We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

32

Trajectory Control of RLED Robot Manipulators Using a New Type of Learning Rule

Hüseyin Canbolat
Mersin University
Turkey

1. Introduction

Rigid Link Electrically Driven (RLED) robot manipulators are used extensively in applications. For RLED manipulators, a hybrid adaptive-learning controller, which do not utilize the velocity measurements, is designed and proved that it can be made semi-global asymptotically stable (Canbolat et al., 1996). The learning part in that work (Canbolat et al., 1996) is based on the results given in (Messner et al. 1991). However (Messner et al., 1991) neglected the electrical dynamics and the velocity measurements are available. In (Canbolat et al., 1996), the system is designed through a high pass filter which produces the surrogates of velocity. Later in another work, (Kaneko & Horowitz, 1997) designed a similar controller for a robot manipulator using a velocity observer neglecting the electrical dynamics. The system in (Canbolat et al., 1996) had not been verified through simulation and experiments. Recently, (Uguz & Canbolat, 2006) published the simulation results of the controller proposed in (Canbolat et al., 1996) for a sinusoidal desired position. However, a typical desired position for a robotic application is not generally sinusoidal. Due to this, more general position vectors should be generated. A general task requires a smooth trajectory, which starts from an initial position to a final position and repeats this over and over again. Such a desired trajectory can be generated in several ways (Fu et al., 1987). In the simulation of the system in (Canbolat et al., 1996), desired functions should satisfy the certain specifications. For this purpose, the polynomial method given in (Fu et al., 1987) is slightly modified in order to accommodate with the requirements of the controller. The modifications are necessary due to the continuous third derivative or jerk requirement in (Canbolat et al., 1996). Here, we also proposed other methods, which utilize transcendental functions. Transcendental function methods give a trajectory that can be continuously differentiable up to any order.

Learning control law is usually used for repetitive tasks in which a certain task should be repeated in each cycle. Indeed, the adaptive and learning control schemes are very similar, since both strategies are based on the estimation of unknown system dynamics. However, the learning control philosophy tries to estimate the unknown time functions instead of estimating the unknown constant parameters of the system as in the adaptive control setup. The aim of the learning control is to improve the tracking performance of the manipulator at

each cycle using the error information obtained during the previous cycles. Thus the tracking of a desired trajectory is expected to improve in a period of the specified task comparing the results in the previous period (Arimoto, 1986; Messner et al., 1991). The control law is adjusted using the tracking error obtained at previous trials. The controller is expected to "learn" the unknown dynamics and make the tracking error goes to zero (Messner et al., 1991). The research on the design of adaptive control laws which tracks a desired trajectory asymptotically for rigid link robot manipulators has been conducted for years. The parametric uncertainties for a given system are inevitable for precise control. The uncertainties considerably affect the control performance of the system. Adaptive controllers, which updates the parameter estimates according to an adaptive update rule, tries to achieve the required specifications in the presence of parametric uncertainties (Lewis et al., 1993). In the case of robot manipulators, the control should be nonlinear due to the nonlinear nature of robot manipulator dynamics. Adaptive control law requires the linear parameterization of the system dynamics (Sadegh et al., 1990). However, the learning controller is generally used for periodic desired trajectories (Arimoto et al., 1985; Bondi et al., 1988; Horowitz et al., 1991; Kaneko & Horowitz, 1992; Kawamura et al., 1988; Kuc et al., 1992; Qu et al., 1993). (Messner et al., 1991) proposed a new learning algorithm. The algorithm is based on the selection of a Hilbert-Schmidt kernel. The uncertainties are modeled as an integral equation, which includes the multiplication of the kernel and a function that represents the system uncertainties. The learning update rule is based on the estimation of the system uncertainties via an update rule for the unknown function in the integral equation in terms of the known system variables. The controller makes the system follow the desired trajectory asymptotically (Canbolat et al., 1996).

The simulation of the learning control scheme (Canbolat et al., 1996) could not be achieved due to the partial derivatives of the control law with respect to the second time variable created by the Hilbert-Schmidt kernel. The two-time variables make the system complicated to simulate using traditional simulation packages, such as MATLAB® Simulink and SIMNON. In order to simulate the system in Simulink, the second time variable is considered to be discrete. Therefore, only the samples of the variables at specified locations on the second axis are estimated instead of a continuum of time. However, this process does not result a discrete-time system. Instead, the process results a higher order nonlinear continuous system through the state variables created due to the time-dynamic nature of the control law in both independent time variables, that is, the controller equations include partial derivatives with respect to both time variables. Since time is not discretized the resulting variables on the second axis has still continuous dynamics with respect to the real time.

In this work, the hybrid adaptive/learning controller proposed by (Canbolat et al., 1996) is simulated. The controller does not need the exact parameter values of the robot manipulator. The parameters of the electrical subsystem are updated according to an adaptive rule; while the uncertainties in the mechanical subsystem are compensated via learning term presented by (Messner et al., 1991) and (Canbolat et al., 1996). The controller was designed using a back-stepping technique and follows the desired trajectory asymptotically. The system used in the simulation is a rigid-link electrically driven (RLED) two-link planar robot manipulator, which is actuated by brushed DC (BDC) motors. The controller does not use the link velocities and compensates the electrical subsystem parameter uncertainties using an adaptive update law, while compensating the

uncertainties in the mechanical subsystem via a learning law. The controller is a partial state feedback controller which uses only the link positions and the actuator currents and forces the system follow the desired trajectory asymptotically (Canbolat et al., 1996). The controller is simulated using the MATLAB® SIMULINK software package. The results of the simulation shows that the proposed controller provides the semi-global asymptotic trajectory following.

Robot manipulators are implemented in various types like rectangular, cylindrical, spherical, revolute and horizontal joints to achieve the desired movements. From an industrial point of view, the Selective Compliance Articulated Robotic Arm (SCARA) type manipulators are utilized in the processes such as pick-and-place, painting, brushing, and peg-in-hole. In general, a SCARA manipulator has four degrees of freedom. Shoulder, elbow and wrist arms are controlled by servo motors while the fourth movement is realized pneumatically.

Various types of robot manipulators are designed according to the required movement types but the design of the controller is as important as the design of the mechanical parts. Several studies are available in the literature related to the design of controllers for robot manipulators employing classical proportional-integral-differential (PID) (Das & Dulger, 2005), adaptive (Queiroz et al., 1997; Kaneko & Horowitz, 1997), learning (Canbolat et al., 1996; Horowitz et al., 1991; Messner et al., 1991) artificial intelligence (Golnazarian, 1995; Jungbeck & Madrid, 2001) and fuzzy logic algorithms (Lewis et al., 1993).

Here, we describe the design of the hybrid adaptive repetitive controllers given in (Canbolat et al., 1996) and (Horowitz et al., 1991) and generate desired position functions, which satisfy the specifications given. However, the computation of derivatives requires the manipulation of highly nonlinear transcendental functions. The physical limitations of the robot manipulator are not considered in generation of desired trajectories. For a thorough position function the physical properties should be considered, such as, maximum velocity, acceleration, and jerk. Then a delayed hybrid adaptive repetitive controller (Sahin & Canbolat, 2007) is designed based on the method of (Horowitz et al., 1991). Also, the controllers are applied to a Serpent-1 model SCARA manipulator used in (Das & Dulger, 2005) in a simulation environment for a desired path generated according to the specifications of the hybrid adaptive-learning controller. Then, the performance of the robot with classical PD controller, learning based controller without electrical dynamics and adaptive/learning based hybrid controller are examined by means of simulations. Based on the simulation results, the performance of learning based controllers and classical PD controller is discussed.

2. Control Objective

The objective of this work is to develop a repetitive link position tracking controller for rigid link electrically driven (RLED) robot manipulators driven by brushed DC motors. The controller compensates for the effects of actuator dynamics. Furthermore, it uses only the link position and motor current measurements while compensating for the parametric uncertainty throughout the entire mechanical system and eliminating the link velocity measurements.

To facilitate the control law development, the position tracking error is defined as

$$e=q_d-q. \tag{1}$$

The parametric uncertainties of the mechanical subsystem are included in $c(\gamma)$ of (11) and the unknown electrical subsystem parameters are represented by the following vector

$$\theta_e = \left[\theta_{e1}^T, \theta_{e2}^T, \ \ldots \ , \theta_{en}^T \right]^T \in \Re^{3n} \tag{2}$$

where

$$\theta_{ei} = \left[L_i, R_i, K_{bi} \right]^T \in \Re^3 \tag{3}$$

in which L_i, R_i, and K_{bi} are the diagonal elements of electrical subsystem matrices L, R, and K_b, respectively. The true values of these parameters are not known except it is assumed that their upper and lower bounds are known. Whenever these upper and lower bounds are referred in the text, we will denote upper and lower bounds of a parameter matrix with the subscripts upper and lower, respectively. For example, $L_{lower} \leq \lambda_{\min}(L)$ denotes the lower bound for the matrix L, where $\lambda_{\min}(L)$ is the minimum eigenvalue of the matrix L.

A dynamic estimate $\hat{\theta}_e \in \Re^{3n}$ is used for θ_e. The parameter estimation error, $\tilde{\theta}_e$ is defined as follows

$$\tilde{\theta}_e = \theta_e - \hat{\theta}_e \, . \tag{4}$$

In the following section, we will give the details of the control design. The controller will be a partial state feedback controller in the sense that it will not utilize link velocity measurements to compensate for parametric uncertainties in the system. It is shown that the designed controller guarantees the semi-global asymptotic link position tracking. The system performance is simulated through a computer code. The code is written for both hybrid adaptive-learning controllers for BDC RLED robot manipulators and for the learning controller designed in (Messner et al., 1991). The results of the simulations show that the controller performs well in terms of error is below some certain value. However, the error does not become zero, but it has some average value. This is because of the complexity of the control law and the minimum information used to achieve the control goal.

3. System Model

3.1 Robot and Actuator Dynamics

The dynamics of an n-link robot manipulator electrically driven by brushed DC (BDC) motors can be expressed as follows:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + F_d\dot{q} = K_\tau I \quad \text{(mechanical subsystem)} \tag{5}$$

$$L\dot{I} + RI + K_b\dot{q} = v \quad \text{(electrical subsystem)} \tag{6}$$

where,

$q, \dot{q}, \ddot{q}$	:nx1 link position, velocity and acceleration vectors, respectively,
$M(q)$	:nxn symmetric, positive definite inertia matrix,
$V_m(q,\dot{q})$	:nxn matrix of centripetal and Coriolis terms,
F_d	:nxn constant, diagonal, dynamic friction matrix,
$G(q)$	:nx1 gravitational effects vector,

τ	:nx1 torque vector,
L	:nxn diagonal inductance matrix,
R	:nxn diagonal resistance matrix,
K_b	:nxn diagonal back-emf matrix,
K_τ	:nxn diagonal torque coefficients matrix, and
v	:nx1 motor input voltages vector.

The periodic desired trajectory $q_d(t)$ and its time derivatives up to 3rd order should be continuous and bounded (Canbolat et al., 1996).
The following properties of robot dynamics were utilized in the stability analysis of the controller:

1. For any given vector, $x(t)$, the inertia matrix, $M(q)$, satisfies the following inequality:

$$M_1 \|x\|^2 \le x^T M(q) x \le M_2 \|x\|^2 \tag{7}$$

where M_1 and M_2 are known positive constants that depend on the mass properties of the specific robot for which the controller is designed.

2. The matrix $\dot{M}(q) - 2V_m(q,\dot{q})$ is skew symmetric, that is, for any given vector x, we have

$$x^T \left(\dot{M}(q) - 2V_m(q,\dot{q}) \right) x = 0 \tag{8}$$

3. The Coriolis-centripetal matrix V_m is bounded as

$$\left\| V_m(q,\dot{q}) \right\|_{i\infty} \le \zeta_c \|\dot{q}\| \tag{9}$$

where ζ_c is a known positive constant.

4. The left-hand side of (5) can be written in terms of the desired trajectory as

$$w(t) = M(q_d)\ddot{q}_d + V_m(q_d,\dot{q}_d)\dot{q}_d + G(q_d) + F_d\dot{q}_d \,. \tag{10}$$

Since the desired trajectories $q_d, \dot{q}_d, \ddot{q}_d$ are periodic with the period T, $w(t)$ of (10) is also periodic. $w(t)$, can be expressed as a linear integral equation as shown by (Horowitz et al., 1991). That is, $w(t)$ can be expressed as follows

$$w(t) = \int_0^T K(t,\gamma) c(\gamma) d\gamma \tag{11}$$

where $K(t,\gamma)$ is a *known* Hilbert-Schmidt kernel and $c(\gamma)$ is an *unknown* influence function. Note that t and γ are independent variables.

5. (Horowitz et al., 1991) used the kernel of the form

$$K(t,\gamma) = f_0 + \sum_{m=1}^{\infty}\left(f_m \cos\left(2\pi mt/T\right)\cos\left(2\pi m\gamma/T\right) + d_m \sin\left(2\pi mt/T\right)\sin\left(2\pi m\gamma/T\right)\right), \quad (12)$$

where f_i and d_i's are scalar constants, which satisfy the conditions,

$Dm^{-2} < |f_m|$ and $Dm^{-2} < |d_m|$ for all $m > N$ with D, N are constants.

If the kernel of the form given in (12) is utilized, then

$$\int_0^T \|c(\gamma)\|^2 \, d\gamma < \mu \quad (13)$$

where μ is a positive constant.

Consider the following kernel, which is a Gaussian distribution function, given by

$$K(t,\gamma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(t-\gamma)^2}{2\sigma^2}\right) \quad (14)$$

where σ is a positive design constant. This function satisfies the conditions given in (12) (Horowitz et al., 1991).

If the kernel defined in (13) is used, then the following relations can be shown:

$$\sup_{t\in[0,T]} \int_0^T K^2(t,\gamma) d\gamma = \kappa < \infty \quad (15)$$

$$\sup_{t\in[0,T]} \int_0^T \left(\frac{\partial}{\partial t} K(t,\gamma)\right)^2 d\gamma = \kappa_d < \infty, \quad (16)$$

where κ and κ_d are positive constants.

3.2 Position Tracking Controller

To achieve the control objective, the methods proposed by (Burg et al., 1996) and (Horowitz et al., 1991) are combined. The design procedure can be summarized as: i) We use a pseudo-velocity filter to generate the signals for use as link velocity, ii) since the developed torque is a function of motor currents, a desired current signal is designed to force the link position to track the desired trajectory (backstepping) and iii) the voltage control input is designed to ensure the motor currents tracks the desired current.

Using the position tracking error defined in (1), the following high pass filter is designed to obtain a velocity dependent signal e_f:

$$\begin{aligned} \dot{p} &= -(k+1)p + (k^2+1)e \\ e_f &= -ke + p \\ p(0) &= ke(0) \end{aligned} \quad (17)$$

where k is a positive gain constant and the auxiliary signal p is used to get two implementable equations for the filter.

The filtered tracking error is defined as follows

$$\eta = \dot{e} + e + e_f \,. \tag{18}$$

Note that, the filtered tracking error, η, cannot be measured, since the link velocities cannot be measured. Based on the dynamics of η, the auxiliary variable $w_1(t)$ is defined as

$$w_1(t) = \int_0^T K(t,\gamma) c_x(\gamma) d\gamma = K_\tau^{-1} \int_0^T K(t,\gamma) c(\gamma) d\gamma \tag{19}$$

Note that the uncertainties in K_τ is now included in $w_1(t)$. The desired current, I_d, is designed to force the filtered tracking error, η, to zero.

The desired current, I_d, is defined as

$$I_d = \hat{w}_1(t) - k e_f + e \tag{20}$$

and the current tracking error, η_I, is defined as

$$\eta_I = I_d - I \tag{21}$$

where $\hat{w}_1(t)$ is the estimate of $w_1(t)$ and is defined as

$$\hat{w}_1(t) = \int_0^T K(t,\gamma) \hat{c}_x(t,\gamma) d\gamma \tag{22}$$

where $\hat{c}_x(t,\gamma) \in \Re^n$ is an estimate of $c_x(\gamma)$. $\hat{c}_x(t,\gamma)$ is updated according to the following rule

$$\begin{aligned} \hat{c}_x(t,\gamma) = & \int_0^t \left(K(\sigma,\gamma) K_L \left(e(\sigma) + e_f(\sigma) \right) \right) d\sigma \\ & + K(t,\gamma) K_L e(t) - K(0,\gamma) K_L e(0) \\ & - \int_0^t \left(\frac{\partial}{\partial \sigma} K(\sigma,\gamma) \right) K_L e(\sigma) d\sigma. \end{aligned} \tag{23}$$

where K_L, is an nxn diagonal, positive definite gain matrix.

The electrical parameter regression matrix is defined as

$$Y_e \theta_e = L w_2 + (k-1) L e_f - k L e + R I + K_b (\dot{q}_d + e + e_f) \tag{24}$$

where

$$w_2 = \int_0^T \frac{\partial K(t,\gamma)}{\partial t} \hat{c}_x(t,\gamma) d\gamma \,. \tag{25}$$

Based on the current tracking error dynamics the voltage control input is designed as

$$v = Y_e \hat{\theta}_e + \left(\left\| K_L \right\|_{i2}^2 \kappa^2 k_{n1} + k^4 k_{n2} + k_{n3} + k_{n4} + k_{n5} + 1 \right) \eta_I \tag{26}$$

where $\left\| K_L \right\|_{i2}$ denotes the induced-2 norm of the matrix K_L and the k_{ni} (i=1,2,...,5) and κ are positive control gains, and the adaptive electrical parameter update rule is defined as

$$\dot{\hat{\theta}}_e = \Gamma_e Y_e^T \eta_I \tag{27}$$

with $\Gamma_e \in \Re^{3n \times 3n}$ is a positive definite, diagonal adaptive gain matrix and the electrical regression matrix $Y_e \in \Re^{3n \times 3n}$ is defined as

$$Y_e = blockdiag \left\{ \begin{bmatrix} w_{2i} + (k-1)e_{fi} - (k+1)e_i \\ I_i \\ \dot{q}_{di} + e_i + e_{fi} \end{bmatrix}^T \right\} \tag{28}$$

where w_{2i} is the ith element of w_2 defined in (25).

The following theorem can be stated for the tracking performance of the proposed controller (Canbolat et al., 1996).

Theorem 1: If the control gains k_{ni} and k_n satisfy the following conditions

$$k_{ni} > 3L_{\max}^2 + K_{\tau \max}^2 + K_{b \max}^2 \tag{29}$$

$$k_n \geq \frac{1}{\lambda_3} \left(\frac{\lambda_2}{\lambda_1} \left\| x(0) \right\|^2 + 1 \right) \tag{30}$$

where

$$\lambda_1 = \min \left\{ K_{\tau lower}, m_1, L_{lower}, \lambda_{\min}(\Gamma_e^{-1}), K_{\tau lower} \lambda_{\min}(K_L^{-1}) \right\} \tag{31}$$

$$\lambda_2 = \max \left\{ K_{\tau upper}, m_2, L_{upper}, \lambda_{\max}(\Gamma_e^{-1}), K_{\tau upper} \lambda_{\max}(K_L^{-1}) \right\} \tag{32}$$

$$\lambda_3 = \min \left\{ K_{\tau lower}, 1 - L_{upper}^2 \left(\frac{1}{k_{n1}} + \frac{1}{k_{n2}} + \frac{1}{k_{n3}} \right) - \frac{K_{bupper}^2}{k_{n4}} - \frac{K_{\tau upper}^2}{k_{n5}} \right\} \tag{33}$$

$$x = \left[e^T \quad e_f^T \quad \eta^T \quad \eta_I^T \quad \tilde{\theta}_e^T \quad \int_0^T \tilde{c}_x^T(t,\gamma) d\gamma \right]^T \in \Re^{8n}, \tag{34}$$

then

$$\lim_{t\to\infty}\|e(t)\| = 0. \tag{35}$$

In the above equations, $(.)_{lower}$ and $(.)_{upper}$ denotes the known lower and upper bounds for the eigenvalues of the corresponding unknown parameter matrices, respectively. Similarly, $\lambda_{\min}(.)$ and $\lambda_{\max}(.)$ denotes the minimum and maximum eigenvalues of the matrix in parentheses, respectively.

The proof of this theorem can be found in (Canbolat et al., 1996). For the sake of brevity, the proof is omitted here.

3.3 Delayed Learning Rule

We define the following delayed update rule for $\hat{w}_1(t)$ in (22) and $w_2(t)$ in (25) similar to (Messner et al., 1991):

$$\hat{w}_1(t) = \int_0^T K(t,\gamma)\hat{c}_x^k(\gamma)d\gamma \quad kT \le t < (k+1)T \tag{36}$$

$$w_2(t) = \int_0^T \left\{\frac{\partial}{\partial t}K(t,\gamma)\right\}\hat{c}_x^k(\gamma)d\gamma \quad kT \le t < (k+1)T \tag{37}$$

where $\hat{c}_x^k(\gamma)$ is defined as

$$\begin{aligned}\hat{c}_x^k(\gamma) = \hat{c}_x^{k-1}(\gamma) &+ K(kT,\gamma)K_L e(kT) \\ &+ \int_{kT-T}^{kT} K(\sigma,\gamma)K_L\left[e(\sigma)+e_f(\sigma)\right]d\sigma \\ &\int_{kT-T}^{kT}\left\{\frac{\partial}{\partial\sigma}K(\sigma,\gamma)\right\}K_L e(\sigma)d\sigma,\end{aligned} \tag{38}$$

with $\hat{c}_x^0(\gamma) = -K(0,\gamma)K_L e(0)$. Note that the form of (22) and (25) are not changed except the estimate of $\hat{c}_x(t,\gamma)$ defined in (23) is replaced with $\hat{c}_x^k(\gamma)$. Similarly, all other equations are same. One can use the same equations for the controller by changing (22), (23) and (25) with (36), (38) and (37), respectively. This definition aims the reduction of computational burden for real time applications. One can show that the controller with the delayed learning rule of (38) is asymptotically stable using the arguments used by (Messner et al., 1991) and (Canbolat et al., 1996).

3.4 Generation of Desired Trajectories

A proper desired trajectory should be generated for the proposed controller, for performance evaluation. A position function for a robot manipulator is a smooth function that starts from a certain initial position and ends at a final position. Generally, the

acceleration and velocity are required to be continuous and smooth. However, in our case the desired trajectory should be continuously differentiable up to the third order. There are several methods to generate the desired trajectories. One common method is to separate the trajectory into three main parts (initial-lift off (IL), lift off-set down (LS), and set down-final (SF)) and impose the continuity conditions at the boundaries. The coefficients of the polynomials are to be solved according to the imposed conditions. (Fig. 1).

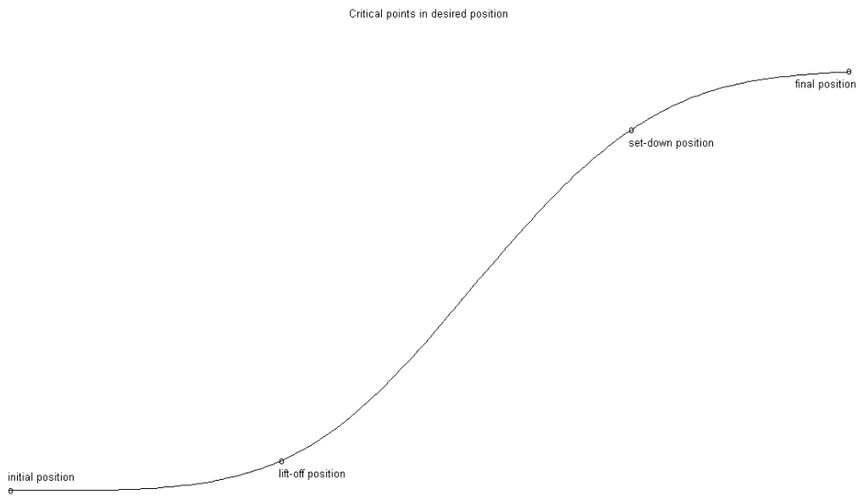

Fig. 1. Critical points for a robot end-effector position

In this section, we propose several methods to determine the desired trajectories. Polynomial methods are the modifications of 3-5-3 and 4-3-4 methods given in (Fu et al., 1987). The naming of the method is based on the degrees of the polynomials, which are valid for the IL, LS and SF parts. Since these methods generate functions continuously differentiable up to the second order, we should have modified the method to generate functions with continuous third time derivatives. The modification is done by increasing the degrees of the polynomials. Following the same convention, the modified methods are named as 4-6-4 and 5-4-5. Detailed formulae can be found in (Fu et al., 1987).
Formulation is carried as in (Fu et al., 1987). Let t_0, t_1, t_2, t_3 be the time boundaries for subtrajectories IL, LS and SF (Fig. 2). Let us assign the numbers 1, 2, 3, 4, 5, and 6 for forward IL, LS, SF and backward IL, LS, SF subintervals, respectively, and define the following dimensionless variable τ for each subinterval as follows

$$\tau = \frac{t - t_{i-1}}{t_i - t_{i-1}} \tag{39}$$

for $t \in [t_{i-1}, t_i]$ and i=1, 2, …, 6. Note that, in a given subinterval $[t_{i-1}, t_i]$ $\tau \in [0,1]$. With the definition given in (39) each subtrajectory can be defined on [0, 1]. Boundary conditions become the conditions at τ=0 and τ=1.

Let h_i be the polynomial in the i^{th} subinterval. The first and k^{th} derivative of h_i can be found as

$$\frac{dh_i}{dt} = \frac{dh_i}{d\tau}\frac{d\tau}{dt} = \frac{1}{t_i - t_{i-1}}\frac{dh_i}{d\tau} = A_i \frac{dh_i}{d\tau}$$
$$\frac{d^k h_i}{dt^k} = A_i^k \frac{d^k h_i}{d\tau^k} \tag{40}$$

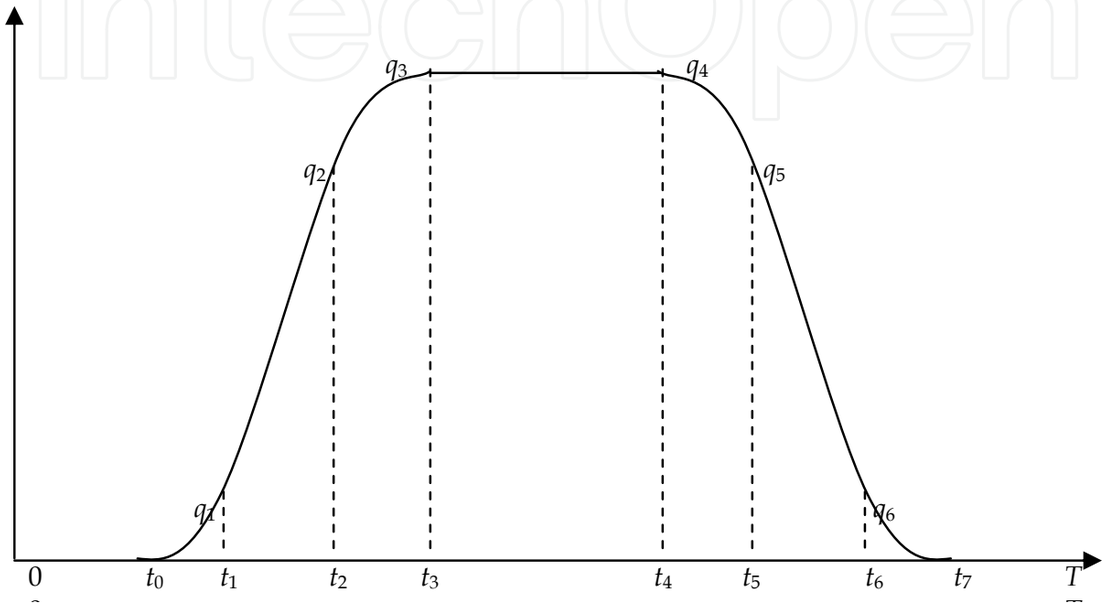

Fig. 2. Critical points of a desired trajectory

where $A_i = \dfrac{1}{t_i - t_{i-1}}$.

Since both methods use polynomials of maximum 6^{th} degree in each subinterval, one can write the following general expressions for h_i and its derivatives:

$$h_i(\tau) = a_{i6}\tau^6 + a_{i5}\tau^5 + a_{i4}\tau^4 + a_{i3}\tau^3 + a_{i2}\tau^2 + a_{i1}\tau + a_{i0}$$
$$h_i^{'}(\tau) = \frac{1}{t_i - t_{i-1}}\left(6a_{i6}\tau^5 + 5a_{i5}\tau^4 + 4a_{i4}\tau^3 + 3a_{i3}\tau^2 + 2a_{i2}\tau + a_{i1}\right)$$
$$h_i^{''}(\tau) = \frac{1}{\left(t_i - t_{i-1}\right)^2}\left(30a_{i6}\tau^4 + 20a_{i5}\tau^3 + 12a_{i4}\tau^2 + 6a_{i3}\tau + 2a_{i2}\right) \tag{41}$$
$$h_i^{'''}(\tau) = \frac{1}{\left(t_i - t_{i-1}\right)^3}\left(120a_{i6}\tau^3 + 60a_{i5}\tau^2 + 24a_{i4}\tau + 6a_{i3}\right).$$

Let q_i, q_l, q_s ve q_f be the positions at the initial, lift-off, set-down and final positions, respectively (Fig. 1). Each polynomial should satisfy the following boundary conditions:

$$\begin{aligned} h_i(1) &= h_{i+1}(0) \\ h_i'(1) &= h_{i+1}'(0) \\ h_i''(1) &= h_{i+1}''(0) \\ h_i'''(1) &= h_{i+1}'''(0) \end{aligned} \tag{42}$$

where primes denote the derivative with respect to time. In (42), i should be 1 or 2. (42) creates 8 equations for the coefficients. At the initial and final positions, the following conditions should be satisfied:

$$\begin{aligned} h_1(0) &= q_b \\ h_1'(0) &= h_1''(0) = h_1'''(0) = 0 \\ h_3(1) &= q_f \\ h_3'(1) &= h_3''(1) = h_3'''(1) = 0 \end{aligned} \tag{43}$$

From (43) we have another set of 8 equations. Thus we have 16 equations for 17 unknown coefficients. In order to have a unique solution, one can use the position at the lift-off or set-down positions. Using the lift-off position, we have the following equation:

$$h_1(1)=q_l \tag{44}$$

Since the desired trajectory is periodic, the manipulator should go back to the initial position at the end of the period. Due to this, the formulation given in (39-44) should be done twice for both reaching the final position and returning to the initial position. The forward and backward trajectories may not be symmetric. That is, we are free to select different time intervals and different lift-off and set-down positions. We can even use different methods in the generation of forward and backward paths. Typically, symmetrical trajectories are easy to use in applications.

There are 8 critical points in a period (Fig. 2). The time instants t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7 and the corresponding positions q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7 are critical points of a desired trajectory. The first 4 points q_0, q_1, q_2, q_3 correspond to initial, lift-off, set-down and final positions of the forward path, and the last 4 points q_4, q_5, q_6, q_7 correspond to initial, lift-off, set-down and final positions of the backward path, respectively. The equalities,

$$q_0=q_7,\ q_3=q_4, \tag{45}$$

should be satisfied for a periodic trajectory. For a symmetrical trajectory, the following constraints in positions,

$$q_1=q_6,\ q_2=q_5, \tag{46}$$

and in time

$$t_0=T-t_7,\ t_1=T-t_6,\ t_2=T-t_5,\ t_3=T-t_4, \tag{47}$$

should be satisfied. For each desired trajectory, the following position values are used:

q_i=0, q_l=0.08, q_s=0.92, q_f=1,
in forward path and
q_i=1, q_l=0.92, q_s=0.08, q_f=0
in backward path. The corresponding time instants are assigned as follows:
t_0=0,1T; t_1=0,15T; t_2=0,25T; t_3=0,3T
t_4=0,7T; t_5=0,75T; t_6=0,85T; t_7=0,9T
All position values are in radians, since we used a two-link robot with revolute joints. It is possible to select closer lift-off and set-down points. However, in this case the subpolynomials may have maxima and minima inside their own subintervals. Typically, a robot path should be smooth and monotone increasing or decreasing.

4-6-4 Method

In this method, IL and SF polynomials are fourth order. Therefore,
a_{i6}=a_{i5}=0
in (41). Furthermore, the initial conditions in (43) requires
a_{10}=q_b, a_{11}=a_{12}=a_{13}=0
for forward path and
a_{40}=q_b, a_{41}=a_{42}=a_{43}=0
for backward path. The conditions in (42), (43) and (44) give the following matrix equality for the unknown coefficients:

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
4A_1 & 0 & 0 & 0 & 0 & 0 & 0 & -A_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
6A_1^2 & 0 & 0 & 0 & 0 & 0 & -A_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4A_1^3 & 0 & 0 & 0 & 0 & -A_2^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 6A_2 & 5A_2 & 4A_2 & 3A_2 & 2A_2 & A_2 & 0 & 0 & 0 & 0 & -A_3 & 0 \\
0 & 0 & 15A_2^2 & 10A_2^2 & 6A_2^2 & 3A_2^2 & A_2^2 & 0 & 0 & 0 & 0 & -A_3^2 & 0 & 0 \\
0 & 0 & 20A_2^3 & 10A_2^3 & 4A_2^3 & A_2^3 & 0 & 0 & 0 & 0 & -A_3^3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 3 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} a_{14} \\ a_{10} \\ a_{26} \\ a_{25} \\ a_{24} \\ a_{23} \\ a_{22} \\ a_{21} \\ a_{20} \\ a_{34} \\ a_{33} \\ a_{32} \\ a_{31} \\ a_{30} \end{bmatrix}
=
\begin{bmatrix} q_i \\ q_l \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ q_f \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{48}$$

where A_i's are defined in (40). (48) is valid for both forward and backward paths. Solving (48) for forward and backward paths, we obtained the following solution (Fig. 3):

$$\begin{bmatrix} a_{14} & a_{44} \\ a_{10} & a_{40} \\ a_{26} & a_{56} \\ a_{25} & a_{55} \\ a_{24} & a_{54} \\ a_{23} & a_{53} \\ a_{22} & a_{52} \\ a_{21} & a_{51} \\ a_{20} & a_{50} \\ a_{34} & a_{64} \\ a_{33} & a_{63} \\ a_{32} & a_{62} \\ a_{31} & a_{61} \\ a_{30} & a_{60} \end{bmatrix} = \begin{bmatrix} 0.0800 & -0.0800 \\ 0.0000 & 1.0000 \\ -3.4695 & 3.4695 \\ 11.6716 & -11.6716 \\ -12.4098 & 12.4098 \\ 2.5600 & -2.5600 \\ 1.9200 & -1.9200 \\ 0.6400 & -0.6400 \\ 0.0800 & 0.9200 \\ -0.0077 & 0.0077 \\ 0.0309 & -0.0309 \\ -0.0463 & 0.0463 \\ 0.0309 & -0.0309 \\ 0.9923 & 0.0077 \end{bmatrix} \tag{49}$$

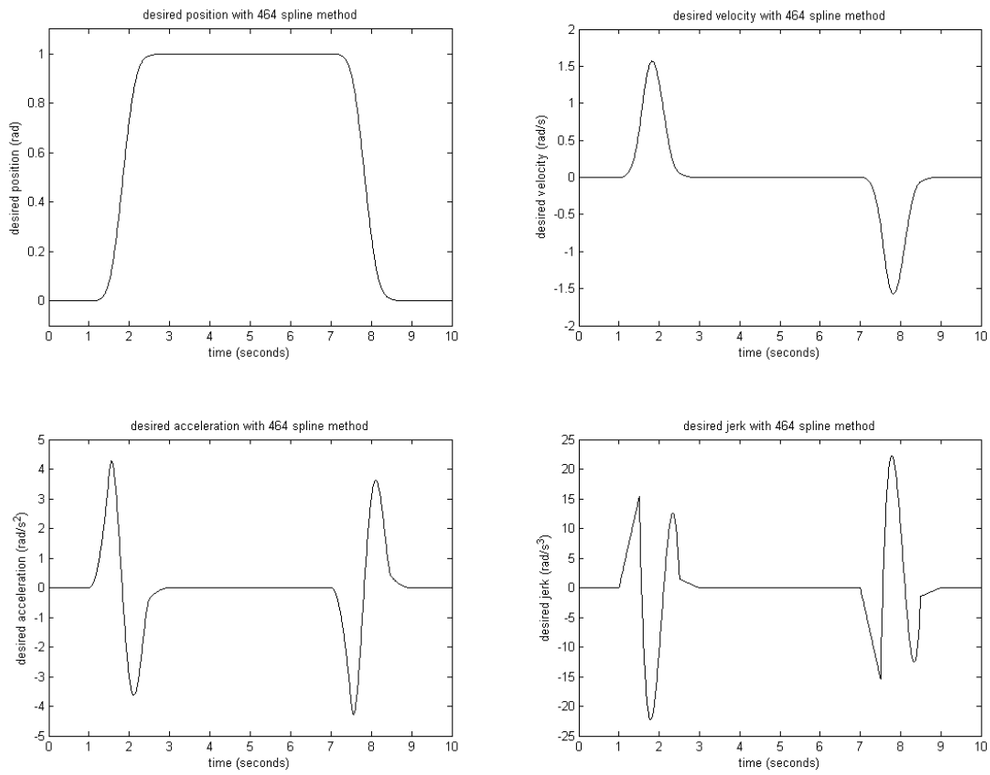

Fig. 3. Desired position and its derivatives with 4-6-4 spline method

5-4-5 Method

In this method, we should have

a_{i6}=0
for IL and SF subintervals in (41) and for the LS subinterval
a_{i6}=a_{i5}=0.
From the initial conditions in (43), we write
a_{10}=q_b, a_{11}=a_{12}=a_{13}=0
for forward path and
a_{40}=q_b, a_{41}=a_{42}=a_{43}=0
for backward path.
The conditions in (42), (43) and (44) give the following matrix equality for the unknown coefficients:

$$\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
5A_1 & 4A_1 & 0 & 0 & 0 & 0 & -A_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10A_1^2 & 6A_1^2 & 0 & 0 & 0 & -A_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10A_1^3 & 4A_1^3 & 0 & 0 & -A_2^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 4A_2 & 3A_2 & 2A_2 & A_2 & 0 & 0 & 0 & 0 & 0 & -A_3 & 0 \\
0 & 0 & 0 & 6A_2^2 & 3A_2^2 & A_2^2 & 0 & 0 & 0 & 0 & 0 & -A_3^2 & 0 & 0 \\
0 & 0 & 0 & 4A_2^3 & A_2^3 & 0 & 0 & 0 & 0 & 0 & -A_3^3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 4 & 3 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 6 & 3 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 4 & 1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
a_{15} \\ a_{14} \\ a_{10} \\ a_{24} \\ a_{23} \\ a_{22} \\ a_{21} \\ a_{20} \\ a_{35} \\ a_{34} \\ a_{33} \\ a_{32} \\ a_{31} \\ a_{30}
\end{bmatrix}
=
\begin{bmatrix}
q_i \\ q_l \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ q_f \\ 0 \\ 0 \\ 0
\end{bmatrix} \quad (50)$$

where A_i's are defined in (40). (50) is valid for both forward and backward paths. Solving (50) for forward and backward paths, we obtained the following solution (Fig. 4):

$$\begin{bmatrix}
a_{15} & a_{45} \\
a_{14} & a_{44} \\
a_{10} & a_{40} \\
a_{24} & a_{54} \\
a_{23} & a_{53} \\
a_{22} & a_{52} \\
a_{21} & a_{51} \\
a_{20} & a_{50} \\
a_{35} & a_{65} \\
a_{34} & a_{64} \\
a_{33} & a_{63} \\
a_{32} & a_{62} \\
a_{31} & a_{61} \\
a_{30} & a_{60}
\end{bmatrix}
=
\begin{bmatrix}
-0.0644 & 0.0644 \\
0.1444 & -0.1444 \\
0.0000 & 1.0000 \\
-0.0381 & 0.0381 \\
-0.5299 & 0.5299 \\
0.8900 & -0.8900 \\
0.5113 & -0.5113 \\
0.0800 & 0.9200 \\
-0.0720 & 0.0720 \\
0.2013 & -0.2013 \\
-0.0853 & 0.0853 \\
-0.2320 & 0.2320 \\
0.2747 & 0.2747 \\
0.9133 & 0.0867
\end{bmatrix} \quad (51)$$

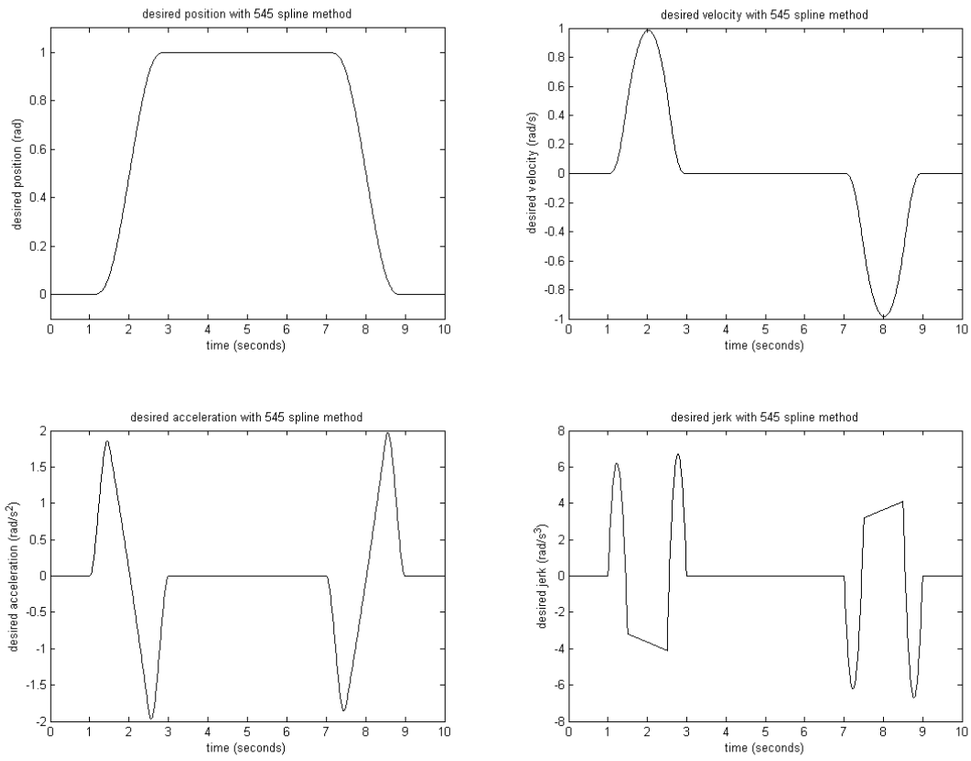

Fig. 4. Desired position and its derivatives with 5-4-5 spline method

Transcendental Function Methods

In these methods, an upper and lower bounded, monotone increasing or decreasing transcendental function, which is continuously differentiable up to the third order, is used. In order to obtain a periodic trajectory, the argument of the transcendental function should be a continuously differentiable periodic function up to the third order. Typically, a sinusoidal function can be selected as the argument. The advantage of using a transcendental function with a periodic argument function is that for each subinterval the same function is used. This reduces the formulation time of the trajectory. Also the trajectory function never has local maxima and minima in the subintervals, since they are monotone increasing or decreasing. However, it is not easy to determine the exact lift-off and set-down points. The derivatives of the function may become very complex, as in the hyperbolic tangent case. Fortunately, we need only the first order derivatives, because the controller uses only the desired trajectory and the desired velocity to compute the input functions, as in (24).

1.Hyberbolic Tangent Method

Let us use the hyperbolic tangent function for the trajectory. In this method, there is no need to consider the forward and backward subintervals of IL, LS, and SF. Instead, we use a hyperbolic tangent function with a continuously differentiable (at least up to the third order) periodic function as the argument. Indeed the method uses the fact that the hyperbolic

tangent function can take values in the interval [-1, 1]. Same function is valid for all times and for each subinterval of the trajectory. However, it is not easy to determine the boundaries for lift-off and set-down positions. There is no general method to determine these points. In this method the desired trajectory is defined as:

$$h(t)=b[a+d\tanh(c\cos(\omega t))] \tag{52}$$

Where b is a weighting constant in radians, a is the constant that determines the initial position, c is the constant that determines the lift-off and set-down positions, d is the constant which determines the difference between the initial (ba–bd) and final (ba+bd) positions, ω is the angular frequency of the desired trajectory. Note that cosine function in the argument is continuously differentiable of any order. The determination method of the constant c is trial and error. Typically, c should be selected large enough so that the trajectory reaches to its final position and remains there for some time without subjecting excessive velocities and accelerations for a pick and place task (Fig. 5). However, a, b, and d can be determined according to the initial and final desired positions. One can easily find the velocity, acceleration and jerk functions by taking the successive derivatives of (52) as

$$v(t)=\frac{d}{dt}h(t)=-bcd\omega\sin(\omega t)\operatorname{sech}^2\left(c\cos(\omega t)\right) \tag{53}$$

$$a(t)=\frac{d}{dt}v(t)=-bcd\omega^2\operatorname{sech}^2\left(c\cos(\omega t)\right)\left[\cos(\omega t)+2c\tanh(c\cos(\omega t)\sin^2(\omega t)\right] \tag{54}$$

$$\begin{aligned} j(t)=\frac{d}{dt}a(t)=bcd\omega^3\sin(\omega t)\operatorname{sech}^2\left(c\cos(\omega t)\right)\left[1-6c\cos(\omega t)\tanh(c\cos(\omega t)\right. \\ \left.+2c^2\sin^2(\omega t)\left\{\operatorname{sech}^2\left(c\cos(\omega t)\right)-2\tanh^2\left(c\cos(\omega t)\right)\right\}\right]. \end{aligned} \tag{55}$$

The jerk expression given in (14) can also be written as follows

$$\begin{aligned} j(t)=\frac{d}{dt}a(t)=bcd\omega^3\sin(\omega t)\operatorname{sech}^2\left(c\cos(\omega t)\right)\left[1-6c\cos(\omega t)\tanh(c\cos(\omega t)\right. \\ \left.+2c^2\sin^2(\omega t)\left\{3\operatorname{sech}^2\left(c\cos(\omega t)\right)-2\right\}\right]. \end{aligned} \tag{56}$$

2.Error Function Method:

Here the trajectory is selected as the integral of Gaussian distribution function, which is known as the error function, defined as

$$\operatorname{erf}(t)=\frac{2}{\sigma\sqrt{\pi}}\int_0^t\exp\left(-\frac{\tau^2}{\sigma^2}\right)d\tau\,. \tag{57}$$

To get a continuous and periodic function, we use a sinusoidal argument as in hyperbolic tangent function. The following function is periodic, continuous and differentiable at least up to the third order:

$$h(t)=B\left[A-D\operatorname{erf}\left(C\cos(\omega t)\right)\right]. \tag{58}$$

The advantage of this function is that the derivatives are in terms of simple exponential and sinusoidal functions. Note that the function given in (58), has the initial value of zero, if one selects $A=D$.

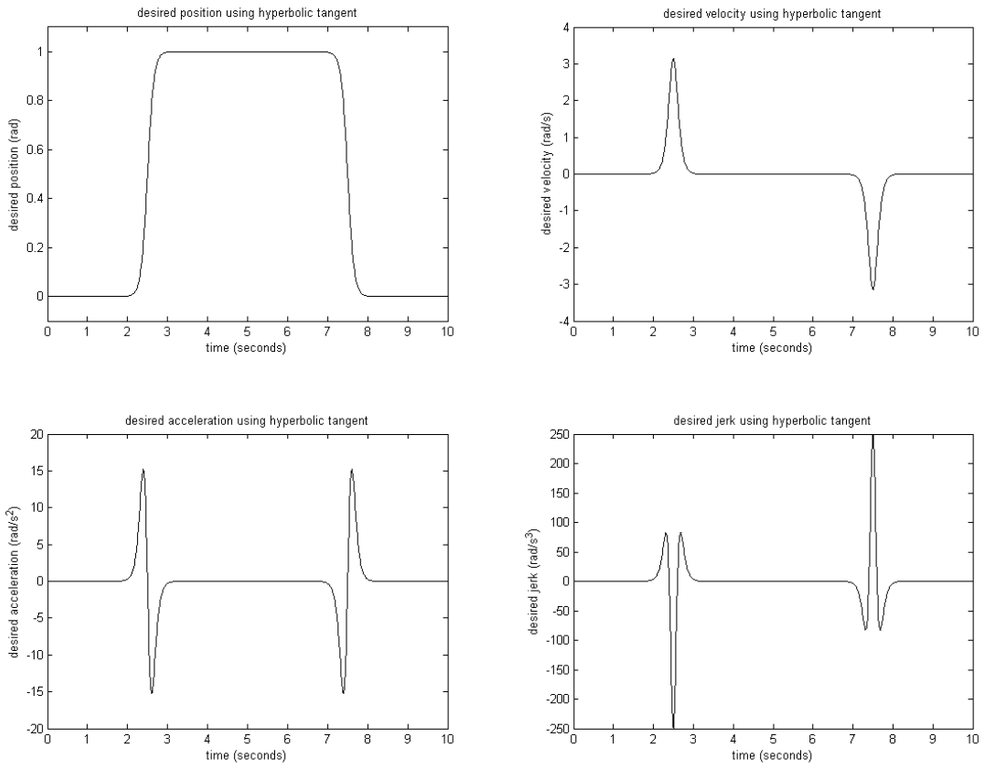

Fig. 5. Desired position with hyperbolic tangent function (c=10)

3.5 PD and Learning Controllers

A frequently used controller in control systems is the classical proportional-derivative (PD) controller (Das & Dulger, 2005). The main advantage of the PD controller is that it can easily be implemented on simple microcontroller architectures. On the other hand, the performance obtained from PD controllers is not satisfying for most of the sensitive applications. In this work, PD and learning controllers (Messner et al., 1991) are simulated along with the hybrid adaptive-learning controller (Canbolat et al. 1996) in order to compare the achieved performance. For this purpose, we repeat the main equations of learning controller below.

First, the main equation of the PD controller is

$$V_k(t) = K_p e(t) + K_d \frac{de(t)}{dt} \tag{59}$$

where $e(t)$ is the error function, K_p is the proportional control coefficient and K_d refers to the derivative control constant.

The learning algorithm proposed by (Messner et al., 1991) has more complex structure than a PD controller. The mechanical part of the robot dynamics can be rearranged as

$$M(q)\ddot{q} = T(t) - V_m(q,\dot{q})\dot{q} - G(q) - F_d(\dot{q}) \tag{60}$$

where $T(t)$ is the control torques applied to the joints and $q(t)=[q_1(t), q_2(t)]^T$ is the position of the manipulator, $\dot{q}$ and $\ddot{q}$ are nx1 vectors of joint velocities, and accelerations, respectively. The other terms are defined in (1). The following function, which includes the mechanical uncertainties, is defined

$$w_r(t) = M(q_d)\ddot{q}_d + V_m(q_d,\dot{q}_d)\dot{q}_d + G(q_d) \tag{61}$$

where $q_d(t)$ denotes the desired periodic trajectory vector of the robot links and the dots denote the differentiation with respect to time, t. Using these two equations, the following error dynamics and the desired compensation control law (DCCL) can be derived

$$\dot{e}(t) = f(t,e) + B(t,e)\left[w_r(t) - \hat{w}_r(t)\right] \tag{62}$$

$$T(t) = \hat{w}_r(t) + F_v e_v(t) + F_p e(t) + d_m(q,\dot{q}) + q_n(e_v) \tag{63}$$

where $e = \left[e_p^T \quad e_v^T\right]^T$ and $\hat{w}_r(t)$ is the estimate of the influence function, $w_r(t)$, that compensates for mechanical uncertainties, F_v and F_p are PD gains, $e_v(t)$ is the reference velocity error vector, $e(t)$ is the position error vector, $d_m(q,\dot{q})$ is the friction compensation. $q_n(e_v)$ is the nonlinear compensation function. As it can be seen from (63), the repetitive-learning algorithm utilizes PD control but also uses $\hat{w}_r(t)$ to compensate for the uncertain parameters, thus providing "learning". The position error is defined as

$$e_p(t) = q_d(t) - q(t) \tag{64}$$

where $q_d(t)$ is the desired trajectory. The reference velocity error function is defined as

$$e_v(t) = \dot{e}_p(t) + \lambda e_p(t)\ , \tag{65}$$

where λ is a positive constant and the nonlinear compensation function is given as:

$$q_n(e_v) = \sigma\left|e_p\right|^2\left|e_v\right| \tag{66}$$

The estimate of the uncertainty function $\hat{w}_r(t)$ is updated with following rules:

$$\hat{w}_r(t) = \int_0^T K(t,\tau)\hat{c}(t,\tau)d\tau \tag{67}$$

$$\frac{\partial \hat{c}(t,\tau)}{\partial t} = K(t,\tau)K_L R^T e(t) \tag{68}$$

where $K(t,\tau)$ is a function that can be selected by the designer as in hybrid controller, $e(t)$ is defined in (63), K_L and R are constant matrices (Messner et al., 1991).

4. SCARA Robot Model

The SCARA manipulator considered in this study is an experimental robot that has DC servo motors for the movements of elbow and shoulder. The third movement is controlled pneumatically. The schematic configuration of the robot is shown in Fig. 6.

The electrical and mechanical dynamical equations of the manipulator are as follows (Das & Dulger, 2005)

$$\begin{bmatrix} L_{a1} & 0 \\ 0 & L_{a2} \end{bmatrix}\begin{bmatrix} \dot{I}_{a1} \\ \dot{I}_{a2} \end{bmatrix} + \begin{bmatrix} R_{a1} & 0 \\ 0 & R_{a1} \end{bmatrix}\begin{bmatrix} I_{a1} \\ I_{a2} \end{bmatrix} + \begin{bmatrix} K_{e1} & 0 \\ 0 & K_{e2} \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} V_{a1} \\ V_{a2} \end{bmatrix} \tag{69}$$

$$\begin{bmatrix} K_{T1} & 0 \\ 0 & K_{T2} \end{bmatrix}\begin{bmatrix} I_{a1} \\ I_{a2} \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{70}$$

$$\begin{bmatrix} A & B \\ E & D \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} C \\ F \end{bmatrix} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{71}$$

The elements A, B, C, and D in (71) are defined as

$$A = J_{m1} + \frac{(J_1 + J_2)}{N_1^2} + \frac{m_1 r_1^2 + m_2 r_2^2 + 4m_2 r_1^2}{4N_1^2} + \frac{m_2 r_1 r_2}{N_1^2}\cos\left(\frac{\theta_{m2}}{N_2}\right) \tag{72a}$$

$$B = \frac{J_2}{N_1 N_2} + \frac{m_2 r_2^2}{4N_1 N_2} + \frac{m_2 r_1 r_2}{2N_1 N_2}\cos\left(\frac{\theta_{m2}}{N_2}\right) \tag{72b}$$

$$C = \frac{-m_2 r_1 r_2}{N_1 N_2}\left(\frac{\dot{\theta}_{m1}\dot{\theta}_{m2}}{N_1} + \frac{\dot{\theta}_{m2}^2}{2N_2}\right)\sin\left(\frac{\theta_{m2}}{N_2}\right) \tag{72c}$$

$$D = J_{m1} + \frac{J_2}{N_2^{\,2}} + \frac{m_2 r_2^{\,2}}{4N_2^{\,2}} \tag{72d}$$

$$E = \frac{J_2}{N_1 N_2} + \frac{m_2 r_2^{\,2}}{4N_1 N_2} + \frac{m_2 r_1 r_2}{2N_1 N_2}\cos\left(\frac{\theta_{m2}}{N_2}\right) \tag{72e}$$

$$F = \frac{m_2 r_1 r_2}{2N_1 N_2}\left(\frac{\dot{\theta}_{m1}^2}{N_1}\right)\sin\left(\frac{\theta_{m2}}{N_2}\right) \tag{72f}$$

where I_{a1} and I_{a2} are motor currents, V_{a1} and V_{a2} are motor voltages, T_1 and T_2 are motor torques, θ_1 and θ_2 are link angles and θ_{m1} and θ_{m2} are motor angles of link 1 and link 2, respectively. Other physical parameters and their values that appear in (72) for the Serpent-1 model SCARA robot are given in Table 1.

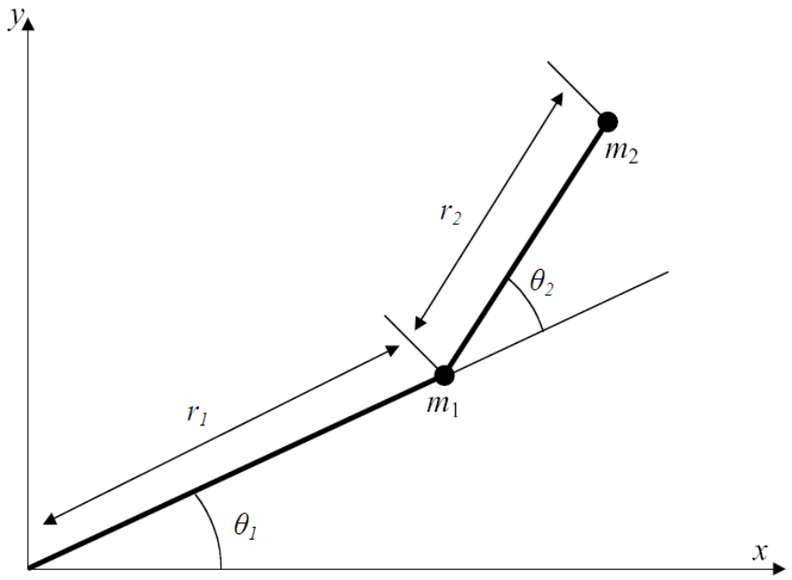

Fig. 6. Upper view of SCARA robot

Parameter	Meaning	Value
L_{a1}, L_{a2}	Armature inductances of motors 1 and 2	1.3mH, 1.3mH
R_{a1}, R_{a2}	Armature resistances of motors 1 and 2	3.5 Ω , 3.5 Ω
K_{e1}, K_{e2}	Inverse emf coefficients of motors 1 and 2	0.047 V.s/rad, 0.047 V.s/rad
K_{T1}, K_{T2}	Torque coefficients of motors 1 and 2	0.047 Nm/A, 0.047 Nm/A

J_1, J_2	Moment of inertias of arms 1 and 2	0.0980kgm², 0.0980kgm²
J_{m1}, J_{m2}	Inertias of motors 1 and 2	3.3.10-6 kgm², 3.3.10-6 kgm²
m_1, m_2	Masses of arms 1 and 2	1.90 kg, 0.93 kg
r_1, r_2	Lenghts of arms 1 and 2	250 mm, 150 mm
N_1, N_2	Gearbox ratios of motors 1 and 2	90, 220

Table 1. Serpent-1 robot parameters and their values

5. Simulation

Dynamics of the SCARA robot and three types of controllers, namely PD, learning and adaptive/learning controllers are modelled in MATLAB Simulink environment. A general simulation model is given in Fig. 7.

In the first simulation, the SCARA is controlled by PD controller. In this case, the electrical dynamics are neglected and the controller block is replaced with a PD controller (Fig.7). The control coefficients are selected as K_{p1}=300, K_{d1}=50, K_{p2}=30, K_{d2}=15 for link 1 and link 2, respectively (Das & Dulger, 2005).

As the second simulation, SCARA is controlled by learning controller. Here the electrical dynamics are again neglected and the controller block is replaced with the learning controller designed by (Messner et al., 1991). In the learning controller, the parameters are selected as;

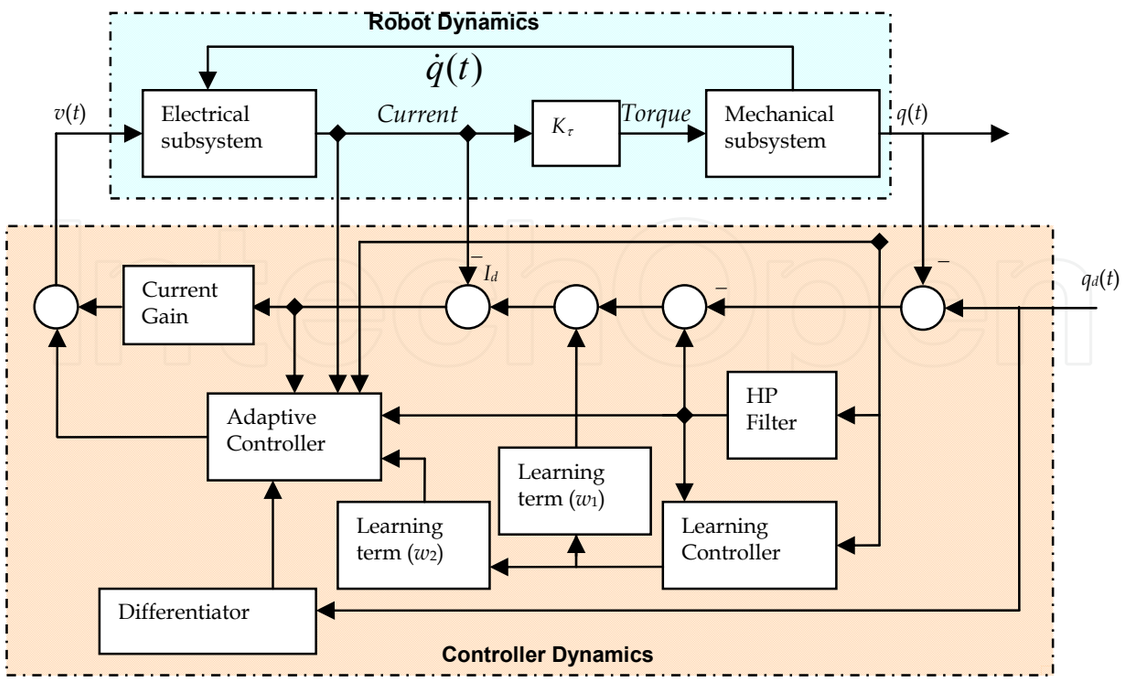

Fig. 7. Detailed Block diagram of robot and controller

$$F_p = \begin{bmatrix} 2000 & 0 \\ 0 & 160 \end{bmatrix} \tag{73}$$

$$F_v = \begin{bmatrix} 200 & 0 \\ 0 & 4 \end{bmatrix} \tag{74}$$

$$K_L = \begin{bmatrix} 2000 & 0 \\ 0 & 175 \end{bmatrix} \tag{75}$$

and λ_p=10, ve σ_n=0, $d_m(x_p)$=0 (Messner et al., 1991). The computation of $\dot{\hat{c}}_x$ and w_r are accomplished by numerical integration with embedded function blocks. The learning controllers have two different independent dynamic (time) variables. The simulation packages do not allow more than one independent simulation variables. To overcome this limitation, the second time variable is defined as a discrete variable and at every discrete point some state variables are introduced according to the dynamics. The differentiation and integration in the second variable are defined through summation and difference equations. The result is a heavy computational burden on the system.

The simulation model of the adaptive/learning hybrid controller is essentially the same as in Fig. 7. The parameters of the adaptive/learning controller are selected as; k=15, σ=12 and

$$K_L = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{76}$$

Again, the computation of $\dot{\hat{c}}_x$, $\dot{w}_1$, w_2 are realized with numerical integrator blocks.

The desired link angle function is chosen as

$$q_d(t) = -0.5 + (-1 + \tanh(10\cos(\omega t))), \tag{77}$$

where ω=1 rad/s.

The function given in (77) is a pick-and-place type task that is widely used in industrial applications. This trajectory function satisfies the periodicity and continuous 3rd order derivative requirements of hybrid/learning controller as discussed in section 3.4.

The desired and achieved link angles when PD controller is used and the link angle errors are given in Fig. 8 and Fig. 9, respectively. The maximum angle errors are 0.4 rad for first link and 0.65 rad for the second link.

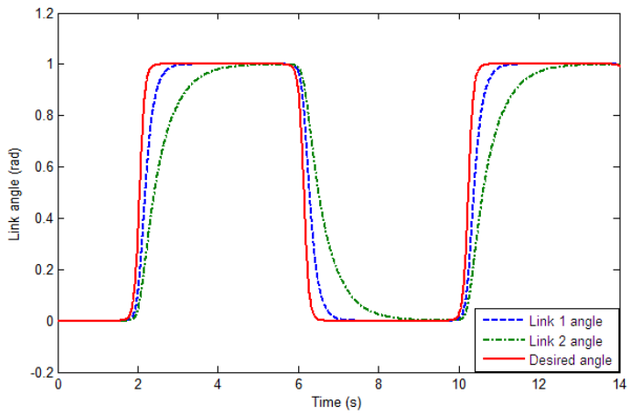

Fig. 8. Desired and simulated link angles when PD controller is utilized

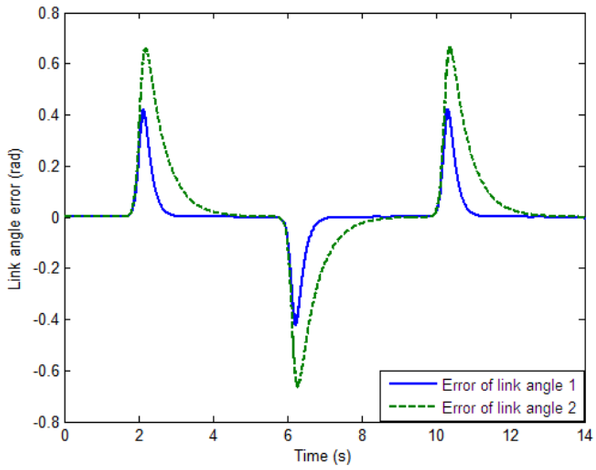

Fig. 9. Link angle errors when PD controller is used

Similarly, the link angle errors for learning controller are plotted in Fig. 10. The maximum angle errors are 0.09 rad for first link and 0.19 rad for the second link. The angle error decreased with respect to PD controller case as it is expected.

The link angle errors are given in Fig. 11 for the hybrid controller. Note that, the maximum link angles are lower compared to learning controller, 0.06 rad for both link 1 and link 2 (the error plots for link 1and 2 are overlapped in Fig. 11). It is worth noting that, the link angle errors have greater average values when hybrid controller is used. We think that the average value is greater for the hybrid controller, since it uses less information for the compensation of the uncertainties comparing with the learning controller given in (63), which uses both link positions and velocities. However the hybrid controller uses the measurements of link positions and motor currents. Furthermore, the learning controller neglects the electrical dynamics and compensates for only mechanical parameter uncertainties. On the other hand, the hybrid controller does not neglect electrical dynamics and compensates for mechanical and electrical parameter uncertainties. That is, the computational burden on the hybrid controller is much more than the learning controller. We think that this fact results more error in the average although the maximum error is less.

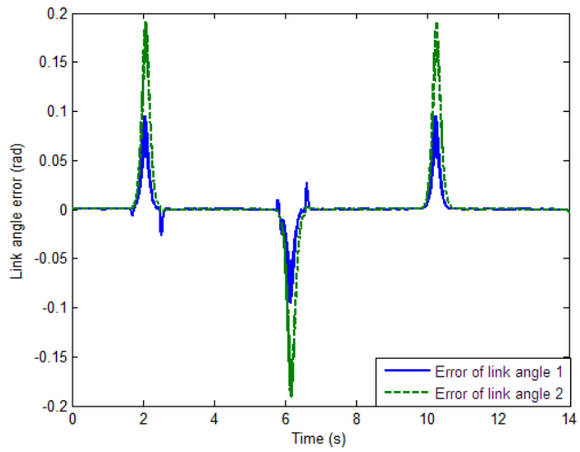

Fig. 10. Link angle errors when learning controller is used

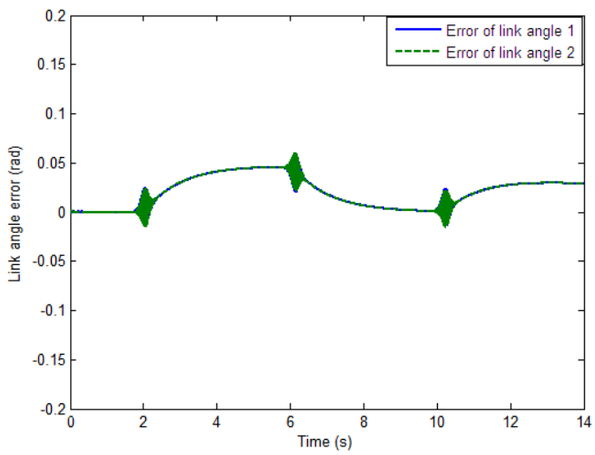

Fig. 11. Link angle errors when adaptive/learning controller is used

6. Conclusion

In this paper, the design of the hybrid adaptive/learning controller is described. Also the design of the learning controller proposed by (Messner et al., 1991) is described shortly along with a classical PD controller. The simulation model of a SCARA robot manipulator is presented and the performance of the controllers are examined through simulation runs. The simulation model and its parameters are based on a physical model of a SCARA robot given in (Das & Dulger, 2005). The simulation model includes the mechanical subsystem, electrical subsystem and the three different types of controllers. The classical PD, learning and adaptive/learning controller schemes are modelled and SCARA robot is simulated with three types of controllers.

The second time variable introduced in learning type controllers results a computational burden in dynamics, since the dynamics of controller is dependent both on the real time variable and the second time variable created via the Hilbert-Schmidt kernel used in learning laws. Moreover, no standard simulation package allows the use of a second

independent time variable in the models. To overcome this difficulty, we discretize the second variable. In order to keep the dynamics with respect to that variable we should have introduced a large number of extra system states at each discrete point of the second variable. Although the simulation is sufficiently fast with a high performance (1.7GHz CPU and 512MB RAM) personal computer, it is not fast enough with a personal computer of lower specifications (667Mhz CPU and 64MB RAM). Considering the much slower computers employed for the single task of controlling industrial robots, a real time application apparently is not possible at this stage. Therefore, the work to reduce the computational burden in the control law is continuing and as soon as this is achieved, an experiment to examine the hybrid controller for a real robot will be performed.

The parameters of a 2-link Serpent-1 model robot are used in simulations and the robot is desired to realize a pick and place type movement. According to the simulation results, the learning and adaptive/learning hybrid controllers provided lower angle errors compared to classical PD controller. Moreover, the maximum angle errors of links when controlled by adaptive/learning controller decreased from 0.09 rad to 0.06 rad for first link and 0.19 rad to 0.06 rad for second link compared to learning controller, which means 33.3% and 63.1% decrement for first link and second link, respectively.

Although the hybrid controller is more complex than PD and learning controllers, its position and velocity errors have smaller maximum values than the learning controller. However its performance is not good in the error averages. We think that the high error averages are due to the fact that the hybrid controller uses partial state information (no link velocities) and compensates for both mechanical and electrical parameter uncertainties, whereas the learning controller uses full state information (both link positions and velocities) though it compensates only for mechanical uncertainties, since it neglects electrical dynamics.

Our work is continuing to develop more powerful computational schemes for the hybrid adaptive/learning controller to reduce the computational burden. Recently, we tried to introduce a low pass filter in the hybrid controller to filter the high frequency components, which effect the tracking performance negatively, in the input voltage. The preliminary results show that the error becomes smoother and its average value reduces.

7. References

Arimoto, S. (1986). Mathematical theory of learning with applications to robot control, In: Adaptive and Learning Systems, K.S. Narendra (Ed.), Plenum Press, ISBN: 0306422638, New York.

Arimoto, S.; Kawamura, S.; Miyazaki, F. & Tamaki, S. (1985). Learning control theory for dynamical systems. Proceedings of IEEE 24th Conference on Decision and Control, 1375-1380, ISBN: 9999269222, Ft. Lauderdale FL, December 1985, IEEE Press, Piscataway NJ.

Bondi, P.; Casalino, G. & Gambardella, L. (1988). On the iterative learning control theory of robotic manipulators. IEEE Journal of Robotics and Automation, Vol. 4, No.1, (February 1988), 14-22, ISSN: 0882-4967.

Burg, T.; Dawson, D. M.; Hu, J. and de Queiroz, M. (1996). An adaptive partial state feedback controller for RLED robot manipulators. IEEE Transactions on Automatic Control, Vol. 41, No. 7, (July 1996), 1024-1030, ISSN:0018-9286.

Canbolat, H.; Hu, J. & Dawson, D.M. (1996). A hybrid learning/adaptive partial state feedback controller for RLED robot manipulators. International Journal of Systems Science, Vol. 27, No. 11, (November 1996), 1123-1132, ISSN:0020 7721.

Das, T. & Dülger, C. (2005). Mathematical Modeling, Simulation and Experimental Verification of a SCARA Robot. Simulation Modelling Practice and Theory, Vol.13, No.3, (April 2005), 257-271, ISSN:1569-190X.

De Queiroz, M.S.; Dawson, D.M. & Canbolat, H. (1997). Adaptive Position/Force Control of BDC-RLED Robots without Velocity Measurements. Proceedings of the IEEE International Conference on Robotics and Automation, 525-530, ISSN:1050-4729, Albuquerque NM, April 1997, IEEE Press, Piscataway NJ.

Fu, K.S.; Gonzalez, R.C. & Lee, C.S.G. (1987). Robotics: Control, Sensing, Vision, and Intelligence, McGraw-Hill, ISBN:0-07-100421-1, New York.

Golnazarian, W. (1995). Time-Varying Neural Networks for Robot Trajectory Control. Ph.D. Thesis, University Of Cincinnati, U.S.A.

Horowitz, R.; Messner, W. & Moore, J. (1991). Exponential convergence of a learning controller for robot manipulators. IEEE Transactions on Automatic Control, Vol. 36, No. 7, (July 1991), 890-894, ISSN:0018-9286.

Jungbeck, M. & Madrid, M.K. (2001). Optimal Neural Network Output Feedback Control for Robot Manipulators. Proceedings of the Second International Workshop on Robot Motion Control, 85-90, ISBN: 8371435150, Bukowy Dworek Poland, October 2001, Uniwersytet Zielonogorski, Instytut Organizacji i Zarzadzania.

Kaneko, K.& Horowitz, R. (1992). Learning control of robot manipulators with velocity estimation. Proceedings of USA/Japan Symposium on Flexible Automation, 828-836, ISBN: 0791806758, M. Leu (Ed.), San Fransisco CA, July 1992, ASME.

Kaneko, K. & Horowitz, R. (1997). Repetitive and Adaptive Control of Robot Manipulators with Velocity Estimation. IEEE Trans. Robotics and Automation, Vol. 13, No. 2 (April 1997), 204-217, ISSN:1042-296X.

Kawamura, S.; Miyazaki, F. & Arimoto, S. (1988). Realization of robot motion based on a learning method. IEEE Transactions on Systems, Man and Cybernetics, Vol.18, No. 1, (Jan/Feb 1988), 126-134, ISSN:0018-9472.

Kuc, T.; Lee, J. & Nam, K. (1992). An iterative learning control theory for a class of nonlinear dynamic systems. Automatica Vol.28, No.6, (November 1992), 1215-1221, ISSN:0005-1098.

Lewis, F.L.; Abdallah, C.T. & Dawson, D.M. (1993). Control of Robot Manipulators, Macmillan, ISBN: 0023705019, New York.

Messner, W.; Horowitz, R.; Kao, W.W. & Boals M. (1991). A new adaptive learning rule. IEEE Transactions on Automatic Control, Vol. 36, No. 2, (February 1991) 188-197, ISBN:0018-9286.

Qu, Z.; Dorsey, J.; Johnson, R. & Dawson, D.M. (1993). Linear learning control of robot motion. Journal of Robotic Systems Vol.10, No.1, (February 1993), 123-140, ISBN: 0741-2223.

Sadegh, N.; Horowitz, ; Kao, W.W. & Tomizuka, M. (1990). A unified approach to the design of adaptive and repetitive controllers for robotic manipulators. ASME Journal of Dynamic Systems, Measurement and Control, Vol.112, No.4 (December 1990), 618-629, ISSN: 0022-0434.

Sahin, V.D. & Canbolat, H. (2007). DC Motorlarla Sürülen Robot Manipülatörleri için Gecikmeli Öğrenme Denetleyicisi Tasarımı (Design of Delayed Learning Controller for RLED Robot Manipulators Driven by DC Motors). TOK'07 Otomatik Kontrol Milli Toplantısı Bildiriler Kitabı (Proc. of TOK'07 Automatic Control National Meeting), 130-133, Istanbul, Turkey, September 2007, Istanbul (Turkish).

Uğuz, H. & Canbolat, H. (2006). Simulation of a Hybrid Adaptive-Learning Control Law for a Rigid Link Electrically Driven Robot Manipulator. Robotica, vol.24, No.3, (May 2006), 349-354, ISSN: 0263-5747.

Advances in Robot Manipulators

Edited by Ernest Hall

ISBN 978-953-307-070-4
Hard cover, 678 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

The purpose of this volume is to encourage and inspire the continual invention of robot manipulators for science and the good of humanity. The concepts of artificial intelligence combined with the engineering and technology of feedback control, have great potential for new, useful and exciting machines. The concept of eclecticism for the design, development, simulation and implementation of a real time controller for an intelligent, vision guided robots is now being explored. The dream of an eclectic perceptual, creative controller that can select its own tasks and perform autonomous operations with reliability and dependability is starting to evolve. We have not yet reached this stage but a careful study of the contents will start one on the exciting journey that could lead to many inventions and successful solutions.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen