A Model-Free Predictive Control Method Based on Polynomial Regression

A Model-Free Predictive Control Method Based on Polynomial Regression

Hongran Li[1†] and Shigeru Yamamoto[2]

[1]Graduate School of Natural Science and Technology, Kanazawa University,
Kakuma, Kanazawa, Ishikawa 920-1192, Japan
(Tel: +81-76-264-6332; E-mail: li@moccos.ec.t.kanazawa-u.ac.jp)
[2]Faculty of Electrical and Computer Engineering, Kanazawa University,
Kakuma, Kanazawa, Ishikawa 920-1192, Japan
(E-mail: shigeru@se.kanazawa-u.ac.jp)

Abstract: This paper proposes a model-free predictive control method for nonlinear systems on the basis of polynomial regression. In contrast to conventional model predictive control, model-free predictive control does not require mathematical models. Instead, it uses the previous recorded input/output datasets of the controlled system to predict an optimal control input so as to achieve the desired output. The novel point in this paper is the improvement of existing model-free predictive control by adopting polynomial regression, which is a generalization of the so-called Volterra series expansion of nonlinear functions.

Keywords: data-driven control, predictive control, just-in-time modeling, polynomial regression, Volterra series.

1. INTRODUCTION

Model-free predictive control is data-driven control that does not explicitly require any mathematical model [1]-[10]. In contrast to standard model predictive control utilizing mathematical models, the model-free predictive control method uses past records of input and output datasets and the current input and output to predict future input and output. The underlying principle is Just-In-Time modeling, which was originally proposed in [11]-[14]; this model aims to adaptively obtain a local linear model using both online measured input/output data and past data [12], [13]. Just-In-Time modeling is also referred to as model on-demand [14], [15], lazy learning [16], or instance-based learning [17]. There are several applications of Just-In-Time modeling including prediction of production processes in the steel industry [18]-[21], PID parameter tuning [22], [23], and soft sensors in industrial chemical processes [24]. In [25], Just-In-Time modeling is also utilized for predictive control; however, only identified local linear models are used for predictive control similar to that in standard model predictive control.

Purely model-free predictive control with no model usage was proposed in [1]-[3]. It basically uses input and output sequences that are cut out into short-length vectors. Although the vectors can be used to identify an auto-regressive model, they are instead used to estimate a short-length vector corresponding to future input sequences by using locally weighted averaging. The idea can also be seen in [4] and can be used to treat discretized input systems [5]. It has also been applied to an inverted pendulum system [6] and a parallel mechanism with pneumatic drives [7]. Recently, in [8], [9] it has been pointed out that locally weighted averaging can be replaced with optimization under a linear algebraic equation that relates to least-norm solutions and ℓ_1 minimization. This yields us a mathematically much simpler model-free predictive control algorithm. The effectiveness of the simplified algorithms is investigated in [10].

So far, model-free predictive control presumes that the controlled system can be locally linearized. Therefore, the short-length vectors linearly contain cut out input and output sequences as a regressor vector of the auto-regressive model. In this paper, to treat nonlinearity in detail, we adopt a polynomial regressor vector for the short-length vectors. We first review a polynomial regression model together with a Volterra series model [26], [27] in Section 2. We also review model-free predictive control and extend it with polynomial regression in Section 3. Section 4 illustrates numerical simulations results, and Section 5 provides the concluding remarks.

2. VOLTERRA AND POLYNOMIAL REGRESSION MODELS

In this section, we review a Volterra model and a polynomial regression model [26], [27].

The so-called Volterra model is a general nonlinear model with an output $y(t)$ and an input $u(t)$ that can be expressed as follows:

$$y(t) = \sum_{p=0}^{P} H_p(\mathbf{x}_1(t)) + e(t) \tag{1}$$

where e is independent and identically distributed noise

$$\mathbf{x}_1(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_L(t) \end{bmatrix} \in \mathbf{R}^L, \tag{2}$$

$$x_i(t) = u(t-i), \quad i = 1, \ldots, L, \tag{3}$$

† Hongran Li is the presenter of this paper.

and

$$H_p(\mathbf{x}_1(t)) = \sum_{i_1=1}^{L} \cdots \sum_{i_p=1}^{L} h_p(i_1, \ldots, i_p) \times \prod_{k=1}^{p} u(t - i_k) \tag{4}$$

where each $h_p(i_1 \cdots i_p)$ is called a Volterra kernel of the system. In general, the expansion order P is infinity. Here, we consider the truncated model, that is, for $p > P$, $|H_p(\mathbf{x}_1(t))|$ is sufficiently small. For $p > 1$, by defining the pth order monomials (homogeneous) regressor vector

$$\mathbf{x}_p(t) = \mathbf{x}_{p-1}(t) \otimes \mathbf{x}_1(t) \in \mathbf{R}^{L^p}, \tag{5}$$

where $\otimes$ denotes the tensor (Kronecker) product; we can rewrite (4) as

$$H_p(\mathbf{x}_1(t)) = \mathbf{x}_p^\top(t)\mathbf{h}_p, \tag{6}$$

where $\mathbf{h}_p$ is a vector containing Volterra kernels $h_p(i_1 \cdots i_p)$. By using (4), another expression of (1) is given as

$$y(t) = \phi^\top(t)\mathbf{h} + e(t), \tag{7}$$

where

$$\phi^\top(t) = \begin{bmatrix} 1 & \mathbf{x}_1^\top(t) & \cdots & \mathbf{x}_P^\top(t) \end{bmatrix} \tag{8}$$

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_P \end{bmatrix} \in \mathbf{R}^{\sum_{p=0}^{P} L^p} \tag{9}$$

By changing the meaning of the index t in (2) so as to neither limit the time nor restrict $x_i(t) = u(t - i)$, we obtain a polynomial regression model. Since we can set $x_i(t) = u(t - i)$, the Volterra model is a special polynomial regression model.

To use polynomial regression, we define another form of the Volterra model. That is, we define

$$\mathbf{x}_1(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_m(t) \\ x_{m+1}(t) \\ \vdots \\ x_L(t) \end{bmatrix} = \begin{bmatrix} u(t-1) \\ \vdots \\ u(t-m) \\ y(t-1) \\ \vdots \\ y(t-n) \end{bmatrix} \tag{10}$$

and $L = m + n$. By adopting this model, we can reduce the truncated order P.

Since the tensor product is a particularly effective method to establish the topological vector space, it yields several duplicate terms. By eliminating these duplicate terms, we define the pseudo-tensor product $\tilde{\otimes}$, for instance,

$$\begin{bmatrix} a & b \end{bmatrix} \tilde{\otimes} \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} a^2 & ab & b^2 \end{bmatrix}, \tag{11}$$

by unifying the duplicated term ab in $\begin{bmatrix} a & b \end{bmatrix} \otimes \begin{bmatrix} a & b \end{bmatrix}$.

When we use the pseudo-tensor product in (5) as

$$\mathbf{x}_p(t) = \mathbf{x}_{p-1}(t)\tilde{\otimes}\mathbf{x}_1(t) \in \mathbf{R}^{\binom{L+p-1}{p}}. \tag{12}$$

the size of ϕ and $\mathbf{h}$ can be reduced to

$$\sum_{p=0}^{P} \binom{L+p-1}{p} = \binom{L+P}{P}. \tag{13}$$

3. MODEL-FREE PREDICTIVE CONTROL

In this subsection, we first summarize the model-free predictive control algorithm. Then we extend it to handle polynomial regression. Subsequently, we assume a system that can be approximated by (1).

3.1. Linear regression case

The control objective is to make the h-step' output trajectory of the system

$$\mathbf{y}_f(t) = \begin{bmatrix} y(t+1) \\ \vdots \\ y(t+h) \end{bmatrix} \in \mathbf{R}^h \tag{14}$$

track the desired reference

$$\mathbf{r}(t) = \begin{bmatrix} r(t+1) \\ \vdots \\ r(t+h) \end{bmatrix} \in \mathbf{R}^h. \tag{15}$$

To achieve the control objective, we predict an h-step' future input sequence

$$\mathbf{u}_f(t) = \begin{bmatrix} u(t) \\ \vdots \\ u(t+h-1) \end{bmatrix} \in \mathbf{R}^h. \tag{16}$$

Model-free predictive control [1] uses the large amount of past data $\{u(t), y(t)\}$ to determine the future input sequence. In particular, it uses for $j = 1, 2, \ldots, N$,

$$\mathbf{a}_j = \begin{bmatrix} \mathbf{y}_p(t_j) \\ \mathbf{y}_f(t_j) \\ \mathbf{u}_p(t_j) \end{bmatrix} \in \mathbf{R}^{n+m+h} \tag{17}$$

$$\mathbf{c}_j = \mathbf{u}_f(t_j) \in \mathbf{R}^h, \tag{18}$$

where

$$\mathbf{y}_p(t) = \begin{bmatrix} y(t-n+1) \\ \vdots \\ y(t) \end{bmatrix} \in \mathbf{R}^n. \tag{19}$$

$$\mathbf{u}_p(t) = \begin{bmatrix} u(t-m) \\ \vdots \\ u(t-1) \end{bmatrix} \in \mathbf{R}^m. \tag{20}$$

Algorithm 1 Model-free predictive control algorithm

Determine n, m, N, h, and the order P. Construct $\mathbf{a}_j$ and $\mathbf{c}_j$ ($j = 1, \ldots, N$). $t \leftarrow 0$.
while $t \leq \max(n, m)$ **do**
  Measure $y(t)$ and apply $u(t)$ with an appropriate value. Increment the time as $t \leftarrow t + 1$.
end while
repeat
  Construct a query vector $\mathbf{b}$.
  Solve (28).
  Compute (23) to obtain $\hat{\mathbf{u}}_f(t)$.
  Apply $u(t) := \hat{u}(t|t)$ to the system.
  $t \leftarrow t + 1$
until a terminate condition is met.

Model-free predictive control [1], [2] utilizes a query vector

$$\mathbf{b} = \begin{bmatrix} \mathbf{y}_p(t) \\ \mathbf{r}(t) \\ \mathbf{u}_p(t) \end{bmatrix} \in \mathbf{R}^{n+m+h} \tag{21}$$

to synthesize

$$\hat{\mathbf{u}}_f(t) = \begin{bmatrix} \hat{u}(t|t) \\ \vdots \\ \hat{u}(t+h-1|t) \end{bmatrix} \tag{22}$$

$$= \mathbf{c}\mathbf{w} \in \mathbf{R}^h, \tag{23}$$

where

$$\mathbf{c} = \begin{bmatrix} \mathbf{c}_1 & \ldots & \mathbf{c}_N \end{bmatrix} \in \Re^{h \times N} \tag{24}$$

$$\mathbf{w} = \begin{bmatrix} w_1 & \ldots & w_N \end{bmatrix}^\top \in \Re^N. \tag{25}$$

The first element $\hat{u}(t|t)$ of $\hat{u}_f(t)$ is only applied into the system as $u(t)$.

The vector $\mathbf{w}$ is originally determined by using the Akaike's final prediction error criterion [1]-[3]. In [8], $\mathbf{w}$ is derived as a least-norm solution of

$$A\mathbf{w} = \mathbf{b}, \tag{26}$$

where

$$A = \begin{bmatrix} \mathbf{a}_1 & \ldots & \mathbf{a}_N \end{bmatrix} \in \Re^{(n+m+h) \times N}. \tag{27}$$

In [9], $\mathbf{w}$ is found by solving an ℓ_1-minimization problem:

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 \text{ subject to } A\mathbf{w} - \mathbf{b} = 0. \tag{28}$$

Many algorithms to solve the ℓ_1-minimization problem have been proposed [28].

The fundamental procedure is summarized as in Algorithm 1.

3.2. Polynomial regression case

We can extend the model-free predictive control method in the previous subsection to the higher order polynomial regression case. First, we must define the vectors

$$\mathbf{a}_1(j) = \begin{bmatrix} \mathbf{y}_p(t_j) \\ \mathbf{y}_f(t_j) \\ \mathbf{u}_p(t_j) \end{bmatrix}, \tag{29}$$

$$\mathbf{b}_1 = \begin{bmatrix} \mathbf{y}_p(t) \\ \mathbf{r}(t) \\ \mathbf{u}_p(t) \end{bmatrix} \tag{30}$$

$$\mathbf{c}_1(j) = \mathbf{u}_f(t_j) \tag{31}$$

$$\mathbf{a}_p(j) = \mathbf{a}_{p-1}(j) \tilde{\otimes} \mathbf{a}_1(j) \tag{32}$$

$$\mathbf{b}_p = \mathbf{b}_{p-1} \tilde{\otimes} \mathbf{b}_1 \tag{33}$$

$$\mathbf{c}_p(j) = \mathbf{c}_{p-1}(j) \tilde{\otimes} \mathbf{c}_1(j) \tag{34}$$

and construct

$$\mathbf{a}_j = \begin{bmatrix} \mathbf{a}_1(j) \\ \vdots \\ \mathbf{a}_P(j) \end{bmatrix} \tag{35}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_P \end{bmatrix} \tag{36}$$

$$\mathbf{c}_j = \begin{bmatrix} \mathbf{c}_1(j) \\ \vdots \\ \mathbf{c}_P(j) \end{bmatrix}. \tag{37}$$

By using these vectors and defining A in (27) and $\mathbf{b}$ in (24), we can use Algorithm 1.

In practical computation, we must introduce a scaling matrix S to avoid blow-up of high-order exponentiation in polynomial regression as

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 \text{ subject to } SA\mathbf{w} - S\mathbf{b} = 0. \tag{38}$$

4. SIMULATION

In this section, we illustrate several simulation results to show the effectiveness of the proposed method. Throughout the simulations, we used the square signal reference

$$r(t) = \begin{cases} 0 & 200k \leq t < 50 + 200k \\ 1 & 50 + 200k \leq t < 100 + 200k \\ 0 & 100 + 200k \leq t < 150 + 200k \\ -1 & 150 + 200k \leq t < 200 + 200k \end{cases}$$
$$k = 0, 1, \ldots \tag{39}$$

4.1. Linear System

We first used the linear system

$$y(t) - 1.7y(t-1) + 0.72y(t-2) = 0.1u(t-1) + 0.2u(t-2) + e(t) \tag{40}$$

with stable poles 0.9 and 0.8 and an unstable zero -2 [29]. To apply a random sequence $e(t)$ according to a Gaussian distribution with zero mean, variance 0.001^2,

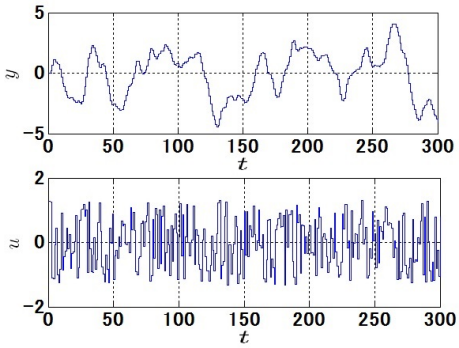

Fig. 1 Stored measurement data of the linear system (40) for model-free predictive control

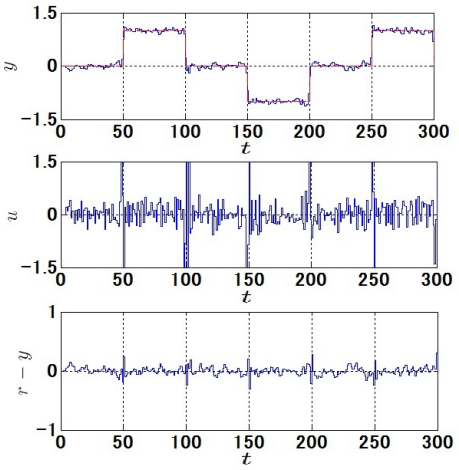

Fig. 2 Simulation result of model-free predictive control for the linear system (40)

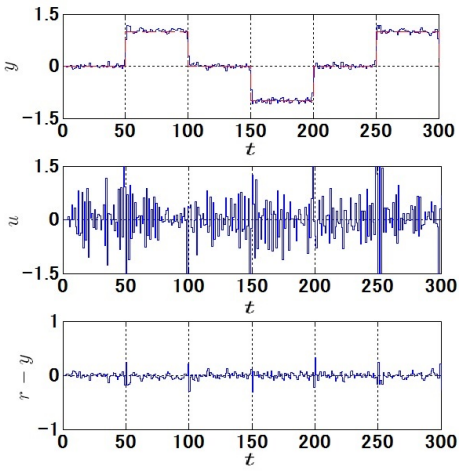

Fig. 3 Simulation result of model-free predictive control with the overestimated order for the linear system (40)

and a random sequence $u(t)$ generated from a uniform distribution $[-2, 2]$, we prepared a dataset containing samples ($N = 300$) of $u(t)$ and $y(t)$, as shown in Fig. 1.

By using parameters for model-free predictive control $m = 3$, $n = 2$, $P = 3$, and $h = 2$ under the noisy condition $e(t) \sim \mathcal{N}(0, 0.001^2)$, we obtained the control result shown in Fig. 2. It shows that the output y can track the reference r.

Next, we used an overestimated order $m = 3$, with

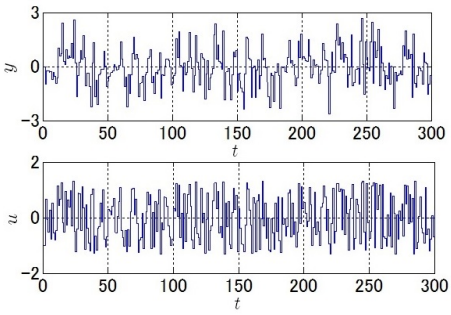

Fig. 4 Stored measurement data of the nonlinear system (41) for model-free predictive control to obtain the control result in Fig. 5

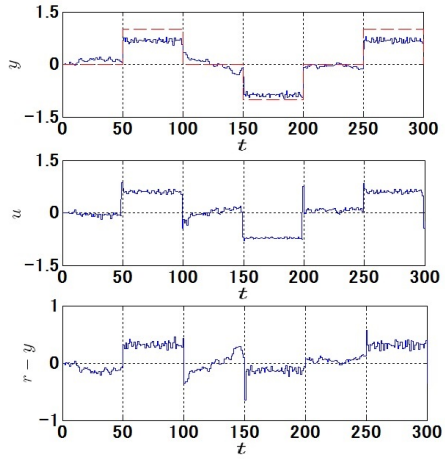

Fig. 5 Simulation result of model-free predictive control for the nonlinear system (41)

other parameters being the same as before, i.e., $n = 2$, $P = 3$, and $h = 2$, and obtained the control result shown in Fig. 3; this is similar to that shown in Fig. 2.

From the two results (Fig. 2 and 3), we see that the proposed method can achieve the desired control performance even when the order of the system is overestimated.

4.2. Nonlinear system

We used the nonlinear system

$$y(t+1) = \frac{y(t)}{1 + y(t)^2} + u(t)^3 + e(t). \tag{41}$$

To obtain a dataset, we apply a random sequence $e(t)$ according to a Gaussian distribution $e(t) \sim \mathcal{N}(0, 0.001^2)$ with zero mean and variance 0.001^2, and $u(t)$ according to a uniform distribution $[-2, 2]$. When we used a dataset containing samples ($N = 300$) of generated $u(t)$ and $y(t)$, as shown in Fig. 4, and parameters $n = 2$, $m = 2$, $P = 2$, and $h = 2$, we obtained a very poor control result (Fig. 5). This may be because considered that only a few values exist in the dataset close to the reference $r = -1, 0$, and 1 (Fig. 6).

To gather $y(t)$ around the reference r, we used PI control only when a dataset was generated. When we use PI

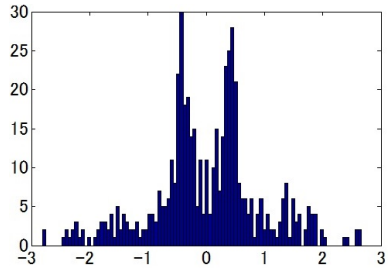

Fig. 6 Histogram of values of output y in the dataset used to obtain the control result in Fig. 5

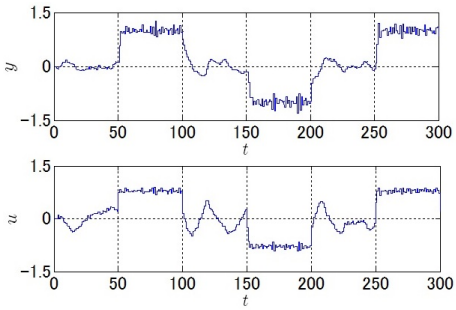

Fig. 7 Stored measurement data of the nonlinear system (41) to obtain the control result in Fig. 9

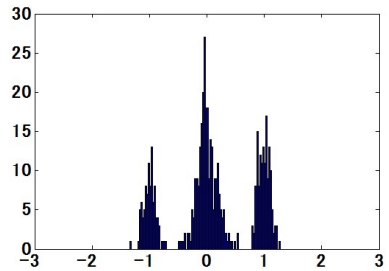

Fig. 8 Histogram of values of output y in the dataset used to obtain the control result in Fig. 9

control

$$u(t) = K_p\epsilon(t) + K_i \sum_{\tau=-\infty}^{t} \epsilon(\tau) \tag{42}$$

$$\epsilon(t) = r(t) - y(t) \tag{43}$$

with the proportional gain $K_p = 0.6$ and the integral gain $K_i = 0.4$, we obtained the input/output shown in Fig. 7 and the histogram of y shown in Fig. 8. In Fig. 7, we see that y roughly tracks r, and there exists much more y around $r = -1, 0, 1$ in Fig. 8 than in Fig. 6. When using this dataset, we obtain a better control result with only model-free predictive control (in this case PI control is not used), as shown in Fig. 9. It shows that the output y can properly track the reference r. From this result, we see that the control performance depends on the dataset.

5. CONCLUSION

In this paper, we examined model-free predictive control using polynomial regression, which is a generalization of the Volterra series. Without estimating the coefficients of polynomial regression (Volterra series), an appropriate control input can be determined by using a

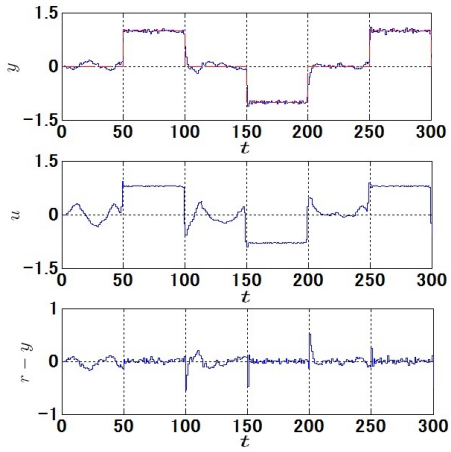

Fig. 9 Simulation result of model-free predictive control for the nonlinear system (41) with the datasets in Fig. 7

dataset containing the input/output data of the controlled system. The obtained control performance depends on the dataset; hence, maintaining a rich dataset is important, that is, the dataset must contain input/output data that is near the desired output. In simulations, we used PI control to maintain the dataset. However, in model-free predictive control, once a rich dataset is obtained, such PI control is not needed. Thus, model-free predictive control yields better control results than PI control.

REFERENCES

[1] D. Inoue and S. Yamamoto, "An Operation Support System Based on Database-Driven On-Demand Predictive Control," *Proceedings of 2004 SICE Annual Conference*, pp. 2024–2027, 2004.

[2] D. Inoue and S. Yamamoto, "A Memory-Based Predictive Control Approach to a Braking Assist Problem," *Proceedings of 2005 SICE Annual Conference*, pp. 3186–3189, 2005.

[3] D. Inoue and S. Yamamoto, "Support for Drivers via Just-in-Time Predictive Control and Fault Detection Based on a Nearest Neighbor Method during Braking to Stop Trains," *Transactions of the Japan Society of Mechanical Engineers. C*, Vol. 72, pp. 2756–2761, 2006.

[4] K. Fukuda, S. Ushida, and K. Deguchi, "Just-in-Time Control of Image-Based Inverted Pendulum Systems with a Time-Delay," *Proc. SICE-ICASE 2006*, pp. 4016–4021, 2006.

[5] E. Konaka, "Design of Discrete Predictive Controller Using Approximate Nearest Neighbor Method," *Proc. the 18th IFAC World Congress*, pp. 10213–10218, 2011.

[6] N. Nakpong and S. Yamamoto, "Just-in-Time Predictive Control for a Two-Wheeled Robot," *Proc. 2012 Tenth Int. Conf. on ICT and Knowledge Engineering*, No. 3, pp. 95–98, 2012.

[7] T. Kosaki and M. Sano, "Networked Just-in-Time Control of a Parallel Mechanism with Pneumatic

Linear Drives," *Communication in Control Science and Engineering (CCSE)*, Vol. 2, pp. 19–25, 2014.

[8] S. Yamamoto, "A New Model-Free Predictive Control Method Using Input and Output Data," *3rd International Conference on Key Engineering Materials and Computer Science (KEMCS 2014)* (Advanced Materials Research Vol.1042, ed.), pp. 182–187, Trans Tech Publication, 2014.

[9] S. Yamamoto, "A Model-Free Predictive Control Method by l_1-minimization," *Proc. 10th Asian Control Conference 2015*, 1570064495, 2015.

[10] H. Saputra and S. Yamamoto, "Comparative Study of Model-Free Predictive Control and its Database Maintenance for Unstable Systems," SICE Journal of Control, Measurement, and System Integration, Vol. 8, No. 6, pp. 390–395, 2015.

[11] G. Cybenko, "Just-in-Time Learning and Estimation," *NATO ASI Series of Computer and Systems Sciences*, Vol. 153, pp. 423–434, 1996.

[12] A. Stenman, F. Gustafsson, and L. Ljung, "Just in Time Models for Dynamical Systems," *Proceedings of the 35th IEEE Conference* on Decision and Control, Vol. 1, pp. 1115–1120, 1996.

[13] A. Stenman, A. V. Nazin, and F. Gustafsson, "Asymptotic Properties of Just-in-Time Models," *Proc. 11th IFAC Symposium on System Identification*, pp. 1249–1254, 1997.

[14] A. Stenman, *Model on Demand: Algorithms , Analysis and Applications*. PhD thesis, Department of Electrical Engineering Linkoping University, 1999.

[15] M. W. Braun, D. E. Rivera, and A. Stenman, "A Model-on-Demand Identification Methodology for Nonlinear Process Systems," *International Journal of Control*, Vol. 74, No. 18, pp. 1708–1717, 2001.

[16] G. Bontempi, M. Birattari, and H. Bersini, "Lazy Learning for Local Modelling and Control Design," *International Journal of Control*, Vol. 72, No. 7-8, pp. 643–658, 1999.

[17] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-Based Learning Algorithms," *Machine Learning*, Vol. 6, No. 1, pp. 37–66, 1991.

[18] Q. Zheng and H. Kimura, "A New Just-in-Time Modeling Method and its Application to Rolling Set-Up Modeling," *Trans. of the Society of Instrument and Control Engineers*, Vol. 37, No. 2, pp. 640–646, 2001 (in Japanese).

[19] Q. Zheng and H. Kimura, "Just-in-Time Modeling for Function Prediction and its Application," *Asian Journal of Control*, Vol. 3, No. 1, pp. 35–44, 2001.

[20] M. Kishi, K. Kimura, J. Ohta, and S. Yamamoto, "Shrinkage Prediction of a Steel Production via Model-on-Demand," *Proc. the 11th IFAC Symposium on Automation in Mining, Mineral and Metal Processing*, pp. 447–450, 2004.

[21] H. Shigemori, M. Kano, and S. Hasebe, "Optimum Quality Design System for Steel Products Through Locally Weighted Regression Model," *Journal of Process Control*, Vol. 21, No. 2, pp. 293–301, 2011.

[22] J. Ohta and S. Yamamoto, "Database-Driven Tuning of PID Controllers," *Trans. of the Society of Instrument and Control Engineers*, Vol. 40, pp. 664–669, 2004 (in Japanese).

[23] T. Yamamoto, K. Takao, and T. Yamada, "Design of a Data-Driven PID Controller," *IEEE Transactions on Control Systems Technology*, Vol. 17, No. 1, pp. 29–39, 2009.

[24] K. Fujiwara, M. Kano, S. Hasebe, and A. Takinami, "Soft-Sensor Development Using Correlation-Based Just-in-Time Modeling," *AIChE Journal*, Vol. 55, No. 7, pp. 1754–1765, 2009.

[25] A. Stenman, "Model-free Predictive Control," *Proceedings of the 38th IEEE Conference on Decision and Control*, Vol. 4, pp. 3712–3717, 1999.

[26] M. O. Franz and B. Scholkopf, "A Unifying View of Wiener and Volterra Theory and Polynomial Kernel Regression," *Neural Computation*, Vol. 18, No. 12, pp. 3097–3118, 2006.

[27] P. Alper, "A Consideration of the Discrete Volterra Series," *IEEE Transactions on Automatic Control*, Vol. 10, No. 3, pp. 322–327, 1965.

[28] A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma, "l_1-minimization Algorithms for Robust Face Recognition," *IEEE Transactions on Image Processing*, Vol. 22, No. 8, pp. 3234-3246, 2013.

[29] D. W. Clarke, "Self-Tuning Control of Nonminimum-Phase Systems," *Automatica*, Vol. 20, No. 5, pp. 501–517, 1984.