Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 67 (2018) 278 – 283

www.elsevier.com/locate/procedia

11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '17

Online learning of stability lobe diagrams in milling

Jens Friedrich[a,*], Jonas Torzewski[a], Alexander Verl[a]

[a]*Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Seidenstraße 36, 70174 Stuttgart, Germany*

* Corresponding author. Tel.: +49-711-685-82416; fax: +49-711-685-72416. *E-mail address:* Jens.Friedrich@isw.uni-stuttgart.de

Abstract

The productivity of milling machines is limited by chatter vibrations. Stability lobe diagrams (SLD) allow the selection of suitable process parameters to maximize the productivity. However, the calculation of SLDs is very time-consuming and requires complex experiments. In this article a new online learning method is presented, which allows the calculation of SLDs during the production process. The algorithm is a combination of reinforcement learning and nearest-neighbor-classification and allows the estimation of the stability border based on measured vibration signals during machining. The proposed algorithm is capable of being continuously trained with sorted input data. A trust criterion is introduced, which allows judging the prediction quality of the algorithm. The algorithm is validated with analytical benchmark functions and with a 2-DOF milling stability simulation.

Keywords: Milling; Stability; Chatter; Learning.

1. Introduction

Chatter vibrations are one of the most important factors limiting the productivity of machining industry. They cause poor surface quality, increase tool wear and can even cause damages on the machine tool. Chatter vibrations are self-excited vibrations, caused by the interaction of the cutting edge with the surface left by the previous cut and the flexibility of the machine structure [1, 2]. The chatter effect is a feedback with time delay caused by the rotational speed of the spindle and excited by the cutting forces. Therefore, the occurrence of chatter in milling operations depends on the process parameters (spindle speed, cutting depth, and cutting width), the tool-material-combination and the dynamics of the machine tool system [3, 4]. To ensure a stable, chatter free machining, the process parameters are often chosen very conservatively.

To avoid chatter, active control strategies can be used for varying the spindle speed in order to avoid the self-excitation [5, 6] or for changing the spindle speed to a spindle speed which is stable [7, 8].

An optimal spindle speed, cutting depth and cutting width can also be selected based on the stability lobe diagram (SLD) [4]. Each combination of spindle speed, cutting depth and cutting width can be either classified as stable or unstable. This information can be represented in a SLD, separating the stable and unstable regions. The knowledge of the stability limit allows the selection of process parameters for maximizing the productivity. There exist several possibilities of generating SLDs. The SLD can be calculated based on the stability of the mathematical model with time delay [9, 10]. This demands a very accurate model. An experimental approach avoids the modelling by applying test cuts for different spindle speeds and cutting depths [11, 12]. The model-based as well as the experimental approach suffer from the additional effort of identifying the model or the SLD. Moreover, the SLD is only valid for the timeframe, in which the experiments were performed, as the machine behavior can change over time [13].

In this paper, an approach to continuously learn the SLD during productive milling is presented. The learning algorithm allows the prediction of the stability border based on measured vibration signals. The application during productive milling leads to sorted input data and an incomplete training set. The algorithm can be trained with incomplete, sorted training data and the proposed trust criterion allows to judge the prediction reliability.

doi:10.1016/j.procir.2017.12.213

In the following section, the requirements for the learning algorithm are defined. Moreover, the theoretical background of the learning algorithm is explained. The results and verification with benchmark functions and milling stability data is presented in Section 3. Finally, a conclusion is given in section 4.

2. Learning algorithm

In this chapter, an overview of the underlying learning algorithm and the specific requirements coming from the application to milling stability prediction is given.

The learning algorithm should generate the SLD from training data measured during the milling process. The spindle speed, the cutting depth and cutting width can be calculated with the method presented in [14] based on the ISO code or the machine positions extracted from the control system. The stability of the process is estimated based on vibration signals as described in [15]. Therefore, we want to construct a function from $\mathbb{R}^m \rightarrow \mathbb{R}$. Where m is the number of input values. In this case $m=3$ (cutting depth x_1, cutting width x_2, spindle speed x_3) which should be assigned to the sensor value y. Consequently, every training point P is a tuple of the input data (x_1, x_2, x_3) and the output data y (see equation (1)).

$$P = (x_1, x_2, x_3, y) \tag{1}$$

The algorithm combines the advantages from reinforcement learning [16, 17] and nearest-neighbor method [18].

2.1. Requirements and assumptions

The first assumption is that the sensor value is steady. Small changes of the process parameters only lead to small changes of the sensor value [15]. The next assumption is that two training points only contain local information, and the larger the distance between them, the smaller the link between them.

The algorithm should collect data during the regular milling process and adapt itself continuously without storing all data. Therefore, it should generalize the data before storing the data in the knowledge base. This implies, that the algorithm must be able to process a sorted, incomplete set of input data, because not all combinations of input parameters are covered during the milling process.

The learning algorithm gives an output for each input vector, regardless of whether it has converged or not. The incomplete input set explained above leads to untrained regions, thus a trust criterion to judge the reliability of the output is essential for the application of the learned SLD.

The next requirement is that the algorithm should be robust against incorrect data, as outliers may occur. Moreover, it should be deterministic and fast enough to run on the control system of the milling machine.

2.2. Knowledge base

To establish the mathematical knowledge base, the definition area is uniformly distributed in states, as it is known from reinforcement learning. Each state represents a memory and stores the information of local training data. Similar to the human learning process, each experience affects suitable local memories. Therefore, the learning process can be described as the calculation of many weighted averages (2).

$$V_{k_1,..,k_m} = \frac{N_{k_1,..,k_m}}{D_{k_1,..,k_m}} = \frac{\sum_{i=1}^{P_{all}} \gamma_{k_1,..,k_m}(x_{1,i}, x_{2,i}, x_{3,i}) \cdot y_i}{\sum_{i=1}^{P_{all}} \gamma_{k_1,..,k_m}(x_{1,i}, x_{2,i}, x_{3,i})} \tag{2}$$

Where the state value $V_{k1,..,km}$ represents the memory at the state $k_1,..,k_m$. The position of a state in the definition area is described by the vector $(k_1,..,k_m)$. The weights $\gamma_{k1,..,km}$ depend on the distance d_k between the state $k_1,..,k_m$ and the input parameters of the training point i. For an easier notation we replace $k_1,..,k_m$ by k. For γ_k we use a decay function, which calculates the weight dependent on the distance d_k between state k and input (x_1, x_2, x_3).

In section 2.1 continuous adaption of the function without storing all training data is requested. This is the reason why we store denominator D_k and numerator N_k separately. Thus, we can add a new summand to the denominator D_k and the numerator N_k and have the same behavior as calculating V_k with all stored training data (equation (2)). The continuous learning process of all states k with a new training point i follows the rule (3)

$$\begin{aligned} N_{k,new} &= N_{k,old} + \gamma_k(x_{1,i}, x_{2,i}, x_{3,i}) \cdot y_i \\ D_{k,new} &= D_{k,old} + \gamma_k(x_{1,i}, x_{2,i}, x_{3,i}) \end{aligned} \tag{3}$$

Using Gaussian function (4) for γ_k, the parameter σ_k controls the influence area of a training point. To ensure a uniform sensitivity for all training data $\sigma_k \in [1, 1.5]$ should be chosen.

$$\gamma_k(x_1, x_2, x_3) = e^{-d_k(x_1,x_2,x_3)^2/\sigma_k^2} \tag{4}$$

For each training step, every state has to be updated. Therefore, the calculation time increases linearly with the number of states. If the resolution of every component (k_1, .., k_m) should be increased, the number of states increases with $O(r^m) = O(r^3)$. To decrease the calculation time only the states near the input can be updated. Every point is multiplied with the weight γ_k. As γ_k is a decay function, it is close to 0 for states, far away from the input. Thus, only the states around the input vector, where $\gamma > \gamma_{min}$ or the distance $d > a \cdot \sigma_k$. are updated. Suitable values are $a = 2,$ which leads to $\gamma_{min} = 0,02.$ The calculation time is now independent from the number of states. With this modification, the Gaussian decay function γ_k is given by (5).

$$\gamma_k(x_1, x_2, x_3) = \begin{cases} e^{-d_k(x_1,x_2,x_3)^2/\sigma_k^2} & d_k \le a \cdot \sigma_k \\ 0 & d_k > a \cdot \sigma_k \end{cases} \tag{5}$$

Other decay functions γ with limited range can be used as well. Especially piecewise defined polynomial function with