Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 46 (2016) 391 – 395

www.elsevier.com/locate/procedia

7th HPC 2016 – CIRP Conference on High Performance Cutting

Iterative learning for machine tools using a convex optimisation approach

Titus Haas[a,*], Natanael Lanz[a], Roman Keller[a], Sascha Weikert[b], Konrad Wegener[a]

[a]*Institute of Machine Tools and Manufacturing, ETH Zurich, Leonhardstrasse 21, 8092 Zurich, Switzerland*
[b]*Inspire AG, Technoparkstrasse 1, 8005 Zurich, Switzerland*

* Corresponding author. Tel.: +41-44-632-4845 ; fax: +41-44-632-1159. *E-mail address:* haas@iwf.mavt.ethz.ch

Abstract

Dynamic, quasi-static and motion control deviations lead to nonlinear but systematic tracking errors. It is shown that these errors can be reduced significantly by adjusting the set points using an optimization based iterative learning approach. This method uses either values obtained from internal encoders or alternatively tool center point measurements. The approach is presented, discussed and validated using simulation and measurement results.

Peer-review under responsibility of the International Scientific Committee of 7th HPC 2016 in the person of the Conference Chair Prof. Matthias Putz

Keywords: Control; Machine tool; Precision

1. Introduction

Dynamic, quasi-static and motion control deviations lead to nonlinear but systematic tracking errors of machine tools. The goal of iterative learning is to learn from systematic and repeatable errors of previous trials and to improve the following trial. Machine tools typically do not learn from previous experience. Especially in high-volume production, where a given part is produced multiple times, it is desirable to reduce tracking errors. A reduction of tracking errors would either allow higher feed rates, leading to the same dynamic deviations, or to reduce contour errors, using the same feed rate. According to Bristow et al. [1], iterative learning control (ILC) modifies the input of the controller of a system and not the controller itself, as for example adaptive control and neural network strategies do. Repetitive control is similar to ILC but for continuous operations, i.e. the next iteration follows immediately and the initial conditions are given by the final conditions of the previous trial.

A lot of work has been done in the field of ILC; a good survey is given by Bristow et al. [1] and Ahn et al. [2].

The basic idea of ILC was published by Garden [3] in 1971. The algorithm was presented for the first time in English by Arimoto et al. [4] in 1984, where an iterative learning scheme for a robot manipulator was proposed.

Feedforward control can compensate tracking errors caused by lag and has good performance if the system is known accurately. Stiction, not modelled nonlinear behavior and disturbances can limit the performance of feedforward control [1]. ILC can compensate any nonlinear, but repeatable disturbance. The performance of ILC is limited by unrepeatable disturbances and noise. The influence of the latter can be reduced by zero-phase filtering, e.g. Butterworth, which is possible without lag. A combination of feedback control and ILC is recommended by [1].

Togai and Yamano [5], e.g., formulated the iterative learning control problem as a quadratic optimization criterion and therefore, reduced the error and additionally the input. Penalizing also the input, the error cannot be reduced to the minimal achievable error. Amann et al. [6], Lee et al. [7] and Barton et al. [8] extended the cost function by the input change instead of the input. Therefore, the ILC algorithm has an integral action in the iteration domain and the minimal achievable error is attainable. It is possible to consider constraints, disturbances and model errors for example.

Kim and Kim [9] presented the proportional, integral and derivative (PID) type ILC algorithm of [4] for a machine tool,

Peer-review under responsibility of the International Scientific Committee of 7th HPC 2016 in the person of the Conference Chair Prof. Matthias Putz
doi:10.1016/j.procir.2016.04.033

where the actual machined path was measured using a roundness measuring instrument. A decrease of 58% of the error between the measued and the desired path is shown for cutting circles with radius of 29.7 mm and a feed rate of 200 mm/min. Tsai and Chen [10] applied the PID type ILC algorithm of [4] to reduce the deviation between the desired cutting and actual fracture trajectory for a CO_2 laser machine tool. Tsai et al. [11] proposed a P-type ILC algorithm with predicted tracking and contour error and compared the performance of the error reduction for different weightings of those errors. An application for improving contour error tracking in precision motion control was presented by Altin and Barton [12]. The norm optimal framework was used to minimize the tracking error, contour error and input change. Using a model of the contour error, Wu et al. [13] proposed an A-type iterative learning cross-coupled control that was based on a contour error model and showed the convergence of it. Khong et al. [14] proposed an extremum seeking approach to iterative learning for nonlinear time-varying systems.

In this paper, the optimization based ILC approach, proposed e.g. by Barton et al. [8], is used and compared to the commonly used PD-ILC algorithm presented by [1]. The quadratic optimization formulation is called convex optimization (CO) ILC in the remainder of this paper.

2. Comparison of ILC algorithms

2.1. Overview of ILC methods

The general application scheme of the ILC algorithm is shown in Fig. 1. The plant **P** represents any dynamic system, e.g. a machine tool servo axis and **C** represents the controller. The tracking error of the iteration j, $\mathbf{e}_j$, is given by the deviation between the desired trajectory, $\mathbf{y}_d$, and the actual measurement, $\mathbf{y}_j$. The input of the iteration j is $\mathbf{u}_j$. Only asymptotically stable closed-loop systems are considered in this paper. Note that $\mathbf{e}_j$, $\mathbf{y}_d$, $\mathbf{y}_j$ and $\mathbf{u}_j$ are vectors of length *N*, where *N* is the number of time discrete trajectory samples. The computation of $\mathbf{u}_j$ is repeated for each iteration. Therefore, ILC can be used online or offline.

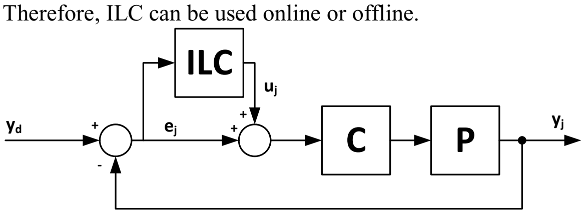

Fig. 1. General application scheme of the ILC algorithm.

In this paper, two types of ILC algorithms have been implemented and compared:

- PD-ILC with a Butterworth low pass filter as presented in [1]
- CO-ILC with a second order model representing the closed loop servo axis behavior of each axis

No model is needed for PD-ILC, whereas for CO-ILC a linear time invariant model of the closed loop system dynamics is required (1). For an initial state $\mathbf{x}_j[0]=0$, the servo loop dynamic matrix $\mathbf{P}_{cl}$, using the state space closed loop dynamics in (2), can be derived as shown in (3). The indices j and n denote the iteration and time sample, respectively.

$$\mathbf{y}_j = \mathbf{P}_{cl}(\mathbf{y}_d + \mathbf{u}_j) \tag{1}$$

$$\begin{cases} \boldsymbol{x}_j[n+1] = \boldsymbol{A}_{cl}\,\boldsymbol{x}_j[n] + \boldsymbol{B}_{cl}\left(\boldsymbol{u}_j[n] + \boldsymbol{y}_d[n]\right) \\ \boldsymbol{y}_j[n] = \boldsymbol{C}_{cl}\,\boldsymbol{x}_j[n] \end{cases} \tag{2}$$

$$\boldsymbol{P}_{cl} = \begin{pmatrix} \boldsymbol{C}_{cl}\boldsymbol{B}_{cl} & 0 & \cdots & 0 \\ \boldsymbol{C}_{cl}\boldsymbol{A}_{cl}\boldsymbol{B}_{cl} & \boldsymbol{C}_{cl}\boldsymbol{B}_{cl} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}_{cl}\boldsymbol{A}_{cl}^{N-1}\boldsymbol{B}_{cl} & \boldsymbol{C}_{cl}\boldsymbol{A}_{cl}^{N-2}\boldsymbol{B}_{cl} & \cdots & \boldsymbol{C}_{cl}\boldsymbol{B}_{cl} \end{pmatrix} \tag{3}$$

2.2. PD-ILC

The PD-ILC algorithm is defined, using the nomenclature in Fig. 1, the gains k_P, k_D and the low pass filter **Q**, as follows:

$$\begin{aligned} \mathbf{u}_{j+1}[n] = \;& \mathbf{Q}[z](\mathbf{u}_j[n] + k_P\mathbf{e}_j[n+1] \\ & + k_D(\mathbf{e}_j[n+1] - \mathbf{e}_j[n])) \end{aligned} \tag{4}$$

A stability criterion and a condition for monotonic convergence is presented in [1]. It is shown that, in order to ensure convergence, a low pass filter, **Q**, is required.

2.3. CO-ILC

CO-ILC is a special case of the optimization based ILC framework presented in [8]. The difference of the input between subsequent iterations is given by

$$\Delta\mathbf{u}_{j+1} = \mathbf{u}_{j+1} - \mathbf{u}_j\,. \tag{5}$$

The predicted error $\mathbf{e}_{j+1}$ of the next iteration, using the linear plant model $\mathbf{P}_{cl}$, is given by

$$\mathbf{e}_{j+1} = \mathbf{e}_j - \mathbf{P}_{cl}\,\Delta\mathbf{u}_{j+1}\,. \tag{6}$$

The cost function of the optimization problem consists of two parts, the tracking error and the weighted velocity of the input difference:

$$\mathbf{J}_1 = \mathbf{e}_j - \mathbf{P}_{cl}\,\Delta\mathbf{u}_{j+1} \text{ and } \mathbf{J}_2 = d\,\mathbf{D}\Delta\mathbf{u}_{j+1}\,. \tag{7}$$

Minimizing $\mathbf{J}_1^T\mathbf{J}_1 + \mathbf{J}_2^T\mathbf{J}_2$, using (5) as state vector and the scalar weighting factor d, the following quadratic program can be stated:

$$\min_{\Delta\mathbf{u}_{j+1}}\left(\frac{1}{2}\Delta\mathbf{u}_{j+1}^T\mathbf{H}\Delta\mathbf{u}_{j+1} + \mathbf{f}^T\Delta\mathbf{u}_{j+1}\right). \tag{8}$$

H and **f** are defined as

$$\begin{aligned} \mathbf{H} &= \mathbf{P}_{cl}^T\mathbf{P}_{cl} + d^2\,\mathbf{D}^T\mathbf{D} \\ \mathbf{f} &= -\mathbf{P}_{cl}^T\mathbf{e}_j \end{aligned} \tag{9}$$

The matrix **D** is given by

$$\mathbf{D} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}, \tag{10}$$

which leads to smooth input signals.