ROBOT TRAJECTORY TRACKING WITH SELF-TUNING PREDICTED CONTROL

Xianzhong Cui and Kang G. Shin

Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

ABSTRACT

A controller that combines self-tuning prediction and control is proposed as a new approach to robot trajectory tracking. The controller has two feedback loops: One is used to minimize the prediction error and the other is designed to make the system output track the set point input. Because the velocity and position along the desired trajectory are given and the future output of the system is predictable, a feedforward loop can be designed for robot trajectory tracking with self-tuning predicted control (STPC). Parameters are estimated on-line to account for the model uncertainty and the time-varying property of the system.

We have described the principle of STPC, analyzed the system performance, and discussed the simplification of the robot dynamic equations. To demonstrate its utility and power, the controller is simulated for a Stanford arm.

Index Terms - Robot trajectory tracking, self-tuning predicted control (STPC), auto-regressive moving average.

1. INTRODUCTION

Because of their flexibility and capability, computer controlled robots have become an essential component of modern manufacturing systems. "Optimal" (in some sense) trajectory control of such a robot is very important for improving manufacturing productivity and product quality.

When the robot moves with high speed or requires accurate positioning or is driven by high torque motors, one must consider its nonlinear characteristics and parameter uncertainties. Moreover, the controller's computational requirements are an important issue for real-time implementation. It is well-known that an industrial robot is a nonlinear, time-varying and highly-coupled multiple input multiple output (MIMO) system. For such a system, adaptive control appears to have advantages in handling the parameter uncertainties and stringent requirements on the positioning accuracy and motion speed.

Adaptive robot control techniques can be categorized into two types: *model reference adaptive control* (MRAC) and *self-tuning control* (STC). For STC, the dynamics of a robotic manipulator must be described by a controlled auto-regressive moving average (ARMAX) model. If the dynamic coupling is neglected, then the model becomes multiple single input single output (SISO) systems. The work described in [1] is a typical example of multiple decoupled STC's, one for each joint. This design method may also be extended to MIMO systems, as discussed in [1] and [2].

Although the STC in [1] has the advantages of implementation simplicity and good performance, the choice of the weight coefficient ε_i is crucial to system performance. It decides not only system transient responses but also system stability. When ε_i=0, the controller becomes a minimum variance controller, which is unstable for non-minimum phase systems. If ε_i is too small, the system will have large oscillations. On the other hand, if ε_i is too large, the system response will be very slow. The only way of choosing ε_i is trial and error. To remedy the difficulty in choosing the weight coefficient, one can attempt to use a pole placement self-tuning controller. Closed-loop system poles are assigned to predesigned positions. The work in [3] is a good example of direct application of the pole placement to control a Stanford arm.

The STC can be realized in either joint or Cartesian space. Because it is the robot's end-effector, not the robot's joints, that performs the task, the performance may not be acceptable at the end-effector, even if it is acceptable at the joints. Therefore, it is highly desirable to formulate a control law directly in the task coordinate frame [4]. An example of hybrid control can be found in [5], where the position and force errors were transformed first from the end-effector's Cartesian coordinates to those in joint space. Then, the position and force errors at each joint were combined into one hybrid error which was eliminated using the pole placement STC.

Some researchers presented the results of combining the STC with other techniques. For example, path control was combined with visual information [6]. The position and orientation of an object were determined from a binary image, and the extracted information was used for a self-tuning controller. The ideas of "resolved motion rate control" and "resolved acceleration control" were used to design a feedforward loop [7]. They resolved the specified positions, velocities and accelerations of the robot's hand into a set of values of joint positions, velocities and accelerations from which the joint torques were computed. The feedback loop was then designed using the STC to reduce the errors in the hand's position and velocity.

The work reported here was supported in part by the NSF under Grant No. ECS-8409938 and the NASA Johnson Space Center under Grant No. NCC-9-16. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors do not necessarily reflect the views of the funding agencies.

The drawback of the pole placement STC is that a time delay appears in the closed-loop characteristic equation, which affects system performance. This is especially true for a system with long delays.

In this paper, we present a new approach to robot trajectory tracking that combines the self-tuning prediction and control (STPC), thus resulting in two feedback loops. One is used to minimize the prediction error and the other is designed to make the system output track the set point input, i.e., the desired robot trajectory. Using a separate predictor, the negative effect of time delay is eliminated from the controller. This controller, unlike the one in [1], does not require to choose any weight coefficient. The performance of the controller depends only on the convergence of the predictor, regardless of whether or not the system is in minimum phase, thereby making the controller more suitable for a time-varying system like industrial robots. Because the future velocity and position along the desired trajectory are given and the future output of the system is predictable, a feedforward loop can be added to speed up the tracking operation. Unlike the resolved motion rate or acceleration algorithms, the computation of this feedforward loop is very simple, which is important for real-time implementation.

In Section 2, an ARMAX model is derived from the robot dynamic equations, on which the STPC will be based. The self-tuning predictor and parameter estimation are described in Section 3. A predicted controller with a feedforward loop is presented in Section 4. Section 5 deals with the analysis of system performance and the determination of the controller parameters. Section 6 presents the simulation results for the STPC of a Stanford arm, and the conclusion follows in Section 7.

2. ROBOT DYNAMIC EQUATIONS AND THEIR SIMPLIFICATION

An ARMAX model of a system has the following form:

$$\mathbf{A}(z^{-1})\,\mathbf{y}(k) = \mathbf{B}(z^{-1})\,\mathbf{u}(k-d) + \mathbf{C}(z^{-1})\,\mathbf{e}(k), \tag{2.1}$$

where $\mathbf{A}(z^{-1}) = \mathbf{I} + \mathbf{A}_1 z^{-1} + \ldots\ldots + \mathbf{A}_{n_a} z^{-n_a}$,

$\mathbf{B}(z^{-1}) = \mathbf{B}_0 + \mathbf{B}_1 z^{-1} + \ldots\ldots + \mathbf{B}_{n_b} z^{-n_b}$,

$\mathbf{C}(z^{-1}) = \mathbf{I} + \mathbf{C}_1 z^{-1} + \ldots\ldots + \mathbf{C}_{n_c} z^{-n_c}$,

$\mathbf{y}(t)$ and $\mathbf{u}(t)$ are respectively the n-dimensional output and input of the system at a discrete time t, d is the time delay, and $\mathbf{e}(t)$ is a random sequence with zero mean and $\sigma^2 = 1$ that describes the model uncertainty. $\mathbf{A}(z^{-1})$, $\mathbf{B}(z^{-1})$ and $\mathbf{C}(z^{-1})$ are $n \times n$ matrix polynomials of z^{-1} with the order of n_a, n_b and n_c, respectively. If the STC is to be used for robot trajectory tracking, the robot dynamic equations should be converted to the same form as Eq. (2.1).

The robot dynamics are described by nonlinear second-order differential equations:

$$\sum_{j=1}^{n} D_{ij}(\boldsymbol{\theta})\,\ddot{\theta}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{\theta})\,\dot{\theta}_j\dot{\theta}_k + G_i(\boldsymbol{\theta}) = \tau_i, \tag{2.2}$$

$1 \le i \le n$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^T$ is the vector representing joint angle displacements, n is the number of joints of the robot, $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$ are the joint velocity and acceleration vectors, respectively, and $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_n)^T$ is the joint torque/force vector. The first term of Eq. (2.2) represents the inertial torques, the second term represents the Coriolis and centrifugal forces and the third term is the gravitational loading. The major difficulty associated with the conversion of Eq. (2.2) to the form of (2.1) is the nonlinear term, $\sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{\theta})\,\dot{\theta}_j\dot{\theta}_k$, and the coupling parameters, $D_{ij}(\boldsymbol{\theta})$, $i \neq j$, and $G_i(\boldsymbol{\theta})$.

Rewrite Eq. (2.2) in the following matrix form:

$$\mathbf{D}(\boldsymbol{\theta})\,\ddot{\boldsymbol{\theta}} + \mathbf{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{\tau}. \tag{2.3}$$

Let the operating point be the current position and velocity, denoted by $\mathbf{Q}_0 = (\boldsymbol{\theta}_0, \dot{\boldsymbol{\theta}}_0)$, and assume that the changes of $\mathbf{D}$ around $\mathbf{Q}_0$ is negligible, i.e., $\mathbf{D}(\boldsymbol{\theta}_0 + \mathbf{q}) \approx \mathbf{D}(\boldsymbol{\theta}_0)$, where $(\mathbf{q}, \dot{\mathbf{q}})$ is a small perturbation around $\mathbf{Q}_0$. Then, using the Taylor series expansion of Eq. (2.3) about $\mathbf{Q}_0$ and neglecting the second and higher order terms, we get a linearized form of Eq. (2.3):

$$\mathbf{D}_0\,\ddot{\mathbf{q}}(t) + \mathbf{H}_0\,\dot{\mathbf{q}}(t) + \mathbf{G}_0\,\mathbf{q}(t) = \mathbf{u}(t), \tag{2.4}$$

where $\mathbf{D}_0 = \mathbf{D}(\boldsymbol{\theta}_0)$, $\mathbf{H}_{0ij} = \left.\dfrac{\partial \mathbf{H}_i}{\partial \dot{\theta}_j}\right|_{\mathbf{Q}_0}$

$\mathbf{G}_{0ij} = \left.\dfrac{\partial \mathbf{H}_i}{\partial \dot{\theta}_j}\right|_{\mathbf{Q}_0} + \left.\dfrac{\partial \mathbf{G}_i}{\partial \theta_j}\right|_{\mathbf{Q}_0}$, and $\mathbf{u} \in \mathbf{R}^n$ is the differential joint torque resulting from the perturbation.

Change the differential equation (2.4) to a difference equation using

$$\dot{\mathbf{q}}(t) = \frac{\mathbf{q}(k+T) - \mathbf{q}(k)}{T}.$$

Assuming the sampling interval $T = 1$, if $\mathbf{D}_0$ is nonsingular, then we can get a MIMO difference equation:

$$\mathbf{A}(z^{-1})\,\mathbf{q}(k) = \mathbf{B}(z^{-1})\,\mathbf{u}(k-d),$$

where $\mathbf{A}(z^{-1}) = \mathbf{I} + \mathbf{A}_1 z^{-1} + \mathbf{A}_2 z^{-2}$

$= \mathbf{I} + \mathbf{D}_0^{-1}(\mathbf{H}_0 - 2\,\mathbf{D}_0) z^{-1} + \mathbf{D}_0^{-1}(\mathbf{D}_0 - \mathbf{H}_0 + \mathbf{G}_0) z^{-2}$

$\mathbf{B}(z^{-1}) = \mathbf{B}_0 = \mathbf{D}_0^{-1}$, and $d = 2$.

According to [8], Eq. (2.3) for a Stanford arm can be written as a simplified set of equations under the assumption that $\dot{\theta}_j\dot{\theta}_k = 0$, $1 \le i, j \le 6$, and $d_{ij}=0$ for all $i \neq j$. Then we get

$$\mathbf{D}(\boldsymbol{\theta})\,\ddot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{\tau} \tag{2.5}$$

where $\mathbf{D}(\boldsymbol{\theta}) = diag\,(d_{ii}) \equiv$ inertial terms, and $\mathbf{G}(\boldsymbol{\theta}) = Col\,(g_i) \equiv$ gravity terms. Again expanding (2.5) with a Taylor series around the nominal position $\boldsymbol{\theta}_0(t)$ and neglecting the second and higher order terms, we get:

$$\mathbf{D}_0\,\ddot{\mathbf{q}}(t) + \mathbf{G}_0\,\mathbf{q}(t) = \mathbf{u}(t) \tag{2.6}$$

where $\mathbf{D}_0 = diag\left[d_{ii}(\boldsymbol{\theta}_0)\right]$, $\mathbf{G}_0 = \left[\left.\dfrac{\partial g_i}{\partial \theta_j}\right|_{\boldsymbol{\theta}_0}\right]$, $1 \le i, j \le 6$.

If the interacting effects of gravity among joints are neglected,[1] i.e., assume $\mathbf{G}_0$ to be a diagonal matrix, Eq. (2.6) will become uncoupled,

$$d_{ii}\,\ddot{q}_i(t) + g_{0ii}\,q_i(t) = u_i(t), \qquad 1 \le i \le 6. \tag{2.7}$$

[1] The error resulting from this approximation will be compensated for by introducing an additional term into each joint ARMAX model.

Replacing differential with difference leads to:

$$d_{ii}\left[\frac{q_i(k+2T)-2\,q_i(k+T)+q_i(k)}{T^2}\right]+g_{0ii}\,q_i(k)=u_i(k).$$

Again by letting the sampling interval $T = 1$, we get

$$A(z^{-1})\,q_i(k)=B(z^{-1})\,u_i(k-d) \tag{2.8}$$

where $A(z^{-1})=1+a_1 z^{-1}+a_2 z^{-2}$, $B(z^{-1})=b_0$, $d=2$, $a_1=-2$, $a_2=\dfrac{d_{ii}+g_{cii}}{d_{ii}}$, and $b_0=\dfrac{1}{d_{ii}}$. (Note that $d_{ii}\neq 0$ for all i.) Considering the modeling error and system disturbance, a disturbance term $C(z^{-1})\,e(k)$ is added to Eq. (2.8), where $e(k)$ is an uncorrelated random sequence with zero mean, and $C(z^{-1})$ is a polynomial of z^{-1} with unknown coefficients. A scalar term $h(k)$ can be added to this equation to express the gravity coupling effects [3]. Then the final equation becomes

$$A(z^{-1})\,q_i(k)=B(z^{-1})\,u_i(k-d)+C(z^{-1})\,e(k)+h(k). \tag{2.9}$$

Because the linearization is achieved around the operating point, if the operating point is changed, the linearization should be repeated, i.e., the parameters of Eq. (2.9) are time-varying. For a time-varying system, the application of adaptive control algorithms is natural. Note, however, that for the STPC we do not linearize the robot dynamic equations every time the operating point is changed. Instead, the change of the operating point is handled by on-line estimation of the model parameters.

3. SELF-TUNING PREDICTOR AND PARAMETER ESTIMATION

For a SISO system described by

$$A(z^{-1})\,y(t)=B(z^{-1})\,u(t-d)+C(z^{-1})\,e(t) \tag{3.1}$$

define the prediction error

$$\varepsilon(t+k)=y(t+k)-\hat{y}(t+k/t) \tag{3.2}$$

Substituting (3.2) into (3.1), we get [9]

$$A\,\varepsilon(t)=B\,u(t-d)-A\,\hat{y}(t/t-k)+C\,e(t). \tag{3.3}$$

Eq. (3.3) represents a new system in which the input is the prediction $\hat{y}(t/t-k)$, the output is the prediction error $\varepsilon(t)$, $u(t-d)$ is a measurable noise and $e(t)$ is an unmeasurable noise. We want to determine an input $\hat{y}(t/t-k)$ to minimize some cost functional resulting from the prediction error.

For the system (3.3), the expected squared prediction error is used as the cost functional, i.e.,

$$J=\mathbf{E}\,[\varepsilon^2(t+k)]. \tag{3.4}$$

Define an identity

$$C=E_0 A+z^{-k}F \tag{3.5}$$

where E_0 and F are polynomials of z^{-1}, and $deg\,(E_0)=k-1$, $deg\,(F)=n-1$ and n is the order of system (3.3). Now, the problem is to derive the optimal input by minimizing J, which is a typical minimum variance control problem. To minimize J, we get the optimal predictor

$$\hat{y}(t+k/t)=\frac{B\,z^{-d}}{A}\,u(t+k)+\frac{F}{E_0 A}\,\varepsilon(t). \tag{3.6}$$

The prediction error associated with Eq. (3.6) is

$$\varepsilon(t+k)=E_0\,e(t+k). \tag{3.7}$$

Since $deg\,(E_0)=k-1$, the prediction error is a $k-1$-th order moving average sum of $e(t)$.

Although Eq. (3.6) gives the optimal predictor, the parameters, A, B, E_0 and F, are unknown. These parameters are estimated on-line as outlined below. If C in Eq. (3.5) is equal to 1, then $A=\dfrac{1-z^{-k}F}{E_0}$ Plugging this into Eq. (3.3), we get

$$\varepsilon(t+k)=-E_0 A\,\hat{y}(t+k/t)+F\,\varepsilon(t)+B\,E_0\,u(t+k-d)+E_0\,e(t+k).$$

Let $F=P$, $E_0 A=-Q$ and $B\,E_0=R$, where $deg\,(P)=n-1$, $deg\,(Q)=n+k-1$, and $deg\,(R)=n+k-1$. Then, the parameter estimation model becomes

$$\varepsilon(t+k)=Q\,\hat{y}(t+k/t)+P\,\varepsilon(t)+R\,u(t+k-d)+\eta(t+k), \tag{3.8}$$

where $\eta(t+k)=E_0\,e(t+k)$. The recursive least squares (RLS) method can be used to estimate the parameters in Eq. (3.8). The number of estimated parameters is $3n+2k-1$. Let $\hat{P}$, $\hat{Q}$ and $\hat{R}$ be the estimates of P, Q, and R, respectively. Then, using the estimated parameters, the predictor Eq. (3.6) becomes

$$\hat{y}(t+k/t)=(1+\hat{Q})\,\hat{y}(t+k/t)+\hat{P}\,\varepsilon(t)+\hat{R}\,u(t+k-d). \tag{3.9}$$

4. THE PREDICTED CONTROLLER WITH A FEED-FORWARD LOOP

For the system (3.1), define a performance index

$$J=\mathbf{E}\left\{\left[\left[\Gamma\,y(t+d)-\Phi\,y_r(t)\right]^2+\left[\Lambda_0\,u(t)\right]^2\right]\Big|_t\right\}. \tag{4.1}$$

The problem is then to choose an input to minimize (4.1). Because $y(t+d)=\varepsilon(t+d)+\hat{y}(t+d/t)$ and $\varepsilon(t+d)$ is only related to the noise input $e(t)$, Eq. (4.1) becomes

$$J=\mathbf{E}\left\{\left[\Gamma\,\hat{y}(t+d/t)-\Phi\,y_r(t)\right]^2+\left[\Lambda_0\,u(t)\right]^2\right\}+\mathbf{E}\left\{\left[\Gamma\,\varepsilon(t+d)\right]^2\right\}\geq\mathbf{E}\left\{\left[\Gamma\,\varepsilon(t+d)\right]^2\right\}.$$

J becomes the minimum when the first term attains the minimum. The first term is only related to the measured system input and output. The problem then becomes to choose an input to minimize

$$J_1=\left\{\Gamma\,\hat{y}(t+d/t)-\Phi\,y_r(t)\right\}^2+\left\{\Lambda_0\,u(t)\right\}^2. \tag{4.2}$$

The control that minimizes J_1 is given by

$$u(t)=\frac{\Phi\,y_r(t)-\Gamma\,\hat{y}(t+d/t)}{\Lambda},\quad\text{where}\quad\Lambda=\frac{\Lambda_0\,\lambda_{01}}{r_0\,\gamma_0}. \tag{4.3}$$

From Eq. (4.3) and considering Eq. (3.6) and (3.1), the closed-loop system can be written as

$$y(t)=\frac{B\,\Phi\,z^{-d}\,y_r(t)}{\Lambda A+\Gamma B}-\frac{F\,\Gamma\,B\,z^{-d}}{A\,E_0\,(\Lambda A+\Gamma B)}\,\varepsilon(t)+\frac{C}{A}\,e(t). \tag{4.4}$$

Using Eq. (3.5) and $\varepsilon(t) = E_0\, e(t)$, we get

$$y(t) = \frac{B\ \Phi\ z^{-d}}{\Lambda A + \Gamma B}\ y_r(t) + \frac{C\ \Lambda + E_0\, B\ \Gamma}{\Lambda A + \Gamma B}\ e(t). \qquad (4.5)$$

The characteristic equation is $\Lambda A + \Gamma B = 0$ for which Λ and Γ can be chosen to make the system stable and provide the desired performance. Block diagrams for the system discussed thus far are given in Fig. 1.

In the case of robot trajectory tracking, the future set point input of the system is given as the desired motion trajectory of the robot's end-effector, and the system's future output is predictable (via the predicted controller). Thus, these two signals can be combined into a feedforward loop, which improves the system's tracking speed and accuracy.

A feedforward loop is added to the controller (4.3) so that the new control becomes

$$u^*(t) = u_1(t) + u_2(t) \qquad (4.6)$$

$$= \frac{\Phi\ y_r(t) - \Gamma\ \hat{y}(t+d/t)}{\Lambda} + \left\{ y_r(t+1) - \hat{y}(t+d+1/t) \right\} G.$$

The linearity of the controlled system allows this loop to be analyzed separately from the rest of the system. For example, one can simply choose $G(z^{-1}) = G_0 = \text{constant}$. Then

$$u_2(t) = \left\{ y_r(t+1) - \hat{y}(t+d+1/t) \right\} G_0, \qquad (4.7)$$

and the closed-loop system becomes

$$y_2 = \frac{B\ z^{-d}}{A} \left\{ y_r(t+1) - \hat{y}(t+d+1/t) \right\} G_0 + \frac{C}{A}\ e(t). \qquad (4.8)$$

The system block diagram is shown in Fig. 2.

5. DETERMINATION OF THE CONTROLLER PARAMETERS

The performance of the closed-loop system (4.5) is determined by the characteristic equation

$$\Lambda A + B\ \Gamma = 0. \qquad (5.1)$$

The pole placement method can be used to move poles in the z–plane. Let the desired characteristic equation be

$$T = 1 + t_1\, z^{-1} + t_2\, z^{-2}. \qquad (5.2)$$

To get the desired performance, choose Λ and Γ such that $\Lambda A + B\ \Gamma = T$. Recall that

$$A(z^{-1}) = 1 + a_1\, z^{-1} + a_2\, z^{-2}, \quad B(z^{-1}) = b_0.$$

We can choose $deg\ (\Lambda) = 0$ and $deg\ (\Gamma) = 1$ to get:

$$(\lambda_0 + b_0\, \gamma_0) + (\lambda_0\, a_1 + b_0\, \gamma_1)\, z^{-1} + (\lambda_0\, a_2)\, z^{-2}$$

$$= 1 + t_1\, z^{-1} + t_2\, z^{-2} \qquad (5.3)$$

$$\text{or} \quad \begin{bmatrix} 1 & b_0 & 0 \\ a_1 & 0 & b_0 \\ a_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \gamma_0 \\ \gamma_1 \end{bmatrix} = \begin{bmatrix} 1 \\ t_1 \\ t_2 \end{bmatrix} \qquad (5.4)$$

The solution of Eq. (5.4) would be the desired parameters.

The stationary gain of the closed-loop system can be determined by the Φ in Eq. (4.7). Apply the final-value theorem of z-transform to $y(t)$:

$$\lim_{t\to\infty} y(t) = \lim_{z\to 1} \frac{z-1}{z} \frac{B\ \Phi\ z^{-d}}{\Lambda A + B\ \Gamma} Z(y_r(t)).$$

Suppose $y_r(t)$ is a unit step input, or $Z(y_r(t)) = \dfrac{z}{z-1}$. Then we get $\lim\limits_{t\to\infty} y(t) = \dfrac{B(1)\,\Phi(1)}{\Lambda(1)\,A(1) + B(1)\,\Gamma(1)}$. If one wants to let the output track the set point input $y_r(t)$, he can choose $\Phi(1) = \dfrac{\Lambda(1)\,A(1) + B(1)\,\Gamma(1)}{B(1)}$

The feedforward gain, G_0, determines the feedforward effects on system performance. The value of G_0 should be chosen experimentally using the amplitude of set point changes and the open-loop gain of the system.

6. SIMULATION RESULTS

The STPC algorithm is applied to control a Stanford arm, which is modeled as multiple SISO systems, one for each joint, as follows.

$$A_i(z^{-1})\, q_i(t) = B_i(z^{-1})\, u_i(t-d) + h_i(t) + e_i(t) \qquad (6.1)$$

where $A_i(z^{-1}) = 1 + a_{i1}\, z^{-1} + a_{i2}\, z^{-2}$, $B_i(z^{-1}) = b_{i0}$

$$d = 2, \quad a_{i1} = -2, \quad 1 \le i \le 6$$

$$a_{i2} = \frac{d_{ii} + g_{0ii}}{d_{ii}}, \quad b_{i0} = \frac{1}{d_{ii}}.$$

Refer to [8] for the values of d_{ii} and g_{0ii}, and note that the values are for the case of "no load" in the robot's hand. Inputs are joint torques/forces and outputs are joint angular positions.

The desired end-effector path is a straight line segment, $P_1 \to P_2 \to P_1$, where $P_1 = (-80.95, 10.6, 40.9,)$ and $P_2 = (-31.2, -1.2, 50.0)$. The time of motion is 0.66 second. The initial and final speeds are set to zero and the sampling interval is chosen to be 0.01 sec.

The predictor for the system (6.1) is written as

$$\hat{q}_i(t+k/t) = \frac{B_i}{A_i}\, u_i(t+k-d) + \frac{1}{A_i}\, h_i(t+k) + \frac{F_i}{E_{0i}\ A_i}\, \varepsilon_i(t). \qquad (6.2)$$

When compared with Eq. (3.6), Eq. (6.2) has only one additional term, $h_i(t)$. Similarly, the parameter estimation model can be obtained as:

$$\varepsilon_i(t) = l_i\ \hat{q}_i(t/t-k) + P_i\ \varepsilon_i(t-k) + R_i\ u_i(t-d) + H_i + \eta_i(t). \qquad (6.3)$$

Since $n_a = 2$ and $n_b = 0$, we get

$$l_i = -1 + l_{i,1}\, z^{-1} + \ldots\ldots + l_{i,k+1}\, z^{-(k+1)}, \qquad P_i = p_{i,0} + p_{i,1}\, z^{-1}$$

$$R_i = r_{i,0} + r_{i,1}\, z^{-1} + \ldots\ldots + r_{i,k-1}\, z^{-(k-1)}, \quad H_i = H_{i0} + H_{i1}\, z^{-1}.$$

Then, the vector of estimated parameters becomes

$$\chi_i = [l_{i,1},\ \ldots,\ l_{i,k+1},\ p_{i,0},\ p_{i,1},\ r_{i,0},\ r_{i,1},\ \ldots,\ r_{i,k-1},\ H_i]^T \qquad (6.4)$$

and the vector of measured values is:

$$\phi_i = [\hat{q}_i(t-1/t-1-k),\ \ldots,\ \hat{q}_i(t-1-k/t-2k-1),\ \varepsilon_i(t-k),$$

$$\varepsilon_i(t-1-k),\ u_i(t-d),\ \ldots,\ u_i(t+1-k-d),\ 1]^T. \qquad (6.5)$$

That is, $\varepsilon_i(t) = -\hat{q}_i(t/t-k) + \phi_i\ \chi_i + \eta_i(t)$. (6.6)

The number of estimated parameters is $3 + 2k + 1$.

Since we have already obtained the one and two step-ahead prediction, to reduce the computation time, a three step-ahead predictor can be obtained as:

$$\hat{q}_i(t+3/t) = -a_{i,1}\ \hat{q}_i(t+2/t) - a_{i,2}\ \hat{q}_i(t+1/t) + b_{i,0}\ u_i(t) + h_i. \quad (6.7)$$

The one step ahead predictor is used for the pole placement calculation based on Eq. (5.4). Because $deg\ (E_{0i}) = k - 1$, if $k = 1$, then $R_i = B_i$ and $l_i = -A_i$. Thus, the parameters in Eq. (6.7) become $\chi_i = [-a_{i,1},\ -a_{i,2},\ p_{i,0},\ p_{i,1},\ b_{i,0},\ H_i]^T$. Two and three step-ahead predictors are respectively used to calculate the feedback and feedforward parts of the controller.

Some of simulation results are plotted in Figs. 3 - 5. Figs. 3(a) and 4 are the trajectories of joint 3 and 4 with the desired closed-loop characteristic equation $T = 1 - 1.5\ z^{-1} + 0.69\ z^{-2}$, and Fig. 3(b) is with $T = 1 - 1.5\ z^{-1} + 0.65\ z^{-2}$. The results for the other joints are similar to these and, thus, omitted. From these plots, it is obvious that the feedforward loop improves the behavior of the controller. The square root average errors of each joint are tabulated below.

Joint	No feedforward	With feedforward
1	0.002 rad	0.001 rad
2	0.010 rad	0.005 rad
3	0.361 cm	0.145 cm
4	0.006 rad	0.001 rad
5	0.012 rad	0.008 rad
6	0.005 rad	0.002 rad

Table 1. Square root average position error of each joint.

Fig. 5 shows the convergence of 1 and 2 step-ahead predictors for joint 3. The 1 step-ahead predictor needed about 6 sampling intervals for settling, whereas the 2 step-ahead predictor needed about 9 intervals. In the beginning of operation, we can even open the controller loop and keep the predictors only. After the predictors have converged, the controller loop is closed so that any large fluctuation is avoided during the initial transient period.

The computation time required by the controller is an important consideration for real-time implementation. Table 2 shows the number of multiplications and additions for each joint controller in one sampling interval. When popular 32-bit microprocessors, such as MC68020 and NS 32132, are used, the required computation time for the STPC algorithm is listed in Table 3.
These figures indicate the feasibility of implementing the STPC algorithms in real-time on popular 32-bit microprocessors.

Subroutine	Multiplication	Addition
1 step-ahead predictor & parameter estimator	99	74
2 step-ahead predictor & parameter estimator	156	121
3 step-ahead predictor	3	3
controller with feedforward	10	8
Total	268	203

Table 2. Number of multiplications and additions.

Micro-processor	Multiplication	Addition	Required computing time
MC68020 (32 bit, 12.5 MHZ)	5.68 μs	1.2 μs	10.6 *ms*
NS32132 (32 bit, 10 MHZ)	3.6 μs	0.8 μs	6.76 *ms*

Table 3. Required computation time.

7. CONCLUSION

The self-tuning predicted control is presented as an attractive method for robot trajectory tracking. Using the information of future position, a feedforward loop is shown to improve the system performance. Because of the separation between the predictor and the controller, the system is easily adjustable. The controller does not require any prior knowledge on the manipulator dynamic parameters. The computation time analysis shows that this algorithm can be implemented in real-time by using a popular 32-bit microprocessor.

REFERENCES

[1] A. J. Koivo and T. H. Guo "Control of Robotic Manipulator with Adaptive Controller", *Proc. of IEEE 1981 Conference on Decision and Control*, pp. 271-276.

[2] H. N. Koivo and J. Sorvari "On-line Tuning of a Multivariable PID Controller for Robot Manipulators", *Proc. of IEEE 1985 Conference on Decision and Control*, pp. 1502-1504.

[3] G. G. Leininger, "Self-tuning Adaptive Control of Manipulators", *Advanced Software in Robotics*, pp. 81-96, 1983.

[4] C. H. Chung and G. G. Leininger "Adaptive Self-tuning Control of Manipulators in Task Coordinate System", *Proc. of IEEE 1984 Int'l Conf. on Robotics and Automation*, pp. 546-555.

[5] P. G. Backes, "Joint Self-Tuning with Cartesian Set-points", *Proc. of IEEE 1985 Conference on Decision and Control*, pp. 1416-1421.

[6] A. J. Koivo, R. Lewczyk and T. H. Chiu "Adaptive Path Control of a Manipulator with Visual Information", *Proc. of IEEE 1984 Int'l Conf. on Robotics and Automation*, pp. 556-561.

[7] C. S. G. Lee and B. H. Lee "Resolved Motion Adaptive Control for Mechanical Manipulators", *Proc. of 1984 American Control Conference*, pp. 314-319.

[8] A. K. Bejczy, "Robot Arm Dynamics and Control", *JPL Technical Report* 33-669, Pasadena, CA, Feb. 1974.

[9] D. Keyser and A. R. Van Cauwenberghe "A Self-Tuning Multistep Predictor Application", *Automatica*, vol. 17, no. 1, pp. 167-174, 1981.

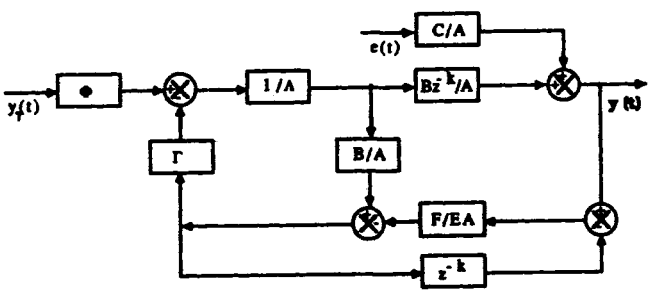

Fig. 1. Self-tuning predicted control system.

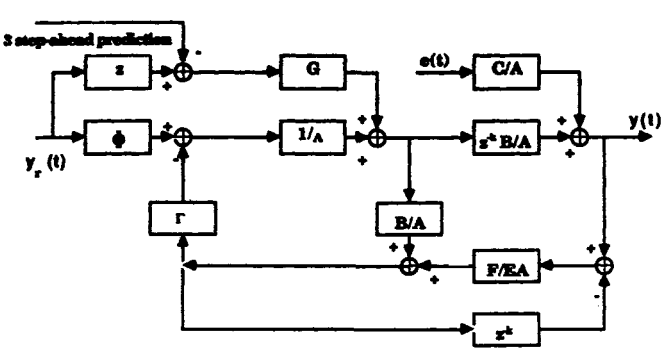

Fig. 2. Self-tuning control system with feedforward.

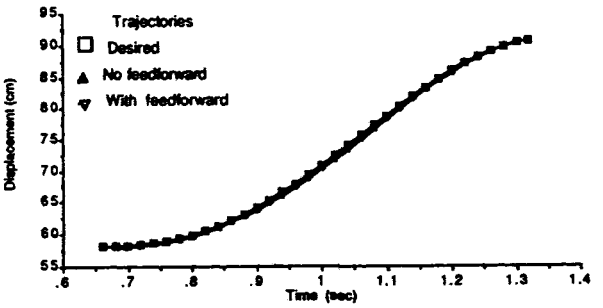

Fig. 3(a). Displacement of joint 3.

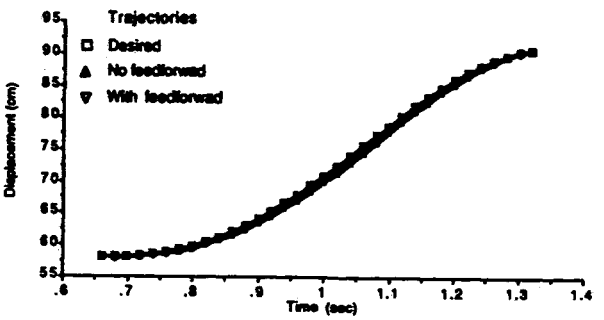

Fig. 3(b). Displacement of joint 3.

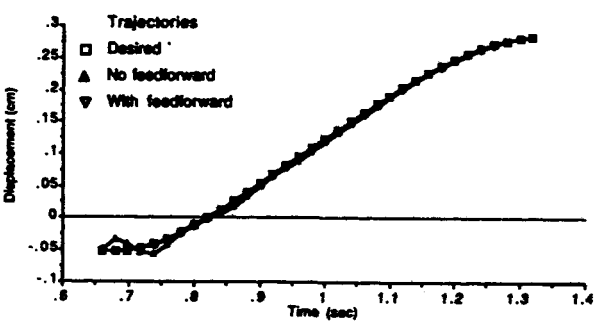

Fig. 4. Displacement of joint 4.

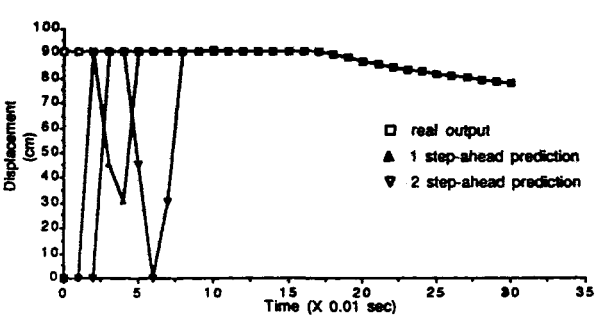

Fig. 5. Outputs of 1 and 2 step-ahead predictors for joint 3.