Computer-Aided Design and Applications

http://www.cadanda.com

Interpolation Parameter Tuning for Parametric Curve Paths Using Learning Automata Method

Syh-Shiuh Yeh[1] and Jin-Tsu Sun[2]

[1]National Taipei University of Technology, ssyeh@ntut.edu.tw
[2]Industrial Technology Research Institute, ricky@itri.org.tw

ABSTRACT

This study focuses on the development of an interpolation parameter tuning method for parametric curve paths in motion control systems. It is usually difficult to design the interpolation parameter of an interpolation algorithm using systematic approaches because some factors, such as mechanical factors and manufacturing factors, can significantly affect the motions of the applied motion plant in real applications. Therefore, in this study, the learning automata tuning method, which operates through interactions with unknown environments, was used to tune the interpolation parameter. Some simulation and motion tests were performed on a die bonder machine to test the proposed approach, and the experimental results show that the learning automata tuning method allows the applied motion control system to achieve a good trade-off between motion accuracy and motion time.

Keywords: interpolation, parameter tuning, learning automata, motion control.
DOI: 10.3722/cadaps.2009.329-339

1. INTRODUCTION

Parametric curve functions are usually applied to motion control systems for planning motion paths for the following reasons:

- A common mathematical model can be used to represent both standard analytic shapes and free-form curves.
- The shapes of curves can be easily modified by manipulating shape-control parameters, such as control points, weights, and knots.

Therefore, an interpolator with an interpolation algorithm is usually required to interpolate the parametric curve functions in the design of motion control systems in industrial applications. Further, the design of the interpolation algorithm must consider the motion dynamics and mechanical factors of the applied motion plant in order to improve motion performances. In recent years, many interpolation algorithms have been developed for multi-axis motion control systems. Algorithms that consider the ACC/DEC properties [9],[14],[26], jerk limitation [14–15], and dynamics [14–15],[27],[32] of machine tools have been developed to improve motion accuracy. Moreover, look-ahead functions [9],[14–15],[26],[28] and feedrate adjustment methods [3–5],[23–24],[30–32] have also been developed to reduce the contouring errors of the motion paths along sharp corners. Tikhon et al. [22] proposed an interpolation algorithm that can achieve a constant material removal rate during cutting, and Ko et al. [10] proposed an interpolation algorithm to control the cutting load, to protect cutting tools and improve machinability. Choi et al. [6] proposed an interpolation algorithm that can control the surface roughness during surface machining. Yeh and Sun [29] proposed an interpolation algorithm that considers the motion control system in the presence of actuator saturation to improve motion

accuracy. Although the existing researches have demonstrated important results, some factors still limit the execution performances of the existing interpolation algorithms in real applications, including:

- Model-based interpolation algorithms require a reference model for the applied motion plant in order to generate motion commands according to the dynamics of the motion plant. However, it is usually difficult to obtain an exact model of the applied motion plant by using system identification processes.
- Some mechanical factors, such as the nonlinear friction and non-uniform backlash of moving tables, usually exist in the applied motion plant and significantly affect motion performances. Thus, the execution performance of an interpolation algorithm is limited if mechanical factors with nonlinear and non-uniform characteristics are not considered.
- Some factors in manufacturing processes, such as the payload conditions of the applied motion plant, affect the actual motions of the motion plant. Interpolation algorithms with fixed interpolation parameters are thus not suitable for manufacturing processes with properties that can be changed on-line.

Therefore, in this study, a model-less interpolation algorithm with an on-line changed interpolation parameter was developed for interpolating parametric curve functions, in order to generate motion commands, such that the applied motion plant can execute good motions under the perturbations caused by mechanical factors and factors in the manufacturing processes. Uniform interpolation [1] is the simplest algorithm for interpolating parametric curves. Eqn. (1.1) shows the iterative equation for computing the parameter sequence in uniform interpolation.

$$u_{k+1} = u_k + \varepsilon \tag{1.1}$$

where u_k is the parameter at the kth step and u_{k+1} is the parameter at the (k+1)th step; ε denotes the parameter step size. Usually, an improper value of ε deteriorates the motion results. For instance, a large value of ε causes high motion speeds and large motion errors because of the large servo-lags and unmatched dynamics of the applied motion axes; in contrast, a small value of ε can cause low motion speeds and small motion errors. Therefore, to maintain motion accuracy and shorten the motion time, the interpolation parameter ε must be set properly before each motion of the applied motion plant to increase the equipment throughput. However, it is usually difficult to design a suitable interpolation parameter ε, for the reasons previously mentioned. In this study, an on-line tuning method was thus considered for tuning the interpolation parameter ε.

In recent decades, because of the rapid developments in computation technologies, model-less tuning methods have been widely and diversely introduced in motion control systems in order to tune the motion parameters regardless of the operation setup, operation environment, and design of the motion control structures [11],[12],[13],[21]. Existing researches have demonstrated important results. However, a tuning method with high complexity is usually difficult to implement and obtaining good solutions with a tedious tuning process is usually time-consuming. Thus, a high-performance personal computer (PC) is required in applications that use advanced tuning approaches. In order to reduce the time needed to tune the motion parameters and reduce the complexity for easier implementations, a tuning process that was automated through the use of the learning automata methodology developed by Narendra and Thathachar [19] was used in this study to tune the interpolation parameter in the uniform interpolation algorithm [1]. The learning automata method operates through interactions with unknown environments by selecting the actions in a stochastic trial and error process and provides additional convergence through probability density functions. Although other tuning methods that integrate the learning automata methodology have been developed to provide better tuning results, including the genetic algorithm approach [7], neural network approach [17], and fuzzy inference system [8], their complexity also limits their industrial applications.

Fig. 1 shows a schematic diagram of the interpolation algorithm combined with the learning automata on-line tuning method. In this study, the interpolation parameter ε is tuned on-line based on the motion time and the actual motion results of the applied motion plant. Therefore, the interpolation algorithm with the on-line tuning method can obtain a suitable interpolation parameter ε such that the applied motion plant can gain a good trade-off between motion time and motion accuracy. Moreover, since the tuning method refers to the actual motions of the motion plant, the motion perturbations induced by mechanical factors of the motion plant and factors in the manufacturing

processes are considered in the interpolation algorithm. The model-less tuning method, which has no need for an exact reference model, makes the use of the interpolation algorithm more feasible in real applications.

The rest of this paper is organized as follows. Section 2 reviews the learning automata method for tuning the interpolation parameter ε. This section also discusses some stochastic properties that are useful for applying the learning automata method. Section 3 presents the developed tuning method based on the learning automata method. This section also details the definitions of objective function and motion errors when the learning automata method is applied to tune the interpolation parameter ε. In section 4, the proposed approach is evaluated through simulations and experiments using a biaxial die bonder machine. Section 5 concludes this study.

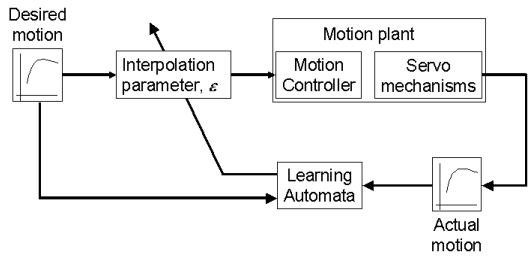


Fig. 1: The proposed interpolation algorithm.

2. REVIEW OF LEARNING AUTOMATA TUNING METHOD

A simple tuning method without the knowledge of motion plant models is usually required in practical tuning applications. In this study, the learning automata method proposed by Narendra and Thathachar [19] was applied to tune interpolation parameter ε of the uniform interpolation [1] because the method is easy to implement and is capable of improving the execution performances of motion systems with unknown dynamics and perturbations. Moreover, the learning automata method can be applied to tune the interpolation parameter using only the input and output data of the motion plant.

Stochastic automata operating under unknown and random environments have previously been proposed as learning models [2],[16]. These automata update their action probabilities in accordance with the inputs received from the environment and improve their own performance during operation. Learning is defined as any relatively permanent change in behavior resulting from past experience, and a learning system is characterized by its ability to improve its behavior with time, in some sense tending toward an ultimate goal. In mathematical psychology, the models of learning systems have been developed to explain patterns among living organisms. These models in turn have been adapted to synthesize engineering systems, which can be considered to show learning behavior. Tsypkin [25] discussed the fact that diverse problems in pattern recognition, control, identification, filtering, and so on can be treated in a unified manner as problems in learning using probabilistic iterative methods.

Mathematically, the goal of a learning automata system is the optimization of an objective function, which is usually used to evaluate the learning performance. An approach that has been used to solve the optimization problem is to reduce the problem to the determination of an optimal set of parameters and then apply stochastic hill-climbing techniques [25]. An alternative approach under investigation is to regard this problem as one of finding an optimal action from a set of allowable actions by using stochastic automata [18,19].

In this study, the tuning process was automated through the use of a learning automata methodology [19], and was used to tune the interpolation parameter in order to minimize the performance objective. The performance objective was usually a simple cost function involving the error over time. The interpolation parameter was initially set by experience. Then, the learning automata method was employed for the interpolation parameter to search the parameter space in order to minimize the specified objective function. The learning automata tuning method operates through an interaction with unknown environments by selecting actions in a stochastic trial and error process. Moreover, in

comparison with other tuning methods, the learning automata method has the advantage of providing additional convergence information through the use of probability density functions.
Assume that the interpolation parameter ε is the variable to be tuned and that $E(\cdot)$ is the objective function to minimize. The learning automata tuning method is summarized as follows:

(1) Given the initial value of the interpolation parameter, ε_0, and the tuning coefficients η, c, and d.
(2) Repeat the following steps: (the i^{th}-iteration)
  (a) Compute the value $E_i = E(\varepsilon_i)$.
  (b) Let $\sigma = c \times E_i + d$.
  (c) Then, use normal distribution to approximate $\varepsilon_i' \sim N(\varepsilon_i, \sigma)$. Here, $N(\mu, \sigma)$ is the normal distribution with mean μ and variance σ.
  (d) Compute $E_i' = E(\varepsilon_i')$.
  (e) Compute $\varepsilon_{i+1} = \varepsilon_i - \eta \cdot (E_i' - E_i) \cdot \dfrac{(\varepsilon_i' - \varepsilon_i)}{\sigma}$.
(3) After the value of the objective function $E(\cdot)$ is less than a threshold value or does not improve any further, the learning process stops. The final value of ε_i is the optimal parameter for minimizing the objective function $E(\cdot)$.

There are two advantages to using a normal distribution to obtain the approximated optimal value in the learning automata method:

- The variance is proportional to the value of the objective function. The variance influences the search range during the learning process, and a small variance will increase the probability of selecting the next parameter near the mean value. However, a large variance will increase the probability of selecting a larger range for the next parameter. Since the variance is proportional to the value of the objective function, when the value of the objective function is small, the searching step size becomes small, and the value of the objective function will be close to the local minimum. Therefore, the value of the objective function changes the search range when the learning automata method is used.
- The learning automata method can supposedly avoid the local minimum. Since the variance is proportional to the value of the objective function, the local minimum can be avoided using the applied learning automata method. When the value of the objective function is larger than the minimal value of the objective function, the next parameter can be selected such that it is far from the mean value, and therefore the learning automata will prevent the tuning process from holding in the local range.

In addition, the tuning method using the learning automata method does not need to identify the system models before learning. Only the input and output data of the tuned control plant are required to evaluate the objective function. Moreover, these input and output data, such as the desired motions and the actual motions of the applied motion plant, are readily available for motion control systems.

3. INTERPOLATION PARAMETER TUNING BASED ON LEARNING AUTOMATA METHOD

The learning automata tuning method is now applied to tune the interpolation parameter ε. Fig. 1 shows the block diagram for tuning the interpolation parameter. The purpose of the tuning method is to tune the interpolation parameter in order to minimize the objective function $E(\varepsilon)$. In this study, since the goal for tuning the interpolation parameter is to make the applied motion plant achieve motions with small motion errors and short motion time, the objective function is defined as

$$E(\varepsilon) = w_{trk} \cdot \sum_{k=1}^{m} \left\| e_k^{trk} \right\| + w_{cnt} \cdot \sum_{k=1}^{m} \left\| e_k^{cnt} \right\| + w_t \cdot t_{mot} \tag{3.1}$$

where $\|\cdot\|$ is a Euclidean norm operator; ε denotes the interpolation parameter to be tuned; e_k^{trk} and e_k^{cnt} are the tracking error and contouring error at the k^{th} sampling instance, respectively; t_{mot} is the motion time of the applied motion plant; w_{trk}, w_{cnt}, and w_t are the weights of the tracking error,

contouring error, and motion time, respectively; and m is the number of sampled data. In this study, the sampling instance of the applied motion control system is synchronized to the interpolation step in the applied interpolation algorithm. Fig. 2 shows the schematic relations between the motion errors, the tracking error and contouring error, and the objective function, $E(\varepsilon)$. T denotes the sampling time. In this study, as shown in Eqn. (3.1), two motion errors, tracking error and contouring error, are used to characterize the motion performances of the applied motion plant. Fig. 3 shows the tracking and contouring errors caused by different motion processes. The tracking error, e_k^{trk}, denotes the distance between the reference position R and the actual position P; however, the contouring error, e_k^{cnt}, denotes the shortest distance from P to the command path. e_k^x and e_k^y are the following errors of the motions on the X-axis and Y-axis, respectively. Conventionally, all the motion axes are controlled independently, with the following errors minimized in order to minimize the tracking errors. However, unmatched dynamics among all the motion axes usually cause large contouring errors [20], and the mechanical factors of the applied motion plant and factors in the manufacturing processes could affect both tracking and contouring errors. Therefore, in this study, the learning automata tuning method, which refers to the object function as in Eqn. (3.1), is applied to tune the interpolation parameter ε such that the applied motion plant can execute good motions with suitable tracking and contouring errors.

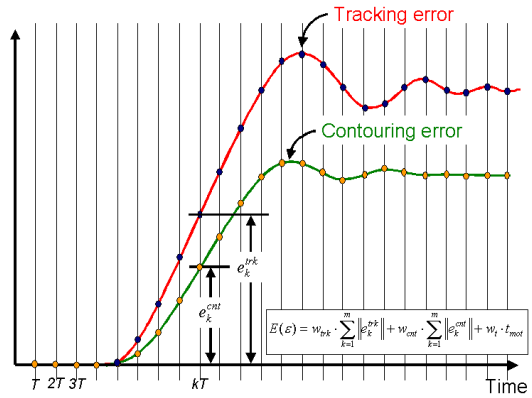


Fig. 2: Objective function and motion errors of the applied motion plant.

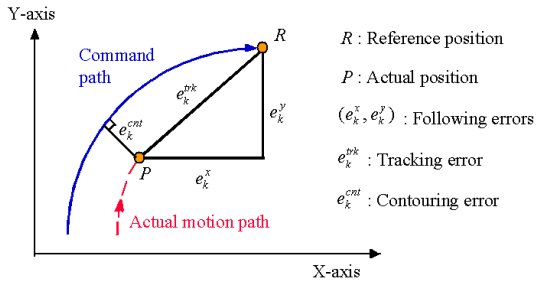


Fig. 3: Tracking error and contouring error.

As shown in Fig. 3, the tracking error e_k^{trk} at the k^{th} sampling instance is obtained by

$$e_k^{trk} = \sqrt{(e_k^x)^2 + (e_k^y)^2} \tag{3.2}$$

where e_k^x and e_k^y are the following errors of the motions on the X-axis and Y-axis at the k^{th} sampling instance, respectively. However, the computation of the contouring error e_k^{cnt} at the k^{th} sampling instance is not direct; thus, an estimation method is required. As shown in Fig. 4, the contouring error

vector, $\vec{e}_k^{cnt}$, is defined as a vector from the actual position P to the nearest point on the command path. The tracking error vector, $\vec{e}_k^{trk}$, is defined as a vector from the actual position P to the reference position R. $\vec{t}_k$ and $\vec{n}_k$ are the tangent vector and the normalized normal vector of the command path at reference position P, respectively. The estimated contouring error vector, $\hat{\vec{e}}_k^{cnt}$, is the vector from the actual position P to the nearest point on the line passing through the reference position P with the tangent $\vec{t}_k$. Therefore, by simple geometric relationships, the magnitude of the estimated contouring error vector, $\hat{e}_k^{cnt}$, is obtained by

$$\hat{e}_k^{cnt} = \left\| \hat{\vec{e}}_k^{cnt} \right\| = \left\| < \vec{e}_k^{trk}, \vec{n}_k > \cdot \vec{n}_k \right\| \tag{3.3}$$

where $<\cdot,\cdot>$ is an inner product operator. Therefore, by using Eqn. (3.2) and Eqn. (3.3), the objective function $E(\varepsilon)$ as in Eqn. (3.1) is modified as

$$E(\varepsilon) = w_{trk} \cdot \sum_{k=1}^{m} e_k^{trk} + w_{cnt} \cdot \sum_{k=1}^{m} \hat{e}_k^{cnt} + w_t \cdot t_{mot} . \tag{3.4}$$

Since m is the number of sampled data, the motion time t_{mot} is obtained as

$$t_{mot} = m \cdot T . \tag{3.5}$$

The summation of the tracking errors, $\sum_{k=1}^{m} e_k^{trk}$, represents the closeness between the reference position and the actual position; the summation of the contouring error, $\sum_{k=1}^{m} \hat{e}_k^{cnt}$, represents the closeness between the command path and the actual motion path; and the motion time t_{mot} represents the time spent during motion. Therefore, the weights, w_{trk}, w_{cnt}, and w_t, can be used to adjust the importance of each characteristic during tuning processes.

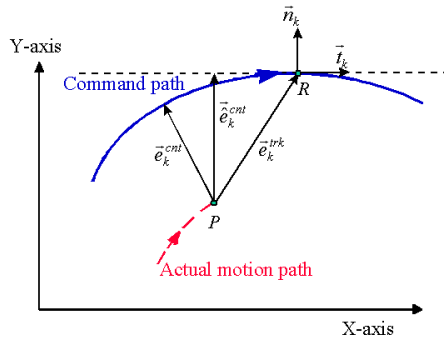


Fig. 4: Estimation method for contouring error.

There are some tuning coefficients, η, c, and d, in the applications of the learning automata tuning method. The tuning coefficients η and c are usually assigned such that the objective function has the same value scale as the tuned parameter. To avoid the error induced by division by zero (when the value of the objective function is zero), the d coefficients are set to be nonzero. However, in practical applications, these tuning coefficients are set based on the experience of the designers and the characteristics of the tuned motion plants.

4. SIMULATION AND EXPERIMENT

Fig. 5 shows the experimental setup of a die bonder machine with two 3-phase Panasonic AC servo motor packs, which was used to test the proposed approach. As shown in Fig. 5, a wafer that is already

cut is placed on the wafer-table such that the suction gripper attached to a moving arm can lift off a die and place it on the prepared lead frame. The motion of the moving arm must be rapid enough for increasing throughput. The applied motion control system mainly consisted of an industrial personal computer (PC) and a DSP-based motion control card. The industrial PC, which had a Pentium IV 2.8 GHz CPU, performed various functions, such as the interface between human and machine operations, the interpreter for motion codes, and the central processor for handling pick-and-place procedures. The DSP-based motion control card, which had a high-performance TI TMS320C32 digital signal processor (DSP), was used to implement the proposed approach in this study, to generate motion commands, and to record signals, such as the motion commands for controllers, position outputs, and the driving forces for the applied AC servo motor packs. The sampling time for motion control and the time step for interpolation were limited to 1 ms. Tab. 1 lists the servo parameters of the motion control system (shown in Fig. 5). K_c and K_e are the electrical gain and mechanical gain of the applied servo system, respectively, and τ is the time constant. K_p and K_d are the proportional gain and derivative gain of the applied PD+ motion controller [29], respectively.

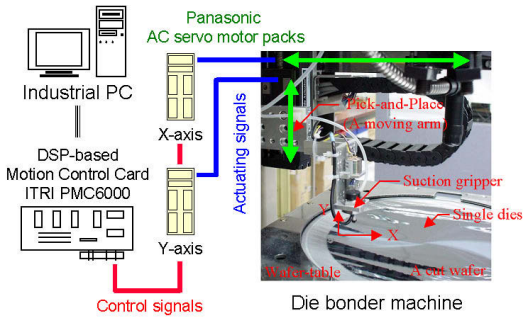


Fig. 5: Experimental setup—a die bonder machine.

Parameter \ Axis	X	Y
τ	0.31	0.135
K_c	2506.17	3156.23
K_e	1591.5494	1591.5494
K_p	0.0136	0.00577
K_d	0.03	0.028

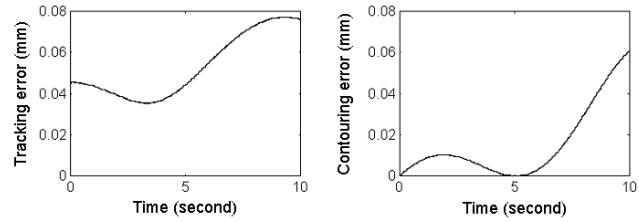
Tab. 1: Servo parameters of the applied motion control system.

In order to test the feasibility of the tuning method using the learning automata method, in this simulation, the learning automata tuning method was applied to tune the interpolation parameter ε of a biaxial motion control system with the servo parameters shown in Tab. 1. Fig. 1 shows the block diagram of the simulation system used in this study. The learning automata tuning method was applied to tune the interpolation parameter ε such that the motion control system performed motions with suitable motion errors and motion time. The objective function was defined as shown in Eqn. (3.1). The weights, w_{trk}, w_{cnt}, and w_t, were set as $w_{trk} = 1.0$, $w_{cnt} = 1.0$, $w_t = 50.0$. The initial value of the interpolation parameter ε was set as $\varepsilon = 0.0001$. The command path for testing the tuning method in this simulation was a circular contour with a 50 mm radius. Fig. 6 and Fig. 7 show the simulation results. Fig. 6(a) shows the tracking and contouring errors of the applied motion control system while the interpolation parameter $\varepsilon = 0.0001$. Obviously, although the motion control system had good tracking and contouring results, a long motion time degraded the motion performance. The maximum value of the tracking error was 0.077 mm, and the maximum value of the contouring error was 0.061 mm. The motion time was 10 seconds. By applying the learning automata tuning method, $\varepsilon = 0.0095$ was obtained as the tuned value of the interpolation parameter. Fig. 6(b) shows the tracking and

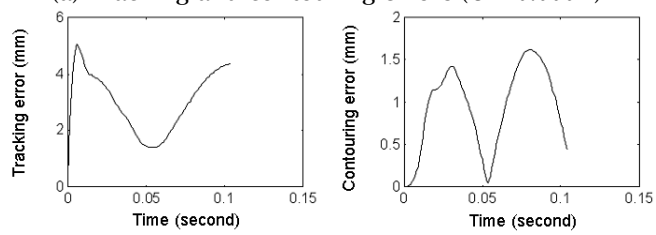
contouring errors of the applied motion control system when the tuned value was set as the interpolation parameter of the applied interpolation algorithm. The motion performance was obviously changed by applying the tuning result. Since the weight of the motion time was far larger than the tracking and contouring error weights, the goal for the applied tuning process was to find an interpolation parameter that allows the applied motion control system to achieve rapid motions. The maximum value of the tracking error became 5.042 mm, and the maximum value of the contouring error became 1.621 mm. The motion time became 0.105 seconds.

Fig. 7 shows the variations in the tuning parameter and the value of the objective function during the tuning process. The value of the objective function is decreasing, and reaches the final value after 74 iterations. The interpolation parameter ε was also varied in order to minimize the value of the objective function. Clearly, increasing the interpolation parameter can significantly shorten the motion time of the applied motion control system. However, this increase in the interpolation parameter also increases the motion errors, as shown in Fig. 6. According to the simulation results, the learning automata tuning method, with its fast learning and simple structure, can be applied to obtain a tuning result with a good trade-off between the motion errors and motion time of the applied motion control system.

In the experiment, the learning automata tuning method was applied to a die bonder machine system in order to tune the interpolation parameter of the applied motion control system, as shown in Fig. 5. The command path applied in this experiment was described by an NURBS function, as shown in Fig. 8.



(a) Tracking and contouring errors ($\varepsilon = 0.0001$).



(b) Tracking and contouring errors ($\varepsilon = 0.0095$).

Fig. 6: Simulation results.

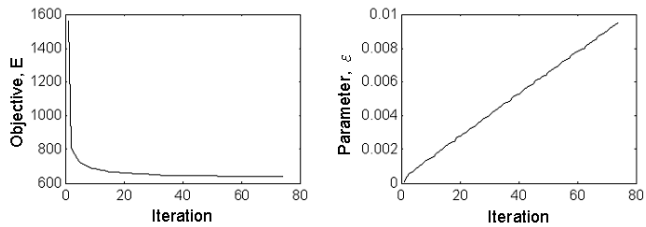


Fig. 7: Variations in the tuning parameter and objective function value (simulation).

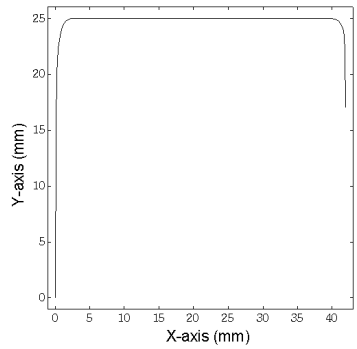


Fig. 8: The applied command path in the experiment.

Fig. 9 shows the tuning process for the applied learning automata tuning method and the experimental results of the applied motion control system using the tuned interpolation parameter. Fig. 9(a) and Fig. 9(b) show the tuning process. The objective function reached a final value after 40 iterations and 0.0033 was obtained as the corresponding interpolation parameter. Fig. 9(c) and Fig. 9(d) show the tracking and contouring errors when applying the tuned interpolation parameter. Although the maximum tracking error was 5.16 mm and the maximum contouring error was 2.034 mm, the motion time was significantly reduced, from 1.0 second to 0.3 second.

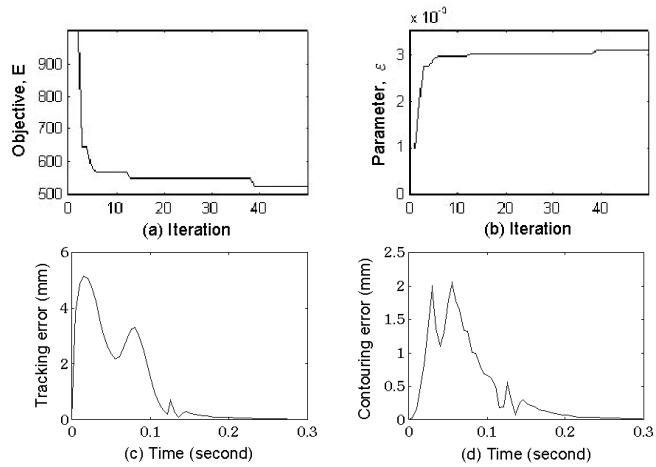


Fig. 9: The tuning process and the motion results.

5. CONCLUSION

In this paper, we investigated the use of the learning automata tuning method for tuning the interpolation parameter of an interpolation algorithm in order to interpolate the motion command paths described by parametric curve functions for motion control systems. Interpolation is important for motion control systems because it directly affects the motion commands for controlling the motions of the applied motion plant, and an improper value for the interpolation parameter in an interpolation algorithm usually deteriorates the motion results. Therefore, it is necessary to design an interpolation parameter that allows the applied motion control system to execute good motion performances, including small motion errors and short motion time. Although some existing researches have demonstrated important results, some factors, such as mechanical factors and manufacturing factors, still limit the executions of those algorithms in real applications. Therefore, this study considered a tuning method for tuning the interpolation parameter that allows the applied motion control system to execute good motions in the presence of the perturbations caused by

mechanical factors and factors in the manufacturing processes. The learning automata tuning method was used in this study for the following reasons:

- The tuning process operates through interactions with unknown environments using a stochastic trial and error process.
- The tuning method provides additional convergence information through probability density functions.
- The tuning process is easy to implement and the computational cost is low.

Some motion tests were performed on a die bonder machine to test the proposed approach. The experimental results showed that it is feasible to use the learning automata tuning method for tuning the interpolation parameter of an interpolation algorithm and that this allowed the motion control system of the applied die bonder machine to achieve a good trade-off between motion accuracy and motion time.

6. REFERENCES

[1] Bedi, S.; Ali, I.; Quan, N.: Advanced techniques for CNC machines, Journal of Engineering for Industry, Transactions of the ASME, 115, 1993, 329-336.

[2] Bush, R. R.; Mosteller, F.: Stochastic models for learning, New York Wiley, 1958.

[3] Cheng, C.-W.; Tsai, M.-C.: Real-time variable feed rate NURBS curve interpolator for CNC machining, International Journal of Advanced Manufacturing Technology, 23(11-12), 2004, 865-873.

[4] Cheng, C.-W.; Tsai, M.-C.; Maciejowski, J.: Feedrate control for non-uniform rational B-spline motion command generation, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 220(11), 2006, 1855-1861.

[5] Cheng, C.-W.; Tseng, W.-P.: Design and implementation of a real-time NURBS surface interpolator, International Journal of Advanced Manufacturing Technology, 30(1-2), 2006, 98-104.

[6] Choi, I.-H.; Yang, M.-Y.; Hong, W.-P.; Jung, T.-S.: Curve interpolation with variable feedrate for surface requirement, International Journal of Advanced Manufacturing Technology, 25(3-4), 2005, 325-333.

[7] Choubey, A. M.; Chan, F. T. S.; Tiwari, M. K.: Solving a fixture configuration design problem using genetic algorithm with learning automata approach, International Journal of Production Research, 43(22), 2005, 4721–4743.

[8] Chtourou, M.; Jemaa, M. B.; Ketata, R.: Learning-automaton-based method for fuzzy inference system identification, International Journal of Systems Science, 28(9), 1997, 889–896.

[9] Du, D.; Liu, Y.; Yan, C.; Li, C.: An accurate adaptive parametric curve interpolator for NURBS curve interpolation, International Journal of Advanced Manufacturing Technology, 32(9-10), 2007, 999-1008.

[10] Ko, T. J.; Kim, H. S.; Park, S. H.: Machineability in NURBS interpolator considering constant material removal rate, International Journal of Machine Tools and Manufacture, 45(6), 2005, 665-671.

[11] Kuo, L.-Y.; Yen, J.-Y.: A genetic algorithm-based parameter-tuning algorithm for multi-dimensional motion control of a computer numerical control machine tool, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 216(3), 2002, 429–438.

[12] Kuo, L.-Y.; Yen, J.-Y.: Servo parameter tuning for a 5-axis machine center based upon GA rules, International Journal of Machine Tools and Manufacture, 41(11), 2001, 1535–1550.

[13] Lee, K.; Ibaraki, S.; Matsubara, A.; Kakino, Y.; Suzuki, Y.; Arai, S.; Braasch, J.: A servo parameter tuning method for high-speed NC machine tools based on contouring error measurement, Laser Metrology and Machine Performance VI, Laser Metrology and Machine Performance VI, 2003, 181–192.

[14] Lin, M.-T.; Tsai, M.-S.; Yau, H.-T.: Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm, International Journal of Machine Tools and Manufacture, 47(15), 2007, 2246-2262.

[15] Liu, X.; Ahmad, F.; Yamazaki, K.; Mori, M.: Adaptive interpolation scheme for NURBS curves with the integration of machining dynamics, International Journal of Machine Tools and Manufacture, 45(4-5), 2005, 433-444.

[16] Luce, R. D.: Individual choice behavior, New York Wiley, 1959.

[17] Mashoufi, B.; Menhaj, M. B.; Motamedi, S. A.; Meybodi, M. R.: Introducing an adaptive VLR algorithm using learning automata for multilayer perceptron, IEICE Transactions on Information and Systems, E86-D(3), 2003, 594–609.
[18] Narendra, K. S.; Viswanathan, R.: Learning models using stochastic automata, Proceedings of 1972 International Conference on Cybernetics and Society, 1972, 9–12.
[19] Narendra, K.; Thathachar, M. A. L.: Learning automata: An introduction, Prentice Hall, 1989.
[20] Poo, A.; Bollinger, J. G.; Younkin, W.: Dynamic error in type contouring systems, IEEE Transactions on Industry Application, IA-8(4), 1972, 477–484.
[21] Qi, S. R.; Wang, D. F.; Han, P.; Li, Y. H.: Grey prediction based RBF neural network self-tuning PID control for turning process, Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2, 2004, 802–805.
[22] Tikhon, M.; Ko, T. J.; Lee, S. H.; Sool Kim, H.: NURBS interpolator for constant material removal rate in open NC machine tools, International Journal of Machine Tools and Manufacture, 44(2-3), 2004, 237-245.
[23] Tsai, M.-C.; Cheng, C.-W.; Cheng, M.-Y.: A real-time NURBS surface interpolator for precision three-axis CNC machining, International Journal of Machine Tools and Manufacture, 43(12), 2003, 1217-1227.
[24] Tsai, M.-C.; Chung, C.-W.: A real-time predictor-corrector interpolator for CNC machining, Journal of Manufacturing Science and Engineering, Transactions of the ASME, 125(3), 2003, 449-460.
[25] Tsypkin, Y. Z.: Adaptation and learning in automatic systems, New York Academic, 1971.
[26] Yan, C.; Du, D.; Li, C.: Design of a real-time adaptive interpolator with parameter compensation, International Journal of Advanced Manufacturing Technology, 35(1-2), 2007, 169-178.
[27] Yau, H.-T.; Kuo, M.-J.: NURBS machining and feed rate adjustment for high-speed cutting of complex sculptured surfaces, International Journal of Production Research, 39(1), 2001, 21-41.
[28] Yau, H.-T.; Wang, J.-B.; Hsu, C.-Y.; Yeh, C.-H.: PC-based controller with real-time look-ahead NURBS interpolator, Computer-Aided Design and Applications, 4(1-6), 2007, 331-340.
[29] Yeh, S.-S.; Sun, J.-T.: NURBS Interpolation for Motion Systems with Actuator Saturation, Computer-Aided Design and Applications, 5(6), 2008, 801-810.
[30] Yeh, S.-S.; Hsu, P.-L.: Adaptive-feedrate interpolation for parametric curves with a confined chord error, Computer Aided Design, 34(3), 2002, 229-237.
[31] Yeh, S.-S.; Hsu, P.-L.: Speed-controlled interpolator for machining parametric curves, Computer Aided Design, 31(5), 1999, 349-357.
[32] Zhiming, X.; Jincheng, C.; Zhengjin, F.: Performance evaluation of a real-time interpolation algorithm for NURBS curves, International Journal of Advanced Manufacturing Technology, 20(4), 2002, 270-276.