Just-In-Time predictive control for a two-wheeled robot

doi: 10.1109/ICTKE.2012.6408578

Just-In-Time Predictive Control for a Two-Wheeled Robot

Nuttapun Nakpong
Graduate School of Natural Science and Technology
Kanazawa University
Kanazawa, Japan
E-mail: nuttapun@moccos.ec.t.kanazawa-u.ac.jp

Shigeru Yamamoto
Faculty of Electrical and Computer Engineering
Kanazawa University
Kanazawa, Japan
E-mail: shigeru@t.kanazawa-u.ac.jp

***Abstract*—We introduce the use of Just-In-Time predictive control to enhance the stability of a two-wheeled robot. Just-In-Time predictive control uses a database which includes a huge amounts of input-output data of the two-wheeled robot and predicts its future movements based on a Just-In-Time algorithm.**

***Keywords*—Just-In-Time predictive control, two-wheeled robot.**

I. INTRODUCTION

While there are many methods to control robots, feedback control is mostly used. To enhance standard feedback control, we propose to use a method called Just-In-Time predictive control which is data-driven control.

The basic principle of Just-In-Time predictive control is to find several similar phenomena in the database to the current observed one, to predict the future behavior based on the similar ones based on the Just-In-Time method [1], [2]. There are many applications related to Just-In-Time predictive control, for example, supporting for drivers via Just-In-Time predictive control and fault detection based on a nearest neighbor method during braking to stop the train [3], [4].

The robot in this paper is a two-wheeled robot which can move with just only two wheel by as an invert pendulum under stabilization by state feedback to keep its balance in both standing and moving. In general, it is not an easy task to obtain the appropriate state feedback gain for a good control performance. To obtain a satisfactory feedback gain, we need a precise model of the robot to design the feedback gain and/or iterative control experiments for gain tuning. Instead of them, in this paper, we additionally apply Just-In-Time predictive control to improve the performance by state feedback which is implemented for stabilization for the two-wheeled robot.

When we use Just-In-Time predictive control, alternative problem comes from the computational time for Just-In-Time predictive control and time-delay in the communication between the computer and the robot. In general, such a time-delay in feedback loop deteriorates the control performance and losses stability. To compensate it, we generally need the exact the time-delay to design the control system. However, when we use Just-In-Time predictive control, we do not explicitly know how long the time-delay is.

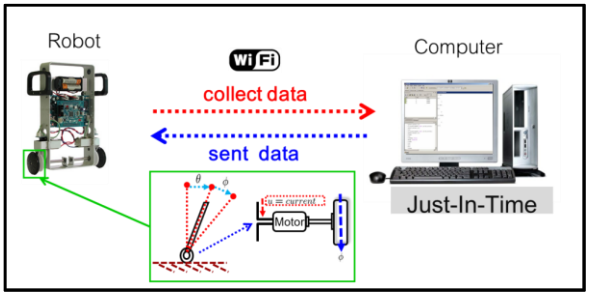

Figure 1. Work flow diagram of Just-In-Time predictive control for a robot.

II. JUST-IN-TIME PREDICTIVE CONTROL

Just-In-Time predictive control is predictive control which uses a database instead of a model. In the database, the input and output of the controlled plant are stored By using a Just-In-Time algorithm we can predict an appropriate control input to compare the current input-output data with that in the database.

We assume that when a plant is controlled with a sampling time T, it can be expressed by a nonlinear autoregressive exogenous (NARX) discrete-time model

$$y[t] = g(z[t]) + e[t], \tag{1}$$

where g() is a nonlinear function of a regressor vector

$$z[t] = \begin{pmatrix} y[t-[m-1]] \\ \vdots \\ y[t-1] \\ u[t-[n-1]] \\ \vdots \\ u[t] \end{pmatrix}, \tag{2}$$

u is the input, y is the output, and e is a modeling error. The regressor vector consists of an m step sequence of the output y and an n step sequence of the input u.

The control purpose is to find a control input $u(t+1)$ such that the output y follow reference r. Then, the so-called P-step-ahead predictive control finds a future input

$$u^f = \begin{pmatrix} u[t+1] \\ u[t+2] \\ \vdots \\ u[t+P] \end{pmatrix} \tag{3}$$

minimizing the difference between the *P*-step-ahead predicted future output

$$y^f = \begin{pmatrix} y[t+1] \\ \vdots \\ y[t+P] \end{pmatrix} \quad (4)$$

and the reference

$$r[t+1] = \begin{pmatrix} r[t+1] \\ \vdots \\ r[t+P] \end{pmatrix}. \quad (5)$$

In Just-In-Time predictive control, such a control input is found from vectors in the database which contains huge amounts of the past observed u and y of (1). More precisely, when we have

$$y^{\mathrm{p}}[t] = \begin{pmatrix} y[\mathrm{t}-[\mathrm{m}-1]] \\ \vdots \\ y[t] \end{pmatrix}, \qquad u^p[t] = \begin{pmatrix} u[t-[n-1]] \\ \vdots \\ u[t] \end{pmatrix}, \quad (6)$$

will be found as a weighted mean of $\tilde{u}_1^f, \ldots, \tilde{u}_k^f$ which are corresponding to $\Psi_1, \ldots, \Psi_k$ where in the database where

$$\Psi_i = \begin{pmatrix} y_i^p \\ y_i^f \\ u_i^p \end{pmatrix}, \quad (7)$$

$$y_i^f = \begin{pmatrix} y[\tau_i+1] \\ \vdots \\ y[\tau_i+P] \end{pmatrix}, \qquad u_i^p = \begin{pmatrix} u[\tau_i-[n-1]] \\ \vdots \\ u[\tau_i] \end{pmatrix}, \quad (8)$$

which are k-nearest neighbor of the information vector

$$\phi[t] = \begin{pmatrix} y^p[t-1] \\ r[t+1] \\ u^p[t] \end{pmatrix}. \quad (9)$$

As the input u at time $t+1$, only $\hat{u}[\tau+1]$ is used.

In the database D, we assume that we have N sets of Ψ_i and u_i^f. A matrix representation of the database is

$$D = \begin{bmatrix} \Psi_1 & \cdots & \Psi_N \\ u_1^f & & u_N^f \end{bmatrix} \quad (10)$$

In this paper, we use the following algorithm to predict $\hat{u}[t+1]$ from the database D when an information vector $\varphi[t]$ is given. In the algorithm, we use the $k_{\max}$-nearest neighbor to reduce the computational time. In this paper, $k_{\max}$ is chosen such that it is less than the size of the information vector φ, that is, $k_{\max} < 4m + n + 4P$.

Just-In-Time Algorithm

$$\hat{u}[t+1] = JIT(D, \phi[t], k_{max}) \quad (11)$$

Step 1

We define the distance between the Information vector $\phi[t]$ and the data vector Ψ_i by the normalized Euclidean distance

$$\mathrm{d}(\phi[t], \Psi_i) = \sqrt{(\phi[t] - \Psi_i)^T W (\phi[t] - \Psi_i)}, \quad (12)$$

where W is the weighting matrix given by

$$W = \left[\mathrm{diag}\left(\frac{1}{l}\sum_{i=1}^{l}(\Psi_i - \bar{\Psi})\,(\Psi_i - \bar{\Psi})^{\mathrm{T}}\right)\right]^{-1}, \quad (13)$$

$$\bar{\Psi} = \frac{1}{l}\sum_{i=1}^{l}\Psi_i\,, \quad (14)$$

where l is the column size of D, an $\mathrm{diag}(A)$ means the operator to generate a diagonal matrix to make off-diagonal elements be zero.

Step 2

Sort rows of D in ascending order according to $\mathrm{d}(\phi[t], \Psi_i)$

Step 3

Find the $k_{\max}$-nearest neighbor of the information vector $\phi[t]$.

$$\Omega_{k_{\max}} = \{\Psi_i | \mathrm{i} = 1, \ldots, k_{\max}\}. \quad (15)$$

Step 4

Determine the optimal size of the k-nearest neighbor of the information vector $\phi[\mathrm{t}]$ to use the FPF (Final Prediction Error) as

$$k_{\mathrm{opt}} = \arg_{1 \le k \le k_{\max}} \min V_k \frac{k+\dim(\phi[t])}{k-\dim(\phi[t])} \quad (16)$$

where V_k is the loss function given by

$$V_k = \frac{1}{k}\sum_{j=1}^{k}\left\|\tilde{u}_j - \hat{u}_k\right\|^2 \quad (17)$$

$$\hat{u}_k = \sum_{j=1}^{k} K\left(\frac{\mathrm{d}(\phi[t], \Psi_j)}{\mathrm{d}(\phi[t], \Psi_k)}\right)\tilde{u}_j. \quad (18)$$

In addition, $K(x)$ is a kernel function defined as

$$K(x) = A\exp(-Bx^2) \quad (19)$$

where A and B are parameters.

Step 5

Find the weighted average of $\tilde{u}_{\mathrm{j}}$

$$\hat{u}[\mathrm{t}+1] = \sum_{j=1}^{k_{\mathrm{opt}}} K\left(\frac{\mathrm{d}(\phi[t], \Psi_j)}{\mathrm{d}(\phi[t], \Psi_{k_{\mathrm{opt}}})}\right)\tilde{u}_j. \quad (20)$$

III. Experiment on a Two-Wheeled Robot

A. *Two-wheeled Robot and state feedback*

In this paper, we use the two-wheeled robot named e-nuvo WHEEL which is equipped with one CPU H8, one motor to drive two wheels, one rotary encoder to measure the angle of the wheel and one gyro sensor to measure the angle of the body.

The robot moves as a two-wheeled inverted pendulum. To do so, state feedback control $u(t) = Kx(t)$ is generally used. This control input is calculated by H8 installed in the robot.

The experiment is about combining Just-In-Time theory to the robot by using computer as the data processor to see how this theory can increase the robot's potential and efficiency then compare with the ordinary two-wheeled robot. The process data is produced by Just-In-Time algorithm via the computer and then send back to the robot to evaluate the robot's efficiency weather it increases or not.

B. Communication between robot and computer

For applying JIT predictive control to the robot, firstly we started with producing the database by the data we have collected before. Secondly, the robot will be working to send other data or values to the computer and then we will use program "Simulink" in order to get another value by "Just-In-Time Algorithm". After we get u_f from the program, this data will be sent back to the robot and then the process will start over just like a loop.

Communication is one of the most important of the process. Normally, robot will send the data to the computer where the data is processed by Just-In-Time Algorithm. Processed data then returns to the robot. All of the basic principle here needs communication in all steps. For this experiment, wireless communication is used together with protocol TCP/IP.

C. Database for JIT predictive control

Just-In-Time predictive control utilizes the database as the main key. To make the database, the data are collected which contain the angle θ, φ, angular velocity $\dot{\theta}$, $\dot{\varphi}$ and random input. As Fig. 3, to collect data, we use the random inputs which are samples from the uniform distribution *U(a,b)* generated by a MATLAB command.

In all experiments, we used $-0.5 \leqslant a < b \leqslant 0.5$, because the robot would be unstable when they exceed the bounds.

After collecting 5000 sample data, they were saved into the database as in figure.

The number of the data *N* can be changed for Just-In-Time predictive control. In general, large *N* requires long computation time of JIT algorithm. In our experiments we used *N= 5000*. All of the data collected are shown in the following figures.

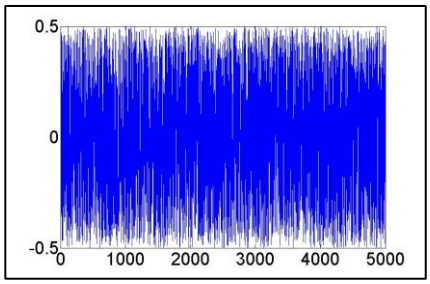

Figure 2. Random signal input to make database.

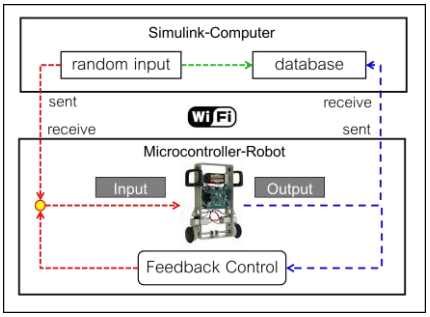

Figure 3. The database contains $z[t]$ and the random signal input.

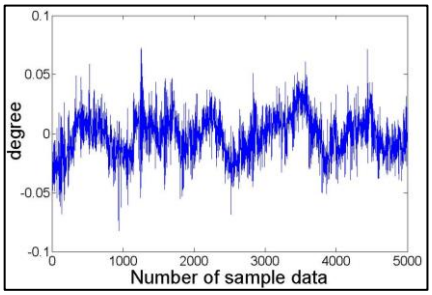

Figure 4. Angle θ.

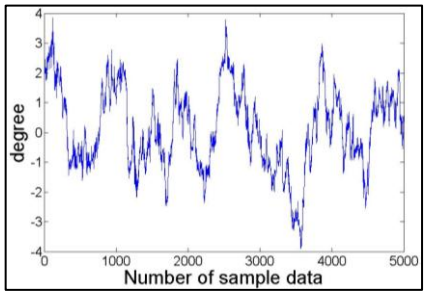

Figure 5. Angle φ.

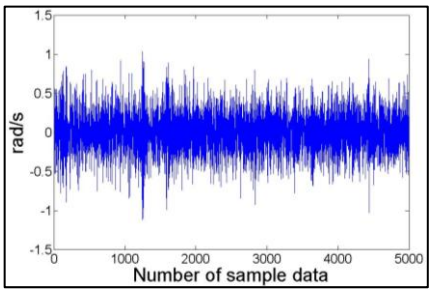

Figure 6. Angular velocity $\dot{\theta}$.

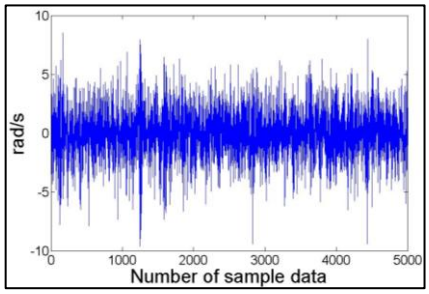

Figure 7. Angular velocity $\dot{\varphi}$.

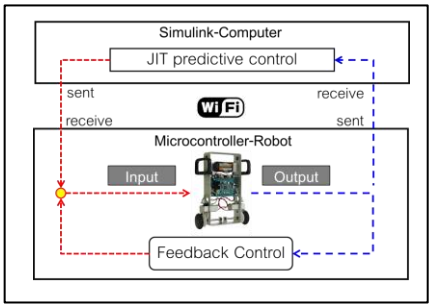

Figure 8. Overall structure of JIT predictive control for the robot.

D. Experiment result

From the experiment time series data, we calculated the "variance" as shown in the table.

TABLE I VARIANCES OF STATE VARIABLES BEFORE APPLYING JIT PREDICTIVE CONTROL

Parameter / Experiment No.	θ	φ	$\dot{\theta}$	$\dot{\varphi}$
X	9.18e-5	7.50e-1	3.00e-3	5.36e-1

TABLE II VARIANCES OF STATE VARIABLES WHEN JIT PREDICTIVE CONTROL IS APPLIED TO

Parameter / Experiment	m	n	p	l	θ	φ	$\dot{\theta}$	$\dot{\varphi}$	$\hat{u}$
A	4	3	2	1000	7.59e-5	5.95e-1	2.50e-3	4.23e-1	1.58e-2
B	20	3	2	1000	1.01e-4	8.17e-1	2.80e-3	4.53e-1	5.70e-3
C	4	15	2	1000	1.163e-4	9.64e-1	3.10e-3	5.21e-1	3.00e-3
D	4	3	10	1000	1.20e-4	9.27e-1	2.40e-3	4.31e-1	5.20e-3
E	4	15	10	1000	9.78e-5	8.97e-1	2.80e-3	4.87e-1	3.60e-3
F	4	3	2	2000	1.13e-4	6.29e-1	2.80e-3	4.57e-1	2.35e-2
G	4	3	2	4000	1.12e-4	7.06e-1	2.70e-3	4.46e-1	2.20e-2
H	3	2	1	1000	1.20e-4	7.73e-1	2.20e-3	3.33e-1	7.50e-3

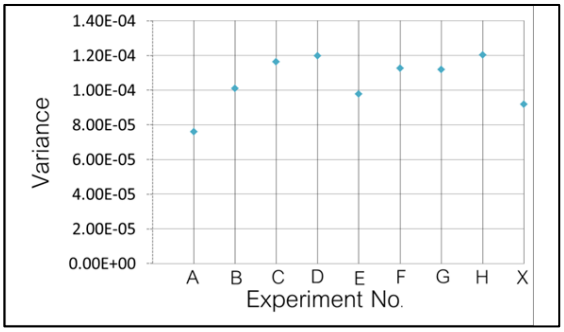

Figure 9. Variance of θ.

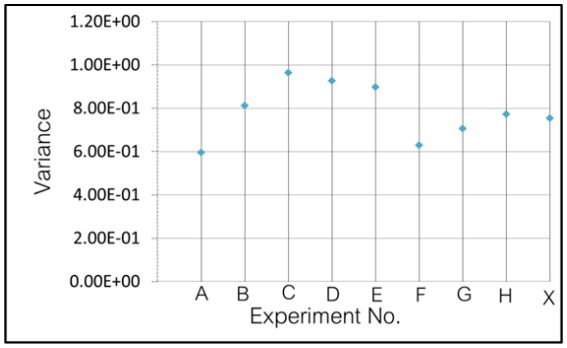

Figure 10. Variance of φ.

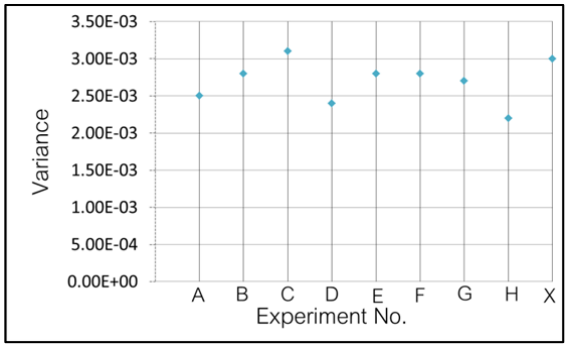

Figure 11. Variance of $\dot{\theta}$.

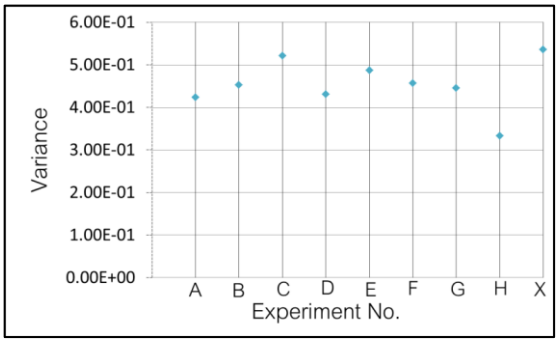

Figure 12. Variances of $\dot{\varphi}$.

From the experiments, experiment A ($m = 4$, $n = 3$, $P = 2$ and $l = 1000$) shows the smallest variance among that is obtained when Just-In-Time predictive control is applied and not applied.

IV. CONCLUSION

We have studied Just-In-Time predictive control for a two-wheeled robot. In this study, the robot is stabilized by state feedback in advance. The purpose to use Just-In-Time predictive control is to enhance the stability of the robot movement.

We have applied Just-In-Time predictive control to a two-wheeled robot which is stabilized state feedback control. Then, we have investigated the variance of the angles of the robot and the wheel and the angular velocities of the robot and the robot and the wheel.

In conclusion, we have verified that Just-In-Time predictive control can improve the robot movement with increased stability and efficiency.

ACKNOWLEDGMENT

This work was partially supported by the JSPC Grant in Aid for Scientific Research (B) No.23360183.

REFERENCES

[1] A. Stenman, "Model on Demand: Algorithms, Analysis and Applications," PhD thesis, Deparetment of Electrical Engineering LinkLNoping University, 1999.

[2] J. Ohta and S. Yamamoto, "Database-Driven Tuning of PID Controllers," Transactions of the Society of Instrument and Control Engineers, Vol. 40, No. 6, pp. 664-669, 2004 (in Japanese)

[3] Daisuke Inoue and Shigeru Yamamoto, "An Operation Support System based on Database-Driven On-Demand Predictive Control," Proceedings of the SICE Annual Conference 2004 (CD-ROM), pp.2024-2027, Sapporo, Japan, August 2004.

[4] Daisuke INOUE, and Shigeru Yamamoto, "A Memory-Based Predictive Control Approach to a Breaking Assist Problem," Proceedings of the SICE Annual Conference 2005 (CD-ROM), pp.3186-3189, Okayama, Japan, August 2005.

[5] K. Fukuda, S. Ushida, and K. Deguchi, Just-In-Time Control of Image-Based Inverted Pendulum System with a Time-Delay, SICE-ICASE International Joint Conference 2006, Busan Korea, October 18-21, 2006. (In Japanese)

[6] Mizukawa, Learning Modern Control Training with Inverted Pendulum Robots (CD-ROM), ZMP INC., 2007. (In Japanese)