that will be shown come from Matlab simulations. Let us introduce a robot joint model expressed by the Laplace transfer function:

$$G_c(s) = \frac{1}{J \cdot s^2}$$

where J is the joint moment of inertia.
From $G_c(s)$ we obtain $G(z)$, that is the z-transform of $G_c(s)$ including the Zero Order Hold (ZOH) :

$$G(z) = \frac{T_s^2}{2J} \frac{z+1}{(z-1)^2}$$

T_s is the sampling time. The joint is controlled by a P.D. feedback controller $F(z)$ and by a feed-forward controller $F_f(z)$.

$$F(z) = k_p + \frac{k_d}{T_s} \frac{z-1}{z} = \frac{k_p T_s + k_d}{T_s} \cdot \frac{z - \dfrac{k_d}{k_p T_s + k_d}}{z}$$

hence

$$F(z) = F_{gain} \cdot \frac{z - F_{zero}}{z}$$

and

$$F_f(z) = \frac{J}{T_s^2} \cdot \frac{(z-1)^2}{z^2} = F_{fgain} \cdot \frac{(z-1)^2}{z^2}$$

$J = 0.0094$ N·s^2 , $k_p = 12.7$, $k_d = 0.4$, $T_s = 0.001$ s.
Figure 2.2 shows the scheme we are considering. $y_d(t)$ is the reference signal we want to track as well as possible and $y_k(t)$, $u_k(t)$, $e_k(t) = y_d(t) - y_k(t)$ are respectively the output, the ILC input and the tracking error of the system at the iteration k. In equations:

$$y_k(t) = T_r(q)y_d(t) + T_u(q)u_k(t) \tag{2.20}$$
$$e_k(t) = y_d(t) - y_k(t) \tag{2.21}$$
$$u_k(t) = Q(q)[u_{k-1}(t) + L(q)e_{k-1}(t)] \tag{2.22}$$

T_u is the transfer operator from $u_k(t)$ to $y_k(t)$ and T_r from $y_d(t)$ to $y_k(t)$.

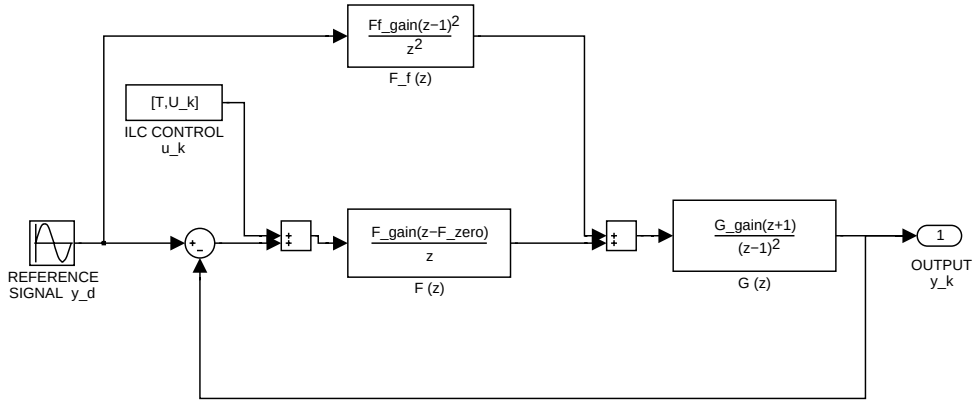

Figure 2.2 ILC applied to the controlled system.

According to the scheme considered we have:

$$T_u(q) = \frac{F(q)G(q)}{1 + F(q)G(q)} \tag{2.23}$$

$$T_r(q) = \frac{(F(q) + F_f(q))G(q)}{1 + F(q)G(q)} \tag{2.24}$$

The controller operating on the joint is already a good controller. See its tracking performance in the simulations results shown in Figure 2.3. The reference choice has been $y_d(t) = \sin(2\pi t)$.

We wish to improve the performance of the controlled system by adding to the system a suitable external control signal, that is ILC.

T_u in our example results:

$$T_u(q) = \frac{0.02195q^2 + 0.00067q - 0.02128}{q^3 - 1.978q^2 + 1.001q - 0.02128} \tag{2.25}$$

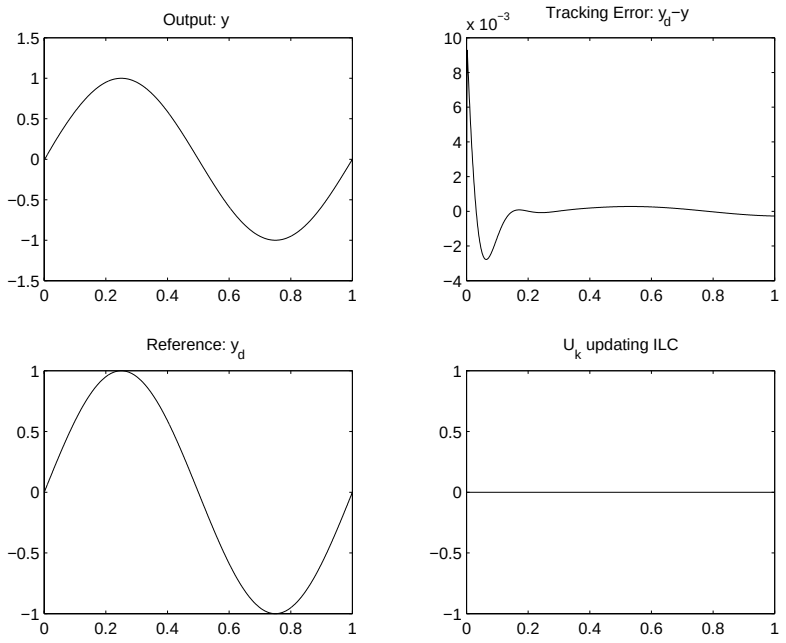

Figure 2.3 Closed loop controlled system behavior without ILC

or equivalently in the zero-pole form:

$$T_u(q) = 0.022\frac{(q+1)(q-0.9692)}{(q-0.9779-0.02299i)(q-0.9779+0.02299i)(q-0.0222)} \tag{2.26}$$

The next subsections will show simulation results using different ILC algorithms. The number of ILC iterations performed is every time 10.

Simulations using Heuristic ILC

In the implementation of this algorithm $Q(q)$ is chosen zero phase low-pass filter and $L(q) = \kappa$. Simulation results after 10 ILC iterations are shown in Figure 2.4 There is a some tracking improvement in comparison with the case where no ILC is applied.

A simulation with different L filter choice is performed, $L(q) = \kappa q$. Observe that the L filter choice is not causal. We are allowed to use anti-causal filters in the ILC design because they operate on already known sequences from the previous iteration. Figure 2.5 displays simulation results after 10 iterations.

The last choice for the filter L results in a better tracking than the previous one.

Simulations using Model Based ILC

Here it is considered an algorithm that uses the knowledge of a system model to design the ILC filters. In simulations the following model $\widehat{T}_u$ of T_u expressed in Equation (2.26) is considered:

$$\widehat{T}_u(q) = 0.022\frac{q(q-0.9692)}{(q-0.9779-0.02299i)(q-0.9779+0.02299i)(q-0.0222)} \tag{2.27}$$

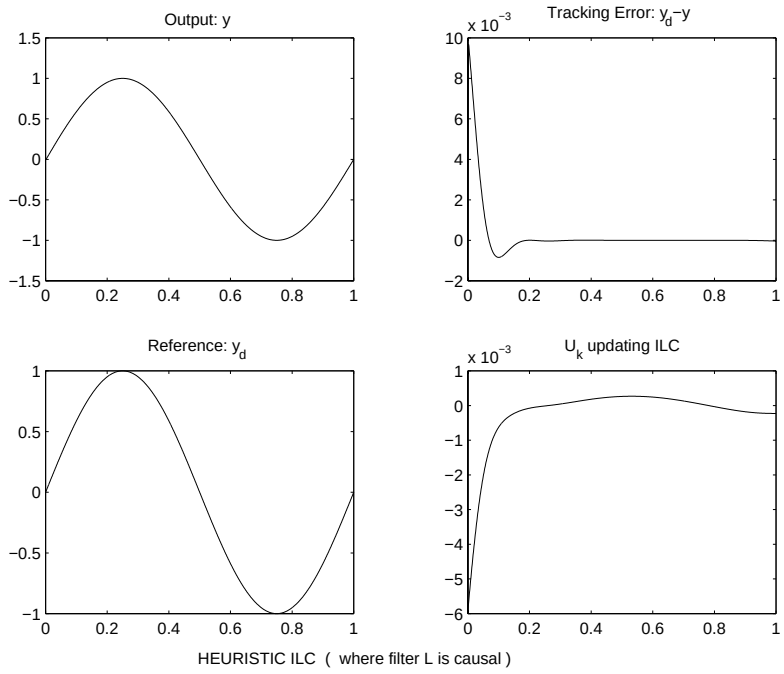

Figure 2.4 Simulation results after 10 iterations of Heuristic ILC with the filter $L(q) = \kappa$.

The filter Q is chosen zero phase low-pass and the aggressive choice $L(q) = \widehat{T}_u^{-1}$ is done. Such L is obtained by imposing the filter $H_b(q) = 0$

Figure 2.6 shows the performance of this algorithm when applied to our joint.

Simulations using Optimization Based ILC

In simulations this technique results to have the best performance in comparison with the previous algorithms shown. The algorithm has been applied in the matrix form consequently the ILC filters are expressed through the matrices $\mathbf{Q}$ and $\mathbf{L}$

Figure 2.7 displays the performance of this algorithm when applied to our simulation problem using the following model choice $\widehat{T}_u(q) = T_u(q)$ from (2.26).

Figure 2.8 shows the performance of the algorithm when $\widehat{T}_u(q)$ is chosen equal to (2.27)

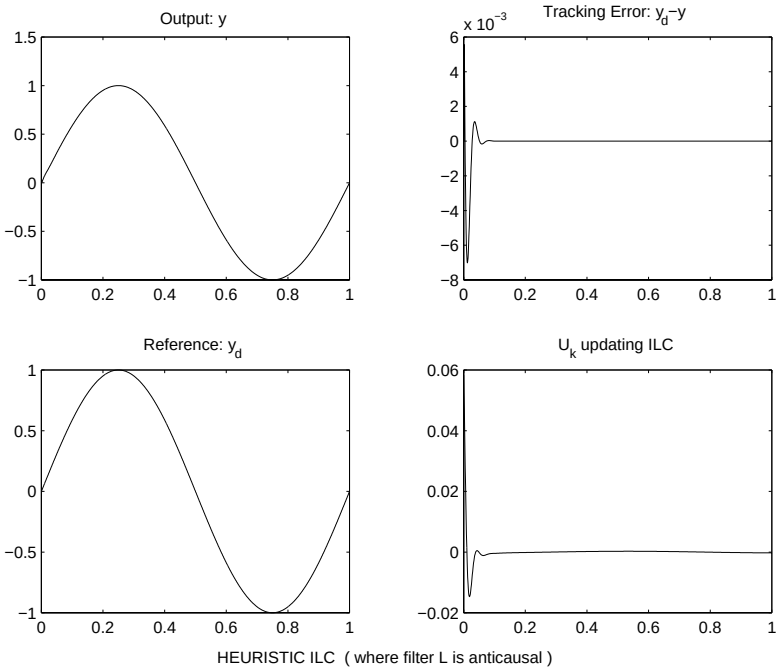

Figure 2.5 Simulation results after 10 iterations of Heuristic ILC with the filter $L(q) = \kappa q$.

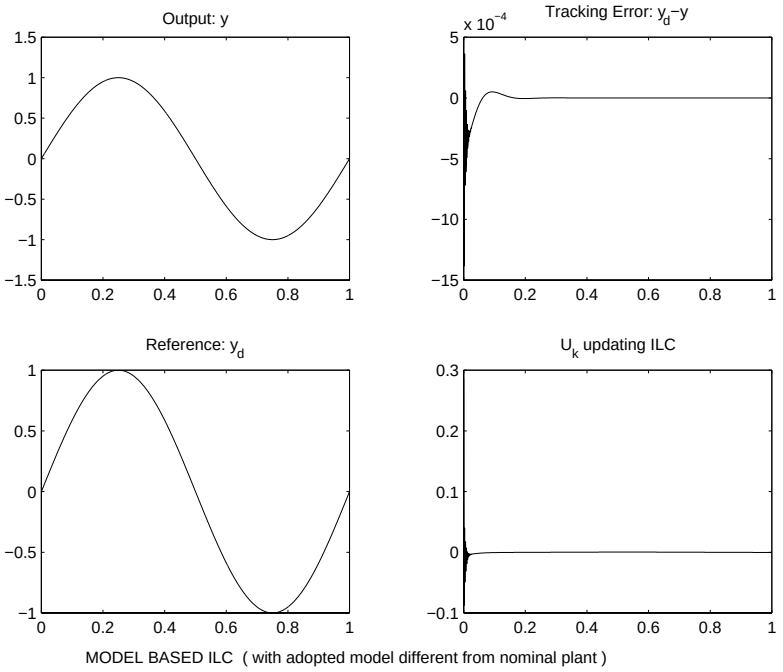

Figure 2.6 Simulation results after 10 iterations of Model Based ILC.

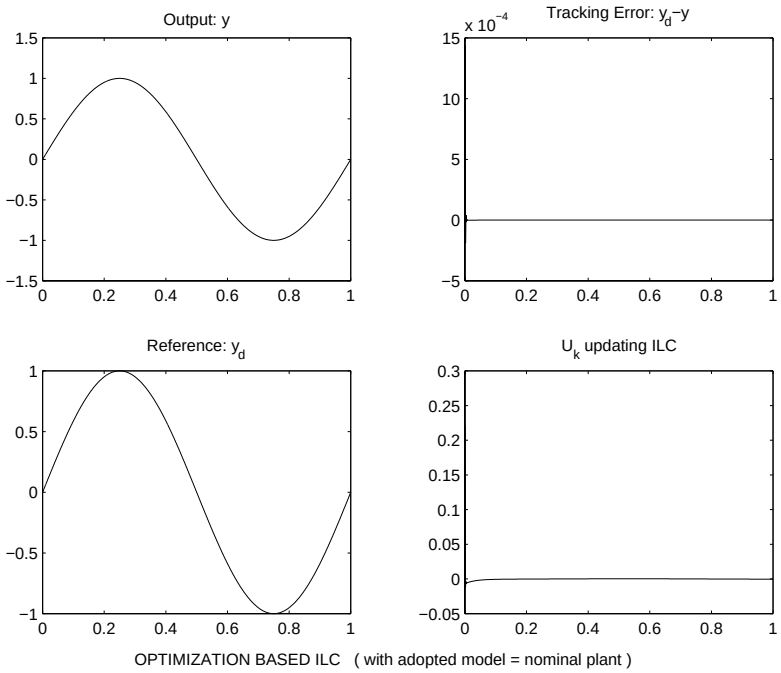

Figure 2.7 Simulation results after 10 iterations of optimization Based ILC, with $\widehat{T}_u(q) = T_u(q)$

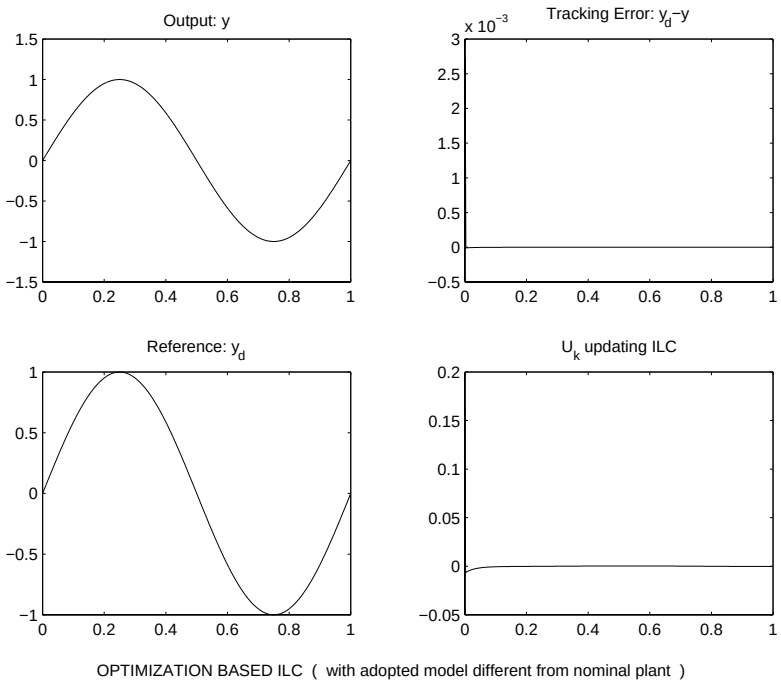

Figure 2.8 Simulation results after 10 iterations of optimization Based ILC, with $\widehat{T}_u(q) \neq T_u(q)$

3. A Brief Description of the Robot System

3.1 The ABB industrial robot Irb-2000

The robot used in the experiments is an Irb-2000, ABB industrial robot. The robot has seven links wich are connected by six joints, as shown in Figure 3.1. It is built up by two big arms and a wrist. Joint 2 (axis B in the figure) is used to move the lower arm back and forth, whereas joint 3 (A) moves the upper arm up and down. Joint 4 (D) is used to turn the wrist unit and joint 5 (E) bends the wrist unit around its center. The sixth joint (F) is used to turn the robot end effector, which is mounted on the tip of the wrist. The end effector is not shown in the figure. Finally joint one (C) turns the entire robot around its base.

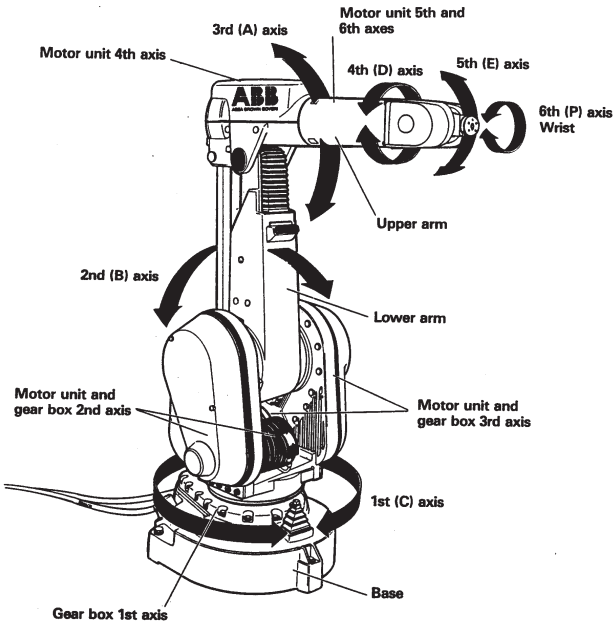

Figure 3.1 The ABB-2000 industrial robot.

The robot system has different built-in controllers, one for the control of each joint angle. These controllers are cascaded PID controllers. The block diagram for a single joint is shown in Figure 3.2.

3.2 The experimental platform

The experimental platform consists of:

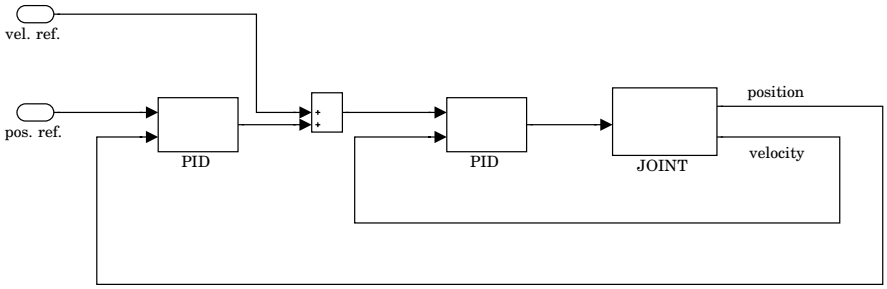

Figure 3.2 Block diagram for a controlled joint. The velocity signal used in the velocity loop is the position signal differentiated and low-pass filtered.

- Reconfigured Irb-2000 robot system (robot and control cabinet).
- VME based board computer system (target system).
- Host computer system consisting of Sun workstations (host system).
- Ethernet connection between host and target.

The Irb-2000 is controlled from VME-based embedded computers [Nilsson, 1996]. Sun workstations are used for software development and control engineering, as well as for robot operator interaction.

Figure 3.3 shows the Irb-2000 part of the laboratory. Signals from in-

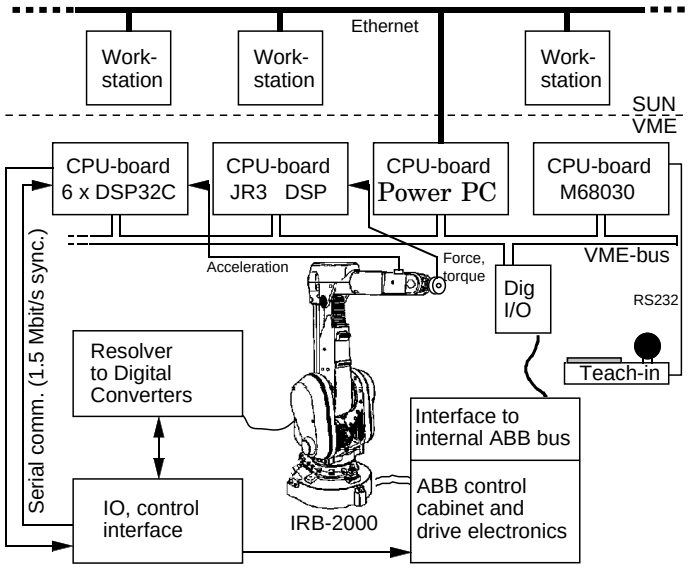

Figure 3.3 The experimental Irb-2000 system.

ternal sensors of the robot to the VME system go via the sensor interface to the DSP board connected to the VME bus.

The master computer in the VME computer is based on Power PC processor. Supervision and safety functions are implemented on a M68030 board, well separated from the rest of the system to prevent damage of the robot. Digital Signal Processors (DSP) are used for low-level control and filtering of sensors signals. Sensors requiring very high data bandwidths

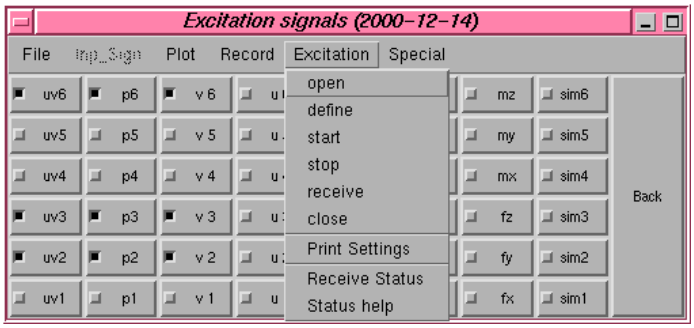

Figure 3.4 The *Exc_handler* Matlab window which makes it possible to assign additional reference trajectories to certain robot links and to record the outputs from the robot sensors.

are connected directly to the DSP boards. An additional DSP board belongs to the force-torque sensor.

3.3 The Matlab-robot connection

By the Sun workstations described in the previous section it is possible the development of programs to control the robot or to send it references to be tracked. That can be done inside the Matlab environment.

A Matlab program called *Exc_handler* is available for simple excitation experiments on the robot. This program can be used to define velocity and position references to the built-in controllers or to directly define torque references to the motors. The inputs can be steps, ramps, sinusoids, noise or other arbitrary signals from the Matlab workspace (see Figure 3.4).

A lot of signals can be recorded during the excitation. These include input torques, position measurements, differentiated position (velocity), and force and torque measurement from the force sensor. The recorded signals can then be exported to the Matlab workspace for plotting and data processing.

4. Experimental Results, ILC Applied to Robot Learning,

4.1 Objectives and assumptions

The robot

Two of the three algorithms discussed in the previous section have been applied to an industrial robot manufactured by ABB Robotics. The robot is an Irb2000 with an Open Control architecture developed at the Dept. of Automatic Control, Lund, Sweden, [Nilsson, 1996]. It has 6 joints and 2 joints (joints 2 and 3) have been used in order to draw bidimensional trajectories.

Objective

ILC algorithms have been applied to the joints involved in the experiments to improve the tracking performance of the robot controllers.
Each joint of the robot is controlled by cascaded PID controllers. We observe that if the frequency range of the references signals sent to the joints covers only low frequencies then the joints controllers work good and there is no need of adding external ILC. If this range covers also higher frequencies then we can reach a situation where the behavior of the joint controllers is not enough. Therefore we need to use ILC in order to improve the robot tracking performance.

Introduction of a model for the controlled joint

Let us assume that the closed loop system from the angular position reference to the angular position response can be approximately described using a low order linear continuous time model. That is for the generic joint i:

$$\widehat{T}_{c,i}(s) = \frac{1}{a_i s + 1} \tag{4.1}$$

See also Figure 4.1.

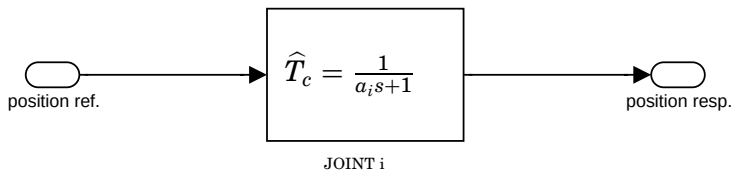

Figure 4.1 Simple model adopted for the controlled joint.

The parameter a_i in Equation (4.1) is identified from the simple step response of joint i.

We will use the joint model just introduced for the implementation of the Model Based ILC algorithm on joints 2 and 3.

4.2 The Spiral experiment

The first experiment done was to draw a spiral from the peripheral starting position to the center and once reached the center to draw back to the starting position following the path already drawn.
The spiral is drawn with constant angular velocity by the robot. The duration time of the operation is 20 seconds.

For both joints the Q-filters were chosen as zero-phase filters $Q_i(q) = \bar{Q}_i(q)\bar{Q}_i(\frac{1}{q})$, where $\bar{Q}_i(q)$ is a second order Butterworth low-pass filter. For each joint involved in the experiment the L-filter was chosen as the inverse of the discretized version of the joint continuous model (4.1), that is $L_i(q) = T_{d,i}(q)^{-1}$. The integer $i : 2,3$ is the index that selects the joint we are considering.

Results from the experiment

The first ILC algorithm applied in the spiral experiment was the Model Based one and the joints involved were joint 2 and joint 3.

Figure 4.2 shows how ILC improves iteration by iteration the tracking of the ideal trajectory (dotted line). In iteration 0 there is no ILC control applied and the figure points out how the joints controllers behave. In the first row of the Figure 4.2 (from left to right) we can see how ILC improves the performance when the spiral is drawn by the robot moving in the anti-clockwise direction. In the second row of Figure 4.2 it is shown the ILC improvement when the second part of the trajectory is drawn, that is when the robot draws from the center to the most peripheral point in the clockwise direction.

Figure 4.3 shows the reference signals for joint 2 and joint 3 (J2 and J3) and the tracking error of the two joints through the iterations.

In all these plots, on the vertical axes there are the motor radians and on the horizontal axes the time in seconds. All the joints signals are expressed in motor radians (i.e. radians on the motor side) because the reference and the ILC control signal have to be applied to the joint motor. Therefore the joints angles expressed in radians are converted through a gear ratio to motor radians.

We observe from Figure 4.3 that in the second ILC iteration the tracking error is really small almost everywhere except for the initial instants on joint2 where it is nearly 0.25 motor-radians. This peak is however acceptable and it decreases if further iterations are executed.

Also the heuristic ILC algorithm was applied to the robot system in the spiral experiment. $Q_i(q)$ was chosen as a zero phase low pass filter and $L_i(q) = k_i \cdot q$. In Figures 4.4 and 4.5 we can see the results of two iterations of ILC.

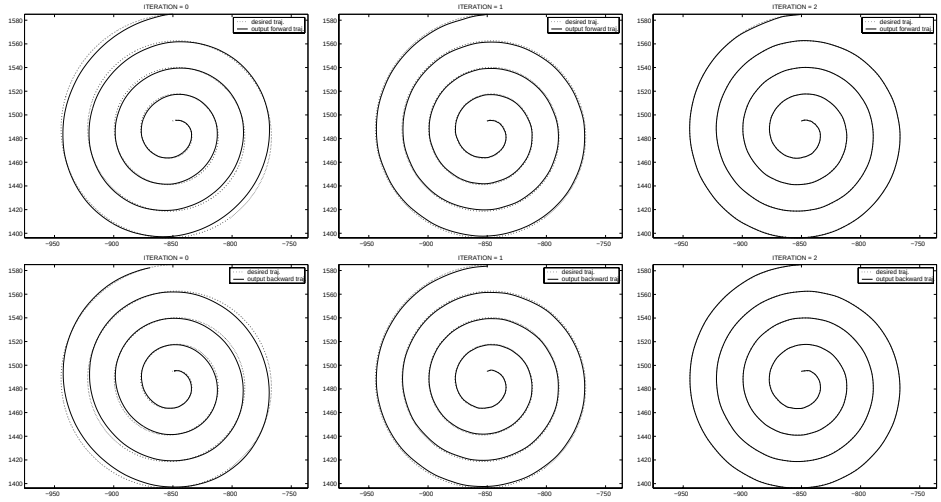

Figure 4.2 Forward and Backwards Spirals in the Model Based experiment. The measurements for the spirals figures are in [mm].

Discussion

Satisfactory results can be seen after only 2 iterations of ILC. If we compare the results shown in the figures regarding the spiral experiments performed using two different ILC algorithms, we realize that the tracking performance after two iterations of the Model Based algorithm is better than the Heuristic one. That is coherent with what we can expect from the ILC theory, and from the simulations in the previous Section 2.5.

4.3 The Christmas Tree experiment

The spiral is a smooth figure and consequently the input references to the joints are smooth. In order to test ILC in different conditions a new experiment using another bidimensional trajectory was performed. This trajectory is the sketch of a Christmas Tree, a figure composed by lines and angles. The Tree with its abrupt changes of trajectory during the drawing represents a good way to test the ILC algorithms. Figure 4.6 is a picture of the robot drawing the Tree. The algorithm used is The Model Based one. ILC was applied to joint 2 and joint 3 with the following choice of filters: Q filters were chosen as a zero-phase filters $Q_i(q) = \bar{Q}_i(q)\bar{Q}_i(\frac{1}{q})$ with $\bar{Q}_i(q)$ as a second order Butterworth low-pass filter. The L filters were chosen as $L_i(q) = T_{d,i}(q)^{-1}$.

Results from the experiment

Figure 4.7 shows some results from the Tree experiment. In iteration zero, without ILC, the angles of the Tree are drawn in a smooth way by the robot. The tracking error on the involved joints is consequently rather big near the angles of the figure (because the desired Tree has sharp angles). ILC operates in order to reduce this error on the Tree angles and it is finally possible for the robot to draw a figure with sharper angles. The

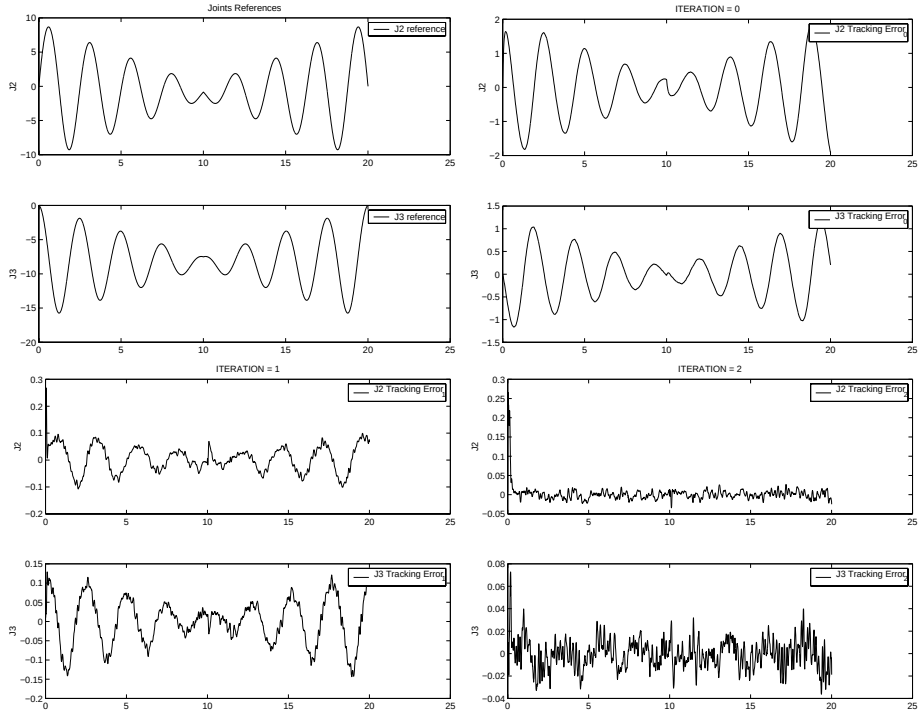

Figure 4.3 References and Position Tracking Errors for joint 2 and joint 3 in the spiral Model Based experiment. The measurements for the References and for the Position Tracking Errors are in [motor rad.] versus [s].

figure points out how the tracking error on joint 2 and joint 3 decreases after two iteration of the learning algorithm.

4.4 ILC applied to the joint velocity input

Up to now we have considered only the position inputs of the joints, following the scheme in Figure 4.8 .

More generally each joint accepts as inputs one position input and one velocity input (see Figure 4.9).

The joint controllers are implemented as cascaded controllers with an inner velocity loop and an outer position control loop. The bandwidth of the velocity controller is higher than the bandwidth of the position controller. It means the joint controller tracks velocity references better and faster than position references, especially for signals containing higher frequency components. Therefore it could be interesting to implement ILC on the joint velocity input according to the scheme shown in Figure. 4.10

Results from the experiment

The velocity ILC algorithm implemented was the heuristic one and it was applied to joint 2 in this experiment. In Figure 4.11 is shown the reference to joint 2. In Figure 4.12 we see how the velocity tracking error is reduced from iteration 0 to iteration 5. In the same figure we compare the velocity

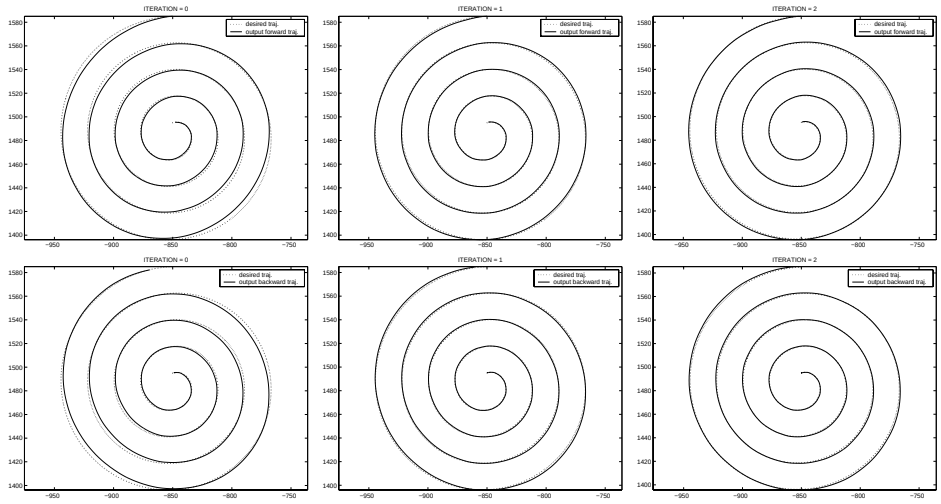

Figure 4.4 Forward and Backwards Spirals in the Heuristic experiment. The measurements for the spirals figures are in [mm].

tracking error in iteration 0 and in iteration 5 and we realize that its variation range has been reduced from ± 3.5 to ± 0.8 motor-rad./s.

Discussion

By analyzing Figure 4.12 we see that through the iterations the tracking error is reduced with respect to iteration zero but higher frequency oscillations appear. In the scheme of Figure 4.10 we can see that the use of velocity ILC updating makes the joint velocity input is not the exact derivative of the position input. It was assumed for the experiment that, since the bandwidth of the velocity controller is larger than that of the position controller, the joint position controller behavior is very slightly affected by velocity ILC updating. The oscillations present through iterations suggest that this assumption does not work completely.

In my opinion this is not the main cause of this poor behavior. The velocity reference sent to joint 2 is fast. The joint increases from zero considerably its velocity and then it has to reduce the velocity in a very short time. The ILC updating applied to velocity can be too abrupt. For example when the joint velocity must decrease very fast to zero it means the relative mechanical link must decrease fast to zero its velocity, it has heavy mass and is not easy to stop a mass moving fast in a short time interval without drawbacks. In this time interval the joint velocity tracking error is not small and consequently velocity ILC tries to compensate. ILC reduces the tracking error range but it is not avoid the presence of oscillations (vibrations). Note that Heuristic ILC was implemented and it does not use too much knowledge about the system.

Probably Model Based and Optimization based ILC can give better performance but they need the introduction of a model for the joint involved. A model describing the joint dynamics when fast movements are performed (taking also into account the joint flexibility problem and the nonlinear coupling terms between the links) would be to prefer but it is not straightforward to do. In conclusion the tracking error is reduced but there are still

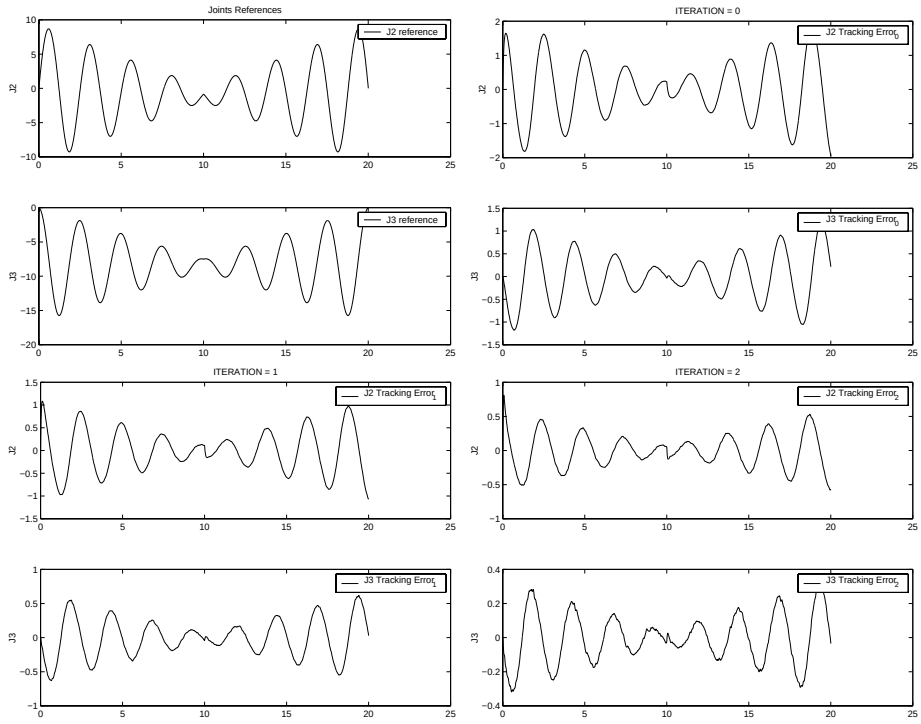

Figure 4.5 References and Position Tracking Errors for joint 2 and joint 3 in the spiral Heuristic experiment. The measurements for the References and for the Position Tracking Errors are in [motor rad.] versus [s].

some vibrations during the motion. Velocity ILC is appealing but there are some unsolved problems that affect the performance.

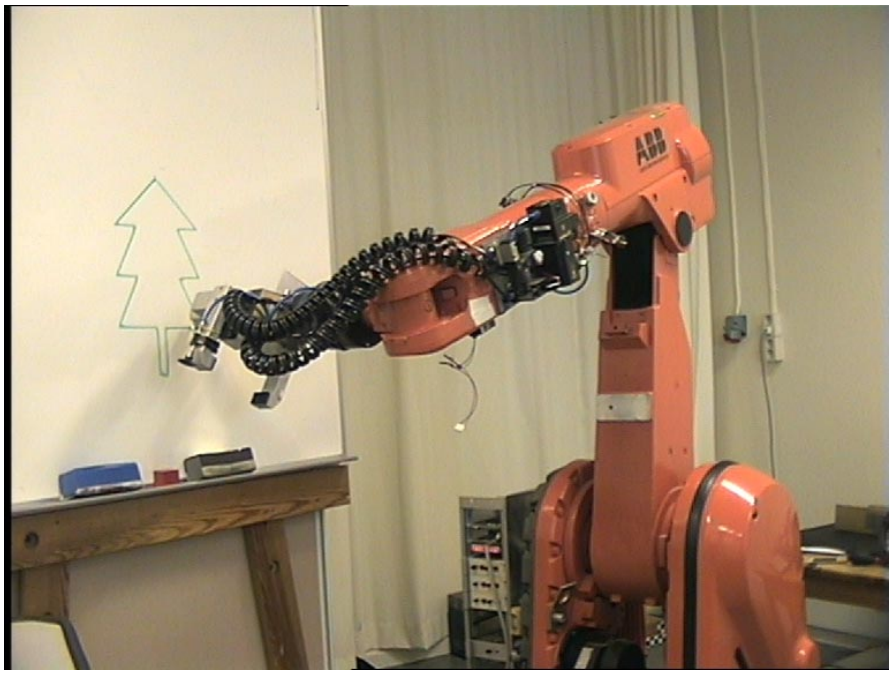

Figure 4.6 The robot draws the Tree using ILC to improve the tracking of the reference.

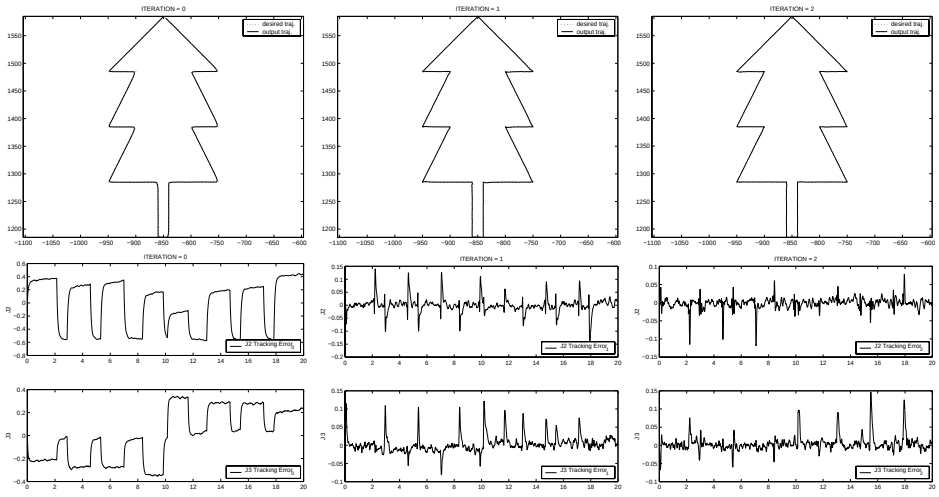

Figure 4.7 Tree plotting and Position Tracking Errors in the tree Model Based experiment. The measurements for the tree figures (first row) are in [mm] and for the Position Tracking Errors (second row) are in [motor rad.] versus [s].

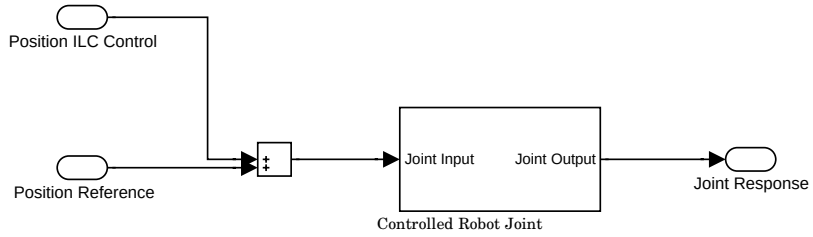

Figure 4.8 position ILC applied to the robot joint.

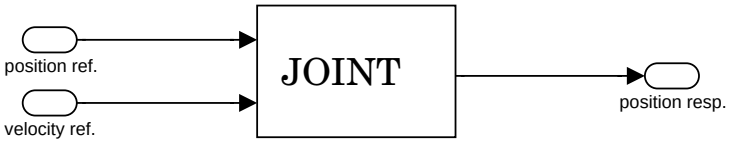

Figure 4.9 General scheme of one robot joint.

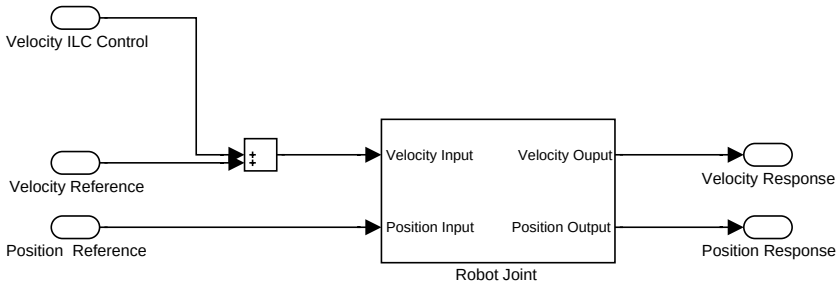

Figure 4.10 Velocity ILC applied to the robot joint with cascaded position and velocity controller.

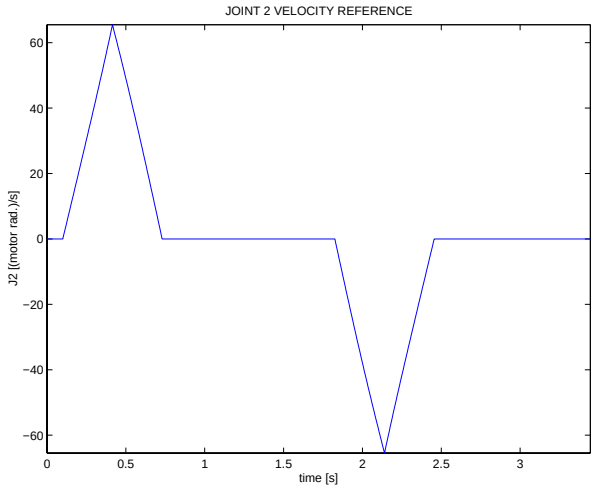

Figure 4.11 Velocity Reference for joint 2.

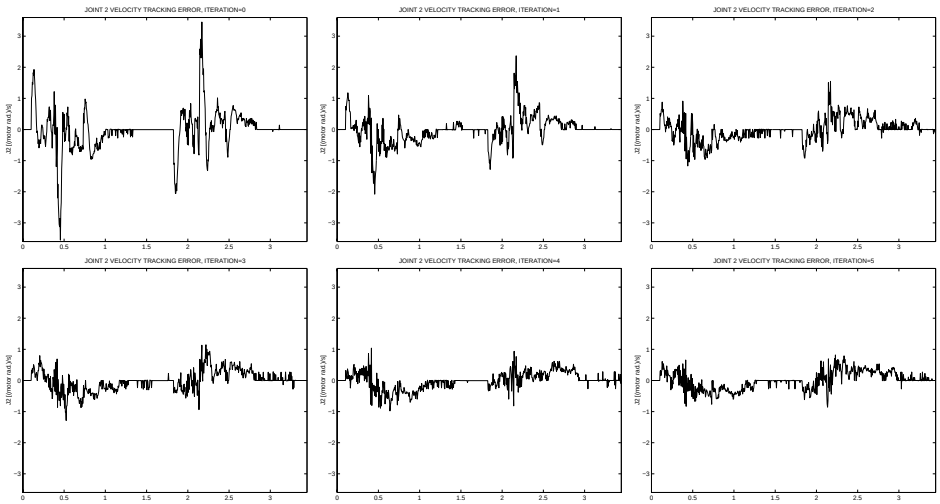

Figure 4.12 The Velocity Tracking Error on joint 2 is reduced trough the ILC iterations but higher frequency oscillation appear. The measurements for the Velocity Tracking Errors are in [(motor rad.)/s] versus [s].

5. Case study: Movement of Open Containers Using a Robot

5.1 Abstract

In this chapter ILC is applied to a robot in order to improve the performance regarding the motion of a container with liquid inside. The purpose is to shorten the motion time of the container and to keep under control the slosh inside. The algorithm for the slosh control is iterative [Grundelius, 1998], [Grundelius and Bernhardsson, 2000], [Grundelius, 2000], [Grundelius and Bernhardsson, 1999b], [Grundelius and Bernhardsson, 1999a]. ILC is also used in an 'inner iteration loop' to improve the robot tracking of the desire acceleration profile. First an introduction and a problem formulation is given. The algorithm for reducing the slosh is given in the next subsection, followed by some simulations and finally the experiments on the robot system are described and presented.

5.2 Problem introduction

The problem of motion of open containers with liquid will be investigated, [Grundelius and Bernhardsson, 1999b]. This is a common problem in the packaging industry. A packaging machine fills the container with liquid and proceeds to seal it. Once filled the package is moved by a holder to a position where it will be sealed. The movement of the package is executed stepwise, the number of steps depends on the machine type. The same movement is applied in every step on all packages. In Figure 5.1 is shown a scheme of packaging machine. The aim is to shorten the motion time of

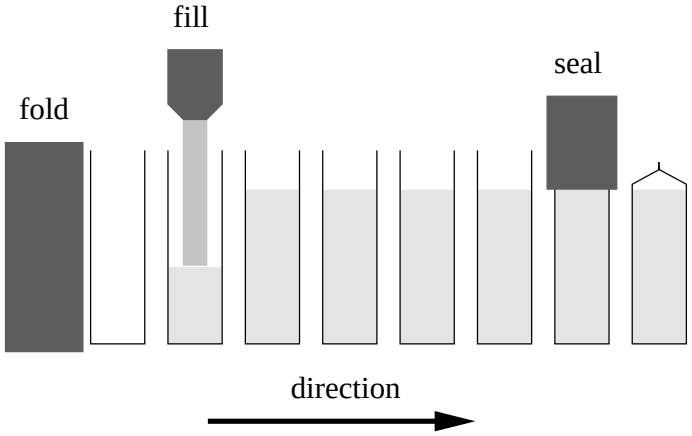

Figure 5.1 Scheme of the packaging machine

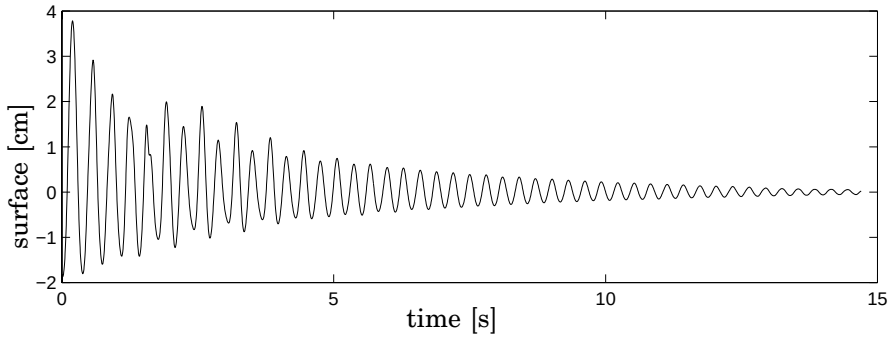

Figure 5.2 The surface elevation displayed is the response to an acceleration impulse applied to the container.

the package because in this way it is possible to increase the production. Increased production rate gives a lower packaging cost and higher profit.

The problem is that we cannot move the container as fast as we want because we must take into account that there is liquid inside and the package movement causes motion in the liquid. We will refer to it as *slosh*. The amount of slosh depends on how the package is accelerated and on the properties of the liquid used. There is the risk that when the container is moved fast the slosh causes some liquid to go outside the container or to wet the carton surfaces that should be sealed. This can result in packages that are not properly sealed and possibly not airtight. If the package is not airtight the storage time is much decreased. Therefore in the problem of minimizing the package motion time we must take into account constraints due to slosh in the liquid (the maximum slosh elevation must be below a certain value).

We want to find the best acceleration profile in order to move the package one step, fulfilling the slosh constraints.

The same acceleration profile is applied at each step. The acceleration must be such that the slosh constraint is not violated when the acceleration profile is repeated. One way to achieve this is to ensure that the slosh is in the same state at the beginning of each movement step. The natural choice of initial state of the slosh is that with the liquid at rest.

The problem is solved by first deriving a model of the slosh and then applying optimal control techniques to calculate the acceleration profile for the container. The choice of slosh model is not trivial. The nonlinear effects are evident for very rapid movements when the surface elevation is large. Figure 5.2 shows the results of an impulse response experiment. In the figure two nonlinear phenomena can be observed: the oscillation is asymmetric and the oscillation frequency is slightly amplitude dependent (the oscillation frequency increases with decreasing amplitude).

A linear slosh model with four states is introduced.

$$\dot{x} = Ax + Bu \tag{5.1}$$

with

$$A = \begin{bmatrix} -2\zeta\omega & -\omega & 0 & 0 \\ \omega & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} a\omega/2g \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

where x_2 is the surface elevation in meters, x_1 is the time derivative of the surface elevation divided by ω, x_3 is the container velocity and x_4 the container position. The following constraints must be considered:

1. Acceleration: $|u(t)| \leq u_{max} = 9.81\ m/s^2$
2. Slosh: $|x_2(t)| \leq s_{max} = 0.035$ m
3. Initial state: $x(0) = [\,0 \quad 0 \quad 0 \quad 0\,]^T$
4. Terminal state: $x(T) = [\,0 \quad 0 \quad 0 \quad d\,]^T$

T is the movement time and $d = 0.2$ m the movement distance. s_{max} is the maximum surface elevation. For a rectangular container with liquid depth h and width a the oscillation frequency of the first harmonic is given by the following expression:

$$\omega_m = \sqrt{\frac{g\pi}{a} \tanh\left(\frac{b\pi}{a}\right)} \tag{5.2}$$

which gives approximately the value $\omega = 21.0$ rad/s. The residual slosh $\rho(t)$ is defined as

$$\rho(t) = s(t + T)\ \forall t \geq 0$$

where $s(t)$ is the surface elevation.

The Minimum Energy Approach

The objective is to calculate the 'best' acceleration profile for the package and one way to do it is to minimize the loss function (5.3) taking into account the constraints previously described.

$$J = \int_0^T u^2(t)\, dt \tag{5.3}$$

In the Minimum Energy Approach the aim is to minimize the total energy put into the system.

We can summarize that if a slosh model is available, the acceleration reference can be calculated using the optimal control technique just shown. However, in the experiments this method requires a very accurate model to be successful. The linear slosh model works enough well when the allowed maximum slosh is small. Nevertheless in the case we consider the allowed maximum slosh is relatively large and the linear model does not fully describe the real slosh behavior. Therefore a different approach is needed in order to calculate the ideal acceleration profile. This approach is the Iterative Learning Control.

5.3 Slosh Iterative Learning Control

The Iterative Learning Control algorithm considered starts using as initial acceleration that derived from the solution of the Minimum Energy Problem presented above and after, at each iteration the acceleration profile is updated taking into account the measured slosh behavior with respect to a slosh given reference [Grundelius and Bernhardsson, 2000]. The usual way to implement ILC is to use the following updating formula for the input signal $u_k(t)$

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{5.4}$$

where $Q(q)$, $L(q)$ are linear filters, not necessarily causal, and $e_k(t) = r(t) - y_k(t)$ is the tracking error. In this application of ILC the input signal $u_k(t)$ is the acceleration reference. The specification on the movement gives the following constraints on $u(t)$

$$\int_0^T u(t)\,dt = 0 \qquad \int_0^T \int_0^t u(s)\,dsdt = d \tag{5.5}$$

Where T is the movement time. We start with one value of T but we will try to have it as small as possible according to the ILC slosh problem performance. These constraints make it hard to choose the filters $Q(q)$ and $L(q)$.

From the linear slosh model and the solution of the Minimum Energy Problem a reference $r(t)$ for the surface elevation on the backward side of the package and an initial acceleration reference $u_0(t)$ are respectively obtained. The surface elevation is measured on the backward and on the forward side of the package giving the measurements $y^1(t)$ and $y^2(t)$. The linear model used when calculating $r(t)$ gives a symmetric surface elevation, hence $-r(t)$ is the surface elevation on the forward side of the package.

The acceleration reference and the surface elevation reference is sampled with sampling period h such that $T = nh$. The surface elevation is augmented with zeros giving $m > n$. This is to make it possible to penalize the residual slosh after the movement. The problem is to find $\delta u_k(t)$ in $u_{k+1}(t) = u_k(t) + \delta u_k(t)$ that minimizes the error in the next iteration $e_{k+1}(t) = r(t) - y_{k+1}(t)$. By supposing that the slosh is approximately described by the linear discrete time operator $G(q)$ this gives

$$\widehat{y}_k^1 = G(q)u_k(t) \qquad \widehat{y}_k^2 = -G(q)u_k(t)$$

Then the error in iteration $k+1$ is approximately given by

$$\begin{aligned} e_{k+1}^1(t) &\approx e_k^1(t) - G(q)\delta u_k(t) \\ e_{k+1}^2(t) &\approx e_k^2(t) + G(q)\delta u_k(t) \end{aligned}$$

definition of the vectors

$$\delta U_k = [\,\delta u_k(0) \quad \delta u_k(h) \quad \dots \quad \delta u_k((n-1)h)\,]^T$$

$$E_k^1 = [\,e_k^1(0) \quad e_k^1(h) \quad \dots \quad e_k^1((m-1)h)\,]^T$$
$$E_k^2 = [\,e_k^2(0) \quad e_k^2(h) \quad \dots \quad e_k^2((m-1)h)\,]^T$$

and the matrix

$$G = \begin{bmatrix} g(0) & 0 & \dots & 0 \\ g(h) & g(0) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ g((n-1)h) & g((n-2)h) & \dots & g(0) \\ \dots & \dots & \dots & \dots \\ g((m-1)h) & g((m-2)h) & \dots & g((m-n)h) \end{bmatrix} \tag{5.6}$$

where $g(t)$ is the impulse response of $G(q)$. We have the following equations

$$U_{k+1} = U_k + \delta U_k \tag{5.7}$$
$$E^1_{k+1} \approx E^1_k - G\delta U_k \tag{5.8}$$
$$E^2_{k+1} \approx E^2_k + G\delta U_k \tag{5.9}$$

Zero-order-hold sampling and normalization of the constraints (5.5) gives

$$A \cdot \delta U_k = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

with

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 \\ n-\frac{1}{2} & n-\frac{3}{2} & \dots & \frac{1}{2} \end{bmatrix}$$

The update of the acceleration reference δU_k is now given as the solution to the following quadratic optimization problem

$$\min_{\delta U_k} \{(E^1_{k+1})^T W_1 E^1_{k+1} + (E^2_{k+1})^T W_2 E^2_{k+1} + (U_{k+1})^T W_u U_{k+1}\} \tag{5.10}$$

subject to
$A\delta U_k = 0$
All δU_k that satisfy the constraints are given by $K\theta$ where K is the Kernel of A and θ is an arbitrary vector. Insertion in (5.10) of (5.7), (5.8), (5.9), $\delta U_k = K\theta$ and differentiation with respect to θ gives

$$\begin{aligned} \delta U_k = K[K^T(G^T(W_1+W_2)G + W_u)K]^{-1} \times \\ K^T(G^T(W_1E^1_k - W_2E^2_k) - W_uU_k) \end{aligned} \tag{5.11}$$

The update law for the ILC is given by (5.7) and (5.11) and can be written as

$$U_{k+1} = QU_k + L_1E^1_k + L_2E^2_k \tag{5.12}$$

5.4 Simulations

The update law in (5.12) is evaluated using simulations. The model used in the update law is given in continuous time by the transfer operator

$$G_c(p) = \frac{a}{2g}\frac{\omega_m^2}{p^2 + 2\zeta_m\omega_m p + \omega_m^2}$$

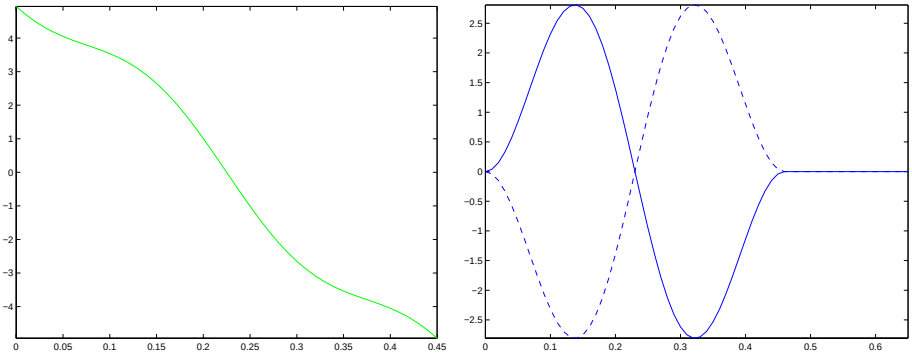

Figure 5.3 Initial acceleration reference $u_0(t)$ (left fig.) and surface elevation reference (right fig.) $r^1(t)$ and $r^2(t)$ (dashed)

with $\zeta_m = 0$ and $\omega_m = \sqrt{\frac{g\pi}{a}\tanh\frac{b\pi}{a}} = 21$ rad/s where $a = 0.07$ m is the package width and $b = 0.2$ m is the liquid depth. The system sampling time is $h = 0.01$ s which gives the discrete time transfer operator $G(q)$ and the matrix G. The surface elevation reference and initial acceleration reference are calculated using this model and minimum energy approach shown previously. The surface elevation reference is augmented with 20 zeros, Figure 5.3 shows the initial acceleration reference and the surface elevation references.

The movement time is $T = 0.46$ s and the movement distance is $d = 0.2$ m. In our simulation we use the following continuous time linear model to describe the slosh process:

$$P(p) = \frac{1.1a}{2g}\frac{\omega_p^2}{p^2 + 2\zeta\omega_p p + \omega_p^2}$$

with $\omega_p = 0.9\omega_m$ and $\zeta = 0.01$. The measurements are given by

$$\begin{aligned} y_k^1(t) &= P(p)u_k(t) \\ y_k^2(t) &= -P(p)u_k(t) \end{aligned}$$

The first choice of the weights is $W_1 = W_2 = I$ and $Wu = \varphi I$ with $\varphi = 0.00001$. This value of φ represents a good tradeoff between error and maximum control signal. Figure 5.4 shows the acceleration reference and the surface elevation after five ILC iterations using the previous weights choice. We can see that the ILC algorithm successfully finds an acceleration reference that fulfills the specifications.

Since the container motion is performed stepwise, a waiting time between the end of each step and the beginning of the next is necessary to let the slosh inside the package be zero. The more the residual slosh is reduced the shorter is the waiting time interval between two steps. Consequently the package motion from the filling station to the sealing station will take shorter time. This means increased production rate and higher profit.

In the algorithm it is possible to reduce the residual slosh by increasing the weights on the last 20 samples. Figure 5.5 shows the surface elevation

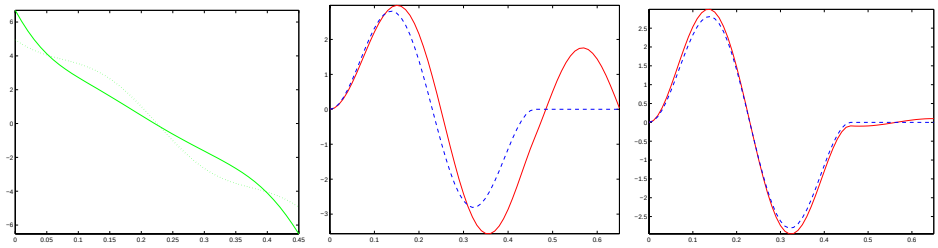

Figure 5.4 Acceleration reference (left fig.), initial (dotted) and after five iterations of ILC. Surface elevation $y^1(t)$ (solid) on the backward side of the package (center fig.) at iteration zero and at iteration five (right fig.). The linear process model has been used for simulations with the weights choice $W_1 = W_2 = I$ and $W_u = 0.00001I$.

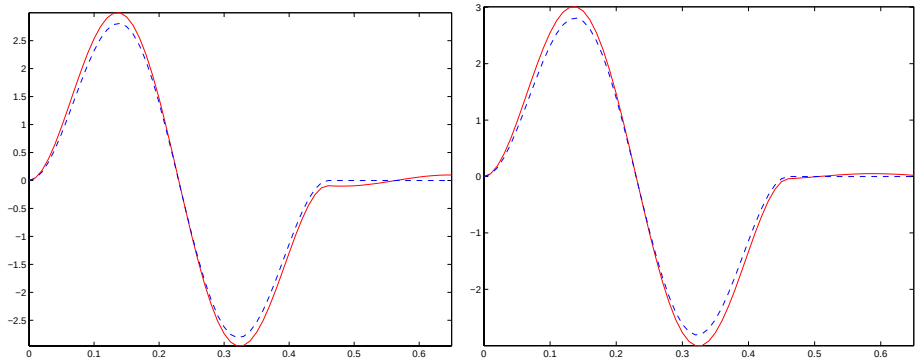

Figure 5.5 Surface elevation $y^1(t)$ (solid) on the backward side of the package at iteration five (right fig.) using the linear model with the weights choice $W_1 = W_2 = I$ and $W_u = 0.00001I$. Surface elevation $y^1(t)$ (solid) on the backward side at iteration five (right fig.) using the linear model with the weights choice $W_1 = W_2 = diag(I_{46}, 10I_{20})$ and $W_u = 0.00001I$. We can see that the residual slosh is reduced in the second case by increasing the weights in the last part of the reference

after five iterations for the weights choice $W_1 = W_2 = I$ and $Wu = 0.00001I$ and $W_1 = W_2 = diag(I_{46}, 10I_{20})$, $Wu = 0.00001I$. The figure shows that the residual slosh is reduced if the weights are increased in the final part. It was observed that the linear slosh model gives a good description of the real slosh when the slosh is small that is when not so fast package movement are considered. When fast package movements are executed that causes big slosh inside the container and the process exhibits nonlinear behavior. Therefore to have an idea how the ILC slosh algorithm works in a nonlinear context a non linear process model is introduced in simulations.This nonlinear process model is chosen to mimic some of the non linear behavior

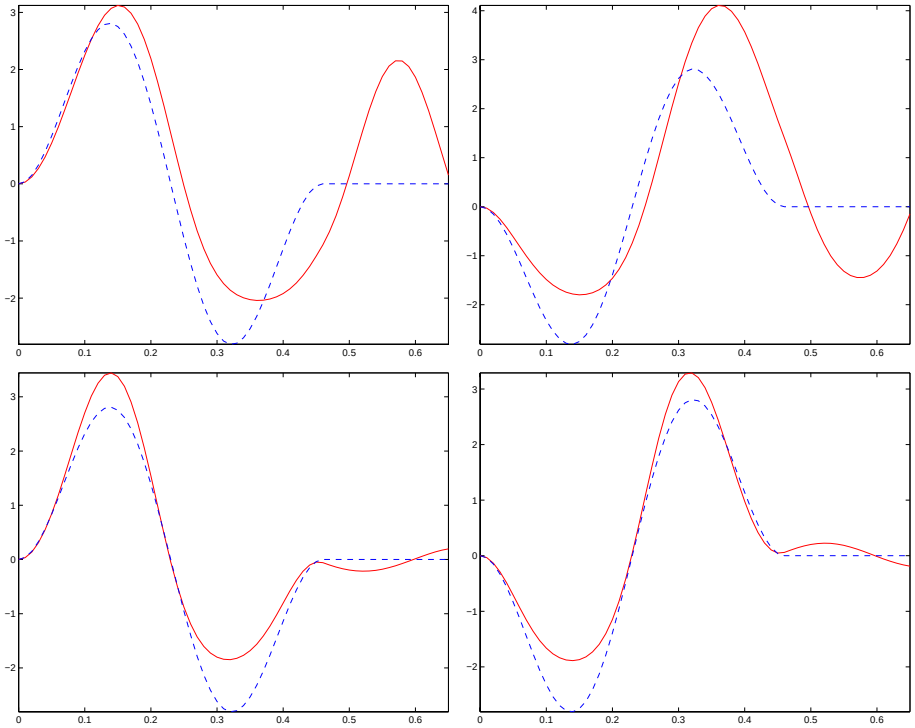

Figure 5.6 In the first row are shown the surface elevations on the backward (left fig.) and forward (right fig.) side of the package at iteration zero, respectively $y^1(t)$ and $y^2(t)$ (solid). In the second row $y^1(t)$ and $y^2(t)$ (solid) are shown after five iterations. The nonlinear process model has been used with the weights choice $W_1 = W_2 = diag(I_{46}, 10I_{20})$ and $W_u = 0.00001I$.

experienced in reality but there is no direct physical meaning of the model.

$$\begin{aligned}
\dot{x}_1(t) &= -2\zeta\omega_m x_1(t) - \omega^2 m\frac{\tanh 30x_2(t)}{30} + \frac{a\omega_m^2}{2g}u_k(t) \\
\dot{x}_2(t) &= x_1(t) \\
y_k^1(t) &= \frac{4}{5}x_2(t) + 7x_2^2(t) \\
y_k^2(t) &= -\frac{4}{5}x_2(t) + 7x_2^2(t)
\end{aligned}$$

where ζ, ω_m, a is the same as before. The term $\frac{\tanh 30x_2(t)}{30}$ will give an amplitude dependent oscillation frequency and the quadratic term in the output equations will give asymmetric oscillation. Figure 5.6 shows the surface elevation on the backward and forward side respectively after five iterations with the nonlinear process model and the weights $W_1 = W_2 = diag(I_{46}, 10I_{20})$ and $W_u = 0.00001I$. The figure points out that the residual slosh is small but there are large differences between the resulting surface elevation and the reference.

With the nonlinear process it is not possible to follow the reference completely. There is a coupling between the surface elevation on the forward and the backward side. Therefore, if the peak is lowered on the backward side the crest will be raised on the forward side which will make

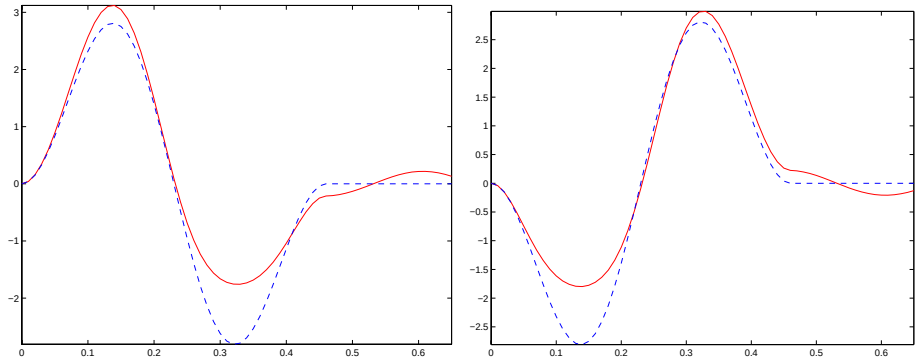

Figure 5.7 Surface elevations $y^1(t)$ and $y^2(t)$ (solid) respectively on the backward and forward side after five iterations of ILC using the nonlinear model with the weights $W_1 = diag(I_{23}, 0_{23}, 10I_{20})$ and $W_2 = diag(0_{23}, I_{23}, 10I_{20})$ and $W_u = 0.00001I$.

the quadratic error larger. Since it is more important to lower the maximum surface elevation one way to deal with the asymmetric behavior is to set the W_1 and W_2 to zero where the reference is negative. That means that we don't weight in the loss function the errors E^1_{k+1} and E^2_{k+1} when the slosh references become negative. Figure 5.7 shows the surface elevation after five iterations with the nonlinear process model and the weights $W_1 = diag(I_{23}, 0_{23}, 10I_{20})$, $W_2 = diag(0_{23}, I_{23}, 10I_{20})$ and $W_u = 0.00001I$, where 0_n is a $n \times n$ zero matrix. Figure indicates that the maximum surface elevation is decreased.

5.5 Experiments using a robot

The robot and the additional equipment

An industrial robot has been used to perform the package motion required by the algorithm that controls the slosh. The robot is an Irb2000, six joints robot, manufactured by the company ABB Robotics (see section 3.1 for a description of the robot system at the Department of Automatic Control).

The container is fixed to the wrist of the robot which provides to move it (see Figure 5.8).

The sampling time for the robot, the sensors and the algorithm in the experiment is $h = 0.005$ sec.

In the experiments one slosh measurement sensor is used to get information about the slosh inside the moving package. It is displaced on the backward side of the container and it measures the distance between the sensor and the liquid surface (see Figure 5.8). The sensor measures this distance by emitting a laser radiation that, once reached the liquid surface, is reflected back to the sensor. The slosh measurement is obtained at each sample instant by subtracting the initial measure (collected when the liquid is at rest) to every measurement collected during the motion.

During the package motion, it is important to keep under control the slosh on both sides (backward and forward). For this purpose the algorithm

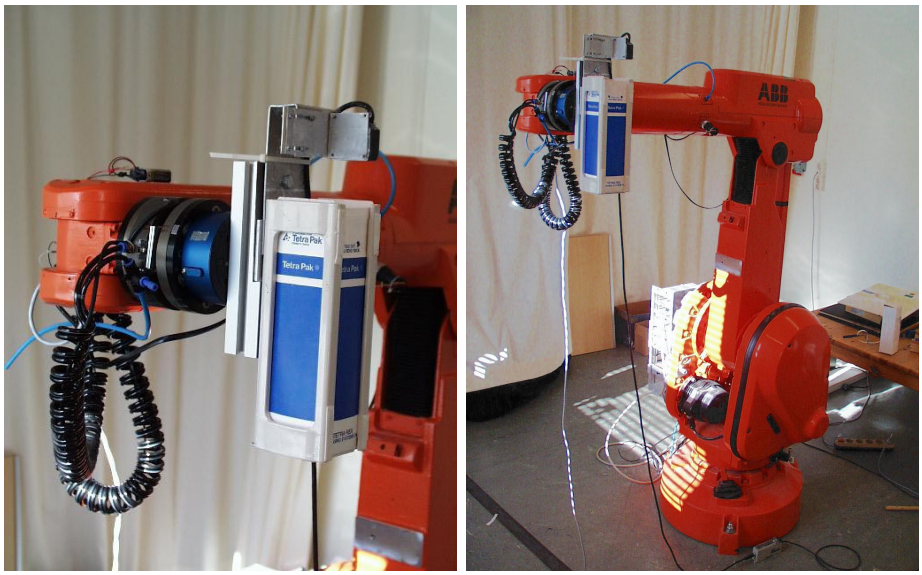

Figure 5.8 The picture on the left displays in detail how the package is fixed to the wrist of the robot. Observe the slosh sensor above the package. In the picture on the right we see the robot-package system in the starting position before the movement is performed.

that controls the slosh uses, at each iteration, the measurements from both sides of the package. We need two sensors to measure the slosh on both sides but in the experiments only one sensor is available (backward side). See Figure 5.9. It is possible to overcome this problem by performing the

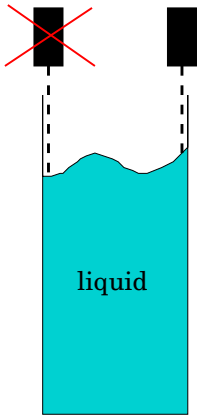

Figure 5.9 Scheme of the container with slosh inside. Above there are two slosh sensors. The sensor on the left measures the slosh on the forward side and the sensor on the right measures the slosh on the backward side. In practical experiments only the right sensor is available

movement two times instead of only one time in case two slosh sensors were available. During the first time the package is moved forward by a spatial acceleration $u_k(t)$ and the backward side surface elevation $y_k^1(t)$ is measured. The distance d has been covered and the robot waits still some seconds, necessary to let the slosh inside the container be zero. After that

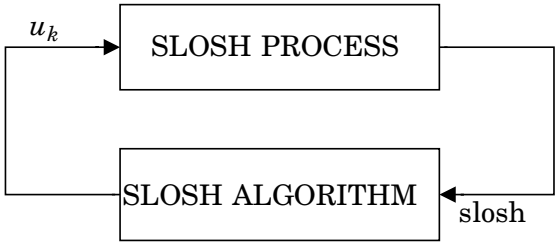

Figure 5.10 Ideal scheme for the Slosh ILC algorithm

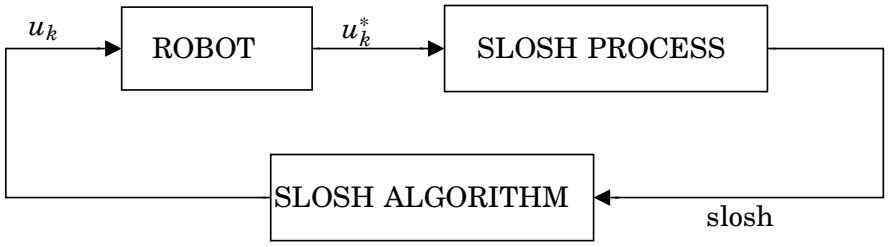

Figure 5.11 Real scheme for the Slosh ILC algorithm

the robot moves backward the container with spatial acceleration $-u_k(t)$ (the acceleration has negative sign because the head of the robot moves backward referring to the Cartesian axes system centered on the base of the robot). The backward motion makes it possible to measure on the backward side the same surface elevation $y_k^2(t)$ present on the forward side when the robot moves forward.

Objective and limitations

The aim in the slosh experiments is to shorten the motion time T according to the fulfillment of the constraints we discussed in the previous sections. **Remark:** we will refer to the algorithm that controls the slosh as Slosh ILC. It is the same algorithm shown in Section 5.3.

Ideally the Slosh ILC algorithm, at iteration k, calculates the acceleration u_k. This acceleration is applied to the package, the slosh is measured and the new measurements are used by the algorithm in order to calculate the new acceleration (see the scheme in Figure 5.10). Such a scheme is obviously ideal because it supposes that the same acceleration u_k calculated by the algorithm is applied to the package and that is not exactly possible in reality. Therefore we need to introduce in the scheme also the robot. The robot receives as input the acceleration u_k but it applies to the container, being also a real machine, an acceleration u_k^* different (see the scheme in Figure 5.11). The situation is the following, the Slosh ILC algorithm calculates at iteration k the acceleration u_k and sends it to the robot, since the robot does not track faithfully the references it will apply to the package a different acceleration u_k^*, which is not the acceleration u_k the slosh algorithm requires. So when the Slosh ILC algorithm receives

the measurements from the slosh sensor it considers them the response to the acceleration u_k sent to the robot and proceeds to calculate the next acceleration u_{k+1} wrongly.

In conclusion, if the robot is not capable to perform a movement with the acceleration as calculated by the ILC slosh algorithm, there is no hope the slosh experiment will work. It is impossible to get positive results from the experiment without modifying the things.

To improve the robot tracking performances Iterative Learning Control algorithms are applied to the joints.
Remark: we will refer to these algorithms as Slosh ILC (algorithms).

The general algorithm used in the experiment includes the use of the Slosh ILC and the Robot ILC according to the following scheme:

General algorithm

1. $k \leftarrow 0$

2. Calculate the initial acceleration u_0 by solving the Minimum Energy Problem and the slosh reference $r(t)$

3. Consider the acceleration reference $u_k(t)$

- **3.1.** Execute an iteration of Robot ILC and measure the acceleration performed $u_k^*(t)$
- **3.2.** If $u_k^*(t)$ does not approximate well $u_k(t)$ go to step 3.1.

4. Execute an iteration of Slosh ILC by reproducing on the robot the acceleration $u_k^*(t)$ performed in the last iteration of Robot ILC.

5. Calculate the new acceleration $u_k(t)$ using the Slosh ILC algorithm.

6. $k \leftarrow k + 1$

7. If the slosh behavior needs to be improved go to step 3.

Robot inverse kinematics solution

At iteration k the Slosh ILC algorithm calculates the acceleration profile u_{k+1}, that must be applied to the package.

Every joint of the robot expects as inputs one angular position reference and one angular velocity reference (see Figure 5.12).

The Slosh ILC algorithm calculates the spatial acceleration to be applied to the head of the robot where the package is mounted. The spatial acceleration cannot directly be used as input to the robot but it is necessary to calculate the input trajectories to the joints involved in the movement. Therefore the next procedure must be applied. The spatial acceleration is time-integrated using the Matlab command *cumtrapz* in order to get the spatial velocity. A further time-integration is necessary to calculate the spatial position. Once the spatial position and spatial velocity references are available we need to calculate the angular position and the angular velocity trajectories for each robot joint involved. That is done by using a function implemented in Matlab called *inverse*3.*m*. It solves the robot inverse kinematics problem and calculates the angular position and velocity references for all the joints involved. The joints position and joints velocity references just calculated are sent to the robot-system which performs the movement.

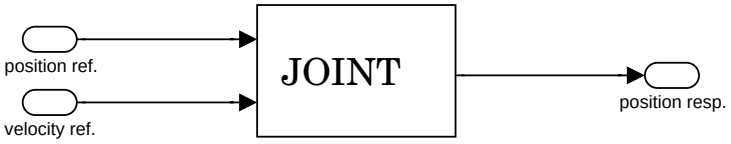

Figure 5.12 General scheme of one robot joint

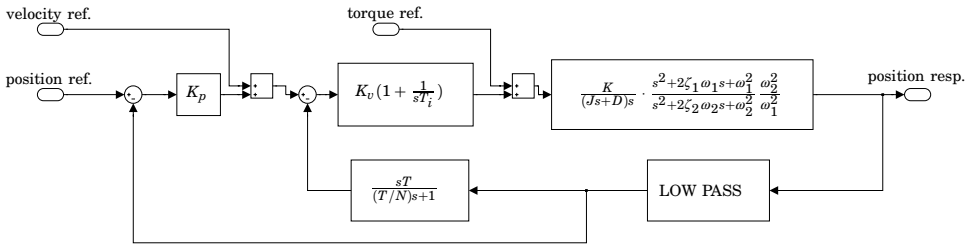

Figure 5.13 Blocks scheme of a robot joint

Robot ILC implementation

We have pointed out that if u_k is the desired acceleration, the actual acceleration performed by the robot is u_k^*. How much u_k and u_k^* are different depends on many causes that are to be investigated. We must try to make the robot produces an u_k^* very close to u_k in order to get significant results from the slosh experiments. That means the tracking performance of the joints involved in the motion must be improved.

We have said that the joint expects a position and a velocity input in order to perform the relative link motion. The joint can be considered a closed loop system if we include its controller. The joint controller is implemented as two cascaded controllers with an inner velocity loop and an outer position control loop. A simplified controlled joint model is displayed in Figure 5.13. In the blocks scheme of Figure 5.13 the biggest block contains a simple model of the system link-motor. To understand better what Robot ILC must improve it is useful to have a look to the results of an experiment perfromed on joint 2. The position input and the velocity input displayed (dashed) in Figure 5.14 are sent to joint 2. The second signal is the time derivative of the first. The figure shows the responses of the joint to the previous inputs (solid line). We observe that in the velocity response there are two undesired ripples produced when the velocity reference goes from nonzero to zero values. In these two areas the velocity reference goes to zero in a very short time interval. That puts in evidence the difficulty of the joint controller to reduce very fast to zero the joint velocity. Moreover the tracking is not so good in proximity of the maximum and the minimum of the velocity reference. Anyway the problem of the ripples is more critical. Actually, if we analyze the ripple on the left, it can be seen that the velocity in a short time interval changes from positive to negative values and then from negative to positive values. This means that the joint moves forward with decreasing velocity, then it moves a bit backward and then again forward.

These effects must be reduced, if it is not possible to avoid, because

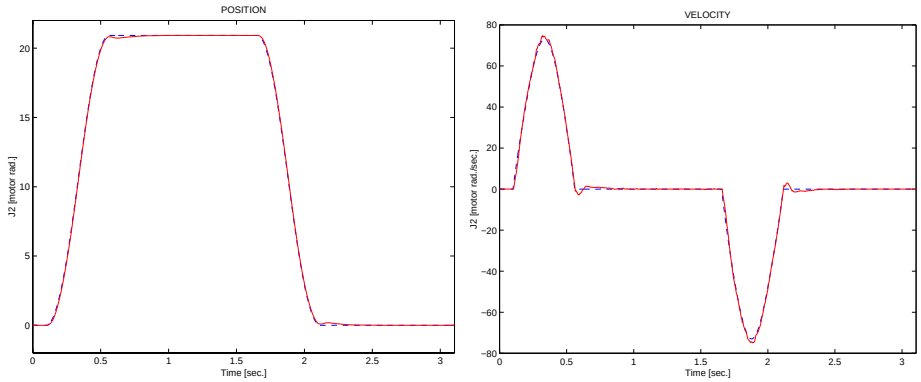

Figure 5.14 Response of joint 2 to position and reference inputs.

they could be very dangerous for the package motion. They could produce undesired shaking in the liquid. The way considered here in order to improve the tracking on the joints involved in the motion is Iterative Learning Control and the heuristic approach has been used.

Before proceeding we observe that, since every joint is fed with position and velocity signals, it is important that the velocity reference be the derivative of the position signal, otherwise there is mismatch and the tracking response of the joint could be not good.

The bandwidth of the joint to velocity reference is larger than the position reference one. Therefore the system is 'faster' to track velocity references than position references. To improve the joint tracking performance ILC can be used either on the velocity reference or on the position references. It is to be avoided the simultaneous use of position and velocity ILC. In fact if velocity and position references are updated, at each iteration, by two different ILC algorithms then the total joint velocity input could be rather different from the derivative of the total position input and as we said earlier such a mismatch causes bad tracking response of the joint.

Initially I applied ILC to the velocity references of joint 2 and 3 but, despite the efforts, there were not good results especially on joint3 which is the more critical joint to control. The velocity ILC introduced vibrations during the motion and that was very bad for the slosh inside the container.

Alternatively position ILC was taken into account, it was applied to joints 2 and 3 of the robot. Since in every iteration ILC algorithm updates the joint position input, it is necessary to send to the joint velocity input the derivative of the position input in order to avoid mismatches. Anyway it is not advisable to send directly the derivative because the discrete differentiation causes the presence of high frequencies components in the velocity reference input. The joint controller tries to track these oscillations caused by the discrete derivative and consequently oscillations appear also in the joint response. The result is that the velocity profile is not well tracked. The presence of these oscillations means undesired vibrations during the liquid motion. Low-pass filtering of the differentiated position signal is needed before sending it to the joint velocity input (see Figure 5.15).

In detail let J_i be the time-indexed vector containing the position inputs to the joint i. The time discrete derivative DJ_i is calculated from J_i using

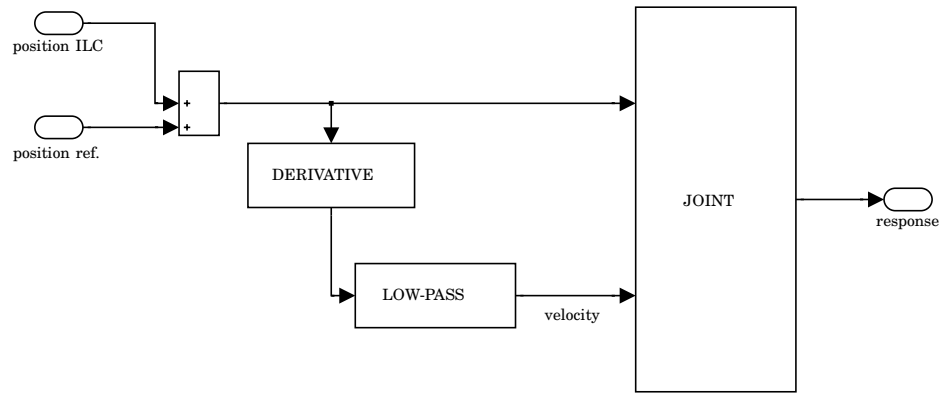

Figure 5.15 Scheme of ILC applied to joints 2 and 3

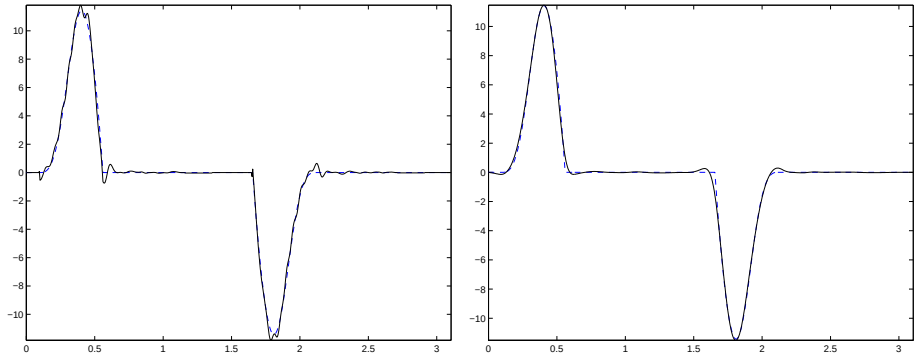

Figure 5.16 Joint velocity input without low-pass filtering (on the left) and actual joint velocity input using low-pass filtering (on the right).

the Matlab command $diff$. After that DJ_i is filtered by a zero phase low-pass filter which is implemented in Matlab using the command Butter to have low pass Butterworth filtering and then using the command $filtfilt$ to have zero phase filtering. After these operations the signal can be applied to the velocity input of the joint i.

Nevertheless the zero-phase filtering just discussed introduces a bit of distortion and that causes the signal does not start exactly from zero, there is a small step from zero to nonzero at the beginning of the velocity input. We must avoid to send steps in input to the joints especially on velocity inputs because as told the system is more sensible to velocity steps (the velocity bandwidth is larger than the position one). This problem is overcome by smoothing the step using an opportune ramp which starts from zero. Figure 5.16 shows (on the left) behavior of the joint velocity input using position ILC without low-pass filtering and (on the right) after low-pass filtering.

If we analyze for example the position reference (dashed line) in Figure 5.14, we see that there is no need to use ILC all over the time interval the reference is sent to the joint. In fact learning is needed where the position reference changes and not in time intervals where it is constant

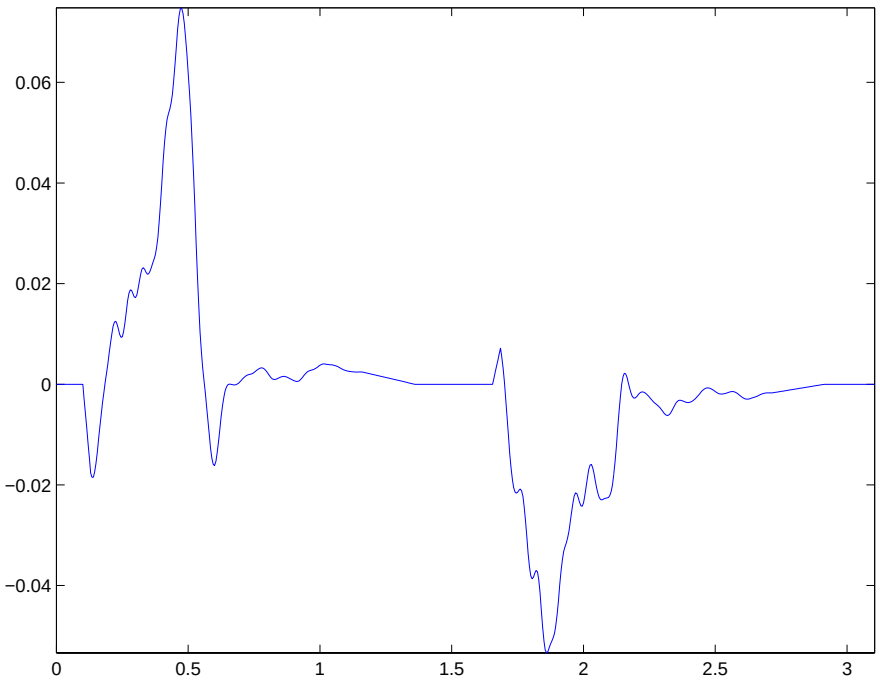

Figure 5.17 Example of ILC control on joint 3 using ramps to smooth the transition between regions of the reference not updated by ILC-algorithm and updated regions of the reference.

(because the joint and the relative link are still and the joint controller holds the position in a very good way). So position ILC is used in the 2 subintervals where the joint moves.

After iteration zero (no ILC is applied), there is the problem that in the time subintervals considered the ILC control signal is different from zero and it is zero outside, consequently in the total ILC control signal there are 4 steps. These steps are very dangerous as we have mentioned in the past because they cause an oscillatory transient in the response, and so they must to be avoided. One way is to smooth them with appropriate ramps from zero to nonzero and from nonzero to zero values (starting from the left to the right).

The problem is more critical when there is a transition from non-zero to zero because it is hard to stop the motion of the robot link involved in a very short time. Therefore the smoothing on the right side of the ILC control signal must be slower than on the left side. That is the duration of the right ramp must be greater than the left one (Figure 5.17). A greater duration time means to shift to the right the right extreme of the ramp, being the left one fixed. The ramp duration must be opportune especially on the left side of the subintervals. If on the left side it is too short the step effects are still present and if it is too long the performance of the learning algorithm is damaged because the ramp extends over a time interval where learning is needed to improve the tracking.

Filters design

The heuristic design procedure is :

1. Choose the Q filter as a low pass with cut-off frequency such that the band-width of the learning algorithm is sufficient.
2. Let $L(q) = \kappa q^{\delta}$.

In the experiment heuristic ILC algorithm has been implemented on joint 2 and on joint 3. The Q-filter considered is a zero phase filter, $Q(q) = \overline{Q}(q)\overline{Q}(\frac{1}{q})$, this operation is implemented in Matlab using the command $filtfilt$. $\overline{Q}(q)$ is a second order Butterworth filter created in Matlab by the command,

$$>> [\overline{Q}_{num}, \overline{Q}_{den}] = butter(2, \omega_i);$$

where ω_i is the normalized cut-off frequency of the filter, it is a fraction of the Nyquist frequency. In detail the following filter parameters choice has been done for joint 2

$$\begin{aligned} \omega_2 &= 0.2 \\ \kappa_2 &= 0.9 \\ \delta_2 &= 4 \end{aligned}$$

and for joint 3 the choice has been

$$\begin{aligned} \omega_3 &= 0.2 \\ \kappa_3 &= 0.9 \\ \delta_3 &= 6 \end{aligned}$$

Experimental Results

Now we can describe the package motion experiment when the movement time is $T = 0.46$ sec.. The Robot ILC is used to improve the tracking on joint 2 and joint 3 of the joints position and velocity signals. Figure 5.18 and Figure 5.19 show how is improved the position and the velocity tracking on joint 2 and joint 3 respectively, at iteration 0 of Slosh ILC (when the joint references are calculated from the solution u_0 of the Minimum Energy Approach in the way described earlier and no Slosh ILC updating is still considered).

Remark:we will refer to Slosh ILC iterations also as outer iterations. In fact in the general algorithm shown previously the Robot ILC loop (inner loop) is inside the Slosh ILC loop (outer loop).

The figures refer to the tracking on the motor side, so position signals are expressed in motor radians and velocity signals in motor radians per second. On the horizontal axes there is the time in seconds. In both figures position signals are on the left and velocity signals on the right.

The robot ILC iterations are shown from top to bottom. The references are in dashed line, the responses in solid line and the position ILC updating in dotted line. The Robot ILC is stopped after 3 iterations because the

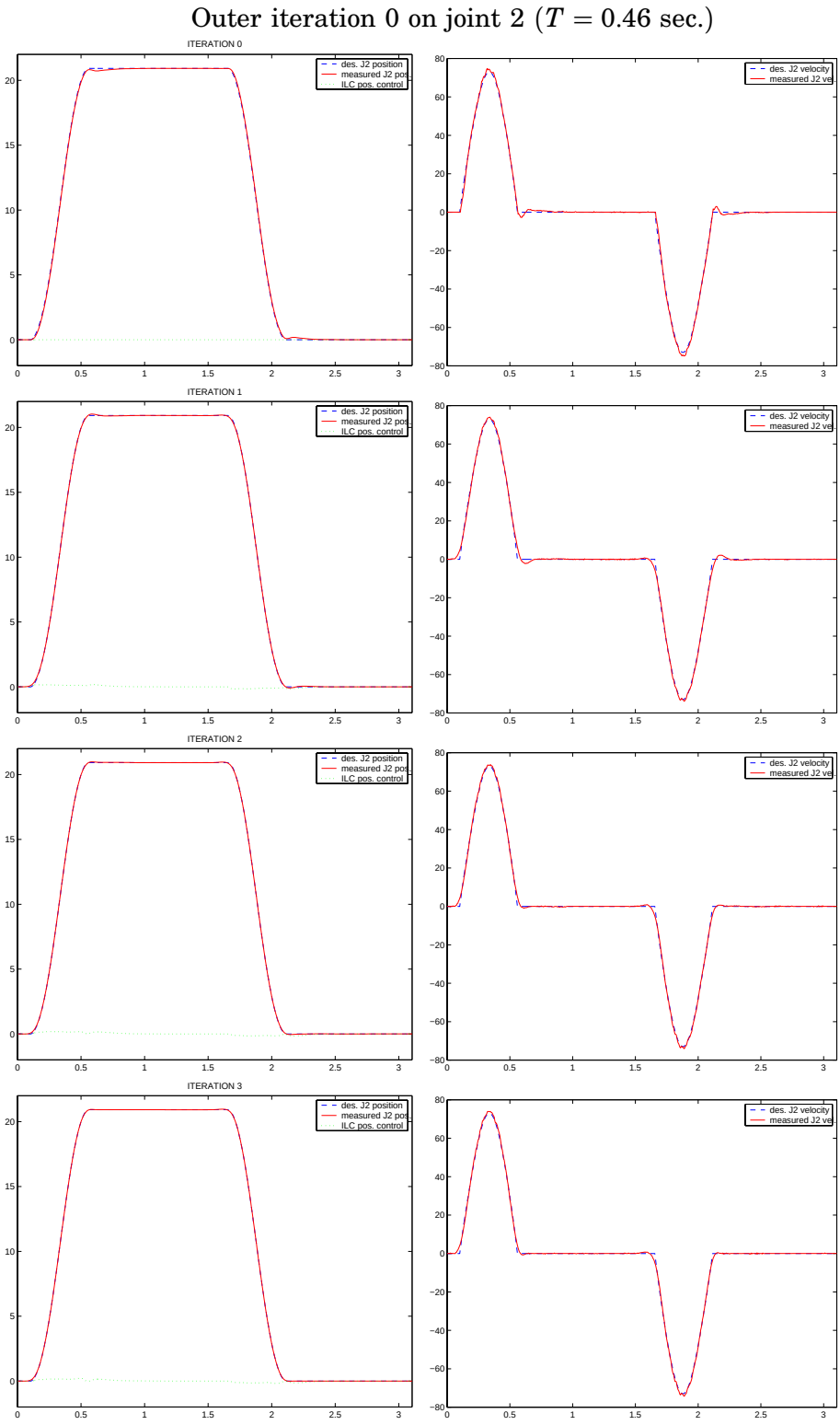

Figure 5.18 Tracking on the motor side of joint 2 applying Robot ILC to the position reference. The figure refers to iteration zero of Slosh ILC and to $T = 0.46$. Dashed line is used for reference signals, solid line for measured output signals, dotted line for ILC signals. Every row represents one ILC iteration, on the left there are the position signals and on the right the velocity signals. The Robot ILC iterations are shown from top to bottom.

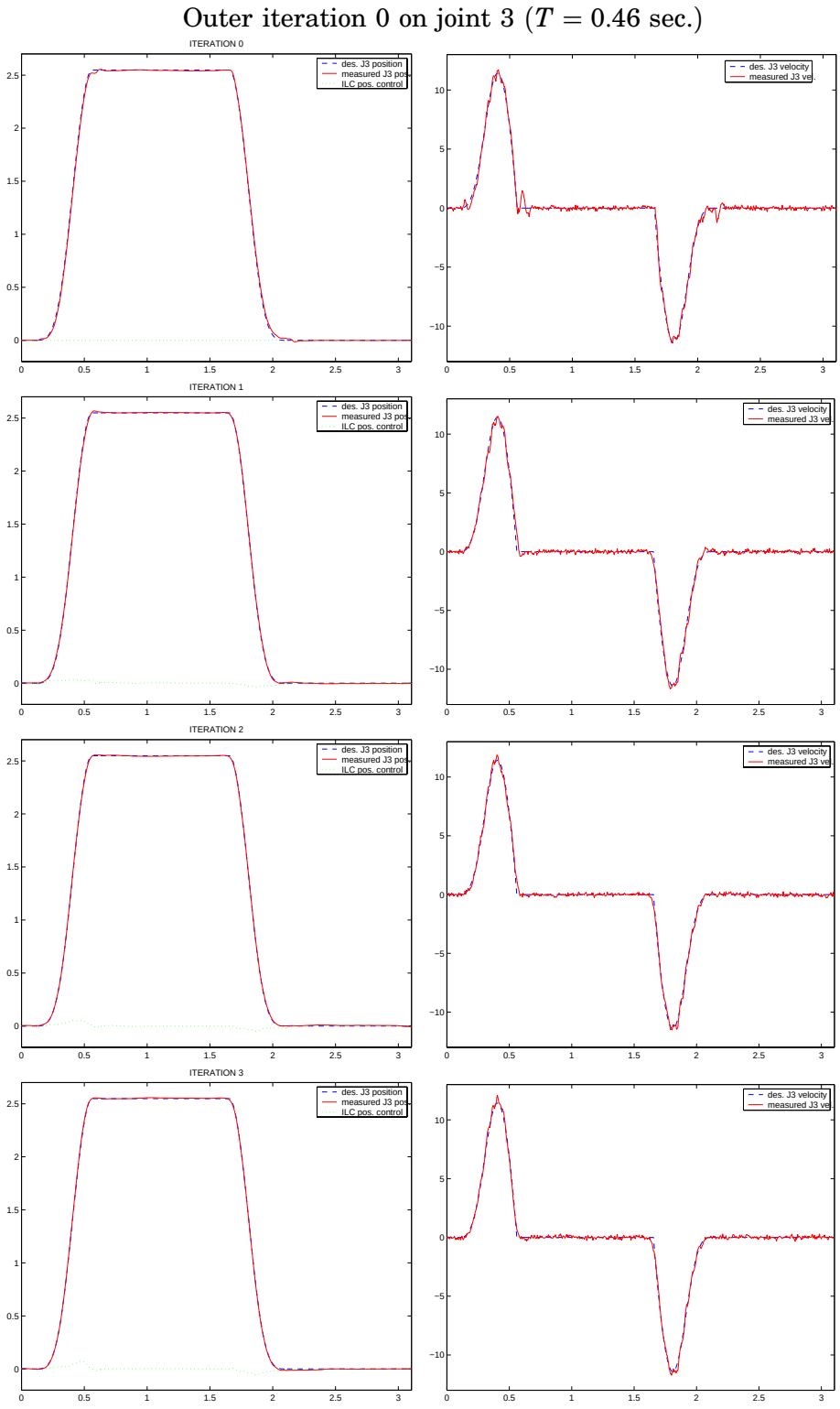

Figure 5.19 Tracking on the motor side of joint 3 applying robot ILC to the position reference. The figure refers to iteration zero of Slosh ILC and to $T = 0.46$. Dashed line is used for reference signals, solid line for measured output signals, dotted line for ILC signals. Every row represents one ILC iteration, on the left there are the position signals and on the right the velocity signals. The Robot ILC iterations are shown from top to bottom

performance does not improve. On joint 2 and on joint 3 we can see how ILC is efficient in order to reduce the initial velocity ripples iteration by iteration.

Figure 5.20 and 5.21 show how is performed Robot ILC tracking on the motor side of joint 2 and joint 3 in the third and last outer iteration executed.

In comparison with outer iteration zero shown in Figure 5.18 and 5.19 we can see how position and velocity references have been changed by the Slosh ILC updating.

Figure 5.22 shows the liquid surface elevation (in cm.) on the backward side (left column) and on the forward side (right column) when ILC is applied (from top to bottom of the figure). Row 1 corresponds to outer iteration zero, row 2 to iteration 1, row 3 to iteration 2 and row 4 to iteration 3. We see on iteration 3 how the peaks on the backward and forward side are lowered with respect to iteration 0. Moreover, also the residual slosh is sensibly reduced through the iterations.

Figure 5.23 shows how the acceleration changes from iteration 0 (dotted line) to iteration 3 (solid line).

Figure 5.24 shows the liquid backward and forward surface elevation through the Slosh ILC iterations when the movement time has been increased to $T = 0.6$ sec.. The peaks are lowered and the residual slosh reduced.

In Figure 5.25 we see the initial acceleration profile (dotted line) and the acceleration profile at outer iteration 2 (solid line).

Figure 5.26 shows the backward and forward liquid surface elevation using Slosh ILC (from top to bottom) when $t = 0.7$ sec. Here there is not the problem of lowering the maximum peaks, the algorithm reduces the residual slosh.

In Figure 5.27 the initial acceleration profile (dotted line) and at iteration 2 (solid line).

Discussion

Previously we have said how is necessary to low-pass filter the joint velocity input before sending it to the robot. At every iteration the position input J_i to the joint i involved is the sum of the position desired reference $J_i^{(des)}$ and the ILC updating $J_i^{(ILC)}$.

$$J_i = J_i^{(des)} + J_i^{(ILC)}$$

The joint velocity input is calculated by diffentiating and low-pass filtering J_i. The low-pass filtering is also needed because the position ILC updating introduces through the differentiation some undesired higher frequency oscillations in the velocity input. Therefore an opportune low pass filter cuts these oscillations. On the other side this filtering causes the velocity input is smooth also in time intervals (when velocity changes from zero to nonzero) where the tracking of $J_i^{(des)}$ is good. Consequently the velocity smoothing, due to low pass filtering, causes the tracking is a bit damaged in these intervals (see for example Figure 5.18).

Probably, a better solution, though not tried, is to avoid low-pass filtering in the intervals where the robot controller tracks good the input signals with higher frequency components. I say probably because when the joint

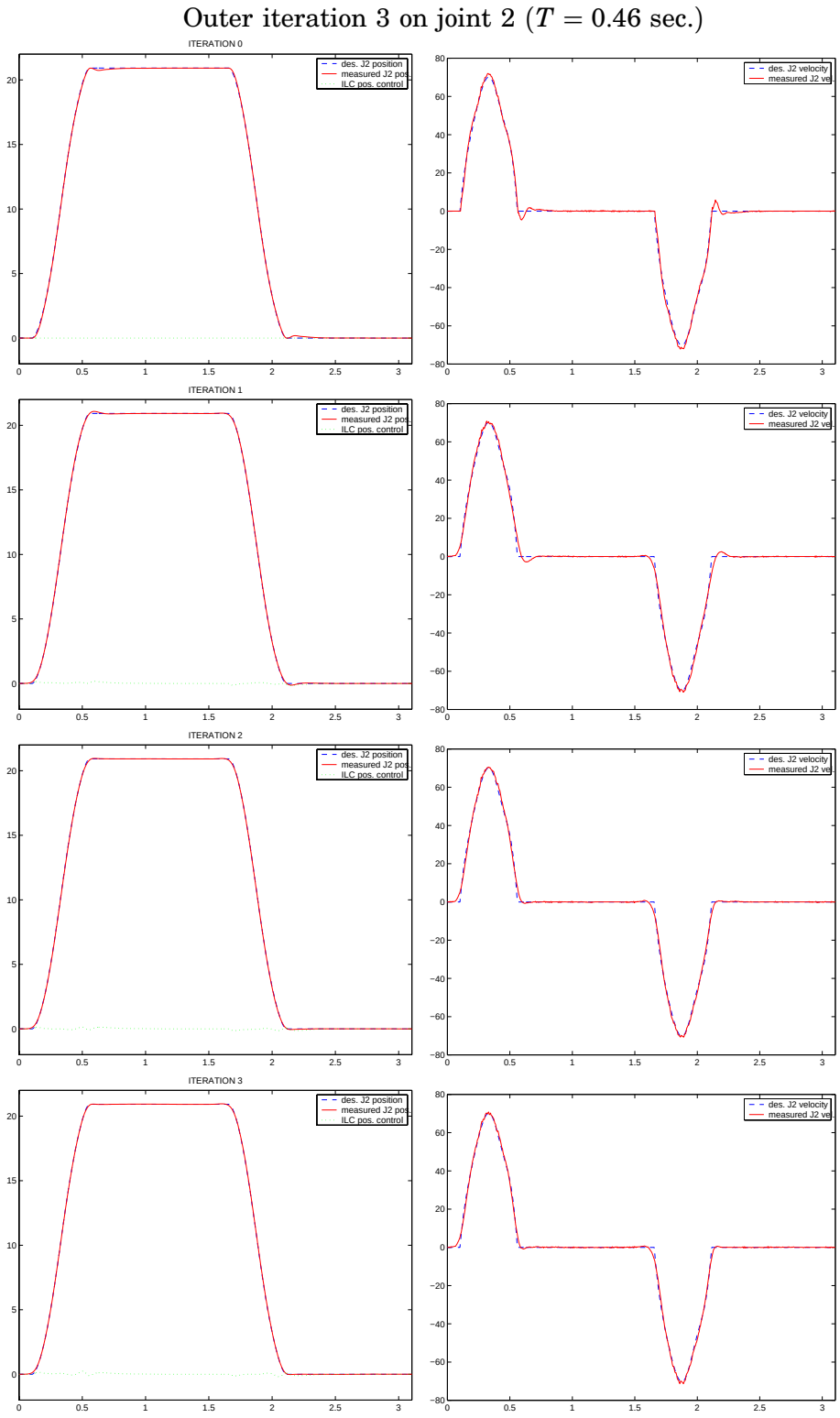

Figure 5.20 Tracking on the motor side of joint 2 applying robot ILC to the position reference. The figure refers to iteration three of Slosh ILC and to $T = 0.46$. Dashed line is used for reference signals, solid line for measured output signals, dotted line for ILC signals. Every row represents one ILC iteration, on the left there are the position signals and on the right the velocity signals. The Robot ILC iterations are shown from top to bottom

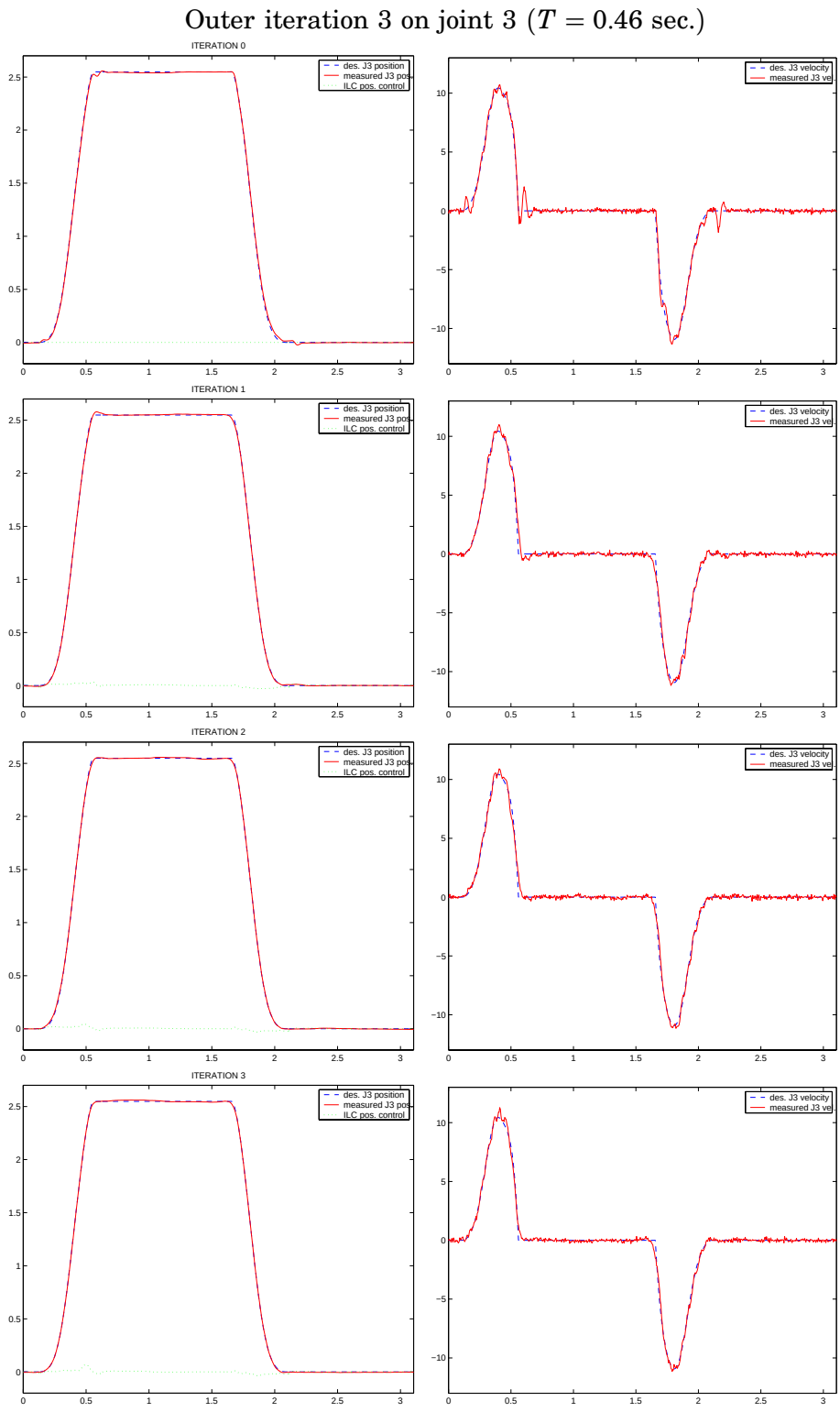

Figure 5.21 Tracking on the motor side of joint 3 applying robot ILC to the position reference. The figure refers to iteration three of Slosh ILC and to $T = 0.46$. Dashed line is used for reference signals, solid line for measured output signals, dotted line for ILC signals. Every row represents one ILC iteration, on the left there are the position signals and on the right the velocity signals. The Robot ILC iterations are shown from top to bottom

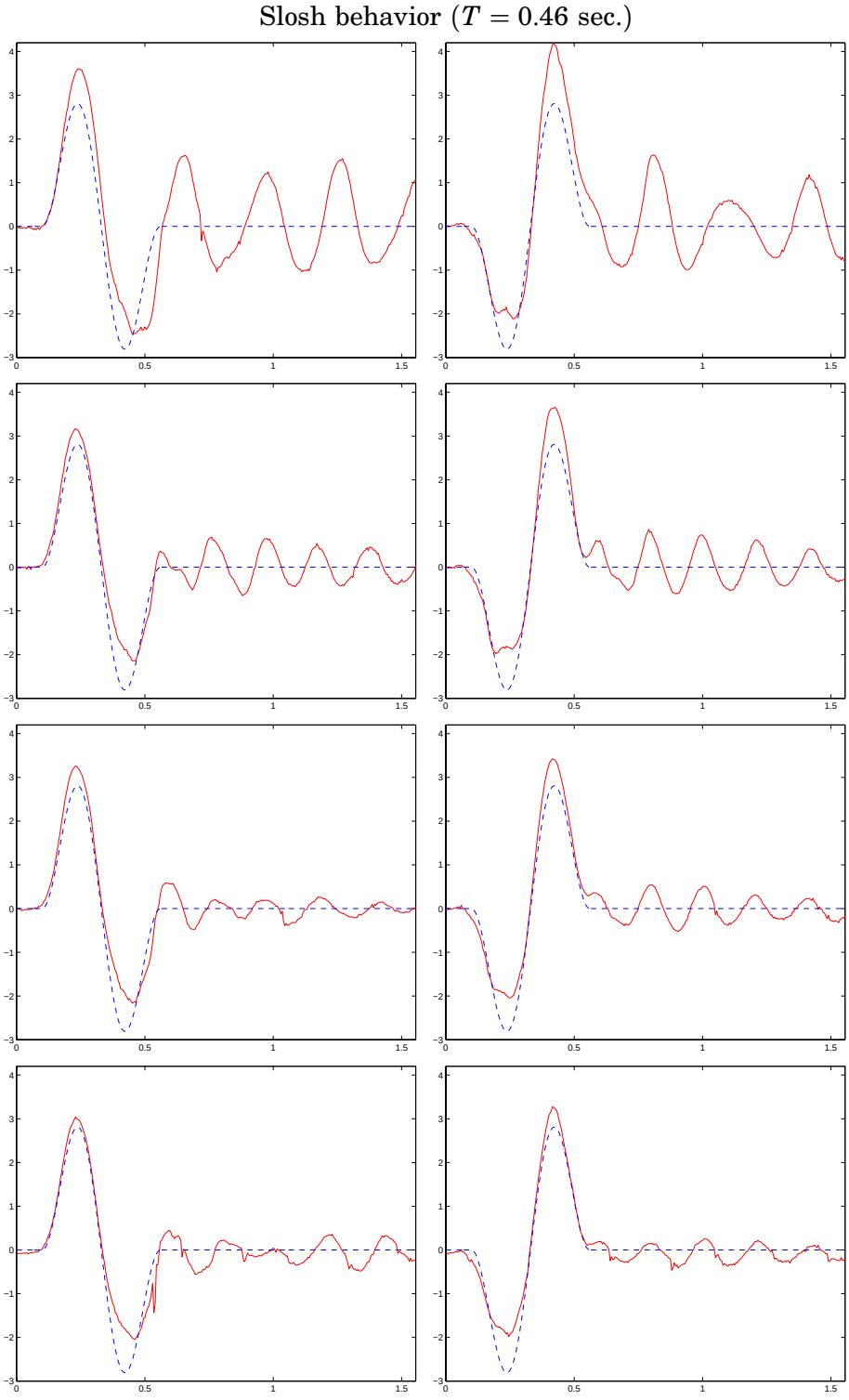

Figure 5.22 Liquid surface elevation on the backward side (left column) and on the forward side (right column) using Slosh ILC (from top to bottom), when $T = 0.46$.

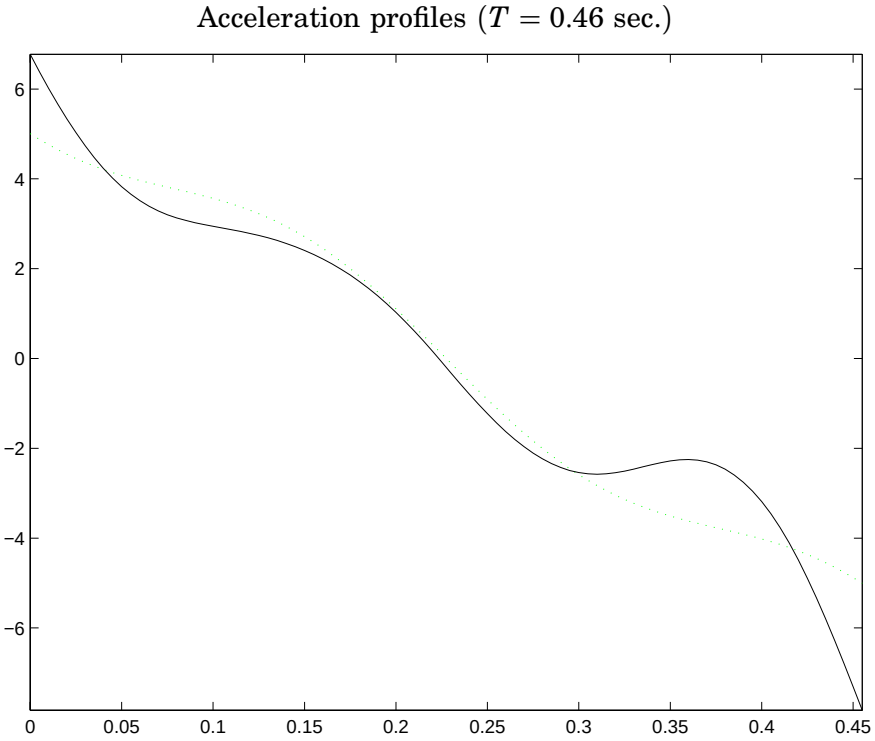

Figure 5.23 Initial acceleration profile (dotted line) and acceleration profile at iteration three of Slosh ILC (solid line), when $T = 0.46$.

velocity changes very fast (high frequency components are present in the input signal) the problem of joint flexibility could come. That is, the angle tracking is well performed on the motor side but not on the arm side, there is an effect similar to that met in the problem of the two masses linked by a spring.

This effect if present damages the performance on the arm side. It causes the package is moved improperly not according to the accelerations produced on the motor side.

Therefore there are situations where it is better not to have very good tracking on the motor side because flexibility causes deviation on the arm side.

Before proceeding we observe that in every iteration zero of Robot ILC the joint velocity input is not the filtered derivative of the position reference. In iteration zero the ILC control signal is null. I don't know exactly how to justify this choice but from experiments done it results better than to use a low pass filter also in iteration zero.

The Slosh algorithm used does not weight the slosh error when the references are negative. In other words, it does not care about the tracking when the slosh references on both sides (backward and forward) are negative. This has been discussed earlier and it gives more freedom to the Slosh algorithm in order concentrate the efforts on how to lower the peaks and how to reduce the residual slosh.

During the motion it is important to lower the maximum peaks as well as to reduce the residual slosh. Remember that since the container motion is performed stepwise, a waiting time between the end of each step and

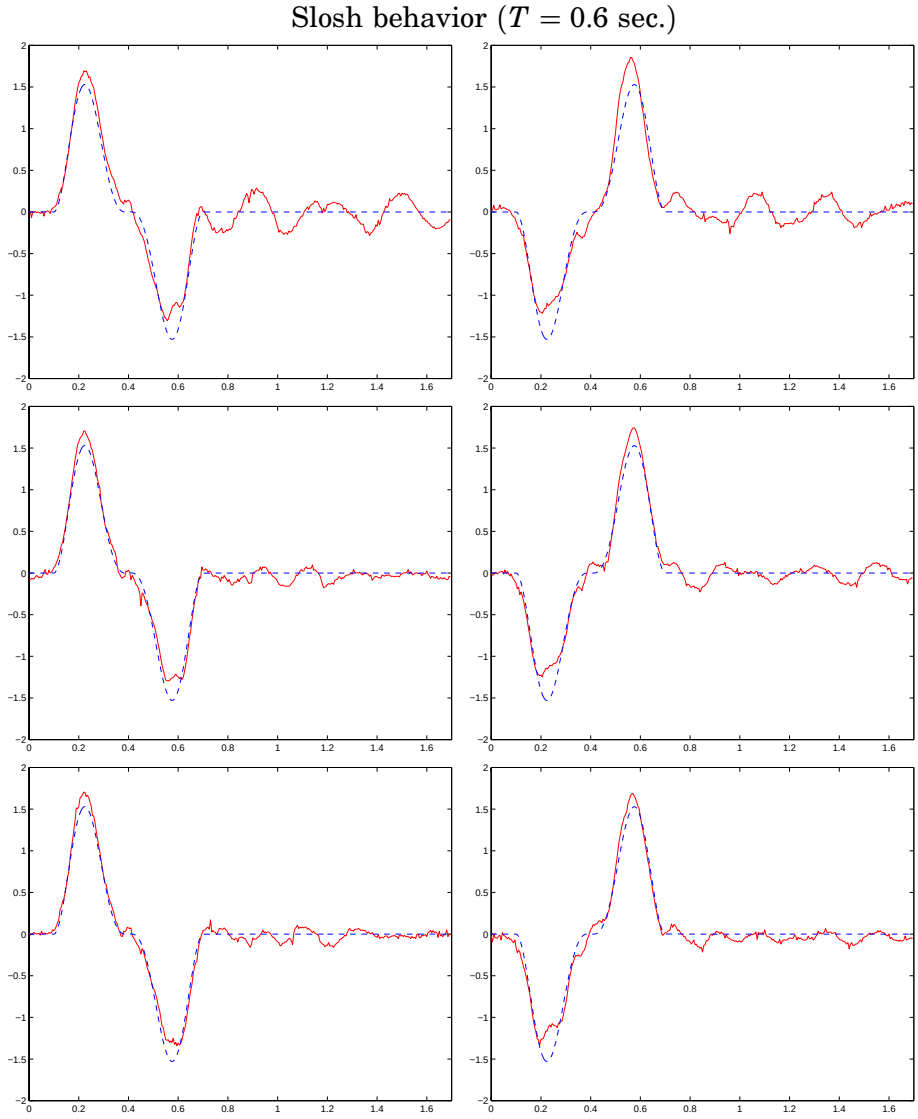

Figure 5.24 Liquid surface elevation on the backward side (left column) and on the forward side (right column) using Slosh ILC (from top to bottom), when $T = 0.6$.

the beginning of the next is necessary to let the slosh inside the package be zero. The more the residual slosh is reduced the shorter is the waiting time interval between two steps. Consequently the package motion from the filling station to the sealing station will take shorter time. This means increased production rate and higher profit.

In the experiment with $T = 0.46$ the Slosh algorithm has been stopped at iteration 3. It does not improve if we run further iterations and there is even the risk to worsen the performance. Let us have some reflections on the causes.

Although the position ILC used improves the tracking performances of the joints controllers with respect to the case without ILC, we cannot say that the tracking performed by ILC is perfect. The Slosh algorithm at each iteration calculates the next acceleration reference from the last acceleration and last slosh measurements. In my opinion the Slosh algo-

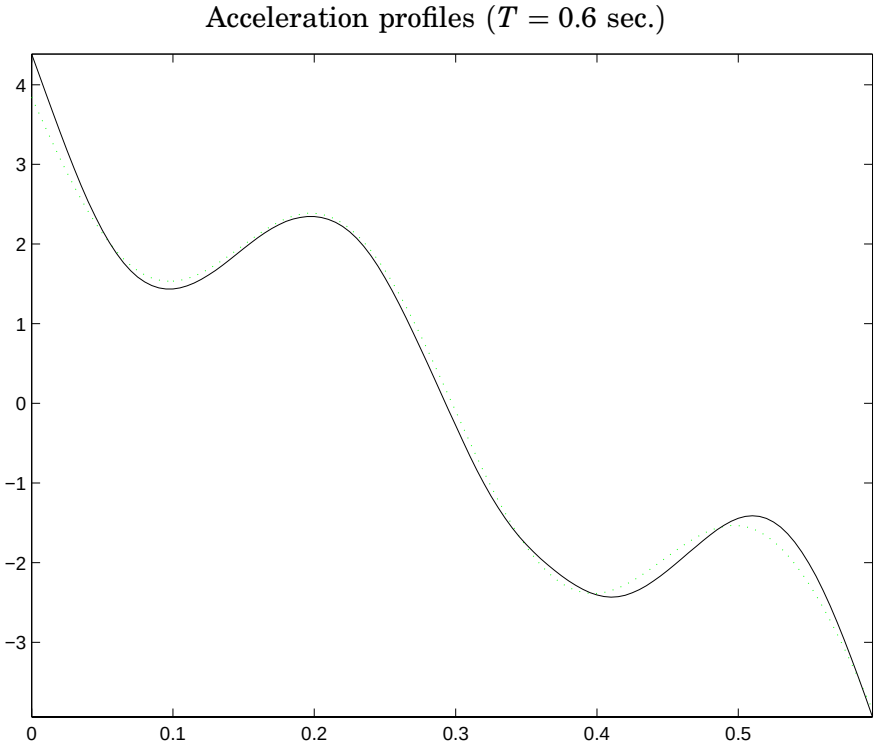

Figure 5.25 Initial acceleration profile (dotted line) and acceleration profile at iteration two of Slosh ILC (solid line), when $T = 0.6$.

rithm does not improve over some iterations $(3 \div 4)$ because, in order to calculate the exact acceleration, it needs from the measured slosh some small details that only a faithful almost perfect reference tracking of the robot can assure.

Moreover there are other damaging aspects to take into account. Joint flexibility is one of them. We have applied ILC to improve the tracking on the motor side of the joint. There is a gear ratio between the motor angle and the relative link angle. If the joint reference inputs are not so fast then the gear ratio is a constant and it is so very easy to determine the link angle from the motor angle. On the contrary if the joint reference inputs are fast (the robot moves fast) the gear ratio is not constant anymore and the relation between motor angle and link angle is not easy to establish.

The robot ILC algorithm implemented improves the tracking on the motor side. In the case the gear ratio is constant if there is good tracking on the motor side there is consequently good tracking on the link side. In the experiment is important the tracking on the link side because the motion of the package is determined geometrically by the the movements of the robot links.

In presence of flexibility we can have a situation with good tracking performed on the motor side but poor tracking on the link side. To overcome the problem of flexibility is not easy.

We need to add one extra sensor (for example an accelerometer) to measure the angles on the robot links and there is the problem of performing the sensor-fusion with the measurements of the motor angle and the Cartesian acceleration measurements. Moreover to make changes to

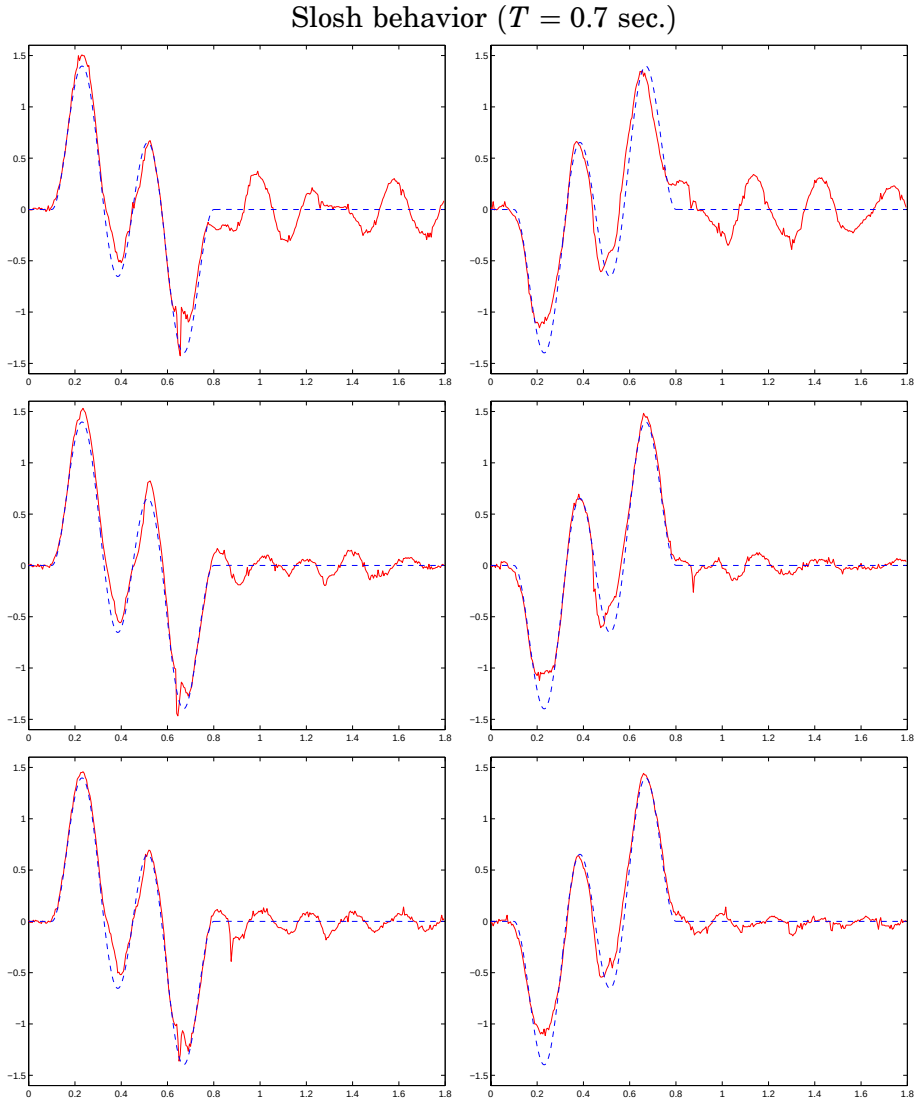

Figure 5.26 Liquid surface elevation on the backward side (left column) and on the forward side (right column) using Slosh ILC (from top to bottom), when $T = 0.7$.

the algorithms used and to implement the new theory subjects that are needed is not simple. There are some aspects that are still open to future research.

Another problem present during the experiments has been the presence of disturbances on the measurement system of the joints position. These disturbances damage the tracking performance of the joint controllers and of ILC. Therefore it is hard to improve further the tracking performance.

In conclusion we can say that, by taking into account the causes explained above, the Slosh algorithm improves up to a limit and after that it is not worth to run further iterations.

We observe that when $T = 0.6$ and $T = 0.7$ sec. the Slosh ILC algorithm changes only slightly the initial acceleration profile. In fact the initial acceleration profile is the solution of the Minimum Energy Approach, which is the best solution for linear process model.

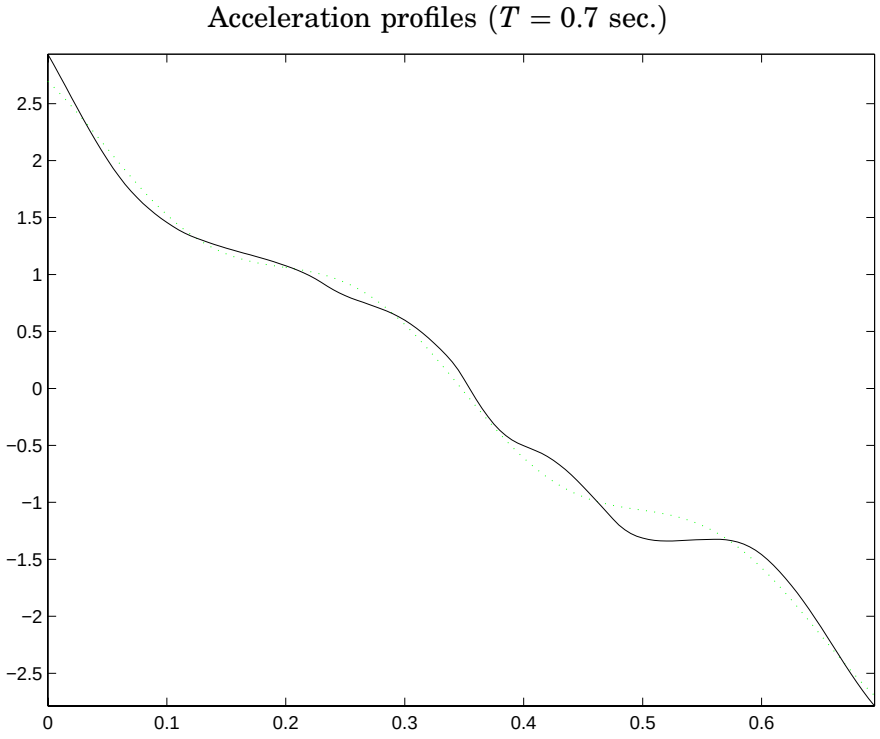

Figure 5.27 Initial acceleration profile (dotted line) and acceleration profile at iteration two of Slosh ILC (solid line), when $T = 0.7$.

For large values of T the slosh inside the container has a behavior close to that described by the linear process model. So being in these conditions, the linear process model is a good model to describe the real process and thus it is obvious that the acceleration profile calculated during the Slosh ILC iterations be very similar to the initial acceleration profile solution of the Minimum Energy Problem.

6. Conclusions and Future Work

6.1 Conclusions

Heuristic, Model Based and Optimization Based ILC algorithms have been implemented in simulations. The simulation environment used has been Matlab/Simulink. The Optimization Based ILC algorithm resulted in the best performance, followed by the Model Based and finally the Heuristic one.

Heuristic and Model Based ILC has been applied to the robot to improve the tracking of a spiral reference. In these experiments Model Based ILC shows a faster convergence than the Heuristic one. That is coherent with what we can expect from ILC theory and from simulations.

The Model Based ILC has been tested in other experiments where the robot had to draw the sketch of a Christmas tree. This figure is composed by lines and angles and represents a good test for the ILC algorithm implemented. ILC works good and it reduces considerably the tracking error in proximity of the angles of the tree.

In the experiments with open containers, Heuristic ILC has been implemented in order to improve the tracking of a desired acceleration profile. ILC has been applied (in simulations and experiments) also for the slosh control inside a moving container. Iteration by iteration it lowers the maximum peaks of the slosh and reduces the residual slosh.

6.2 Future work

ILC has been applied to improve the tracking on the motor side but during some experiments the problem of joint flexibility has appeared. Especially in the experiments with the slosh when fast robot movements have been performed. In presence of flexibility we can have a situation with good tracking on the motor side but bad tracking on the arm side. To overcome the problem of joint flexibility extra sensors (accelerometers) must be added in order to get information about the position and the velocity of the robot links. ILC algorithms must be consequently modified to take into account both motor side measurements and arm side measurements.

In the experiments where Model Based ILC has been applied, probably the models adopted for the controlled joints are too simple. More detailed models are needed to improve the performance of the ILC algorithm. In particular joint 3 requires a model which takes into account the effects of gravity.

Only Heuristic ILC has been implemented in the experiments with slosh. This was done according to a simple but important 'rule' which says: "Try simple things first". Therefore before other things it is advisable to apply the simplest algorithm. The other two ILC approaches require the in-

troduction of a model for each joint involved. This model must describe the joint dynamics when fast movements are performed. Unfortunately there was not enough time for the implementation of these remaining ILC algorithms because there were many problems to face during the development of slosh experiments. It would be interesting to apply them and see how they work.

7. Bibliography

Arimoto, S., S. Kawamura, and F. Miyazaki (1984): "Iterative learning control for robot systems." In *Proceedings of IECON*. Tokyo, Japan.

Casalino, G. and G. Bartolini (1984): "A learning procedure for the control of movements of robotic manipulators." In *IASTED Symposium on Robotics and Automation*.

Craig, J. (1984): "Adaptive control of manipulators through repeated trial." In *Proc. of ACC*. San Diego, CA.

Craig, J. (1988): *Adaptive Control of Mechanical Manipulators*. Addison-Wesley Publishing Company.

Grundelius, M. (1998): *Motion Control of Open Containers with Slosh Constraints*. Lic Tech thesis ISRN LUTFD2/TFRT--3222--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Grundelius, M. (2000): "Iterative optimal control of liquid slosh in an industrial packaging machine." In *Proceedings of the 39th IEEE Conference on Decision and Control*. Sydney, Australia.

Grundelius, M. and B. Bernhardsson (1999a): "Control of liquid slosh in an industrial packaging machine." In *Proceedings 1999 IEEE Int. Conf. Control Applications and the Symp. Computer Aided Control Systems Design (CCA'99&CACSD'99)*. Kohala Coast, Hawaii.

Grundelius, M. and B. Bernhardsson (1999b): "Motion control of open containers with slosh constraints." In *Preprints 14th World Congress of IFAC*, vol. L, pp. 487–492. Beijing, P.R. China.

Grundelius, M. and B. Bernhardsson (2000): "Constrained iterative learning control of liquid slosh in an industrial packaging machine." In *Proceedings of the 39th IEEE Conference on Decision and Control*. Sydney, Australia.

Gunnarsson, S., O. Rousseaux, and V. Collignon (1999): "Iterative feedback tuning applied to robot joint controllers." In Chen *et al.*, Eds., *Proc. of the 14th World Congress of IFAC*, vol. I, pp. 451–456. Elsevier Science, Beijing, P.R. China.

Nilsson, K. (1996): *Industrial Robot Programming*. PhD thesis ISRN LUTFD2/TFRT--1046--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Norrlöf, M. (1998): "On analysis and implementation of iterative learning control." Licentiate thesis LIU-TEK-LIC-1998:62 Linköping Studies in Science and Technology. Licentiate Thesis No 727. Department of Electrical Engineering, Linköping University.

Norrlöf, M. and S. Gunnarsson (1999): "A model based iterative learning control method applied to an industrial robot." In *Second Conference on Computer Science and Systems Engineering in Linköping*.

Norrlöf, M. (2000): *Iterative Learning Control: Analysis, Design and Experiments*. PhD thesis, Department of Electrical Engineering, Linköping University. Sweden.

Norrlöf, M. and S. Gunnarsson (1998): “Some results on iterative learning control with disturbances.” In *Proceeding First Conference on Computer Science and Systems Engineering in Linköping*, pp. 193–202. ECSEL.

Åström, K. and B. Wittenmark (1984): *Computer Controlled Systems: Theory and Design*. Prentice-Hall.