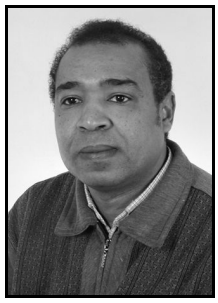
Ali A. LESEWED [1], **Jerzy E. KUREK** [2]
[1] WARSAW UNIVERSITY OF TECHNOLOGY
[2] POLITECHNIKA WARSZAWSKA, INSTYTUT AUTOMATYKI I ROBOTYKI

Design of iterative learning control for simple robot based on neural network robot model

M.Sc. Eng. Ali A. LESEWED

Received his B.Sc. degree in control engineering. from higher institute of electronics engineering, Libya in 1984 and his M.Sc. degree in Mechatronic from Hull University, U.K in 1993. He is currently a PhD student in Mechatronics Faculty at Warsaw University of Technology. His research interests include intelligent control, robotics, neural networks and fuzzy systems.



e-mail: ali_lesewed@yahoo.com

Prof. dr hab. inż. Jerzy KUREK

Profesor zwyczajny Pol. Warszawskiej (PW). W kadencji 2002-2005 Dziekan Wydziału Mechatroniki PW. Absolwent Wydziału Mechaniki Precyzyjnej PW (obecnie Wydział Mechatroniki). Jego zainteresowania naukowe dotyczą m. in. wyznaczania nowych algorytmów sterowania, stabilności układów automatyki, wyznaczania modeli neuronowych układów dynamicznych, szczególnie robotów. Jest członkiem Control System Society IEEE, Inc. oraz członkiem zarządu Polskiego Stowarzyszenia Pomiarów, Automatyki i Robotyki POLSPAR.

e-mail: jkurek@mchtr.pw.edu.pl

Abstract

Design of the iterative learning control (ILC) for robot manipulator with 2 degree of freedom based on model of the robot approximated by neural network is presented. The robot model has form of the Lagrange-Euler equation and neural network was trained to estimate the model parameters. Then, the estimated model was used for synthesis of ILC.

Keywords: neural model of industrial robot, iterative learning control.

Synteza iteracyjnie uczącego się sterowania prostego robota na podstawie neuronowego modelu robota

Streszczenie

W pracy przedstawiono syntezę iteracyjnie uczącego się sterowania dla robota o 2 stopniach swobody na podstawie modelu aproksymowanego przy pomocy sieci neuronowych. Model robota ma formę równań Lagrange'a-Eulera, którego nieliniowe funkcje zostały wyznaczone przez odpowiednio wytrenowaną sieć neuronową. Aproksymowany model został następnie wykorzystany do syntezy regulatora.

Słowa kluczowe: model neuronowy robota przemysłowego, iteracyjnie uczące się sterowanie.

1. Introduction

The robot manipulators have complex nonlinear dynamics that might make accurate and robust control difficult [6]. It is very hard to obtain the exact mathematical model of robot, the nonlinear model structure is well known but the parameters like inertia momentums are unknown. To overcome this problem, we have used the neural network to approximate unknown nonlinear functions in model of robot arm dynamics [7, 8].

The ability of neural network (NN) to approximate nonlinear functions and to learn through examples makes it the main tool in many disciplines, including robot control [11]. Many researchers have designed NN controllers in robot motion control with substantial success but a few of them have used NN for identification tasks [10]. In this paper, we consider the design of NN in order to approximate nonlinear functions of the robot. It should be underlined that the design of NN doesn't require the exact knowledge of functions that describe the model.

In the last two decades, a great deal of progress has been made in the research of ILC, e.g. [3, 9]. Several design methods were proposed based on conventional control system design principles.

We present in this paper a synthesis of advanced ILC of robot arm based on robot mathematical model whose parameters are approximated by trained neural network. The structure of the discrete-time NN model has been designed based on the Lagrange-Euler equation.

2. Mathematical model of robot

The mathematical model of robot can be represented by the discrete time model derived from the Lagrange-Euler equation [1, 2] as follows:

$$y(k+1) = 2y(k) - y(k-1) - T_p^2 M^{-1}[y(k)]\{D[y(k), y(k-1)] + G[y(k)] - \tau(k)\} \tag{1}$$

where $y \in R^n$ is the vector of joint position, $\tau \in R^n$ is a vector of generalized torque, $M[y(k)] \in R^{n \times n}$ is inertia matrix, $D[y(k), y(k-1)] \in R^n$ is a vector of Coriolis and centrifugal forces, $G[y(k)] \in R^n$ is a vector of gravity loading, k is a discrete time and T_p is sampling time, $t=kT_p$.

The above model can be also presented in the following form

$$y(k+1) = 2y(k) - y(k-1) + A_m[y(k), y(k-1)] + B_m[y(k)] + C_m[y(k)]\tau(k) \tag{2}$$

where

$$A_m = -T_p^2 M^{-1}[y(k)] D[y(k), y(k-1)]$$

$$B_m = -T_p^2 M^{-1}[y(k)] G[y(k)]$$

$$C_m = T_p^2 M^{-1}[y(k)]$$

Our purpose is to approximate the unknown nonlinear parameters of A_m, B_m and C_m in model (2) using neural network.

The presented model (2) can be rewritten in state space form as follows

$$\begin{aligned} x(k+1) &= A(x,k) + B(x,k)u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3}$$

where

$$x(k) = \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix},$$

$$A(x,k) = \begin{bmatrix} 2y(k) - y(k-1) + A_m(k, k-1) + B_m(k) \\ x_2(k) \end{bmatrix},$$

$$B(x,k)=\begin{bmatrix} C_m(k) \\ 0 \end{bmatrix}, \quad C=\begin{bmatrix} I & 0 \end{bmatrix}$$

and $I \in R^{n\times n}$ is identity matrix , $0 \in R^{n\times n}$ is zero matrix.

3. Neural model of robot

In this work, we setup the parameters of discrete time equation (2) for two degree of freedom robot, n=2, with revolute joints [10]. These parameters can be estimated by one-hidden layer feedforward neural network (FFNN) shown in fig. 1. The FFNN has been divided into four sub-networks. Every sub-network has two layers: the hidden layer has nonlinear neurons (NL) and output layer has linear neurons (L). The number of neurons in hidden layer was set to 4 neurons while the number of neurons in output layer was equal to the number of elements in vectors A_N, B_N, C_{N1} and C_{N2}, respectively 2,2,2,2. The presented structure of the neural network was chosen after a number of experiments.

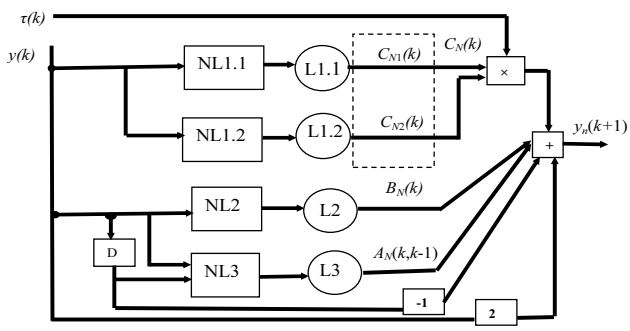


Fig. 1. The structure of feed-forward neural network (D is memory unit)
Rys. 1. Struktura sieci neuronowej feed-forward (D jest elementem pamięci)

The neurons in the hidden layers are described by the hyperbolic tangent function

$$z=f(v)=\frac{e^{2v}-e^{-2v}}{e^{2v}+e^{-2v}} \tag{4}$$

where

$$v=\sum_{i=1}^{n} w_i x_i + b \tag{5}$$

and x_i, w_i, b, z are input signals, weights, bias and output of the neuron, respectively.

In the output layers, the neurons are described by linear function

$$z=f(v)=v \tag{6}$$

We have assumed that the input signals to every layer are connected with all neurons in this layer. Input signals to the network is $y(k)$.

The performance function for learning of the neural network was selected as the sum of squared error, then, the conjugate gradient method [4, 5, 7] was chosen for learning of the NN off-line.

$$F=\frac{1}{2}\sum_{k=1}^{m}\sum_{j=1}^{n}[y_j(k)-y_{nj}(k)]^2 \tag{7}$$

where $y_n \in R^n$ is an output vector of the network, k is a number of the training pattern and m is the length of learning data.

4. Iterative learning control of robot

The ILC algorithm may be briefly described in the following way: given nonlinear state space model of robot in discrete time (3) can be presented in the state-space as follows

$$\begin{aligned} x(k+1,i) &= A(x,k,i)+B(x,k,i)u(k,i) \\ y(k,i) &= Cx(k,i) \end{aligned} \tag{8}$$

where i denotes the iterative learning iteration. A system whose dynamics propagates along two independent directions is known as 2-D system [3]. One process is described by variable k which represents the discrete-time and the second process is described by variable i which refers to the learning iterations.

Then, the general ILC rule can be given as follows

$$u(k,i+1)=u(k,i)+\Delta u(k,i) \tag{9}$$

where Δu denotes updating of the control input. Thus, (8) and (9) describe a 2-D model of ILC process. In ILC the both dynamical processes are independent of each other.

Then, assume that we have given the reference output in finite discrete time interval $k \in [0,N]$, where N is an integer determined by working task specification. Our task is to find control input such that robot output y follows the given reference output y_r. In the ILC after several learning iterations, $u(k,i)$ is modified such that $y(k) \approx y_r(k)$, in general $e(k,i) = y_r(k) - y(k,i) \to 0$ if $i \to \infty$. Fig. 2 shows the general structure of ILC using 2-D descriptions, the robot is the controlled system described by (8).

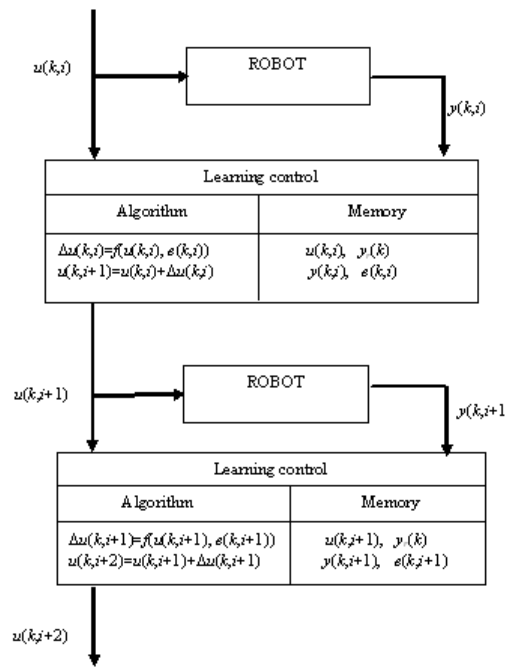


Fig. 2. Flowchart of ILC operation
Rys. 2. Schemat działania układu iteracyjnie uczącego się sterowania

Based on the state space model (8), the time-variable learning control rule can be presented, [9], as follows

$$\begin{aligned} u(k,i) = u(k,i-1) &+ K_1(x,k,i)[y_r(k)-y(k,i)] \\ &+ K_2(x,k,i)[A(x,k,i)-A(x,k,i-1)] \\ &+ K_3(x,k,i)u(k,i-1) \end{aligned} \tag{10}$$

where matrices of learning gain K_1, K_2 and K_3 are calculated the following way

$$\begin{aligned} K_1(x,k) &= [CB(x,k,i)]^{-1} \\ K_2(x,k,i) &= -K_1(x,k,i)C \\ K_3(x,k) &= K_1(x,k,i)CB(x,k,i-1) \end{aligned} \tag{11}$$

Since usually robot model parameters are unknown we have used the trained FFNN as a part of control system. The FFNN calculates the robot parameters (2) which are then used in model (8) for calculation of the ILC learning gain (11).

In this case, the estimated matrices of the nonlinear state space model (3) of robot can be rewritten as follows

$$A(x,k) = \begin{bmatrix} 2y(k) - y(k-1) + A_N(k,k-1) + B_N(k) \\ x_2(k) \end{bmatrix},$$

$$B(x,k) = \begin{bmatrix} C_N(k) \\ 0 \end{bmatrix} \text{ and } C = \begin{bmatrix} I & 0 \end{bmatrix}$$

where A_N, B_N and C_N are robot model parameters approximated by FFNN.

5. Computer simulation

In order to learn and test the designed neural network, the data for learning and testing of NN were generated by simulation of the 2-DOF PUMA robot [10], whose model parameters (1) are described as follows

$$M[y(k)] = \begin{bmatrix} p_1 + p_2 + 2p_3\cos(y_2(k)) & p_2 + p_3\cos(y_2(k)) \\ p_2 + p_3\cos(y_2(k)) & p_2 \end{bmatrix}$$

$$d_{11} = -p_3 y_1(k-1) y_2(k-1)\sin(y_2(k)) - p_3 y_2(k-1)[y_1(k-1) + y_2(k-1)]\sin(y_2(k))$$

$$d_{21} = p_3 y_1^2(k-1)\sin(y_2(k))$$

$$D[y(k), y(k-1)] = \begin{bmatrix} d_{11} \\ d_{21} \end{bmatrix}$$

$$G[y(k)] = \begin{bmatrix} p_4\cos(y_1(k)) + p_5\cos(y_1(k) + y_2(k)) \\ p_5\cos(y_1(k) + y_2(k)) \end{bmatrix}$$

The coefficients p_i of robot are

$$\begin{aligned} P &= \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix} \\ &= \begin{bmatrix} 1.6 & 0.5 & 0.6 & 3.7 & 1.2 \end{bmatrix} \end{aligned}$$

The NN was learned using the conjugate-gradient method [4, 5, 7]. Input signals to the network were robot position $X=\{y(1),\ldots,y(m)\}$ and robot torque $U=\{\tau(2),\ldots,\tau(m)\}$ while $Y_n=\{y_n(3),\ldots,y_n(m)\}$ was NN output and $Y=\{y(3),\ldots,y(m)\}$ was desired output. The learning trajectory (desired) of the robot output is described by the following formula

$$y = A\left(1 - \cos\left(\frac{\omega}{T}t\right)\right) \tag{13}$$

where $A = \dfrac{10}{\pi}$ [°/rad] is the amplitude, $\omega = 2\pi$ and the period T=18.8 [sec]. The neural model output had to follow the above given trajectory in time t=5 [sec] and sampling time T_p=0.01 [sec].

Task of the NN learning was to find the network parameters such that the network output signal $Y_n \approx Y$. This can be achieved by calculating of the error between the actual and desired outputs after each learning iterations in order to update the weights of NN according to the conjugate gradient learning method rule [4, 5, 7]

$$w(h+1) = w(h) + \eta\frac{\partial F(h)}{\partial w(h)} \tag{14}$$

where h is learning iterations and η is learning parameter and was set arbitrarily to 0.003. The obtained results for the desired trajectory and NN output signals are illustrated on Fig. 3.

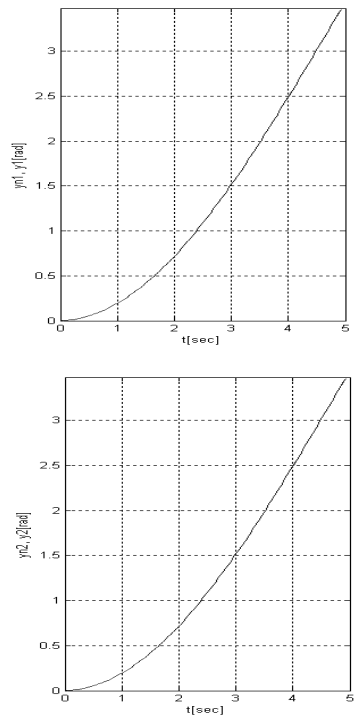


Fig. 3. The desired trajectory y and the NN output y_n (Learning stage)
Rys. 3. Zadana trajektoria y i wyjście sieci NN (nauka)

The maximum absolute errors between the desired trajectory and NN output are given in table 1.

Tab. 1. Maximum absolute error between desired trajectory y and NN output y_n (learning stage)
Tab. 1. Maksymalne błędy absolutne między zadaną trajektorią y oraz wyjściem NN y_n (uczenie)

Joint number	1	2
Error [°]	0.0320	0.0361

After the NN was trained it was incorporated within ILC system, the reference trajectory at each joint for control calculation was according to [10] as follows

$$y_r = A\sum_{i=1}^{3}\sin\left(\frac{\omega_i}{T}t\right) \tag{15}$$

where $A = \frac{10}{\pi}$ [°/rad] is the amplitude and the period is T=2.5 [sec]. The frequencies are ω_1=1.0 Hz, ω_2=2.0 Hz, and ω_3=4.0 Hz. The robot arm has to follow the given reference trajectory in time t=5 [sec] with sampling time T_p=0.01 [sec], i.e. N=500 time instants.

The control system was then tested to follow the given reference trajectory. The obtained results for robot trajectories, reference trajectories and control signals are presented on Fig. 4 and 5.

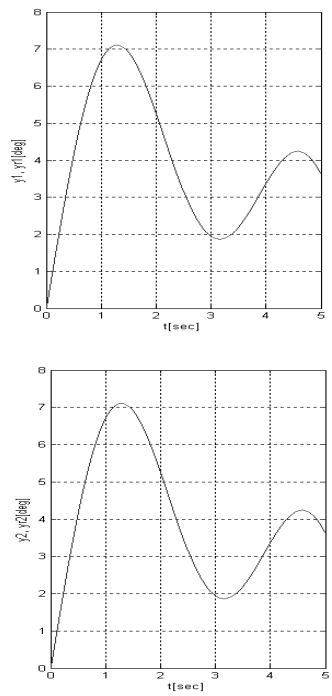


Fig. 4. The reference trajectory y_r and the actual robot position y
Rys. 4. Sygnał zadany y_r i sygnał pozycji robota y

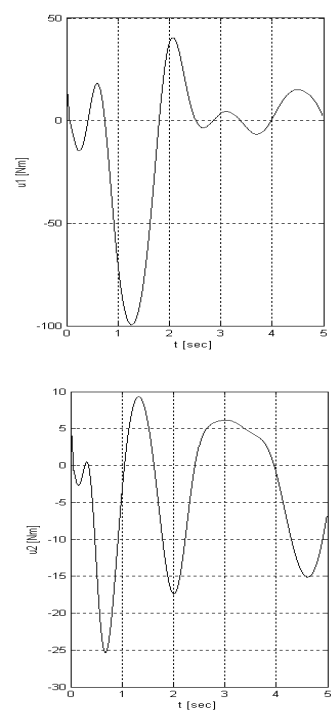


Fig. 5. The control signal τ_i for robot joint no. 1 and 2
Rys. 5. Sygnał sterujący τ_i przegubu 1 oraz 2 robota

The absolute control errors between the reference trajectory and actual robot trajectory are given in table 2.

Tab. 2. Maximum absolute control error between reference trajectory y_r and robot output y
Tab. 2. Maksymalne absolutne błędy sterowania między sygnałem zadanym y_r oraz wyjściem robota y

Joint number	1	2
Error [°]×10^{-3}	0.210	0.500

It is easy to find that the algorithm of ILC works efficiently although the robot parameters are identified by the neural network. In the presented results, we found out that the maximum absolute error for joint no. 1 and 2 not exceed 0.00021 and 0.0005 [°], respectively, and, the satisfactory quality control was obtained after three learning control iteration (i=3).

6. Conclusion

The design of iterative learning control for 2-DOF Puma robot based on model in the form of Lagrange-Euler equation which parameters were approximated parameters obtained by trained FFNN was presented. The satisfactory control has been obtained in a few learning iterations.

However, we have encountered some problems during NN learning and testing such as the length of time required for learning, and selection of the NN structure and learning method to do this task properly. One can expect more problems for control of robot with more degree of freedom. Thus, more research is required in order to increase the efficiency of learning algorithms and simplify the NN structure and at the same time without loose of specified performance level.

7. References

[1] R. P. Paul, Robot Manipulators: Mathematics and Control, Cambridge, Massachusetts, MIT Press, 1981.
[2] K. S. Fu, R. C. Gonzalez, C. S. Lee, Robotics, Control, Sensing, Vision and Intelligence, McGraw-Hill, 1987.
[3] Jamashidi M. O., Learning control system analysis and design based on 2-D system theory, Journal of Intelligent and Robotics Systems, pp. 17-26, 1990.
[4] Haykin S., Neural network: A comprehensive foundation, Macmillan Publishing Company, Englewood Cliffs.
[5] Demuth H., Beale M., Neural network toolbox user's guide, The MathWorks, Inc., Massachusetts, USA, 1994.
[6] Kurek J. E., Calculation of robot manipulator in the form of a recurrent neural net, European Control Conference'99, Karlsruhe Germany, pp. 354-359, 1999.
[7] Lesewed A., Kurek J. E., Comparison of four robot neural models using different learning methods, MMAR'06 conference, Miedzyzdroje, Poland, 2006.
[8] Lesewed A., Kurek J. E., Calculation of robot parameters based on neural nets, Robot Motion and Control RoMoCo'05, Poznań, Poland, 2005.
[9] Kurek J. E., Zaremba M. B., Iterative learning control synthesis based on 2-D system theory, IEEE Trans. on Automatic Control, Vol. 38, No.1, pp. 121-125, 1993.
[10] Meddah D. Y., Benallegue A., A stable Neuro-adaptive controller for rigid robot Manipulator, Journal of Intelligent and Robotic Systems, pp. 181-193, 1997.
[11] Limin P., Woo P., Neural-fuzzy control system for robotic manipulator, IEEE Control Systems Magazine, Feb. 2002.

Artykuł recenzowany