Available online at www.sciencedirect.com
ScienceDirect
Control Engineering Practice 16 (2008) 377–391
CONTROL ENGINEERING PRACTICE
www.elsevier.com/locate/conengprac

Lifted system iterative learning control applied to an industrial robot

W.B.J. Hakvoort[a,*], R.G.K.M. Aarts[b], J. van Dijk[b], J.B. Jonker[b]

[a]*Netherlands Institute for Metals Research, Delft, The Netherlands*
[b]*University of Twente, Enschede, The Netherlands*

Received 17 October 2005; accepted 11 May 2007
Available online 21 June 2007

Abstract

This paper proposes a model-based iterative learning control algorithm for time-varying systems with a high convergence speed. The convergence of components of the tracking error can be controlled individually with the algorithm. The convergence speed of each error component can be maximised unless robustness for noise or unmodelled dynamics is needed. The learning control algorithm is applied to the industrial Stäubli RX90 robot. A linear time-varying model of the robot dynamics is obtained by linearisation of the non-linear dynamic equations. Experiments show that the tracking error of the robot joints can be reduced to the desired level in a few iterations.

Keywords: Learning control; Robot dynamics; Linearisation; Time-varying systems; Singular value decomposition; Convergence analyses

1. Introduction

Laser welding has several advantages over conventional welding, e.g., the high depth-to-width ratio of the weld, the relatively small heat input and the high processing speed. To obtain defect free welds, the laser beam should typically track the weld seam with an accuracy in the order of 0.1 mm (Duley, 1998) at speeds beyond 100 mm/s. The demands on the orientation of the laser beam with respect to the weld seam are in the order of several degrees and not as restrictive as the demands on the linear tracking accuracy. Nevertheless, the required linear tracking accuracy puts high demands on the manipulator that moves the laser beam with respect to the weld seam. The industrial applicability of laser welding will be increased considerably by the use of commercially available six-axes industrial robots, as these robots can access complicated three-dimensional seam geometries. However, using standard industrial controllers the tracking accuracy of these robots appears to be insufficient for laser welding at high speeds. Since the (dynamic) repeatability of the robots is much better than their tracking accuracy it is expected that the tracking performance of these robots can be improved considerably with iterative learning control (ILC).

ILC is a learning control technique for systems making repetitive movements. Each trial a feedforward is computed based on measurements of the error in the previous trials, such that the error converges to a small value. Since the pioneering work of Arimoto, Kawamura, and Miyazaki (1984) a vast amount of applications and implementations of ILC have been proposed. An overview of the work until 1998 is given by Moore (1998).

From the beginning ILC has been used to improve the tracking accuracy of robotic manipulators. Two main approaches can be distinguished. The first approach, applied by, e.g., Arimoto et al. (1984), Arimoto, Nguyen, and Naniwa (2000), Elci, Longman, Phan, Juang, and Ugoletti (2002), Guglielmo and Sadegh (1996), Longman (2000), is to use a feedforward that is proportional to the error in the previous run, its time derivative or its time integral (PID-like ILC). Thus no explicit model of the robot is used to compute the feedforward with this type of ILC. Convergence of the error, for a certain choice of the learning gains, is proven by, e.g., passivity analyses (Arimoto et al., 2000) or Lyaponov analysis (Guglielmo &

*Corresponding author. University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands. Tel.: +31 (0)53 4895442; fax: +31 (0)53 4893631.
E-mail address: w.hakvoort@nimr.nl (W.B.J. Hakvoort).

0967-0661/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2007.05.002

Sadegh, 1996). Although the computation of the feedforward is straightforward, the convergence speed of these algorithms is often limited, i.e., the reduction of the tracking error requires many iterations. The other approach, applied by, e.g., Gunnarsson, Norrlöf, Rahic, and Özbek (2004), Kavli (1993), Norrlöf and Gunnarsson (2002), Poo, Lim, and Ma (1996), is to use some kind of model of the robot dynamics in the ILC algorithm (model-based ILC). An approximate inverse of the dynamics is used to compute the feedforward from the error measured in the previous run. Although the model-based algorithms require a model of the robot, the convergence rate of the error is generally much higher than for the PID-like ILC algorithms.

In this paper the goal is to achieve the tracking error required for laser welding. Furthermore, the number of iterations to achieve this error should be limited, i.e., a high convergence speed is required. For this reason the model-based approach is adopted. The dynamics of an industrial robot are non-linear and since industrial robot feedback controllers are often simple PID-controllers, the closed-loop dynamics are non-linear as well. Using a linear time-invariant (LTI) approximation of the dynamics (Gunnarsson et al., 2004; Kavli, 1993; Norrlöf & Gunnarsson, 2002) or only information on the mass matrix (Poo et al., 1996) in the ILC algorithms results in a converging error, but it is expected that the convergence speed can be increased by including more information on the robot dynamics. Therefore, the well-known non-linear equations of the rigid robot-dynamics are used for model-based ILC in this paper. This model information is used to increase the convergence speed of ILC. To avoid the use of the complicated non-linear model for ILC, the model is linearised for small deviations from a reference trajectory. The resulting linear-time variant (LTV) robot model is combined with the known controller dynamics to obtain an LTV closed-loop dynamic model. These linearised equations of motion were previously used by Arimoto et al. (1984) for convergence analyses of ILC, but to the authors knowledge these equations have not been used for model-based ILC so far.

The ILC algorithm proposed in this paper is formulated in the lifted system description that is very well suited for analyses and design of ILC for LTV systems. In the lifted system description all time samples of a signal are stacked into a single vector and a linear (time-varying) system is represented by the system matrix that maps the input vector to the output vector. Likewise ILC maps the measured error vector to the feedforward vector in the subsequent trial. The term "lifted" is taken from the work of Dijkstra (2004), Tousain (2001). Equivalent formulations were used by many others (Elci et al., 2002; Gunnarsson et al., 2004; Longman, 2000; Norrlöf & Gunnarsson, 2002). The lifted system description is often combined with an optimisation based approach to ILC (Q-ILC). In this approach the applied feedforward minimises a weighted norm of the predicted error and the feedforward (see, e.g., Dijkstra, 2004; Gunnarsson et al., 2004; Kim, Chin, Lee, & Choi, 2000; Norrlöf & Gunnarsson, 2002; Tousain, 2001).

In this paper a slightly different approach is adopted. The singular value decomposition of the system matrix is used. The tracking error is projected on the left singular vectors of the system matrix to decouple the system dynamics. The evolution of each of the projected error components from trial to trial can be controlled individually. It is shown that the control of each error component is a trade-off between convergence speed and robustness for unmodelled dynamics and noise. A learning control algorithm is proposed to maximise convergence speed of each error component unless it has to be reduced to guarantee robustness for noise or unmodelled dynamics. The advantage of the approach in this paper over Q-ILC is that the robustness needed for error components affected by noise or unmodelled dynamics does not limit the convergence speed of the other error components. Kim et al. (2000) used the singular value decomposition of the lifted system matrix to analyse the properties of Q-ILC and to reduce the size of the optimisation problem. However, the singular value decomposition was not used to be able to control each error component individually.

The learning controller that is proposed in this paper is applied to an industrial robot. The error components that are affected by noise or unmodelled robot dynamics are not compensated by ILC, while the remaining components are eliminated in one trial. Experiments show reduction of the tacking error and the convergence speed that result from the application of the learning controller on the joint tracking error of an industrial robot, the Stäubli RX90 robot.

In the first section lifted system ILC is introduced, a learning control algorithm is proposed and its convergence and robustness properties are analysed. In the second section a linearised model of the dynamics of the industrial Stäubli RX90 robot is derived. Experimental results of the application of the proposed learning controller to the Stäubli RX90 robot are presented in the third section.

2. Lifted system ILC

An ILC algorithm based on the lifted system description is proposed in this section. The first subsection introduces the lifted system description. The second subsection discusses lifted system ILC (LSILC) and shows the analogy with conventional feedback control. In the third subsection the system dynamics are decoupled using SVD and a learning algorithm is proposed based on the decoupled system equations. Robustness of the proposed learning algorithm is analysed in the subsequent subsections. The last subsection gives guidelines for tuning the proposed ILC algorithm for specific applications.

2.1. The lifted system description

Consider the following state space representation of a linear time-varying (LTV) system:

$$\begin{aligned} \boldsymbol{x}(l+1) &= \boldsymbol{A}_l\boldsymbol{x}(l) + \boldsymbol{B}_l\boldsymbol{u}(l),\\ \boldsymbol{y}(l) &= \boldsymbol{C}_l\boldsymbol{x}(l) + \boldsymbol{D}_l\boldsymbol{u}(l), \end{aligned} \tag{1}$$

where l is the discrete time index, $\boldsymbol{x}(l)$ is the state vector, $\boldsymbol{u}(l)$ is the input and $\boldsymbol{y}(l)$ the output.

In the lifted system description all samples of a (multi-dimensional) discrete time signal are put into a single column vector. For example, the lifted input vector denoted by $\bar{\boldsymbol{u}}$ is defined as

$$\bar{\boldsymbol{u}} = [\boldsymbol{u}(1)^{\mathrm{T}}, \boldsymbol{u}(2)^{\mathrm{T}}, \ldots, \boldsymbol{u}(N)^{\mathrm{T}}]^{\mathrm{T}}, \tag{2}$$

where N is the number of samples. The lifted representation of the LTV system in Eq. (1) is

$$\bar{\boldsymbol{y}} = \bar{\boldsymbol{H}}\bar{\boldsymbol{u}} + \bar{\boldsymbol{y}}_0, \tag{3}$$

where the system matrix $\bar{\boldsymbol{H}}$ and the effect of the initial states $\bar{\boldsymbol{y}}_0$ are given by

$$\bar{\boldsymbol{H}} = \begin{bmatrix} \boldsymbol{D}_1 & 0 & \cdots & 0 \\ \boldsymbol{C}_2\boldsymbol{B}_1 & \boldsymbol{D}_2 & \ddots & 0 \\ \boldsymbol{C}_3\boldsymbol{A}_2\boldsymbol{B}_1 & \boldsymbol{C}_3\boldsymbol{B}_2 & \ddots & 0 \\ \boldsymbol{C}_4\boldsymbol{A}_3\boldsymbol{A}_2\boldsymbol{B}_1 & \boldsymbol{C}_4\boldsymbol{A}_3\boldsymbol{B}_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{C}_N \prod_{i=N-1}^{2} \boldsymbol{A}_i\boldsymbol{B}_1 & \boldsymbol{C}_N \prod_{i=N-1}^{3} \boldsymbol{A}_i\boldsymbol{B}_2 & \cdots & \boldsymbol{D}_N \end{bmatrix}$$

$$\bar{\boldsymbol{y}}_0 = \begin{bmatrix} \boldsymbol{C}_1\boldsymbol{A}_0 \\ \boldsymbol{C}_2\boldsymbol{A}_1\boldsymbol{A}_0 \\ \boldsymbol{C}_3\boldsymbol{A}_2\boldsymbol{A}_1\boldsymbol{A}_0 \\ \vdots \\ \boldsymbol{C}_N \prod_{i=N-1}^{0} \boldsymbol{A}_i \end{bmatrix} \boldsymbol{x}(0). \tag{4}$$

The system matrix $\bar{\boldsymbol{H}}$ describes the effect of the lifted input $\bar{\boldsymbol{u}}$ on the lifted output $\bar{\boldsymbol{y}}$. Element $\bar{\boldsymbol{H}}(i,j)$ of matrix $\bar{\boldsymbol{H}}$ is the response of $\boldsymbol{y}$ at time i to an impulse of $\boldsymbol{u}$ at time j. Note that the lifted vectors and the lifted system matrix are both denoted with an overbar.

The tracking error $\bar{\boldsymbol{e}}$ is defined as

$$\bar{\boldsymbol{e}} = \bar{\boldsymbol{r}} - \bar{\boldsymbol{y}} + \bar{\boldsymbol{v}} = \bar{\boldsymbol{r}} - \bar{\boldsymbol{H}}\bar{\boldsymbol{u}} - \bar{\boldsymbol{y}}_0 + \bar{\boldsymbol{v}}, \tag{5}$$

where $\bar{\boldsymbol{r}}$ denotes the lifted reference and $\bar{\boldsymbol{v}}$ denotes the effect of all disturbances on the output.

2.2. Lifted system ILC

Suppose the system described in the previous section makes repetitive movements (trials) and the reference, the disturbance and the initial states are the same for each run. Using superscript k to denote the trial number, Eq. (5) can be rewritten as

$$\bar{\boldsymbol{e}}^k = \bar{\boldsymbol{d}} - \bar{\boldsymbol{H}}\bar{\boldsymbol{u}}^k, \tag{6}$$

where $\bar{\boldsymbol{d}}$ is defined as

$$\bar{\boldsymbol{d}} = \bar{\boldsymbol{r}} - \bar{\boldsymbol{y}}_0 + \bar{\boldsymbol{v}}, \tag{7}$$

and contains all effects that are assumed to be constant for each trial. Note that superscript k is omitted for $\bar{\boldsymbol{d}}$ since $\bar{\boldsymbol{d}}$ is trial independent.

ILC compensates for the effect of the trial independent disturbance $\bar{\boldsymbol{d}}$ on the error $\bar{\boldsymbol{e}}^k$ by an input $\bar{\boldsymbol{u}}^k$ computed from measurements of the error in the previous run(s). The ILC design problem is to find a suitable update equation that computes an input from the errors in the previous run(s). As shown by Dijkstra (2004) ILC design can be written in a form analogous to conventional feedback controller design using the concepts of the *trial delay operator* and the *trial domain*. The *trial delay operator* $\mathscr{Z}^{-1}$ is defined by

$$\bar{\boldsymbol{u}}^k = \mathscr{Z}^{-1}\bar{\boldsymbol{u}}^{k+1}. \tag{8}$$

The *trial delay* operator is the discrete delay operator in the *trial domain*, in contrast to the *time delay operator* z^{-1} in the *time domain* that is usually considered for systems analysis and design.

With the definition of the trial domain and trial delay operator, the ILC update law can be written as

$$\bar{\boldsymbol{u}}^k = \bar{\mathscr{L}}(\mathscr{Z})\bar{\boldsymbol{e}}^k, \tag{9}$$

where $\bar{\mathscr{L}}(\mathscr{Z})$ is a trial domain transfer function, referred to as the learning function. Eq. (9) and (6) form a closed-loop system in the trial domain as depicted in Fig. 1.

The LSILC design problem can thus be formulated as the design of a trial domain feedback controller $\bar{\mathscr{L}}(\mathscr{Z})$ that rejects the effect of disturbance $\bar{\boldsymbol{d}}$ on the error $\bar{\boldsymbol{e}}$. The internal model principle states that in order to reject a signal of a specific class, the feedback loop should at least include a model of a generator of that signal class. The disturbance $\bar{\boldsymbol{d}}$ is trial-independent, i.e., constant in the trial domain. A constant signal can be generated by an integrator. Therefore, a trial domain integrator is taken

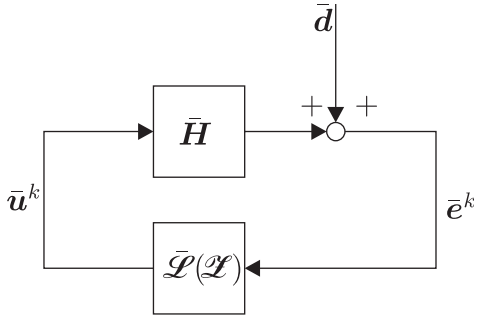

Fig. 1. Closed-loop system.

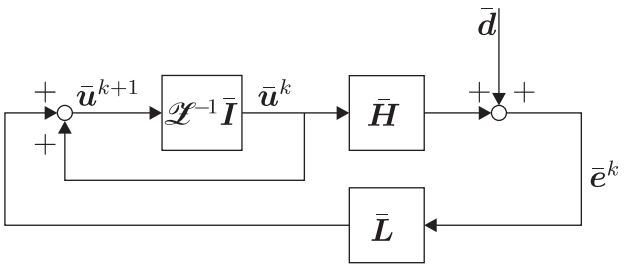

Fig. 2. Closed-loop system, integrating update law.

for $\mathscr{L}(\mathscr{Z})$, resulting in the update equation

$$\bar{u}^k = \frac{1}{\mathscr{Z}-1}\bar{L}\bar{e}^k, \tag{10}$$

or in a more conventional notation

$$\bar{u}^{k+1} = \bar{u}^k + \bar{L}\bar{e}^k, \tag{11}$$

where $\bar{L}$ is the learning matrix. The resulting trial domain feedback system is depicted in Fig. 2. The learning matrix is analogous to the integral gain in integral feedback control. An implementation of the learning matrix is proposed in the next section. Combination of the update Eq. (11) with the system Eq. (6) gives the equation for the propagation of the error

$$\bar{e}^{k+1} = (\bar{I} - \bar{H}\bar{L})\bar{e}^k. \tag{12}$$

Monotonic convergence of the error is guaranteed if

$$\sigma^*(\bar{I} - \bar{H}\bar{L}) < 1, \tag{13}$$

where the operator σ^* returns the largest singular value (Longman, 2000).

2.3. Decoupling

In this section an implementation of the learning matrix $\bar{L}$ is proposed. The learning matrix is based on the decoupled system equations that are formulated using the singular value decomposition of the system matrix $\bar{H}$, denoted by

$$\bar{H} = \bar{U}\bar{\Sigma}\bar{V}^{\mathrm{T}}, \tag{14}$$

where $\bar{\Sigma}$ is a diagonal matrix with the singular values $\sigma(i)$ on its diagonal, sorted such that $\sigma(i) \geqslant \sigma(i+1)$. Matrices $\bar{U}$ and $\bar{V}$ are orthogonal matrices containing the left and right singular vectors of the system matrix. Since these matrices are orthogonal they can be used as a basis for projection of the inputs, errors and disturbances

$$\bar{u}_d = \bar{V}^{\mathrm{T}}\bar{u}, \quad \bar{e}_d = \bar{U}^{\mathrm{T}}\bar{e}, \quad \bar{d}_d = \bar{U}^{\mathrm{T}}\bar{d}. \tag{15}$$

The elements of $\bar{u}_d$, $\bar{e}_d$ and $\bar{d}_d$ are referred to as the input components, the error components and the disturbance components, respectively, in the remainder of this paper. Using Eq. (14), the definitions (15) and the orthogonality property of $\bar{U}$ and $\bar{V}$, the system equation (6) can be rewritten to the following decoupled system equation:

$$\bar{e}_d^k(i) = \bar{d}_d - \sigma(i)\bar{u}_d^k(i). \tag{16}$$

Note that the error components corresponding to $\sigma(i) = 0$ cannot be affected by the input.

In line with Section 2.2 an integrating ILC update law is proposed for each input component

$$\bar{u}_d^{k+1}(i) = \bar{u}_d^k(i) + \lambda(i)\bar{e}_d^k(i), \tag{17}$$

where $\lambda(i)$ is the integral gain. For further discussion it is useful to define the learning gains $\kappa(i)$ as

$$\kappa(i) = \lambda(i)\sigma(i). \tag{18}$$

The gains $\kappa(i)$ determine the convergence and robustness properties of the learning algorithm. The effect of the value of $\kappa(i)$ on the convergence and robustness is analysed hereafter. The analysis is concluded with guidelines for the choice of $\kappa(i)$ in Section 2.6.

Using the definitions (15) it can be shown that the update law in Eq. (17) is equivalent to the update law in Eq. (11) where the learning matrix is of the form

$$\bar{L} = \bar{V}\,\mathrm{diag}(\lambda(1)\ldots\lambda(N))\bar{U}^{\mathrm{T}}. \tag{19}$$

Combining update Eq. (17) with Eq. (16) gives the decoupled equation for the propagation of the error:

$$\bar{e}_d^{k+1}(i) = (1 - \kappa(i))\bar{e}_d^k(i). \tag{20}$$

The propagation of the error components can thus be controlled individually by the value of $\kappa(i)$. The error component $\bar{e}_d(i)$ converges to zero if $0 < \kappa(i) < 2$. Taking $\kappa(i) = 1$ results in maximum convergence (analogous to dead-beat control for discrete time systems). Note that this is only possible if $\sigma(i) \neq 0$. The error components $\bar{e}^d(i)$ corresponding to $\sigma(i) = 0$ cannot be changed by any input, therefore, $\lambda(i) = \kappa(i) = 0$ for these error components to minimise the corresponding input component.

2.4. Effect of noise

In the previous subsections it was assumed that the reference, the disturbances and the effect of the initial states were trial independent. All these effects were contained in the trial independent disturbance vector $\bar{d}$. It was argued that the effect of a trial independent disturbance vector on the error vector can be rejected using the integrating update law of equation (11). In practical applications the disturbance or the initial states are often partly trial dependent due to noise. This subsection analyses the effect of a trial dependent disturbance vector on the error.

The effect of a trial dependent disturbance vector is analysed by writing the system equation (6) as

$$\bar{e}^k = \bar{d}^k - \bar{H}\bar{u}^k, \tag{21}$$

where all lifted vectors, including $\bar{d}^k$, are assumed to be trial dependent. Substitution of update equation (10) in the system equation gives

$$\left(\bar{I} - \frac{1}{\mathscr{Z}-1}\bar{H}\bar{L}\right)\bar{e}^k = \bar{d}^k. \tag{22}$$

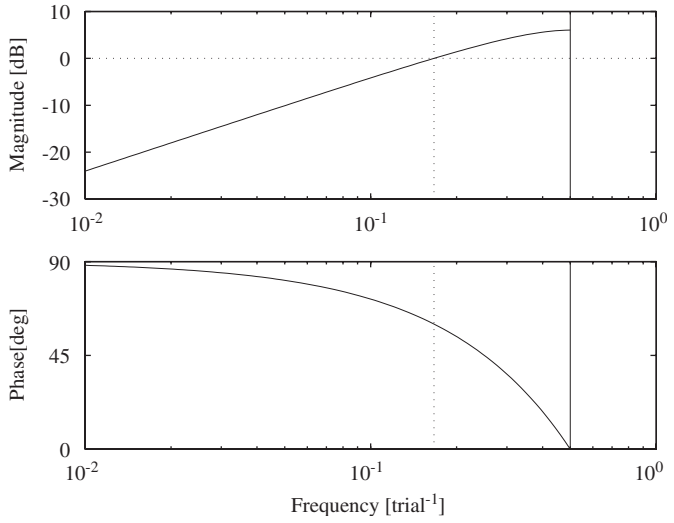

Fig. 3. Trial domain Bode-plot of $(1-\mathscr{Z}^{-1})$.

Substitution of the proposed learning matrix, given in Eq. (19), the singular value decomposition of the system matrix (14) and the definitions (15) gives

$$\bar{\boldsymbol{e}}_d^k(i)=\frac{\mathscr{Z}-1}{\mathscr{Z}-1+\kappa(i)}\bar{\boldsymbol{d}}_d^k(i). \tag{23}$$

This equation relates the error components to the disturbance components. Trivially $\kappa(i)=0$ results in $\bar{\boldsymbol{e}}_d^k(i)=\bar{\boldsymbol{d}}_d^k(i)$. For $\kappa(i)=1$ Eq. (23) becomes

$$\bar{\boldsymbol{e}}_d^k(i)=(1-\mathscr{Z}^{-1})\bar{\boldsymbol{d}}_d^k(i). \tag{24}$$

In this case a trial independent disturbance, i.e., $\bar{\boldsymbol{d}}_d^k(i)=\mathscr{Z}^{-1}\bar{\boldsymbol{d}}_d^k(i)$, results in $\bar{\boldsymbol{e}}_d(i)=0$. The effect of trial dependent disturbances is visualised by the Bode-plot of $(1-\mathscr{Z}^{-1})$, shown in Fig. 3. As $\mathscr{Z}^{-1}$ is a delay in the trial-domain, the frequency-axis requires a special interpretation. The frequency-axis of the Bode-plot corresponds to a time scale expressed in trials, so the frequency specifies how often a signal changes per trial. The zero frequency corresponds to a trial-independent signal and the Nyquist-frequency (at $\frac{1}{2}$ trial^{-1}) to a signal that changes sign each trial. Fig. 3 shows that $\kappa(i)=1$ attenuates the effect of disturbances that change less than $\frac{1}{6}$ trial^{-1} (e.g., slowly varying temperature effects) but amplifies the effect of signals that change more often. The worst case is a disturbance that change sign each trial, the effect of these disturbances is doubled. It can be shown that disturbances that change more than $\frac{1}{6}$ trial^{-1} are amplified less for $0\leqslant\kappa(i)<1$. Thus $\kappa(i)$ should be chosen smaller than 1 for the error components that are caused by disturbances that change more than $\frac{1}{6}$ trial^{-1}.

2.5. Effect of modelling errors

In the previous subsections it was assumed that a perfect model of the system is available to construct a learning matrix that decouples the propagation of the error components. Suppose that a multiplicative error exists between the modelled system matrix $\bar{\boldsymbol{H}}$ and the dynamics of the real system $\bar{\mathscr{H}}$, i.e.,

$$\bar{\mathscr{H}}=(\bar{\boldsymbol{I}}+\bar{\boldsymbol{\Delta}})\bar{\boldsymbol{H}}. \tag{25}$$

Substitution in Eq. (12) gives

$$\bar{\boldsymbol{e}}^{k+1}=(\bar{\boldsymbol{I}}-\bar{\mathscr{H}}\bar{\boldsymbol{L}})\bar{\boldsymbol{e}}^k=(\bar{\boldsymbol{I}}-(\bar{\boldsymbol{I}}+\bar{\boldsymbol{\Delta}})\bar{\boldsymbol{H}}\bar{\boldsymbol{L}})\bar{\boldsymbol{e}}^k. \tag{26}$$

Substitution of the proposed learning matrix, given in Eq. (19), the singular value decomposition of the system matrix (14) and the definitions (15) gives

$$\begin{aligned}\bar{\boldsymbol{e}}_d^{k+1}&=(\bar{\boldsymbol{I}}-\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{\Lambda}})\bar{\boldsymbol{e}}_d^k+\bar{\boldsymbol{U}}^{\mathrm{T}}\bar{\boldsymbol{\Delta}}\bar{\boldsymbol{U}}\bar{\boldsymbol{\Sigma}}\bar{\boldsymbol{\Lambda}}\bar{\boldsymbol{e}}_d^k\\&=(\bar{\boldsymbol{I}}-\bar{\boldsymbol{K}})\bar{\boldsymbol{e}}_d^k+\bar{\boldsymbol{U}}^{\mathrm{T}}\Delta\bar{\boldsymbol{U}}\bar{\boldsymbol{K}}\bar{\boldsymbol{e}}_d^k,\end{aligned} \tag{27}$$

where

$$\bar{\boldsymbol{K}}=\mathrm{diag}(\kappa(1)\ldots\kappa(N)). \tag{28}$$

In general $\bar{\boldsymbol{U}}^{\mathrm{T}}\bar{\boldsymbol{\Delta}}\bar{\boldsymbol{U}}$ is not diagonal, so the propagation of the error components is coupled by the model error. Note that the first term in Eq. (27) represents the propagation of the error for the nominal model and the second term is the contribution by the model error. The contribution of an error component to the second term is reduced if the corresponding value of $\kappa(i)$ close to zero. However, a small value of $\kappa(i)$ decreases the (nominal) convergence of the error component due to the first term. Thus the larger the model error, the smaller the (nominal) convergence. As in Eq. (13) the condition for monotonic convergence is

$$\sigma^*(\bar{\boldsymbol{I}}-\bar{\boldsymbol{K}}+\bar{\boldsymbol{U}}^{\mathrm{T}}\Delta\bar{\boldsymbol{U}}\bar{\boldsymbol{K}})<1. \tag{29}$$

2.6. Guidelines for tuning $\kappa(i)$

In Section 2.3 a learning matrix is proposed that decouples the propagation of the error components. The propagation of the error components can be controlled individually by the choice of $\kappa(i)$. Robustness of the proposed learning matrix is analysed in the previous subsections. From the convergence and robustness properties the following guidelines for tuning $\kappa(i)$ can be given:

- Convergence of $\bar{\boldsymbol{e}}_d(i)$ requires $0<\kappa(i)<2$, where $\kappa(i)=1$ gives maximum convergence of the error component (Section 2.3).
- Error components corresponding to $\sigma(i)=0$ cannot be controlled. The corresponding input components are minimised by $\kappa(i)=0$ (Section 2.3).
- The effect of disturbance components that changes more than $\frac{1}{6}$ trial^{-1} is amplified for $\kappa(i)=1$. The amplification is reduced for $0<\kappa(i)<1$ (Section 2.4).
- The effect of error components $\bar{\boldsymbol{e}}_d(i)$ on the growth of the error due to unmodelled dynamics is reduced by taking $\kappa(i)$ close to zero (Section 2.5).

The choice of $\kappa(i)$ thus depends on several properties of controlled system. In Section 4.2 the values of $\kappa(i)$ are tuned for the Stäubli RX90 robot.

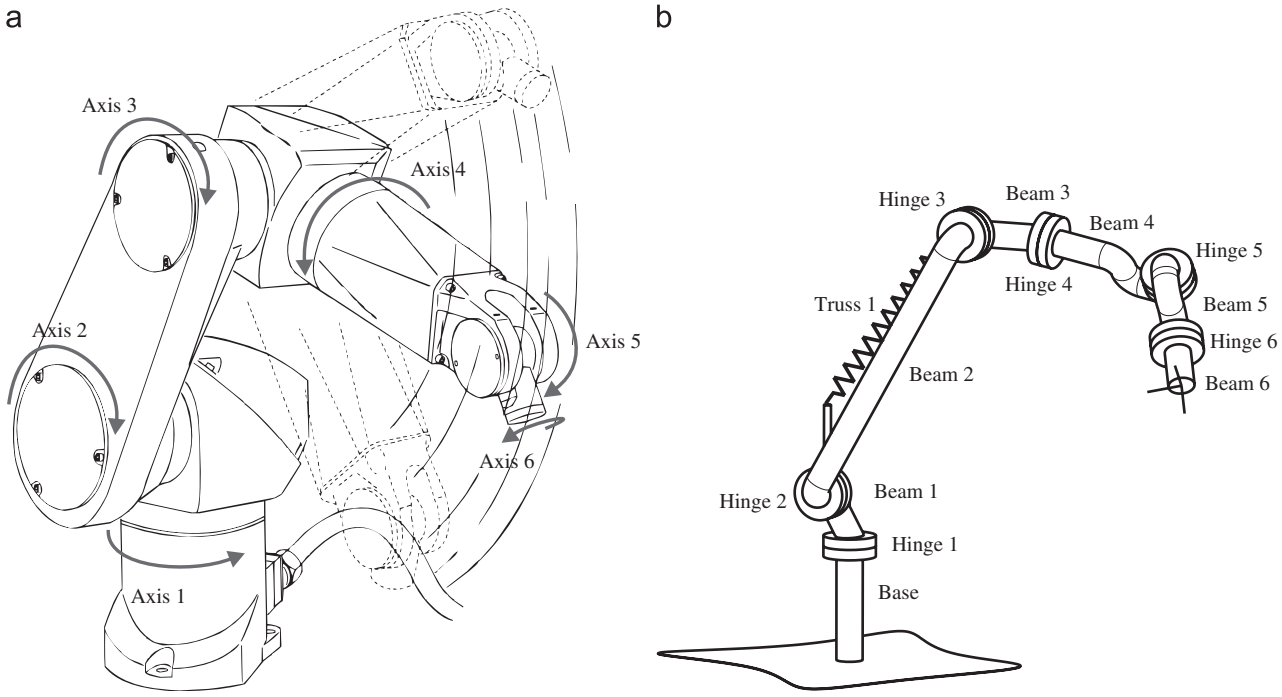

Fig. 4. (a) Stäubli RX90 robot and its (b) finite element model.

3. Closed-loop robot model

In this section the closed-loop dynamic equations of the Stäubli RX90 robot are derived to be able to apply the ILC algorithm proposed in Section 2 to the robot.

In the first subsection the non-linear equations of motions of the robot manipulator are formulated. For small perturbations around a nominal trajectory, these equations can be approximated by the linearised equations of motion, which are derived in the second subsection. In the third subsection the linearised equations of motion are combined with a model of the industrial CS8-controller to obtain a model of the closed-loop system. Finally the model is used to formulate the closed-loop dynamic equations in a form suitable for LSILC.

3.1. Non-linear equations of motion

Fig. 4(a) shows the six-axes Stäubli RX90 robot. A non-linear finite element method is used to formulate the dynamic equations of the robot mechanism. The finite element model is illustrated in Fig. 4(b). The manipulator arms are modelled by beam elements. The beam elements are interconnected with cylindrical hinge elements, representing the joints of the manipulator. Finally the robot is equipped with a gravity compensation spring that is modelled as a slider-truss element. For a detailed description of the finite element model of the Stäubli RX90 robot the reader is referred to Waiboer, Aarts, and Jonker (2005a). The model yields the following well-known equations of motion for the robot manipulator:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \boldsymbol{\tau}, \tag{30}$$

where $\boldsymbol{q}$ are the joint angles, $\boldsymbol{M}(\boldsymbol{q})$ is the configuration dependent mass matrix of the robot, $\boldsymbol{\tau}$ are the joint driving torques and $\boldsymbol{N}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ describes the effects of the velocity dependent virtual inertia forces, the gravity force and the gravity compensation spring.

The robot is driven by servo motors that are connected to the robot joints via gear transmissions. The rotation of the servo motors $\boldsymbol{\phi}$ is related to the joint angles $\boldsymbol{q}$ by the transmission matrix $\boldsymbol{T}$:

$$\boldsymbol{\phi} = \boldsymbol{T}\boldsymbol{q}. \tag{31}$$

The drives are equipped with permanent magnet, three-phase synchronous motors, yielding a linear relation between motor current and motor torque.

The joint driving torques are expressed by

$$\boldsymbol{\tau} = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{(m)}\boldsymbol{i} - \boldsymbol{T}^{\mathrm{T}}\boldsymbol{J}^{(m)}\ddot{\boldsymbol{\phi}} - \boldsymbol{\tau}^{(f)}, \tag{32}$$

where $\boldsymbol{C}^{(m)}$ is a diagonal matrix with the motor constants and $\boldsymbol{J}^{(m)}$ is a diagonal matrix with rotor inertias. The friction torque $\boldsymbol{\tau}^{(f)}$ is described by a friction model that relies on insights from tribological models. The friction model consists of a viscous part $\boldsymbol{\tau}^{(f,v)}$ and a Stribeck part $\boldsymbol{\tau}^{(f,a)}$ resulting from the asperity contacts inside the gears and roller bearings of the robot joints. The pre-sliding regime is not taken into account, because the effect is expected to be small for the high joint velocities considered in this paper. Both the viscous and the asperity part of the friction torque are described as non-linear functions of the joint velocities, i.e.,

$$\boldsymbol{\tau}^{(f)} = \boldsymbol{\tau}^{(f,v)}(\dot{\boldsymbol{q}}) + \boldsymbol{\tau}^{(f,a)}(\dot{\boldsymbol{q}}). \tag{33}$$

For a detailed description of the friction model the reader is referred to Waiboer, Aarts, and Jonker (2005b).

Combining the models of the robot mechanism in Eq. (30), the drives in Eq. (32) and the friction in Eq. (33) gives

$$\boldsymbol{M}^{(n)}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\dot{\boldsymbol{q}},\boldsymbol{q}) + \boldsymbol{\tau}^{(f,v)}(\dot{\boldsymbol{q}}) + \boldsymbol{\tau}^{(f,a)}(\dot{\boldsymbol{q}}) = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{(m)}\boldsymbol{i}, \tag{34}$$

where

$$\boldsymbol{M}^{(n)}(\boldsymbol{q}) = \boldsymbol{M}(\boldsymbol{q}) + \boldsymbol{T}^{\mathrm{T}}\boldsymbol{J}^{(m)}\boldsymbol{T}. \tag{35}$$

The non-linear robot model above relates the motor current $\boldsymbol{i}$ to the joint motion $(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$. The inertia properties of the links, parameters of the gravitation compensation spring and the friction parameters have been estimated with parameter identification (Waiboer et al., 2005a). The links of the robot are considered rigid and also flexibility in the gear transmission and the joints are not modelled.

3.2. Linearised equations of motion

The non-linear dynamic equations of the robot (34) are approximated by linear time variant equations using the perturbation method. In this method, deviations from the nominal motion $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0,\ddot{\boldsymbol{q}}_0)$ are modelled as first-order perturbations $(\delta\boldsymbol{q}_0,\delta\dot{\boldsymbol{q}}_0,\delta\ddot{\boldsymbol{q}}_0)$ of the nominal motion, such that the actual motion is of the form:

$$\boldsymbol{q} = \boldsymbol{q}_0 + \delta\boldsymbol{q}, \tag{36a}$$

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_0 + \delta\dot{\boldsymbol{q}}, \tag{36b}$$

$$\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_0 + \delta\ddot{\boldsymbol{q}}, \tag{36c}$$

where the prefix δ denotes a perturbation and the subscript 0 refers to the (desired) nominal trajectory. The actual motor current is of the form:

$$\boldsymbol{i} = \boldsymbol{i}_0 + \delta\boldsymbol{i}, \tag{37}$$

where the nominal motor current $\boldsymbol{i}_0$ is defined as the motor current required to move the manipulator model along the nominal trajectory $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0,\ddot{\boldsymbol{q}}_0)$ and is computed by substituting the nominal trajectory in Eq. (34).

LTV equations are obtained by linearising the non-linear equations off motion (34) around the nominal trajectory. The resulting equations of motion for the perturbation of the joint motion $\delta\boldsymbol{q}$ are

$$\boldsymbol{M}_0\delta\ddot{\boldsymbol{q}} + \boldsymbol{C}_0\delta\dot{\boldsymbol{q}} + \boldsymbol{K}_0\delta\boldsymbol{q} = \delta\boldsymbol{\tau} - \delta\boldsymbol{\tau}^{(f,a)}, \tag{38}$$

where

$$\delta\boldsymbol{\tau} = \boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{(m)}\delta\boldsymbol{i}, \tag{39}$$

$$\delta\boldsymbol{\tau}^{(f,a)} = (\boldsymbol{\tau}^{(f,a)}(\dot{\boldsymbol{q}}) - \boldsymbol{\tau}^{(f,a)}(\dot{\boldsymbol{q}}_0)), \tag{40}$$

$\boldsymbol{M}_0$ is the mass matrix, $\boldsymbol{C}_0$ is the velocity sensitivity matrix and $\boldsymbol{K}_0$ is the stiffness matrix. The mass matrix $\boldsymbol{M}_0$ is equal to $\boldsymbol{M}^{(n)}(\boldsymbol{q}_0)$ in Eq. (35). Matrix $\boldsymbol{C}_0$ includes the effect of the viscous friction and the centrifugal and coriolis effects. Matrix $\boldsymbol{K}_0$ includes the dynamic and geometric stiffness effects and the structural stiffness of the gravity compensation spring. $\boldsymbol{M}_0$, $\boldsymbol{C}_0$ and $\boldsymbol{K}_0$ depend on the nominal position and velocity $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0)$ in the linearisation points. Since the nominal position and velocity are known as a function of time, these matrices can be considered time dependent. The friction torques due to asperity contacts $\boldsymbol{\tau}^{(f,a)}$ are not linearised as these are discontinuous around zero velocity, they will be considered as a disturbance for control. For a detailed derivation of the linearised equations of motion the reader is referred to Jonker and Aarts (2001).

The linearised equations of motion (38) can be transformed to the linearised state differential equation of motion

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_c(t)\boldsymbol{x}(t) + \boldsymbol{B}_c(t)(\delta\boldsymbol{\tau} - \delta\boldsymbol{\tau}^{(f,a)}), \tag{41}$$

where

$$\boldsymbol{x}(t) = \begin{bmatrix} \delta\boldsymbol{q}(t) \\ \delta\dot{\boldsymbol{q}}(t) \end{bmatrix}, \tag{42}$$

$$\boldsymbol{A}_c(t) = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I} \\ -\boldsymbol{M}_0^{-1}\boldsymbol{K}_0 & -\boldsymbol{M}_0^{-1}\boldsymbol{C}_0 \end{bmatrix}, \tag{43}$$

$$\boldsymbol{B}_c(t) = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{M}^{-1} \end{bmatrix}. \tag{44}$$

Discretisation results in the following discrete state difference equation

$$\boldsymbol{x}(l+1) = \boldsymbol{A}_d(l)\boldsymbol{x}(l) + \boldsymbol{B}_d(l)\big(\delta\boldsymbol{\tau} - \delta\boldsymbol{\tau}^{(f,a)}\big), \tag{45}$$

where

$$\boldsymbol{A}_d(l) = \exp(\boldsymbol{A}_c(l)T_v), \tag{46}$$

$$\boldsymbol{B}_d(l) = \left(\int_0^{T_v} \exp(\boldsymbol{A}_c(l)\eta)\,\mathrm{d}\eta\right)\boldsymbol{B}_d(l), \tag{47}$$

l is the discrete time index and T_v is the sample time. The transfer function representation of the discrete state difference Eq. (45) is denoted by $\boldsymbol{R}(z,l)$.

3.3. Closed-loop model

The Stäubli RX90 robot is controlled by the industrial CS8-controlled. The joints of the robot are controlled independently by six joint controllers. Apart from the integral gains, which are set to zero, the native joint controllers are used for this work. Conventionally position and velocity setpoints for the six joint controllers are generated by the trajectory generator of the CS8 controller. For this work Stäubli provided a tool to send setpoints for the position, velocity, torque feedforward and velocity feedforward to the joint controllers directly. With this tool the torque feedforward from the ILC algorithm can be combined with the standard position and velocity setpoints.

A block scheme of the closed-loop model is shown in Fig. 5. The dotted lines outline the perturbation model of the robot. The input to the robot model is the motor current $\boldsymbol{i}$. The nominal motor current $\boldsymbol{i}_0$ is subtracted from the motor current to obtain the perturbed motor current $\delta\boldsymbol{i}$.

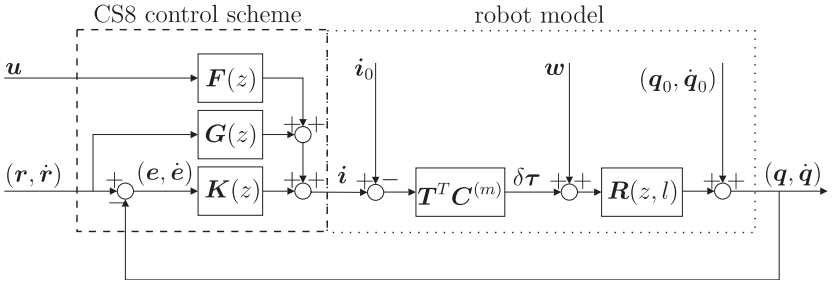

Fig. 5. Closed-loop block scheme.

Multiplication of the motor current with $\boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{(m)}$ gives the perturbed torque $\delta\boldsymbol{\tau}$. A disturbance $\boldsymbol{w}$ is added to the torque vector. This disturbance includes the asperity part of the friction torque $\delta\boldsymbol{\tau}^{(f,a)}$ but could also include other torque disturbances. The sum of the perturbed torque and the disturbance is the input of the LTV robot model $\boldsymbol{R}(z,l)$. The output of $\boldsymbol{R}(z,l)$ is the perturbed motion $(\delta\boldsymbol{q},\delta\dot{\boldsymbol{q}})$. The nominal trajectory $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0)$ is added to the perturbed motion to obtain the actual trajectory $(\boldsymbol{q},\dot{\boldsymbol{q}})$.

A slightly simplified model of the joint controllers of the CS8-controller is outlined by the dashed lines. The joint controllers consist of three parts, the feedback part, the feedforward part and a low-pass filter, represented by the transfer functions $\boldsymbol{K}(z)$, $\boldsymbol{G}(z)$ and $\boldsymbol{F}(z)$, respectively. The feedback part $\boldsymbol{K}(z)$ is a PID-controller. The input is the tracking error $(\boldsymbol{e},\dot{\boldsymbol{e}})$, which is the difference between the actual trajectory $(\boldsymbol{q},\dot{\boldsymbol{q}})$ and the reference trajectory $(\boldsymbol{r},\dot{\boldsymbol{r}})$. The feedforward part $\boldsymbol{G}(z)$ generates a velocity and acceleration feedforward current. The low-pass filter $\boldsymbol{F}(z)$ filters the feedforward $\boldsymbol{u}(l)$. The sum of the outputs of the controller parts $\boldsymbol{K}(z)$, $\boldsymbol{G}(z)$ and $\boldsymbol{F}(z)$ is the motor current.

The nominal trajectory $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0)$, used for the linearisation of the robot model, is taken equal to the reference trajectory $(\boldsymbol{r},\dot{\boldsymbol{r}})$, because the robot moves close to this reference trajectory.

The feedforward $\boldsymbol{u}$ will be used to reduce the joint tracking error $\boldsymbol{e}$ with LSILC. Therefore, the transfer function from feedforward to the tracking error has to be known. This relation is derived from the block scheme and can be written as

$$\boldsymbol{e} = \boldsymbol{v} - \boldsymbol{H}(z,l)\boldsymbol{u}, \tag{48}$$

where

$$\boldsymbol{H}(z,l) = [\boldsymbol{I}\,\boldsymbol{0}](\boldsymbol{I} + \boldsymbol{R}(z,l)\boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{m}\boldsymbol{K}(z))^{-1}\boldsymbol{R}(z,l)\boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{m}\boldsymbol{F}(z), \tag{49}$$

$$\begin{aligned}\boldsymbol{v} = & -[\boldsymbol{I}\,\boldsymbol{0}](\boldsymbol{I} + \boldsymbol{R}(z,l)\boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{m}\boldsymbol{K}(z))^{-1}\boldsymbol{R}(z,l)\\ & \times(\boldsymbol{w} + \boldsymbol{T}^{\mathrm{T}}\boldsymbol{C}^{m}(\boldsymbol{i}_0 + \boldsymbol{G}(z)[\boldsymbol{r}\,\dot{\boldsymbol{r}}]^{\mathrm{T}})).\end{aligned} \tag{50}$$

The lifted representation of $\boldsymbol{H}(z,l)$ is the system matrix that is used to implement LSILC for the Stäubli RX90 robot. The experimental results will be shown in the next section.

Finally it is mentioned that the integral gain of the feedback controller $\boldsymbol{K}(z)$ is set to zero during the experiments with LSILC. A non-zero integral gain would reduce the low-frequency gain of $\boldsymbol{H}(z,l)$ and hence large feedforwards would be required to compensate for low-frequency errors.

4. Experimental results

This section shows the results of the application of the LSILC algorithm proposed in Section 2 to the Stäubli RX90 robot. The robot model of Section 3 is used for the implementation of the learning controller.

The first subsection presents the reference trajectory that is used for the experiments. The robot is moved along this trajectory repeatedly to test the effect of LSILC. In the first run no feedforward is applied and in the subsequent runs the feedforward is updated according to Eq. (11). The computation of the learning matrix is discussed in the second subsection. The experimental results are presented in the last subsection.

4.1. Reference trajectory

The goal of the application of ILC to the Stäubli RX90 robot is to be able to use the robot for laser welding. Therefore, a weld seam trajectory is used to test the implementation of LSILC. Fig. 6(a) shows the weld seam that should be tracked by the focus of the laser welding head attached to the robot. The welding head is held in a fixed orientation relative to the trajectory, so it has to turn in the curved part of the trajectory. The velocity profile is trapezoidal with a maximum velocity of 350 mm/s that is reached in 0.02 s. The duration of the motion is 2.044 s (511 samples of 4 ms), including 0.5 s at the start and the end of the trajectory where $\dot{\boldsymbol{q}}_0 = 0$. The reference for the joint motions is computed from the weld seam geometry, the kinematic robot model and the geometry of the welding head and is shown in Fig. 6(b).

4.2. LSILC implementation for the Stäubli RX90 robot

The feedforwards for the robot are updated according to Eq. (11). In Section 2.3 a learning matrix is proposed that decouples the propagation of the error components. Computation of this learning matrix requires the singular values and vectors of the system matrix and the values of $\kappa(i)$.

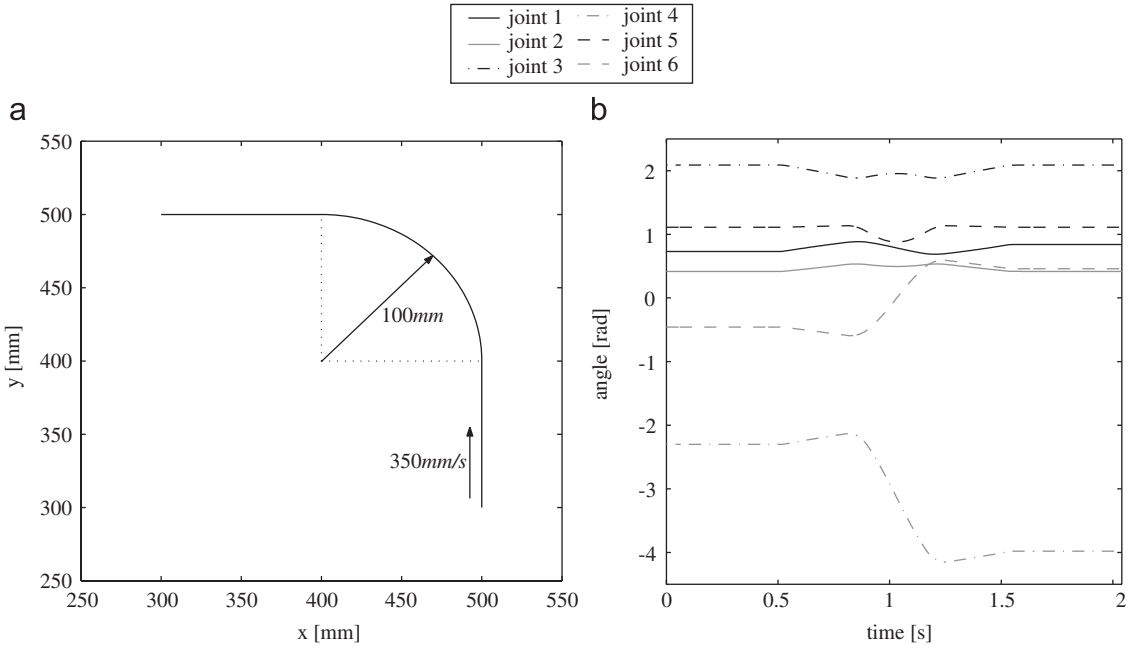

Fig. 6. Reference trajectory: (a) world coordinates, (b) joint coordinates.

The singular values and vectors of the system matrix are obtained in three steps. First the closed-loop dynamic Eq. (48) are derived for small motions around the reference trajectory of Section 4.1. Secondly the LTV model is rewritten in the lifted system representation resulting in

$$\bar{\boldsymbol{e}}^k = \bar{\boldsymbol{d}}^k - \bar{\boldsymbol{H}}\bar{\boldsymbol{u}}^k, \tag{51}$$

where $\bar{\boldsymbol{H}}$ is the lifted system matrix and $\bar{\boldsymbol{e}}^k$, $\bar{\boldsymbol{d}}^k$ and $\bar{\boldsymbol{u}}^k$ are, respectively, the lifted error, the lifted disturbance and the lifted feedforward in trial k. The lifted disturbance vector $\bar{\boldsymbol{d}}^k$ contains the effect of the disturbance $\boldsymbol{v}$ and the initial states on the error. The dimension of the lifted vectors is 3066, equal to the number of samples of the trajectory (511) times the number of joints (6). Finally numerical SVD analyses yield the singular values and vectors of the system matrix.

The values of $\kappa(i)$ are tuned according to the guidelines given in Section 2.6. The values of $\kappa(i)$ corresponding to $\sigma(i) = 0$ are set to zero and the other values of $\kappa(i)$ depend on the expected non-repetitive disturbances and model errors. Eq. (50) shows that the disturbance $\boldsymbol{v}$ depends on the reference trajectory, the nominal motor current and the disturbance $\boldsymbol{w}$. The reference trajectory and the nominal motor current are equal for each trial. Disturbance $\boldsymbol{w}$ models the effect of the asperity part of the joint friction torque and other torque disturbances. The asperity part of the friction depends on the actual trajectory and is thus trial dependent, although its variation is small and will only be noticeable around joint velocity reversals. Other low-frequency torque disturbances are expected to be predominantly trial independent and high-frequency disturbances (noise) to be predominantly trial dependent. The dominant model error is probably the absence of flexibilities in the robot model that may cause high-frequency-resonant vibrations.

The observations above and the guidelines in Section 2.6 indicate that the values of $\kappa(i)$ should be small for high-frequency error components and about 1 for low-frequency error components. This is practically implemented by adding an additional low pass filter to $\boldsymbol{F}(z)$ (see Fig. 5) and taking $\kappa(i) = 1$ for large singular values and $\kappa(i) = 0$ for small singular values. The low-pass filter decreases the effect of the feedforward on high-frequency error components and thus decreases the singular values of the system matrix corresponding to high-frequency error components. Since $\kappa(i) = 0$ for small singular values the learning controller will not try to compensate for high-frequency error components. The convergence of the remaining error components is maximised by $\kappa(i) = 1$. The only tuning parameter left is the number of singular values for which $\kappa(i) = 1$, which is denoted by N_s.

4.3. Results

Five series of experiments have been done, with an increasing number of singular values for which $\kappa(i) = 1$, namely $N_s = \{250, 500, 750, 1000, 1500\}$. These series of experiments are denoted as experiment 1–5, respectively. Each experiment consists of an initial trial and five learning trials. In the initial trial (trial 0) the feedforward is zero and in the subsequent five trials (trial 1 to 5) the feedforward is computed according to Eq. (11) using the error measured in the previous trial.

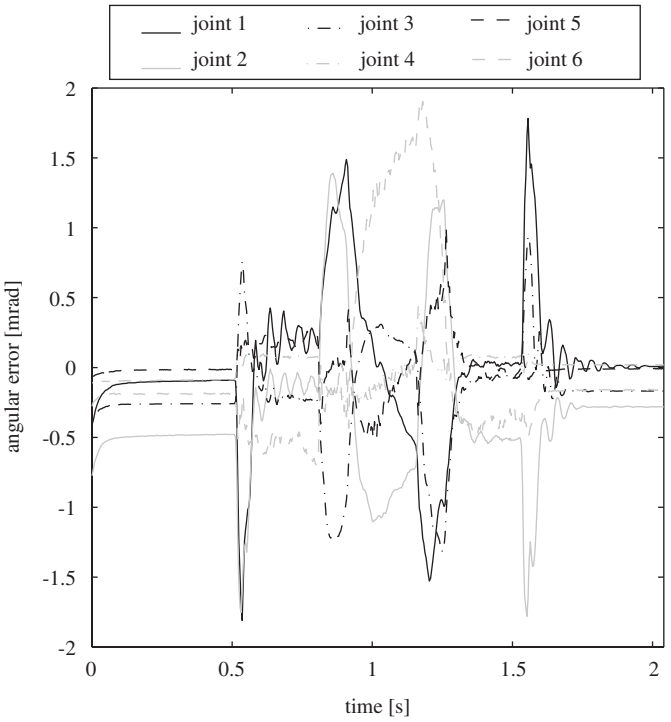

Fig. 7. Angular errors of all joints without learning.

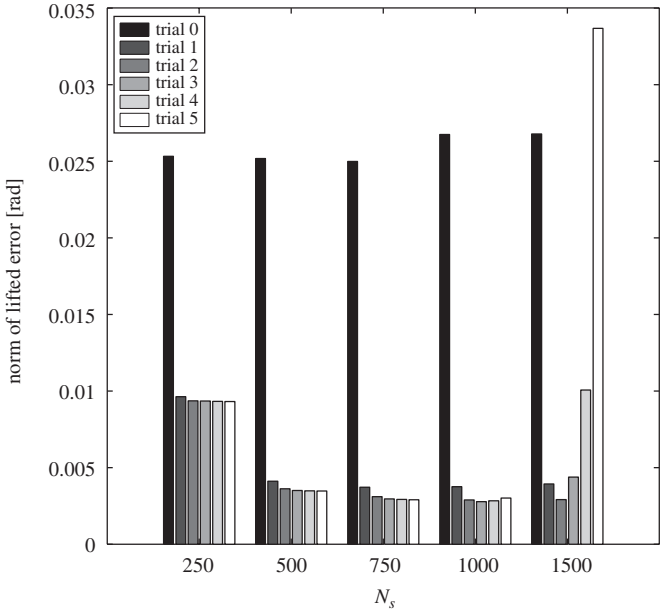

Fig. 8. Norm of lifted error for the six trials of the five series of experiments.

Fig. 7 shows the angular error for trial 0 (no learning applied). Three types of errors are visible:

- A static error before the start of the motion (0.5 s) and after the end of the motion (1.544 s). This error results from the absence of an integrating action in the feedback controller.
- Large low-frequency errors at the start and the end of the motion and in the curved part of the trajectory. These errors are caused by the high accelerations and decelerations of the joints.
- High-frequency errors during the start and the end of the motion. These errors are most likely caused by resonant vibrations.

Fig. 8 shows the norm of the lifted error vector for the six trials of each experiment. The figure shows that the error decreases significantly in the first trial of each experiment and decreases a little more in subsequent trials for experiments 1–3 ($N_s = \{250, 500, 750\}$). In experiment 4 and 5 ($N_s = \{1000, 1500\}$) the error appears to increase after the third trial.

More detailed results are given in Fig. 9, which shows the tracking error of joint 2 in trial 1 and 5. The error in the first few samples of each trial is caused by a delay in the closed-loop system that does not allow a feedforward correction to be applied in the first few samples. In experiments 1–3 the errors in trial 1 and 5 are quite similar, only the static error that is present before the start and after the end of the motion in trial 1, is absent in trial 5. In experiment 1 the error after five trials is a little larger than in experiments 2 and 3. In experiment 4 some high-frequency error components increase slightly after 5 trials. These high-frequency errors are even larger in trial 5 of experiment 5 and in addition some low-frequency errors that were not present in trial 1 appear in trial 5.

The observed behaviour can be explained by looking at the size of the error components $\bar{\boldsymbol{e}}_d(i)$. Note that these error components are computed by projection of the error on the left singular values of the system matrix (Eq. (15)). The propagation of these error components is decoupled as analysed in Section 2.3. Due to the low-pass filter $\boldsymbol{F}(z)$ the error components with a low index correspond to low frequencies, while the error components with a high index correspond to high frequencies. The size of the error components is plotted in Fig. 10 for the six trials of each experiment. The vertical axis corresponds to the index of the error component, the horizontal axis to the trial number and the grey level indicates the size of the absolute value of the error component. The norm of the lifted error $\bar{\boldsymbol{e}}$, shown in Fig. 8, is equal to the norm of the vector of error components.

For experiments 1–3 the sizes of the error components indexed below $N_s = [250, 500, 750]$, respectively, diminish to zero in the first two trials and the components indexed above N_s are constant as expected from the convergence analyses in Section 2.3. Because part of the error components decreases and the rest is constant, the norm of the lifted error decreases to a non-zero value as shown in Fig. 8.

In experiment 4 the error components indexed below 750 disappear as in the first three experiments. This makes the norm of the error decrease initially. However, some of the error components indexed between 750 and 1000 increase, which make the norm of the error increase after the third trial. Considering the exponential growth of the diverging

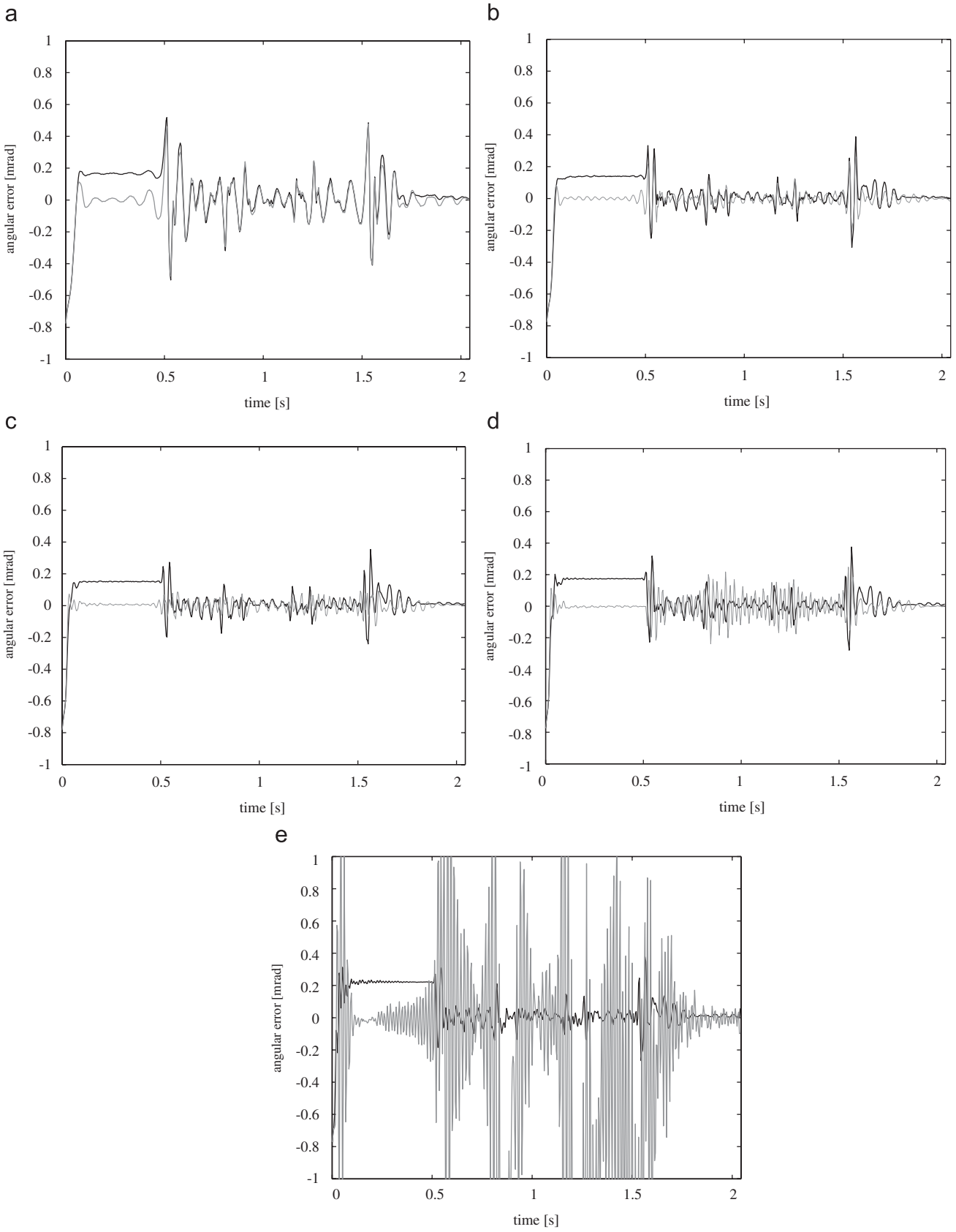

Fig. 9. Angular tracking error of joint 2 for trial 1 (black) and trial 5 (grey): (a) exp. 1, $N_s = 250$, (b) exp. 2, $N_s = 500$, (c) exp. 3, $N_s = 750$, (d) exp. 4, $N_s = 1000$ and (e) exp. 5, $N_s = 1500$.

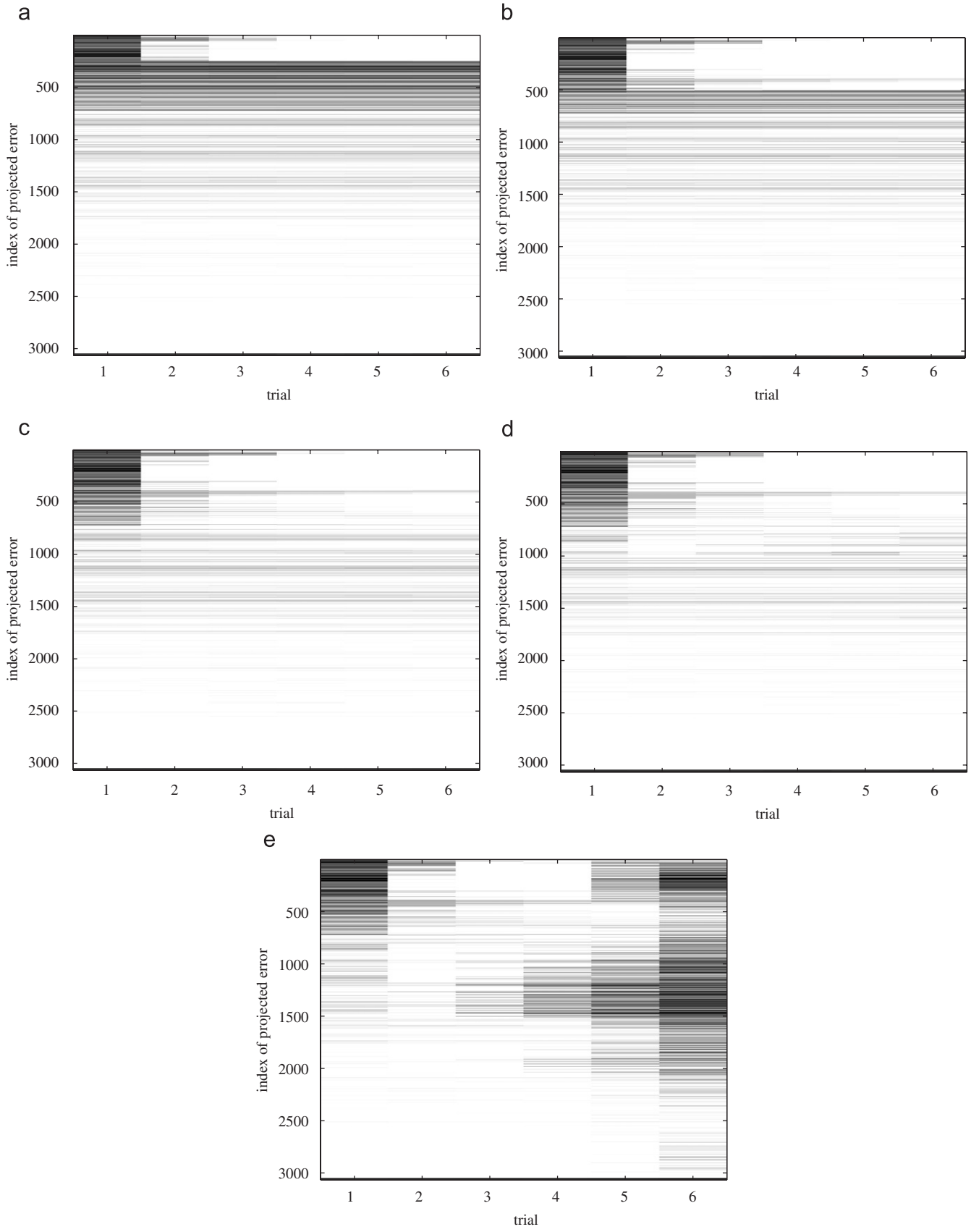

Fig. 10. Size of the error components for all six trials: (a) exp. 1, $N_s = 250$, (b) exp. 2, $N_s = 500$, (c) exp. 3, $N_s = 750$, (d) exp. 4, $N_s = 1000$, (e) exp. 5, $N_s = 1500$. Black corresponds to large values of the error components, grey to medium values and white to small values.

error components, the convergence condition (Eq. (13)) is apparently not met for these error components. This could be caused by modelling errors as discussed in Section 2.5. The most obvious modelling error is the absence of joint or link flexibilities in the robot model. Compensation for resonant vibrations using a model that does not include the effect of flexibilities might increase the resonance vibrations instead of cancelling them. This suspicion is confirmed by the Fig. 9(d), which shows that indeed high-frequency vibrations increase.

In the first two trials the observed behaviour in experiment 5 is equal to the behaviour in experiment 4. However, after the second trial some of the error components indexed below 750 and above 1500 start to increase. As explained in Section 2.5, modelling errors can couple the propagation of the error components. Probably the propagation of the error components indexed between 750 and 1500 is slightly coupled to the propagation of the other error components. Since the first set of error components grows very large in experiment 5, the size of the other error components also grows (see Fig. 10(e)).

The experimental results demonstrate the high convergence speed of the proposed learning control algorithm and the reduction of the tracking error that can be achieved with the available robot model. The robot model is accurate in the low-frequency range and in this range the tracking error is reduced substantially. The model is not accurate in the high-frequency range, since it does not include the elasticity in the robot mechanism. The learning algorithm should be tuned such that it does not compensate at high frequencies to prevent divergence, this means that the value of κ should be zero for the error components corresponding to high frequencies. The inaccuracies in the model are then handled by the algorithm by refraining from compensation of the high frequent error components.

In practice the frequency content of the tracking error (without ILC compensation) and the maximum tracking error allowed by the application dictate the frequency range in which the error should be compensated. The algorithm requires a model that is able to describe the robot dynamics sufficiently accurate in this frequency range. In case the error should be compensated at high frequencies as well, the model should be extended to describe the high-frequency dynamics of the robot more accurately, possibly by including the effect of elasticity in the robot mechanism. Subsequently the gain $\kappa(i)$ should be set to one for frequencies inside the validity range of the model and to zero for frequencies outside this range to guarantee convergence. The frequency of the error components corresponding to each singular value of the system matrix can be obtained from the Fourier transform of the corresponding left singular vector of the system matrix.

The divergence of high-frequency components of the tracking error is encountered more often in practical applications of ILC to industrial robots. A recent example is given by Tayebi and Islam (2006), who presented the experimental results of the application of an adaptive iterative learning control scheme to an industrial robot. The algorithm, proposed by Tayebi (2004), iteratively adapts the time-varying gain of a feedback controller. The time derivative of the tracking error in the previous trials is used for the computation of the gain and convergence of the tracking error is guaranteed for the application of the algorithm to a robot without elasticity. The experimental application of the algorithm results in divergence of high-frequency components of the tracking error. Tayebi and Islam (2006) ascribe the divergence of the high-frequency components to the noise amplification by the computation the time-derivative of the tracking error. Another reasonable explanation for the divergence of the high-frequency error components could be of the elasticity in the robot mechanism, which is not considered in the model that is used for the convergence proof of the algorithm.

For laser welding the difference between the weld seam and the trajectory of the laser focus is of primary importance. This difference is denoted as the tip tracking error. The tip tracking error is estimated from the measured joint tracking error using the nominal kinematic model of the Stäubli RX90 robot. Note that the real tip tracking error probably differs from the estimate because of flexibilities of the robot and errors in the kinematic models. The best error reduction of the joint tracking error was achieved in experiment 3. The estimated tip tracking error in trial 0 and 5 of experiment 3 is plotted in Fig. 11. The estimated tip tracking error decreases considerably. In the fifth trial the tracking error along the trajectory is close to the accuracy of 0.1 mm required for laser welding.

5. Conclusions

This paper presents a model-based iterative learning control algorithm for time-varying systems with a high convergence speed. The algorithm is based on the lifted system description. Singular value decomposition of the lifted system matrix is used to decouple the system equations. Analyses show that robustness of the learning controller for non-repetitive disturbances and model errors can be created at cost of convergence speed of the error components. A learning controller is proposed to control the components of the error individually. The convergence of part of the error components is limited by insufficient model knowledge or noise, while the convergence speed of other errors can be maximised.

Using the standard industrial controller the tracking accuracy of the Stäubli RX90 robot is insufficient for laser welding at high velocities. Therefore, it is investigated if the performance can be increased with learning control. An LTV model of the robot is obtained by linearising its non-linear dynamics for small perturbations around a nominal trajectory. Disturbances and unmodelled robot dynamics are expected to affect the high-frequency error components. The proposed learning scheme is implemented for the robot

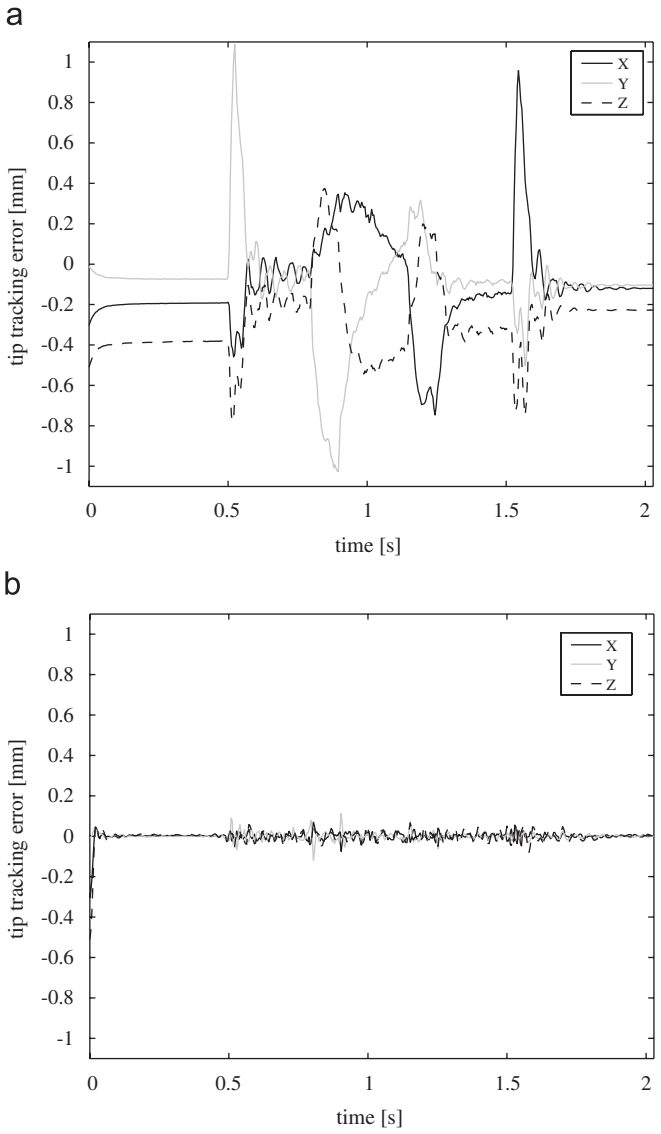

Fig. 11. Estimated tip tracking error in experiment 3: (a) trial 0 and (b) trial 5.

such that the high-frequency components of the error are not compensated, while the remaining components are eliminated in one run. Experimental results show that most of the low-frequency errors can indeed be eliminated in one run, while the high-frequency components remain constant. After five iterations the accuracy of the tip-motion, estimated from the joint motion, is almost sufficient for high demanding laser welding tasks. Trying to compensate for the high-frequency error components as well results in diverging error components, probably due to modelling errors.

6. Discussion

The experiments show that LSILC is suitable to decrease the joint tracking error of the Stäubli RX90 robot. However, accurate joint motion does not necessarily imply accurate tip motion due to joint and drive flexibilities of the robot and errors in the geometric robot model. LSILC could be used to improve the tip motion of the robot as well. This requires measurements of the tip motion and extending the robot model to describe the effect of the feedforward on the tip motion.

Drive and joint flexibilities of the robot are not modelled. This results in the inability of LSILC to compensate for resonant vibrations. Incorporating joint flexibilities in the robot model may enable the learning controller to compensate for these vibrations as well, provided the model matches well with the real robot dynamics. Incorporating the effect of flexibilities could be especially important if LSILC is used to improve the motion of the tip of the robot.

The use of SVD of the system matrix allows the design of an algorithm with maximum convergence speed for certain components of the tracking error and robustness to modelling errors and noise that affect other error components. Furthermore, it allows clear analyses of the convergence and robustness of the learning controller for each of the error components. However, the disadvantage of the SVD of the system matrix is the amount of memory that is required to store the singular vectors and the lengthy computation time. The memory scales with the square of the trajectory length and the computation time grows even faster. The required memory and computation time are acceptable for short trajectories as used for the experiments of this paper but could be a severe limitation for the application of the proposed learning controller for longer trajectories. The required computation time and memory required for lifted ILC can possibly be reduced by the application of the Q-ILC algorithm proposed by Dijkstra (2004). Exploring this possibility involves further research as this algorithm is only applicable to linear time-invariant systems whereas application to robotics requires extension of the algorithm to linear time-varying systems.

Non-repetitive disturbances and modelling errors are expected to affect the high-frequency components of the tracking error of the Stäubli RX90 robot. In this paper a low pass filter is used to decrease the effect of the feedforward on high-frequency components of the error and components of the error that are hardly affected by the feedforward are not compensated for by the learning controller. A more fundamental approach to robustness of LSILC requires further research.

Acknowledgements

This research was carried out under project number MC8.03161 in the framework of the Strategic Research programme of the Netherlands Institute for Metals Research in the Netherlands (www.nimr.nl).

The authors appreciate the information on the CS8 controller provided by Stäubli.

References

Arimoto, S., Kawamura, S., & Miyazaki, F. (1984). Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronic systems. *Journal of Robotic Systems*, *1*(2), 123–140.

Arimoto, S., Nguyen, P., & Naniwa, T. (2000). Learning of robot tasks on the basis of passivity and impedance concepts. *Robotics and Autonomous systems*, *32*, 79–87.

Dijkstra, B. (2004). *Iterative learning control with applications to a wafer-stage*. Ph.D. Thesis, Delft University of Technology, The Netherlands.

Duley, W. (1998). *Laser welding*. New York: Wiley.

Elci, H., Longman, R., Phan, M., Juang, J., & Ugoletti, R. (2002). Simple learning control made practical by zero-phase filtering: Applications to robotics. *IEEE Transactions on Circuits and Systems*, *49*(6), 753–767.

Guglielmo, K., & Sadegh, N. (1996). Theory and implementation of a repetitive robot controller with cartesian trajectory description. *Journal of Dynamic Systems, Measurement and Control*, *118*(2), 15–21.

Gunnarsson, S., Norrlöf, M., Rahic, E., Özbek, M. (2004). Iterative learning control of a flexible robot arm using accelerometers. In *Proceedings of the IEEE international conference on control applications*. (pp. 1012–1016), IEEE.

Jonker, J., & Aarts, R. (2001). A perturbation method for dynamic analysis and simulation of flexible manipulators. *Multibody System Dynamics*, *6*(3), 245–266.

Kavli, T. (1993). Frequency domain synthesis of trajectory learning controllers for robot manipulators. *Modeling, Identification and Control*, *14*(3), 161–174.

Kim, W., Chin, I. S., Lee, K. S., & Choi, J. (2000). Analysis and reduced-order design of quadratic criterion-based iterative learning control using singular value decomposition. *Computers and Chemical Engineering*, *24*, 1815–1819.

Longman, R. (2000). Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, *73*(10), 930–954.

Moore, K. (1998). Iterative learning control: An expository overview. *Applied and Computational Controls, Signal Processing, and Circuits*, *1*(1).

Norrlöf, M., & Gunnarsson, S. (2002). An adaptive iterative learning control algorithm with experiments on an industrial robot. *IEEE Transactions on Robotics and Automation*, *18*(2), 245–251.

Poo, A., Lim, K., & Ma, Y. (1996). Application of discrete learning control to a robotic manipulator. *Robotics and Computer-Integrated Manufacturing*, *12*(1), 55–64.

Tayebi, A. (2004). Adaptive iterative learning control for robot manipulators. *Automatica*, *40*(7), 1195–1203.

Tayebi, A., & Islam, S. (2006). Adaptive iterative learning control for robot manipulators: Experimental results. *Control Engineering Practice*, *14*, 843–851.

Tousain, R., van der Meché, E., & Bosgra, O. (2001). Design strategies for iterative learning control based on optimal control. In *Proceedings of the 40th IEEE conference on decision and control*, December 2001, Orlando, FL, USA (pp. 4463–4468) New York: IEEE.

Waiboer, R., Aarts, R., Jonker, J. (2005a). Modelling and identification of a six axes industrial robot. In *Proceedings of IDETC/CIE. ASME*.

Waiboer, R., Aarts, R., & Jonker, J. (2005b). Velocity dependence of joint friction in robotic manipulators with gear transmissions. In J. Goicolea, J. Cuadrado, & J. G. Orden (Eds.), *ECCOMAS thematic conference on multibody dynamics* (pp. 1–19). Spain: ECCOMAS, Madrid [CDROM].