Nonlinear Generalized Predictive Controller Based on Artificial Neural Network for Robot Control

B. Durmus and N. Yumusak

1Department of Electric-Electronic Engineering, Sakarya University, 54187 Adapazari, Turkey
2Department of Computer Engineering, Sakarya University, 54187 Adapazari, Turkey

Abstract: This study deals with the tracking control problem of a robotic manipulator with changing dynamics. A multiple-input multiple-output (MIMO) artificial neural network based generalized predictive control (GPC) controller was designed for a six-degrees-of-freedom (6-DOF) robotic manipulator random disturbances and changing load. A three-layered neural network was used in the controller design to predict robotic manipulator inputs which track a desired trajectory. Standard back propagation (BP) algorithm was used as a learning algorithm to minimize the difference between actual trajectory and that predicted by the neural network. GPC controller was compared the conventional GPC under different control conditions. Results show that proposed control improved the ability of GPC under uncertainties.

Key words: Robotic manipulator control, trajectory planning, generalized predictive control, artificial neural network

INTRODUCTION

Evaluation of robot actuator inputs to track a desired trajectory is one of the most basic and important problems in robotic (Sun et al., 2001; Behera et al., 1994). A robotic manipulator is highly nonlinear system because of interactions between its links. These interactions make it hard to control. Several conventional methods such as adaptive control, proportional-plus-integral-plus-derivative (PID) control, self-tuning control, self-tuning PID control and generalized predictive control have been used in robot control (Chen et al., 1999; Mallick et al., 1997; Vega and Prada, 1991; Alonge et al., 2003; Nasisi and Carelli, 2003; Kanev and Verhaegen, 2000). However, because of high level interactions and nonlinear dynamics, industrial manipulators which use conventional linear control systems cannot be used over certain velocity limits and the productivity is limited. Furthermore, increasing performance needs also require improved manipulator techniques. Therefore, it is necessary to apply some advanced control techniques to provide high quality tracking control. One of these control techniques is generalized predictive control (GPC).

GPC is based on the use of a model which includes the prediction of the future outputs over a certain horizon (Clarke et al., 1987a, b). Thus, it can predict future changes of the measurement signal and base control actions on this prediction. GPC belongs to the class of model-based predictive control (MPC) techniques has become popular over the past two decades as a powerful tool for solving many problems. Today, this technique is still used widely because of their ability to systematically take into account real plant constraints in real-time (Mahfouf et al., 1997; Bordons and Camacho, 1998; Normey-Rico and Camacho, 2000; Zhang et al., 2004).

On the other hand, the conventional GPC algorithms use linear models of the process to predict the output of the process over a certain horizon and to evaluate a future sequence of control signals in order to minimize a certain cost function that represents the future output prediction errors over a reference trajectory (Clarke et al., 1987a, b). However, if the process is nonlinear, use of linear models becomes impractical and the identification of nonlinear models for control becomes necessary.

In recent years, the use of neural networks (NNs) for nonlinear system identification has proved to be extremely successful (Huang and Lewis, 2003; Zamarreño and Vega, 1999). NNs have been shown to possess good function approximation capabilities and have been applied successfully by many researchers in modeling some poorly understood systems or processes. The results demonstrated the feasibility of identification and control of nonlinear dynamic systems (Tsai et al., 2002; Lu and Tsai, 2004). Therefore, artificial neural network (ANN) was used in the predictive controller design to improve the ability of GPC.

Corresponding Author: Burhanettin Durmus, Department of Electric-Electronic Engineering, Sakarya University, 54187 Adapazari, Turkey Tel: (+90) 264 295 58 28 Fax: (+90) 264 295 56 08
NN based controller has been proposed by many researchers (Palos et al., 2001; Laabidi et al., 2008). Gupta and Sinha (2000) have presented an intelligent control system using a PD controller and ANN. Takahashi and Yamada (1994) have presented a study based on designing a NN controller, to control the tip of the angular position of a single-link flexible arm. Yildirim (2004) has proposed an adaptive robust neural controller for two-degrees-of-freedom robotic manipulator. Proposed neural controller has been shown to perform better than the conventional control schemes. Temurtas et al. (2005) have presented a controller ANN based on GPC for three joint robotic manipulator. Their study can be cited as closest to the work undertaken in this paper. In their study, three independent single-input single-output (SISO) controllers were used for control of joints and mechanical vibrations in the links were neglected. In practical robot applications, mechanic vibrations in links must not be neglected since they make hard to robot control. On the other hand, if a number of controllers are used in control design, many more hardware equipments will be required. Therefore, this paper focuses on a multiple-input multiple-output (MIMO) controller design. In this study, MIMO NGPC controller designed for 6-DOF robotic manipulator with random disturbances and load effect. Conventional GPC was also used for comparison. The controllers generate joint torques that will cause the robotic manipulator to follow a desired trajectory for a given trajectory. In the GPC, linear model was used to predict robot actuator inputs whereas prediction was carried out over a three-layered NN in the NGPC. The curves of trajectory and velocity belonging to joints were examined and end-effector position errors were computed for different control status. The results obtained by using NGPC were compared with those of conventional GPC.

GENERALIZED PREDICTIVE CONTROL

Generalized Predictive Control (GPC) belongs to the class of digital control methods called Model-Based Predictive control (MBPC) and was first introduced by Clarke and his co-workers in 1987 (Clarke et al., 1987a, b). MBPC techniques have been analyzed and implemented successfully in process control industries since the end of the 1970's and continue to be used because they can systematically take into account real plant constraints in real-time.

The GPC system for the robot control is given in Fig. 1. It consists of three components, the robotic manipulator, controller and parameter estimator.

![Block diagram of the GPC system for robotic manipulator](image)

Fig. 1: Block diagram of the GPC system for robotic manipulator

In the GPC algorithm, the process is supposed to be represented by a CARIMA (Controlled Auto-Regressive Integrated Moving Average) model (Clarke et al., 1987a, b):

$$A(q^{-1}) y(t) = B(q^{-1}) u(t-1) + \xi(t) / \Delta$$

(1)

where, $y(t)$ denotes the output of the process, $u(t)$ denotes its input and $\xi(t)$ denotes an uncorrelated random noise.

$A(q^{-1})$ and $B(q^{-1})$ are polynomials of $q^{-1}$, the backward shift operator:

$$A(q^{-1}) = 1 + \sum_{i=1}^{n_a} a_i q^{-i}$$

(2)

$$B(q^{-1}) = \sum_{i=1}^{n_b} b_i q^{-i}$$

(3)

$$\Delta = 1 - q^{-1}$$

In the prediction process, future outputs are predicted by using past inputs and past outputs. Predict step is selected short since using a great number of elements for prediction causes high computation overhead. In this study, the prediction was made for future three outputs. Therefore, $n_a, n_b$ are selected as 2, 1 respectively. Initial values were used as follows: $a_i = a_0 = 0$, $b_i = 1$, $b_0 = 0$.

The prediction process is executed as follows: We rewrite equation 1,

$$A(q^{-1}) \Delta y(t) = B(q^{-1}) \Delta u(t-1) + \xi(t)$$

(4)

$$y(t) = (1 - a_1) y(t-1) + (a_0 - a_2) y(t-2) + a_2 y(t-3)$$

$$+ b_1 \Delta u(t-1) + b_2 \Delta u(t-2) + \xi(t)$$

(5)

The first step predicted output ($\hat{y}(t)$):

$$\hat{y}(t + 1) = (1 - a_1) \hat{y}(t) + (a_0 - a_2) \hat{y}(t-1)$$

$$+ a_2 \hat{y}(t-2) + b_1 \Delta u(t) + b_2 \Delta u(t-1)$$

(6)
The second predicted output $\hat{y}(t + 2)$;

\[
\hat{y}(t + 2) = (1 - a_1 - a_2 + a_2^2) y(t) + a_1 (1 - a_1 - a_2) y(t - 1) + a_1 (1 - a_1) \hat{y}(t) + [b_0 (1 - a_1) + b_1] u(t) + b_1 (1 - a_1) \Delta u(t - 1)
\]

(7)

and the third predicted output $\hat{y}(t + 3)$;

\[
\hat{y}(t + 3) = [a_1 (1 - a_1 - a_2 + a_2^2) + a_2 (2 a_2 - a_2^2)] y(t) + [a_1 (1 - a_1 - a_2 + a_2^2) + a_2 (2 a_2 - a_2^2)] \hat{y}(t - 1) + a_1 (1 - a_1 - a_2 a_1^2) y(t - 2) + [b_0 (1 - a_1 - a_2 + a_2^2) + b_1 (1 - a_1) \Delta u(t - 1) + b_2 (1 - a_1)] u(t) + b_1 (1 - a_1 - a_2 + a_2^2) \Delta u(t - 1)
\]

(8)

A ($q^{-1}$) and B ($q^{-2}$) parameters in equation (1) should be updated each step of control. For this process, a parameter estimator is needed to update. In our application, Recursive Least Square (RLS) was used to parameter estimation and detailed computational issues of the RLS are addressed in Durmus et al. (2008).

The GPC strategy minimizes a weighted sum of square predicted future errors and square control signal increments; the cost function for MIMO GPC is defined as follows:

\[
J(N_i, N_d, N_u) = \sum_{k=N_i}^{N_d} \left[ (\hat{y}(t+j) - y(t+j))^2 + \lambda (\Delta u(t+j))^2 \right] + \sum_{j=1}^{N_u} \left[ \frac{s}{u(n+j) - u_{min} + \varepsilon} + \frac{s}{u_{max} - u(n+j) + \varepsilon} \right]
\]

(9)

where, $N_i$ is the minimum costing horizon, $N_d$ is the maximum costing horizon, $N_u$ is the control horizon, $y$ is a predicted output, $y_i$ is the reference output. $\lambda$ is the control input weighting factor and it is selected very small (as $\lambda = 10^{-4}$). Detailed computational issues of the GPC are addressed in Clarke et al. (1987a, b).

**NONLINEAR GENERALIZED PREDICTIVE CONTROL FOR ROBOTIC MANIPULATORS**

The nonlinear generalized predictive control (NGPC) system for the robotic manipulator is given in Fig. 2. It consists of four components, the robotic manipulator, a tracking reference signal that specifies the desired trajectory of the manipulator, an artificial neural network for prediction and the Cost Function Minimization (CFM) algorithm that determines the input needed to produce the desired trajectory of the manipulator.

The NGPC algorithm operates in two modes, prediction and control. For realizing this aim, a double pole double throw switch is used. The CFM algorithm produces an output which is either used as an input to the robotic manipulator or the manipulator’s neural network model. The switch position is set to the robotic manipulator when the CFM algorithm has solved for the best input, $u(n)$, that will minimize a specified cost function. Between samples, the switching position is set to the manipulator's neural network model where the CFM algorithm uses this model to compute the next control input, $u(n+1)$, from predictions of the response from the manipulator's model. Once the cost function is minimized, this input is passed to the manipulator (Durmus et al., 2008).

The cost function in the NGPC is defined as following equation:

\[
J(N_i, N_d, N_u) = \sum_{k=N_i}^{N_d} \left[ (\hat{y}(t+j) - y(t+j))^2 + \lambda (\Delta u(t+j))^2 \right] + \sum_{j=1}^{N_u} \left[ \frac{s}{u(n+j) - u_{min} + \varepsilon} + \frac{s}{u_{max} - u(n+j) + \varepsilon} \right]
\]

(10)

Where:

- $j = 1, 2, \ldots, \text{#iterations}$
- $N_i = \text{The minimum costing horizon}$
- $N_d = \text{The maximum costing horizon}$
- $N_u = \text{The control horizon}$
- $y_i = \text{A reference trajectory}$
- $\hat{y}_i = \text{The predicted output of the neural network}$
- $\lambda = \text{The control input weighting factor, it is selected very small (as } \lambda = 10^{-4})$
- $\Delta u(t+j) = \text{The change in } u \text{ and is defined as } u(n+j) - u(n+j-1)$
- $u_{max} \text{ and } u_{min} = \text{Maximum and minimum control inputs}$

The minimization algorithm used in the NGPC is Newton-Raphson method because it requires less iteration numbers for convergence. Detailed computational issues of the Newton-Raphson method are addressed in Durmus et al. (2008).

**Robotic manipulator neural network model for NGPC:** A multi-layer feed-forward network (MLFN) with tapped time delays was used for the robotic manipulator NN model.
The network structure is given in Fig. 3. The torque, $u$, is the control input to the manipulator system and the trajectory, $y$ is the output. The network inputs are past values of the torque and the past values of the manipulator's trajectory. The network has a single hidden layer with multiple hidden layer nodes and a single output node.

The number of neurons in the input and output layers was decided according to the structure of GPC. The past 2 inputs and past 3 outputs were used for prediction since over past data using leads to over computation. In this state, $n_u$, $d_u$ and $n_y$ were selected as 2, 3 and 5 respectively. Consequently, the designed neural network has 36 inputs and 6 outputs.

This network is multi-layer network (input layer, hidden layer and output layer). Whereas the hidden layer neurons use tan-sigmoid activation functions, linear activation function is used for output layer neurons. Equations used in the neural network models are given in Eq. 11-13.

Outputs of hidden layer neurons are:

$$\text{net}_j(n) = b_j + \sum_{i=0}^{n_u} w_{ji} u(n-i) + \sum_{i=1}^{n_y} w_{ji} y(n-i)$$ (11)

$$O_j(n) = f(\text{net}_j(n)) = \frac{e^{\alpha_j n} - e^{-\alpha_j n}}{e^{\alpha_j n} + e^{-\alpha_j n}} - 1 + \frac{2}{1 + e^{2\alpha_j n}}$$ (12)

Linear activation function outputs are:

$$y_i(n) = b_i + \sum_{j=0}^{n_u} w_{ij} O_j(n)$$ (13)

where, $j = 0$ to hid-1 and hid is the number of hidden layer nodes, $\text{net}_j(n)$ is the activation level of the $j^{th}$ hidden node, $f(\cdot)$ is the activation function for the $j^{th}$ hidden layer node, $n_u$ is the number of input nodes associated with past $u(n)$, $d_y$ is the number of input nodes associated with past $y(n)$, $w_{ij}$ is the weight connecting the $i^{th}$ hidden node to the output node, $w_{ij}$ is the weight connecting the $i^{th}$ input node to the $j^{th}$ hidden node, $y_i(n-i)$ is the delayed output of the manipulator's joint used as an input to the network and $u(t-i)$ is the input to the network and it's delays.

A training set was prepared by using the results of conventional GPC. The robotic manipulator was controlled for different trajectories to generate the training and test sets. To obtain the torque value at time $t$ as a output, values of torque at time $(t-1)$, $(t-2)$, $(t-3)$ and values of reference trajectory at time $(t-1)$, $(t-2)$, $(t-3)$ were used in the input stage as 36 elements. These data have been generated using GPC controller for different trajectories selected uniformly.

Back propagation (BP) was used for training neural network. 20, 000 input and output vector sets were generated with GPC algorithm, using the robotic manipulator simulation software. The training process was completed in approximately 2, 000, 000 iterations.

**ROBOTIC MANIPULATOR MODEL**

Used the robotic manipulator (6-DOF) is shown in Fig. 4 (Tam et al., 1993). The dynamic model of robot includes kinematics equations, friction effects and effect of carrying load at the end-effector for given robot.

There are six interactive second-order nonlinear differential equations which give the dynamic behaviors of 6-DOF robotic manipulator. The fourth-order Runge-Kutta integration method was used to solve these differential equations. The motion equations of robotic manipulator were developed by Lagrange-Euler as follows (Spong and Vidyasagar, 1989).

$$D(\theta) \ddot{\theta} + H(\theta, \dot{\theta}) + G(\theta) + \tau_r(t) + \tau_l(t) = \tau(t)$$ (14)

where, $\theta$, $\dot{\theta}$ and $\ddot{\theta}$ are n-dimensional vectors, indicating the joint acceleration, velocity and position, respectively. $D(\theta)$ is a n-dimensional symmetric inertia matrix. $H(\theta, \dot{\theta})$ is a n-dimensional vector, represents Coriolis and centrifugal torques. $G(\theta)$ is a n-dimensional vector, represents torque due to gravity. $\tau_r(t)$ is n-dimensional vector, represents torque due to friction effects. $\tau_l(t)$ is load effect, $\tau(t)$ is, n-dimensional, the generalized torque vector.

Robot arm friction model includes static, kinetic and fluid friction used in this study (Rodrigo et al., 2002). The friction model is given following:

$$\tau_{fric} = f_c \left( \frac{\sin(\alpha)}{\sqrt{1 + \frac{\theta}{N_s}}} \right) + f_c \tan \theta \mu(t) + k_w \theta$$ (15)
where, $f_s$, $x_s$, $f_k$, and $k_m$ are static friction, constant to correct static friction due to Stribeck Effect, kinetic friction and fluid friction, respectively (Rodrigo et al., 2002).

In this study, position reference and velocity reference trajectory for each joint were determined according to the sinusoidal trajectory principle. The motion for the sinusoidal trajectory is given in Fig. 5.

The equations for the sinusoidal trajectory (Spong and Vidyasagar, 1989) are,

$$\dot{\theta}(t) = a + b\cos(wt)$$

(16)

$$\ddot{\theta}(t) = -bw\sin(wt)$$

(17)

$$\dddot{\theta}(t) = -bw^2 \cos(wt)$$

(18)

where

$$w = \frac{\pi}{b - a} = \frac{\theta_a + \theta_b}{2}$$

$$\theta[i] = (\theta[a] + \theta[b]) \cos\left(\frac{\pi i}{n}\right) - \frac{\theta[a] - \theta[b]}{2} i = 0 \ldots n$$

(19)

$$\theta[i] = (\frac{\theta[a] - \theta[b]}{2}) \cos\left(\frac{\pi i}{n}\right)$$

(20)

$$\theta[i] = (\frac{\theta[a] - \theta[b]}{2}) \sin\left(\frac{\pi i}{n}\right)$$

(21)

Fig. 5: The motion trajectory planning

The coordinate of end-effector for robot arm is computed as follows:

$$x = \left[ d_s (\cos(\theta_2 + \theta_3) \cos \theta_4 \sin \theta_5 + \sin(\theta_3 + \theta_5) \cos \theta_4) \ight. \left. + d_s \sin(\theta_2 + \theta_3) + a \cos(\theta_2 + \theta_3) + a \cos \theta_4 \right]$$

(22)

$$y = \left[ d_s (\cos(\theta_2 + \theta_3) \cos \theta_4 \sin \theta_5 + \sin(\theta_3 + \theta_5) \cos \theta_4) \ight. \left. + d_s \sin(\theta_2 + \theta_3) + a \cos(\theta_2 + \theta_3) + a \cos \theta_4 \right]$$

(23)

$$z = \left[ d_s (\cos(\theta_2 + \theta_3) \cos \theta_4 \sin \theta_5 + \sin(\theta_3 + \theta_5) \cos \theta_4) \ight. \left. + d_s \sin(\theta_2 + \theta_3) + a \cos(\theta_2 + \theta_3) + a \cos \theta_4 \right]$$

(24)

THE DESIGN OF CONTROLLER USING NGPC FOR 6-DOF ROBOTIC MANIPULATOR CONTROL

Here, the design of the NGPC based on ANN controller is given. The designed controller has 36
inputs and 6 outputs. To obtain the torque value at time \( t \) as a output, values of torque at time \( (t-1) \), \( (t-2) \), values of trajectory at time \( (t-1) \), \( (t-2) \), \( (t-3) \) and values of reference trajectory at time \( (t-1) \) are used in the input stage as 36 elements for 6-DOF robotic manipulator. Outputs are the torques which will be applied to joints to track the desired trajectories. These data also have been used for prediction and learning of artificial neural network in the NGPC. The block diagram of the control system is shown in Fig. 6.

The NGPC starts with the input signal, \( \theta_i \) (entered angle of joints), which is presented to the trajectory planning model. This model produces a tracking reference signal, \( y_i(n) \), which is used as an input to the CFM block. The CFM block produces an output which is both used as an input to the plant (robotic manipulator) and to neural network for prediction. The neural network serves to predict the plant outputs from \( N_i \) to \( N_i \) future time steps in Eq. 10 by using past data. The predicted outputs passed to CFM. The CFM minimizes cost function that represents errors between reference signal and predicted outputs of the plant until desired minimization is achieved. Once the cost function is minimized, this input is passed to the manipulator. This process repeats for each control step.

For comparison of control algorithms, the following control states and values were used as a scenario of control.

The control states used in this study:

- There were disturbance (between -0.5 and +0.5 Newton*meters), falling load and friction effects in the control simulation.

The control values used in the simulation:

- The total simulation time is 10 second and total step number is 10000.
- The end-effector of robotic manipulator carries 5 kg load in the state of carrying and falling load. And the load is falling at the 4000th control step (at 4th second) in the state of falling load.
- Random disturbances between -0.5 and +0.5 Nm were added to the torques computed at the end of each step for control states.

RESULTS AND DISCUSSION

Some sample control results of the 6-DOF robotic manipulator which uses GPC algorithm at the state of \( A \) were given in Fig. 7. And some sample control results of the 6-DOF robotic manipulator which uses NGPC algorithm at the state of \( A \) were also given in Fig. 8. The same trajectory is used for both methods. The start and final angles of the joints are given in Table 1.

There are errors at the beginning of motion results from inertia of motionless robot arm (Fig. 7, 8). However, angular velocity and torque errors are bigger in the GPC than those in the NGPC. This can be because the neural network improves the predicted trajectory at the beginning of the motion. So, motion of the manipulator is more smooth and flexible in the NGPC.

There were similar situations at the load changes. In the Fig. 9 and 10, angular velocity and torque errors are seen at time falling load (at 4th second) result from load change. If the parameters of the system change in the control process, the performance of control system is
hardly affected. The errors were bigger in the GPC than those of the artificial neural network implemented NGPC. It is shown that the influence of load change to the NGPC is less than that of the GPC. This means that the NGPC is much stable than GPC for the load changes. It was seen that the proposed controller has better robustness in resisting against the changes of the parameters of the control system. The neural network provided quick adaptation for NGPC whereas GPC could not adapt quickly. There were also similar situations with random disturbances (Fig. 11, 12).

Comparisons of the GPC and NGPC algorithms control results are summarized and demonstrated in Table 2 and 3 by using angle location errors and x, y, z axis errors of end-effector for all of the control status. End-effector coordinates of robotic manipulator were computed using Eq. 22-24.

Angle location errors and x, y, z axis errors of the end-effector are smaller in the NGPC than those in the GPC (Table 2, 3). The difference between control results for the carrying load, with random disturbances and the falling load states can be easily shown. These values for the
Fig. 8: Some angle and velocity graphs of joints using neural network implemented NGPC (State A: with load and with friction)

Table 2: Angle location errors of control algorithms for control status

<table>
<thead>
<tr>
<th>Joint</th>
<th>Controller</th>
<th>Control status (rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPC</td>
<td>0.004173537500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0041735351978</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173516747</td>
</tr>
<tr>
<td>2</td>
<td>GPC</td>
<td>-0.000062396165</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.000062553161</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>3</td>
<td>GPC</td>
<td>0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>4</td>
<td>GPC</td>
<td>-0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>5</td>
<td>GPC</td>
<td>0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>6</td>
<td>GPC</td>
<td>-0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>NGPC</td>
<td></td>
<td>0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
<tr>
<td>NGPC</td>
<td></td>
<td>0.000014089886</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000014089839</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000173575015</td>
</tr>
</tbody>
</table>

3790
Fig. 9: Some torque and velocity graphs of joints using GPC (State B: with friction and falling load)

Table 3: The end-effector axis errors for different control status

<table>
<thead>
<tr>
<th>Control state</th>
<th>Controllers</th>
<th>x (mm)</th>
<th>y (mm)</th>
<th>z (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>GPC</td>
<td>-0.103896501760</td>
<td>2.32835706459</td>
<td>-0.505426004190</td>
</tr>
<tr>
<td></td>
<td>NGPC</td>
<td>0.364929292884</td>
<td>-0.032855065385</td>
<td>-0.263357821119</td>
</tr>
<tr>
<td>B</td>
<td>GPC</td>
<td>-0.2798830286579</td>
<td>2.3278973566565</td>
<td>-0.364074534578</td>
</tr>
<tr>
<td></td>
<td>NGPC</td>
<td>0.1624385373309</td>
<td>-0.03414165537</td>
<td>-0.129771968526</td>
</tr>
<tr>
<td>C</td>
<td>GPC</td>
<td>4.067840656383</td>
<td>0.010870719423</td>
<td>-3.391902606906</td>
</tr>
<tr>
<td></td>
<td>NGPC</td>
<td>0.451026191143</td>
<td>0.076745334388</td>
<td>-0.364679197600</td>
</tr>
</tbody>
</table>

NGPC are less than that of the GPC. NGPC reached at the targeted point with less error.

In this study, GPC and NGPC algorithms were designed and applied to 6-DOF robotic manipulator for joint control. Nonlinear prediction model was proposed instead of linear model to improve the ability of GPC. Conventional GPC and NGPC algorithms were compared according to different scenarios of control. When the parameters of the system change in the control process, the performance of control system is hardly affected. It was seen that the proposed controller has better robustness in resisting against the changes of the parameters of the control system and its trajectory tracking performance was observed higher than
Fig. 10: Some torque and velocity graphs of joints using neural network implemented NGPC (State B: with friction and falling load)

Fig. 11: Torque and velocity graphs of joint 2 using GPC (State C: with falling load and random disturbances)
Fig. 12: Torque and velocity graphs of joint 2 using neural network implemented NGPC (State C: with falling load and random disturbances)

conventional GPC’s under uncertainties. Artificial neural network improved the accuracy of predicted trajectory at the beginning of motion, load changes and with random disturbances and provided quick adaptation.

REFERENCES


