An Adaptive Model Following Control of Industrial Manipulators

BYUNG KOOK KIM, Member, IEEE

KANG G. SHIN, Senior Member, IEEE
The University of Michigan

Using the adaptive model following control (AMFC) technique, a control system is developed to obtain high performance of an industrial manipulator that has wide variations in its payload and spatial configuration. The main advantage of this technique over other methods is simplicity in computation, which provides the capability of real-time control calculation with today's microprocessors. Moreover, it guarantees automatically the stability of the overall system without requiring any additional stability analysis.

In order to demonstrate the capability of this technique, we have conducted a computer simulation for the Unimation PUMA 600 series manipulator. The simulation results agree well with the expected high performance of the control system: (1) computational requirements are much less than other existing ones, and (2) it provides the capability of adapting to large variations in payload and spatial configuration. Also presented is a comparison of the amount of its required computation with that of other methods.

Manuscript received December 10, 1982; revised June 7, 1983.

The work reported here is supported in part by the Postdoctoral Program of the Korea Science and Engineering Foundation, Republic of Korea, U.S. AFOSR Contract No. F49620-82-C-0089, and by Robot Systems Division, Center for Robotics and Integrated Manufacturing (CRIM), University of Michigan, Ann Arbor.

Authors' address: Department of Electrical and Computer Engineering, The University of Michigan, Ann Arbor, MI 48109.

0018-9251/83/1100-0805 $1.00 © 1983 IEEE

I. INTRODUCTION

Application of robots to industrial automation has emerged as a major research and development task in both industry and universities. This trend is motivated mainly by a desire for increasing productivity and for improving the worker's life as a whole. However, there are many technical issues and problems to be solved before the desire is met. One of these problems is to design robotic manipulator controllers that can drive manipulators more effectively and efficiently than the contemporary controllers (e.g., in terms of operating speed, use of energy, capability of dealing with various tasks, accuracy, etc.). The contemporary controllers used by robot manufacturers are largely based on simple conventional feedback control techniques that are unable to cope with complex, uncertain dynamics of the manipulator and its interaction with other machinery. On the other hand, most of the proposed advanced controllers are either based on unrealistic approximations or on requirements that demand too much computation to implement on inexpensive computers.

The recent development of small, low-cost, high-performance microprocessors and memories using the fast-developing semiconductor technology has widened the application horizon of real-time computer control such as numerical control, process control, etc. Such microprocessors and memories can also be used for implementing the controllers (in real time) which enable manipulators to perform complex manufacturing tasks automatically with high speed, accuracy and low energy. This paper considers the design of such a controller that (1) can cope with the complex manipulator dynamics, and (2) requires only a simple computation structure for which microcomputers are suitable.

The manipulator control problem can be classified into four different categories depending upon the need of collision avoidance and adherence to a given trajectory [13]. However, it is frequently specified as the problem of finding appropriate forces/torques to drive the associated actuators so that the manipulator may follow a prespecified trajectory with a given velocities schedule (this is called the *inverse problem*).

Presently there are three well-known methods that provide position control of a manipulator and are all kinematically oriented. In resolved motion position control (RMPC) [8], the desired joint positions are determined directly by solving the inverse kinematic equations, and then the related joint velocities and accelerations are calculated from the positions. In resolved motion rate control (RMRC) [11, 12], the linear/angular velocities needed to maintain the desired end-effector position and orientation are mapped into joint rates by the inverse Jacobian matrix, and then the joint accelerations are computed from the velocities. In resolved motion acceleration control (RMAC) [6], corrective Cartesian accelerations are calculated and resolved into joint accelerations, and the joint positions

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

and rates are measured. All of these motion control schemes resolve the control into generalized joint coordinates denoted by q, $\dot{q}$, and $\ddot{q}$. Given q, $\dot{q}$, and $\ddot{q}$, joint torques can be obtained by the Lagrangian formulation [9] or Newton–Euler formulation [7] of manipulator dynamics. Although recently more efficient algorithms for computation of dynamics have been proposed [3, 7], these methods still require a considerable amount of computation during the motion. Furthermore, these methods may include modeling error—discrepancies between the dynamic model and actual manipulator dynamics—thus requiring additional intelligent controllers to compensate for this error. Also, note that even if accurate modeling is possible the dynamic model has to be a function of the task being performed (e.g., payload and positions of the end effector). This fact results in either very complex dynamic models or inaccurate but simplified models; neither of the two is desirable.

One possible technique to reduce the complexity of the motion equations is the parameterization of these equations using a table look-up technique [1, 10]. This method is limited by its inherent finite dimensionality and, in addition, requires a prohibitively large storage space if payload changes are to be included [16].

As a remedy for the problems mentioned above, we have employed an adaptive control technique. This control technique has an important advantage in that (1) it does not require an exact dynamic model of the manipulator, and (2) it requires much less computation than the most efficient existing methods. Note that attempts with a similar notion have been made by others. Koivo used the self-tuning adaptive scheme to control the manipulator [4], which is composed of a system parameter identifier and a controller based on the identified system parameters. No results are given on the effects of payload and no comparison is made with other control methods. Horowitz and Tomizuka [14] employed an *explicit adaptation* in which the manipulator parameters are identified with a double integrator reference model and then used for an adaptive control, compensating for the nonlinearity and decoupling the manipulator dynamics. This method requires computations for both parameter identification and adaptive control law. Also, the gravity effect was omitted in their study. An adaptive control scheme using a reference model was also proposed in [2], where the controller is to drive the manipulator to follow the reference model as closely as possible. It employed the steepest descent method in the adaptation mechanism, and the stability analysis was done separately using a linearized model. However, this design method cannot be applied in general practice since stability analysis for each application is necessary but difficult to perform.

In this paper, we have developed a control scheme to overcome the above problems using the adaptive model following control (AMFC) technique [5] which provides computational speed and automatic system stability regardless of variations in payload and spatial configuration. First, a class of control structures which assure overall system stability is determined. Then, out of these structures a manipulator controller is designed that has high performance and automatic stability yet requires only a small amount of computation. This control scheme has an advantage over [2] in that the stability of the overall system can be assured automatically using the hyperstability and positivity concepts. Thus one can eliminate the need of separate study of the system stability. Also it has an advantage over [14] in that an *implicit adaptation* scheme is employed, thereby not requiring any computation for parameter identification. The reference model in the present scheme is a general second-order linear system which may be more suitable than the pure double integrator for specifying the desired manipulator performance such as rise time, overshoot, damping, etc. Note that the manipulator dynamics are represented as a nonlinear second-order system.

This paper is organized as follows. Section II gives the problem and a solution algorithm for the adaptive model following control, Section III presents computer simulation of the Unimation PUMA 600 manipulator in order to demonstrate the capability of the method developed, Section IV analyzes the amount of require computation and compares with other methods, and Section V contains the conclusion.

II. ADAPTIVE MODEL FOLLOWING CONTROL

The first step in the development of any manipulator control system is to derive an analytical model of manipulator dynamics. Using the Lagrangian mechanics [9], one can derive explicit dynamic equations which represent generalized forces/torques in terms of the joint positions, velocities, and accelerations:

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u} \tag{1}$$

Where $\boldsymbol{u}$ is an $n \times 1$ generalized force/torque vector and $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ are vectors of generalized coordinates, velocities, and accelerations, respectively. $\boldsymbol{D}(\boldsymbol{q})$ is an $n \times n$ inertial matrix, $\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is an $n \times 1$ Coriolis and centrifugal force vector, $\boldsymbol{g}(\boldsymbol{q})$ is an $n \times 1$ gravitational loading vector, and n is the number of joints in the manipulator. The inertia, the gravity loading, and the Coriolis and centrifugal terms depend on the position of each joint as well as on the mass, the first moment, and the inertia of each link. Also note that these terms are functions of manipulator payload (i.e., tool and parts). The inertia and the gravity terms are dominant at slower operating speeds. At faster speeds the Coriolis and centrifugal effects become significant [7]. In general, the design of manipulator controllers has been hampered by the inherent nonlinearity and the joint couplings as evidenced in (1).

The dynamics equation (1) can be converted into a state-variable representation with $2n$-dimensional state vector $\boldsymbol{y} = [\boldsymbol{y}_p^{\mathrm{T}} \boldsymbol{y}_v^{\mathrm{T}}]^{\mathrm{T}} = [\boldsymbol{q}^{\mathrm{T}} \dot{\boldsymbol{q}}^{\mathrm{T}}]^{\mathrm{T}}$ (T denotes transpose) as follows:

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

$$\dot{\boldsymbol{y}} = \boldsymbol{A}_p\boldsymbol{y} + \boldsymbol{B}_p(\boldsymbol{y})\boldsymbol{u} + \boldsymbol{C}_p(\boldsymbol{y}) \tag{2}$$

where

$$\boldsymbol{A}_p = \begin{bmatrix} 0 & \boldsymbol{I} \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{B}_p(\boldsymbol{y}) = \begin{bmatrix} 0 \\ \boldsymbol{D}^{-1}(\boldsymbol{y}) \end{bmatrix}$$

$$\boldsymbol{C}_p(\boldsymbol{y}) = \begin{bmatrix} 0 \\ -\boldsymbol{D}^{-1}(\boldsymbol{y})[\boldsymbol{g}(\boldsymbol{y}) + \boldsymbol{h}(\boldsymbol{y})] \end{bmatrix}.$$

These equations show that the manipulator dynamics are highly nonlinear coupled functions of positions and velocities of the manipulator joints. The nonlinear coupled characteristics of the manipulator dynamics cause the design of any controller to be complex and computationally demanding, thereby making its real-time implementation extremely difficult if not impossible. To overcome this difficulty without appealing to complex controllers, we have adopted the adaptive model following control (AMFC) method [5] in this paper. Considering the manipulator dynamics (1), the reference model is chosen to be a set of n uncoupled linear time-invariant second-order systems (i.e., one for each of n joints), which are described as follows:

$$\ddot{x}_i + 2\zeta_i\omega_i\dot{x}_i + \omega_i^2 x_i = \omega_i^2 r_i \qquad \text{for } i = 1,2, \ldots, n \tag{3}$$

where for each joint i, x_i is the position produced by the reference model (i.e., the desired position), r_i is the command input to the reference model, ζ_i is the damping ratio, and ω_i is the natural frequency.

Note that the order and the structure of the reference model are the same as those of the manipulator dynamics in (1). The reference model can be regarded as a means of supplying the manipulator positions and velocities with desired characteristics of rise time, overshoot, and damping. Then the AMFC problem is to design an adaptive controller that drives the manipulator to follow the desired positions and velocities generated by the reference model as closely as possible. In such a case, the reference model can be considered as a trajectory planner that provides the manipulator controller with information on a preplanned path and its timing. Hence we can apply command inputs r_i to the reference model so that the reference model generates the desired positions and velocities. For example, we can simply apply a (soft) *saturated ramp input* $r(t)$[1] to generate a trajectory for the manipulator to move from one point to another so that the velocity profile of the reference model has segments of acceleration, cruise, and deceleration (similar to a trapezoidal velocity profile) by selecting adequate values of ζ_i, ω_i (see Fig. 3(a) for this). In the foregoing sense the model reference adaptive controller can be regarded as similar to most conventional controllers solving the inverse problem. It is, however, important to observe that the model reference adaptive controller requires neither accurate modeling of manipulator dynamics nor solving the dynamics equations (which is known to be difficult and computationally demanding). This is the novelty of the present method.

The reference model equations can be converted into a state variable form as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_m\boldsymbol{x} + \boldsymbol{B}_m\boldsymbol{r} \tag{4}$$

where

$$x_{n+i} = \dot{x}_i \qquad \text{for } i = 1,2, \ldots, n$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{2n} \end{bmatrix}$$

$$\boldsymbol{A}_m = \begin{bmatrix} 0 & \boldsymbol{I} \\ -\omega^2\boldsymbol{I} & -2\zeta\omega\boldsymbol{I} \end{bmatrix}$$

$$\boldsymbol{B}_m = \begin{bmatrix} 0 \\ \omega^2\boldsymbol{I} \end{bmatrix}.$$

Notice that letter $\boldsymbol{I}$ in the above expressions is used to mean the following: $\boldsymbol{I}$ is an $n \times n$ identity matrix

$$\omega^2\boldsymbol{I} = \text{diag}[\omega_1^2 \ \ldots \ \omega_n^2]$$

$$\zeta\omega\boldsymbol{I} = \text{diag}[\zeta_1\omega_1 \ \ldots \ \zeta_n\omega_n].$$

The AMFC problem is now to design an adaptive controller that guides the manipulator to follow the reference model as closely as possible. There are three basic approaches to the AMFC design problem. The first approach uses local parametric optimization to derive an adaptive law but provides no guarantee on the stability of the resulting adaptive system, thus requiring a separate stability analysis which is often very difficult if not impossible. The work reported in [2] is an example of this approach. The second approach is based on the use of Lyapunov functions in order to design stable adaptive systems. However, it is in practice impossible to find a Lyapunov function with the best performance. In the third approach one can obtain a large family of stable adaptive systems using the hyperstability and positivity concepts [5]. Then an adaptive controller with the best performance can be selected from them. On the other hand, an adaptation mechanism for the AMFC problem may require an explicit model (this is called an *explicit adaptation*) or may not require such a model at all (this method is called an *implicit adaptation*), depending upon the design objectives. The former requires an explicit model to be identified as a generating part of the control law. This approach normally requires additional time for the model identification and was adopted in [14]. The latter specifies the design objective *only* for the computation of the control law without requiring any explicit model.

[1] A saturated ramp input $r(t)$ is defined as

$$r(t) = \begin{cases} ct/T & \text{for } 0 \le t < T \\ c & \text{for } t \ge T \end{cases}$$

where c is the distance to move, T is the time interval of interest, and t stands for actual time.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

Since our present main objective lies in the design of a manipulator controller which can (1) compensate for the nonlinearity, joint couplings, and uncertainty in the dynamics, and (2) be implemented in real time on inexpensive microcomputers, we have chosen here the third approach with implicit adaptation.

The generalized state error $\boldsymbol{e}$ is defined by

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{y}. \tag{5}$$

The adaptive model following control signal $\boldsymbol{u}$ is

$$\boldsymbol{u} = [-\boldsymbol{K}_y + \Delta\boldsymbol{K}_y(\boldsymbol{e},t)]\boldsymbol{y} + [\boldsymbol{K}_r + \Delta\boldsymbol{K}_r(\boldsymbol{e},t)]\boldsymbol{r} \tag{6}$$

where $\boldsymbol{K}_y$ and $\boldsymbol{K}_r$ represent linear gains, and $\Delta\boldsymbol{K}_y$ and $\Delta\boldsymbol{K}_r$ represent adaptive gains. Note that the terms with subscript y constitute feedback gains for position and velocities of the manipulator, and those with subscript r feedforward gains for the command input. The generalized error is used here as a driving force to the adaptation mechanism.

The objective of the adaptation mechanism for generating two time-varying matrices $\Delta\boldsymbol{K}_y(\boldsymbol{e},t)$ and $\Delta\boldsymbol{K}_r(\boldsymbol{e},t)$ is to assure that the generalized state error $\boldsymbol{e}$ goes to zero as $t \to \infty$ under certain conditions. The adaptation mechanism assuring the above convergence can be derived by using the approach in [5] as follows:

$$\boldsymbol{v} = \boldsymbol{H}\boldsymbol{e} \tag{7}$$

$$\Delta\boldsymbol{K}_y(\boldsymbol{e},t) = \Delta\boldsymbol{K}_y(\boldsymbol{v},t) = \int_0^t \varphi_1(\boldsymbol{v},t,\tau)\,d\tau + \varphi_2(\boldsymbol{v},t) + \Delta\boldsymbol{K}_y(0) \tag{8}$$

$$\Delta\boldsymbol{K}_r(\boldsymbol{e},t) = \Delta\boldsymbol{K}_r(\boldsymbol{v},t) = \int_0^t \psi_1(\boldsymbol{v},t,\tau)\,d\tau + \psi_2(\boldsymbol{v},t) + \Delta\boldsymbol{K}_r(0) \tag{9}$$

where

$$\Delta\boldsymbol{K}_y(0) = \int_{-\infty}^0 \varphi_1(\boldsymbol{v},t,\tau)\,d\tau$$

$$\Delta\boldsymbol{K}_r(0) = \int_{-\infty}^0 \psi_1(\boldsymbol{v},t,\tau)\,d\tau.$$

In these equations, the integrand matrices φ_1 and ψ_1 represent a nonlinear time-varying integral relation between the two matrices $\Delta\boldsymbol{K}_y(\boldsymbol{v},t)$ and $\Delta\boldsymbol{K}_r(\boldsymbol{v},t)$ and the values of $\boldsymbol{v}(\tau)$ for $0 \le \tau \le t$, which reflects the memory of the adaptation mechanism. This integral adaptation assures zero steady-state errors of the manipulator position and velocity. Matrices φ_2 and ψ_2 denote a nonlinear time-varying proportional relation between the two matrices $\Delta\boldsymbol{K}_y(\boldsymbol{v},t)$ and $\Delta\boldsymbol{K}_r(\boldsymbol{v},t)$ and the values of $\boldsymbol{v}(t)$, which are transient terms vanishing at the end of the adaptation process. (That is, the adaptation process ends when $\boldsymbol{v} = 0$ and therefore $\varphi_2(0,t) = 0$ and $\psi_2(0,t) = 0$ for all t.) Also note that $\varphi_1(0,t,\tau) = \psi_1(0,t,\tau) = 0$ for all t and τ.

The above adaptation law can be actually designed by

$$\Delta\boldsymbol{K}_y(\boldsymbol{v},t) = \int_0^t \boldsymbol{F}\boldsymbol{v}(\boldsymbol{G}\boldsymbol{y})^{\mathrm{T}}\,dt + \boldsymbol{F}'\boldsymbol{v}(\boldsymbol{G}\boldsymbol{y})^{\mathrm{T}} + \Delta\boldsymbol{K}_y(0) \tag{10}$$

$$\Delta\boldsymbol{K}_r(\boldsymbol{v},t) = \int_0^t \boldsymbol{M}\boldsymbol{v}(\boldsymbol{N}\boldsymbol{r})^{\mathrm{T}}\,dt + \boldsymbol{M}'\boldsymbol{v}(\boldsymbol{N}\boldsymbol{R})^{\mathrm{T}} + \Delta\boldsymbol{K}_r(0) \tag{11}$$

where $\boldsymbol{F}$, $\boldsymbol{M}$, $\boldsymbol{G}$, and $\boldsymbol{N}$ are positive definite matrices, and $\boldsymbol{F}'$ and $\boldsymbol{M}'$ are positive semidefinite matrices with appropriate dimensions. The stability of this system can be assured by choosing a matrix $\boldsymbol{H}$ that satisfies

$$\boldsymbol{H} = \boldsymbol{B}_m^{\mathrm{T}}\boldsymbol{P} \tag{12}$$

where $\boldsymbol{P}$ is a positive definite matrix satisfying

$$\boldsymbol{A}_m^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A}_m = -\boldsymbol{Q} \tag{13}$$

for a given positive definite matrix $\boldsymbol{Q}$. Note that the above choice of $\boldsymbol{H}$ guarantees the hyperstability and positivity of the system (see [5] for a detailed discussion and proof).

A block diagram of the adaptive model following control system of the manipulator discussed thus far is given in Fig. 1 with $\boldsymbol{K}_y = [\boldsymbol{K}_p, \boldsymbol{K}_v]$. The values of the

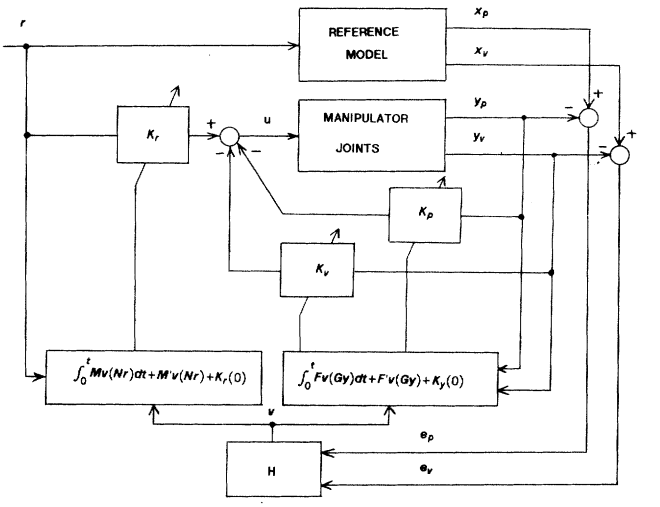


Fig. 1. Adaptive model following control of a manipulator.

linear model following gains $\boldsymbol{K}_y$, $\boldsymbol{K}_r$ are computed from the linear model following criterion for a typical manipulator position:

$$\boldsymbol{K}_y = -\boldsymbol{B}_p^{+}(\boldsymbol{A}_m - \boldsymbol{A}_p)$$

$$\boldsymbol{K}_r = \boldsymbol{B}_p^{+}\boldsymbol{B}_m$$

where $\boldsymbol{B}_p^{+} = (\boldsymbol{B}_p^{\mathrm{T}}\boldsymbol{B}_p)^{-1}\boldsymbol{B}_p^{\mathrm{T}}$ is the left Penrose pseudo-inverse of $\boldsymbol{B}_p$ [5]. By choosing appropriate values of the design variables $\boldsymbol{F}$, $\boldsymbol{F}'$, $\boldsymbol{M}$, $\boldsymbol{M}'$, $\boldsymbol{G}$, and $\boldsymbol{N}$, one can form a separate decoupled adaptation mechanism for each joint. A high ratio of the proportional gain to the integral gain leads to a high speed in reducing the error $\boldsymbol{e}$, which, however, requires larger control energy. The larger the proportional and integral gains the faster adaptation becomes. However, the increase of these gains is limited by the saturation on the control input and also by the

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

imperfect characterization of the manipulator dynamics.[2] It is therefore necessary to have a tune-up stage for choosing the design (gain) parameters in the adaptation mechanism.

More significantly, each separate adaptation mechanism automatically takes care of the interjoint coupling effects on the corresponding joint. The coupling effects are implicitly compensated in the gain adaptation (our simulation results in the next section demonstrate this capability) by using the actual manipulator position and velocity in $\boldsymbol{e}$. This is a considerable advantage of the present method, particularly because any direct handling of the couplings results in a very complex controller which is too computationally demanding to implement in real time on inexpensive microprocessors.

III. SIMULATION RESULTS

As a numerical example, a PUMA 600 series manipulator is simulated on DEC VAX-11/780 to demonstrate the capability of the AMFC method discussed thus far. The PUMA manipulator is manufactured by Unimation and consists of six rotational joints, each of which is driven by a dc servomotor.

Using the Lagrangian formulation the dynamics of the first three joints of the manipulator (i.e., (1)) can be described by

$$\sum_{j=1}^{n} d_{ij}(\boldsymbol{q})\ddot{\boldsymbol{q}}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{q})\dot{\boldsymbol{q}}_j\dot{\boldsymbol{q}}_k + g_i(\boldsymbol{q}) = u_i, \qquad i = 1,2, \ldots, n \quad (14)$$

where $d_{ij}(\boldsymbol{q})$ is the inertial term acting on joint i due to joint j, $h_{ijk}(\boldsymbol{q})\dot{\boldsymbol{q}}_j\dot{\boldsymbol{q}}_k$ represents the Coriolis or centrifugal force on joint i caused by joint j and k, and $g_i(\boldsymbol{q})$ is the gravitational force acting on joint i. Some of the typical terms have the following form:

$$\begin{aligned} d_{11}(\boldsymbol{q}) &= J_{111} + J_{133} + J_{233} + 2J_{234}d_2 + J_{244}d_2^2 \\ &\quad + J_{322} + J_{333} + J_{334}d_2^2 \\ &\quad + (J_{211} + 2J_{214}a_2 + J_{244}a_2^2 + J_{344}a_2^2)C_2^2 \\ &\quad + J_{222}S_2^2 + (J_{311} - J_{333})C_{23}^2 \\ &\quad + 2J_{334}a_2C_2S_{23} \\ h_{211} &= (J_{211} - J_{222} + 2J_{214}a_2 + J_{244}S_2 + J_{344}a_2^2)S_2C_2 \\ &\quad + (J_{311} - J_{333})C_{23}S_{23} \\ &\quad - J_{334}a_2(C_2C_{23} - S_2S_{23}) \\ g_2 &= -(a_2m_3 + a_2m_2 + m_2\bar{x}_2)C_2g - m_3\bar{z}_3S_{23}g \end{aligned}$$

[2]Note that, strictly speaking, the AMFC theory is valid only for time-invariant systems, although it can be extended for practical purpose to time-varying systems without theoretical proof (e.g., slowly time-varying systems). So, time-varying systems like the present one are characterized by some (approximate) models for which the AMFC method works.

where J_{ijk} is the (j,k)th element of the 4×4 inertia tensor J_i for the ith joint; $\bar{x}_i$, $\bar{y}_i$, $\bar{z}_i$ represent the coordinate of the center of mass of joint i with respect to its coordinate frame. d_2, a_2 are lengths related to the arm coordinate frame, and for $i,j = 1,2, \ldots, n$, $C_i = \cos(q_i)$, $S_i = \sin(q_i)$, $C_{ij} = \cos(q_i + q_j)$, $S_{ij} = \sin(q_i + q_j)$. The remaining three joints are not considered here for computational convenience. Although these three joints may affect the first three joints through the joint couplings, their main functions are to orient the end effector and, therefore, do not play as an important role as the first three joints in the area of our present concern (i.e., inertia changes due to its payload and spatial configuration). Also, note that the simulator is not a part of the controller and will be replaced by the manipulator in case of actual implementation.

The simulator computes the current position and velocity of each joint with nonlinear coupled dynamics of the manipulator given by (14), and these values are supplied to the adaptive controller. The adaptation mechanism generates adaptive feedback and feedforward gains which are then used by the controller. (That is, it is an implicit adaptation since the system parameters are not directly computed or used.) This procedure is executed once every sampling interval.

Numerical values used in this simulation are as follows.

(1) The mass, center of mass, and inertia of each joint of the PUMA manipulator are set as in Table I.

(2) $\boldsymbol{Q}$ is a diagonal matrix with $diag\ \boldsymbol{Q} = [10\ \ 10\ \ 10\ \ 1\ \ 1\ \ 1]$. This value of $\boldsymbol{Q}$ will result in more weight on the position following than the velocity following. Namely, accuracy in positioning is more emphasized than that in velocity. Depending upon the application tasks one can adjust relative weightings between the accuracies in position and velocity.

(3) Parameters in the reference model are $\zeta_i = 1.0$, and $\omega_i = 3.0$ (5.0 for the saturated ramp response) for $i = 1, 2, \ldots, 6$ to obtain critically damped responses for each joint.

Note that the choice of these parameters is somewhat arbitrary for the sake of numerical demonstration and any of such choices does not change the basic performance of our manipulator control method.

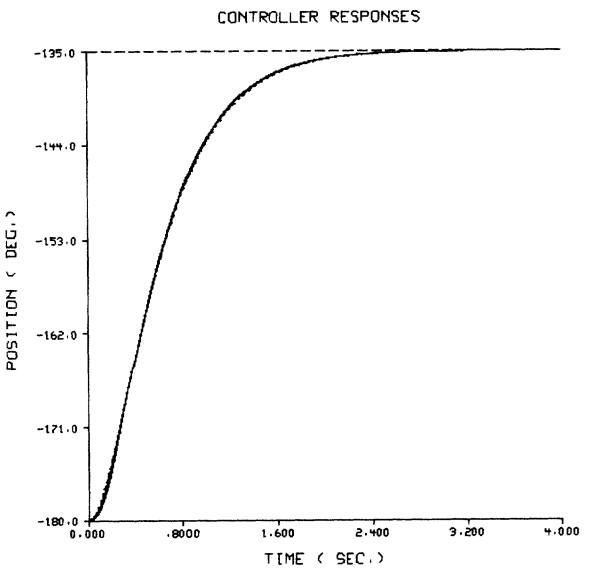
Responses for the Step Input. The first simulation shows the results when a step command input $\boldsymbol{r}$ is applied from an initial joint angle ($-60°$, $-180°$, $90°$) (i.e., a horizontally flat position) to a final joint angles ($-120°$, $-135°$, $135°$) (i.e., a position with the lower arm 45° upward and the upper arm vertically standing) to demonstrate the performance of the proposed adaptive control scheme.

Fig. 2(a) shows the response of the proposed adaptive controller. It shows the response of joint 2 and other joints have shown similar characteristics and hence are

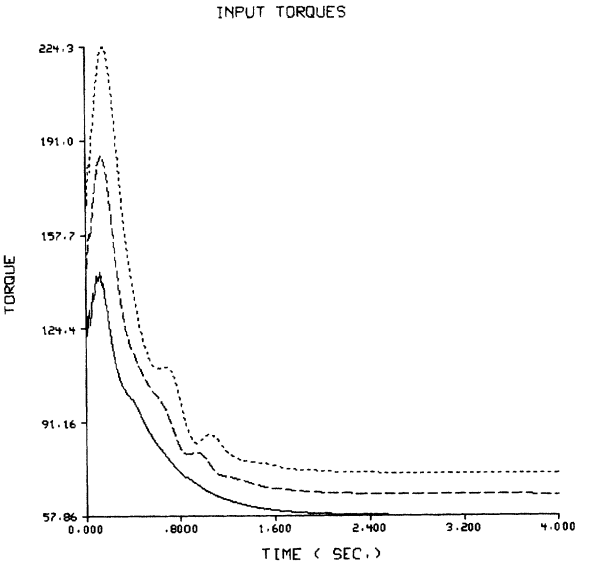
Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

TABLE I
Mass, First Moments, and Inertias of the First Three Joints for the PUMA 600 Manipulator

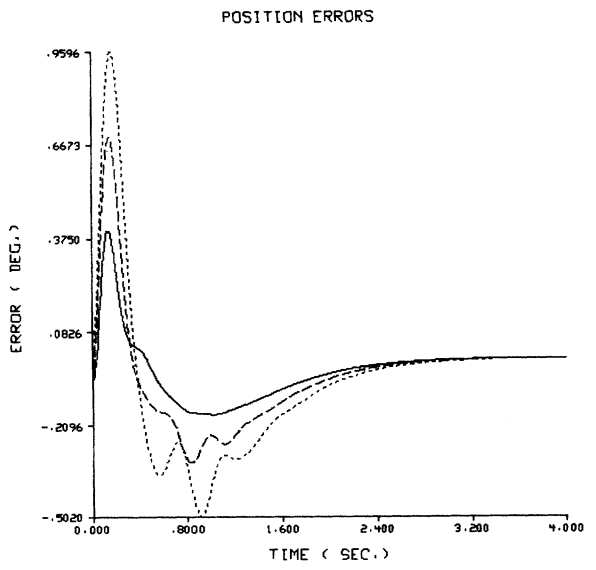
	Mass	Center of Mass			Inertia		
Link	M (kg)	$\bar{x}$ (m)	$\bar{y}$ (m)	$\bar{z}$ (m)	I_x (kg·m^2)	I_y (kg·m^2)	I_z (kg·m^2)
1	2.27	0.0	0.0	0.075	0.00376	0.00376	0.0169
2	15.91	−0.216	0.0	0.0	0.9897	0.1237	0.1237
3	11.36	0.0	0.0	0.216	0.0074	0.0074	0.7067



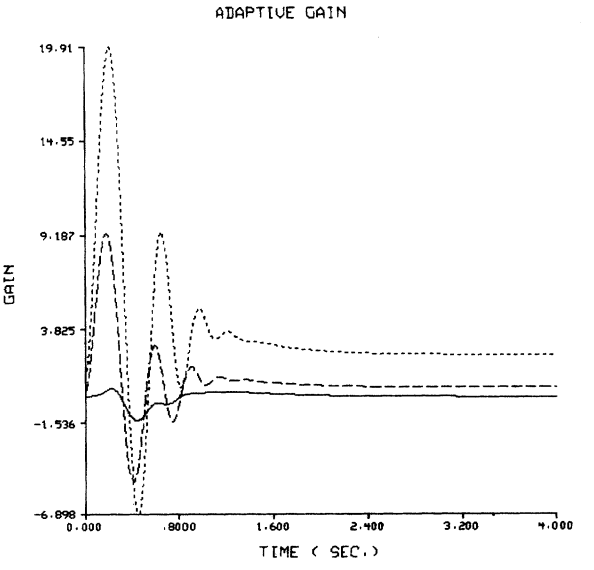
(a)



(c)



(b)



(d)

Fig. 2. (a) Controller responses of joint 2 with step command. – – – – : command input; - - - - - - - - : response of the reference model ———— : response of the manipulator. (b) Position errors on magnified scale with step command. ———— : no load; – – – – : load of 2.5 kg; - - - - - - - : load of 5 kg. (c) Control inputs for joint 2 with step command. ———— : no load; – – – – : load of 2.5 kg; - - - - : load of 5 kg. (d) Adaptive feed-forward gain of joint 3 with step command. ———— :no load; – – – – : load of 2.5 kg; - - - - : load of 5 kg.

omitted here. The gross dotted line represents the step command input that describes the position and timing information for the reference model to follow, and the fine dotted line depicts the response of the reference model. We also plotted the manipulator responses with solid lines for three cases:

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

(1) when there is no load
(2) when there is a maximum load of 2.5 kg (about 5 lb, given by the manufacturer[3])
(3) when a load of 5 kg is considered.

The result shows that the three solid lines are virtually indistinguishable from the fine dotted line, demonstrating that the manipulator closely follows the behavior of the reference model regardless of the load it is carrying. Case (3) is included to show the adaptive capability of our method, although it may bear little practical importance. The errors between the model and the manipulator responses are magnified for the above three load conditions in Fig. 2(b) (again, only the errors for joint 2 are presented), showing that the errors are always less than 1 degree regardless of the load condition. The maximum position tracking errors for each joint and each load condition are summarized in Table II. Fig. 2(c)

TABLE II
Maximum Position Tracking Errors with a Step Command

Load (kg)	Joint 1 (deg)	Joint 2 (deg)	Joint 3 (deg)
0	0.123	0.399	0.223
2.5	0.712	0.692	0.458
5	1.407	0.959	0.793

depicts the corresponding control inputs for joint 2. Fig. 2(d) shows a typical adaptive gain of $\Delta \boldsymbol{K}_{r3}$ (the feedforward gain of the joint 3), indicating the adaptation capability of the proposed controller. (The choice of joint 3 for this is arbitrary.)

Responses for the Saturated Ramp Input. In this simulation, the first three joints of the manipulator are commanded with a saturated ramp input from initial joint angles ($-60°$, $-180°$, $90°$) to final joint angles ($-120°$, $-135°$, $135°$), with time interval T of 2 seconds as in Fig. 3(a). This command input $\boldsymbol{r}$ drives the reference model to move to the target point within about 3 s.

Fig. 3(a) shows the response of the proposed adaptive controller for joint 2. Again, this result exhibits that the manipulator closely follows the behavior of the reference model regardless of its load condition. Fig. 3(b) shows the position error in a magnified scale, showing the errors always less than 1 degree for this joint irrespective of its load condition. The maximum position tracking errors for each joint and each load condition are summarized also in Table III. The maximum error in positioning is always less than 1.5 degrees for all joints and load conditions. Fig. 3(c) depicts the corresponding control inputs for joint 2. Fig. 3(d) shows an adaptive gain of $\Delta \boldsymbol{K}_{r3}$.

As a whole this simulation exhibits that the proposed control method has a potential for high performance with a very simple structure.

[3]As one referee pointed out, the allowable maximum load for a simulated manipulator may be much higher than this due to the inaccuracy in the simulated frictions.

TABLE III
Maximum Position Tracking Errors with a Ramp Command

Load (kg)	Joint 1 (deg)	Joint 2 (deg)	Joint 3 (deg)
0	0.089	0.303	0.244
2.5	0.259	0.511	0.467
5	0.592	0.734	0.731

IV. COMPUTATIONAL COMPLEXITY

The Lagrangian formulation or the Newton–Euler formulation may be used to compute the applied torque/force to control the manipulator. With both the recursive Lagrangian formulation and the recursive Newtonian formulation of the dynamics, the number of additions and multiplications varies linearly with the number of joints, whereas the Lagrangian formulation requires the computation of order of n^4 [3]. However, they require computations of order of 1000 for $n = 6$, which is still a challenging task for today's microprocessors to perform in real time.

For the explicit adaptation scheme proposed by Horowitz and Tomizuka [14], the major computational burdens were on the parameter identification (i.e., $N = N_a + N_m$) and also on the matrix computation for the inertia and Coriolis terms which results in $O(n^3)$ dependencies.

For the adaptive model following control technique the number of computations also varies linearly with the number of joints. However, the technique does not require the solution of the inverse problem at all but instead calculates adaptively both the feedforward and feedback gains (i.e., implicit adaptation). The AMFC algorithm reported in [2] is based on the steepest descent optimization, and its minimal computational requirement is about twice as high as that of the adaptive algorithm developed in this paper. Note that this comparison states only a lower bound since the method in [2] also has to include additional overhead for the needed separate analysis of stability.

The discrete-time version of [2] is presented in [15], where the simplified version of the adaptive control in [2] is presented. However, there is only one adaptation loop in this scheme, whereas three adaptation loops are employed in our scheme, namely, $\Delta \boldsymbol{K}_r$, $\Delta \boldsymbol{K}_p$, $\Delta \boldsymbol{K}_v$, making the discrete version [15] less capable yet requiring an approximately the same amount of computation as the present method.

For comparison purposes computational requirements for different methods are given in Table IV.

V. CONCLUSION

We have presented the design of a manipulator control system on the basis of the adaptive model following concept. The design has focused on the following three important features. First, unlike most

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

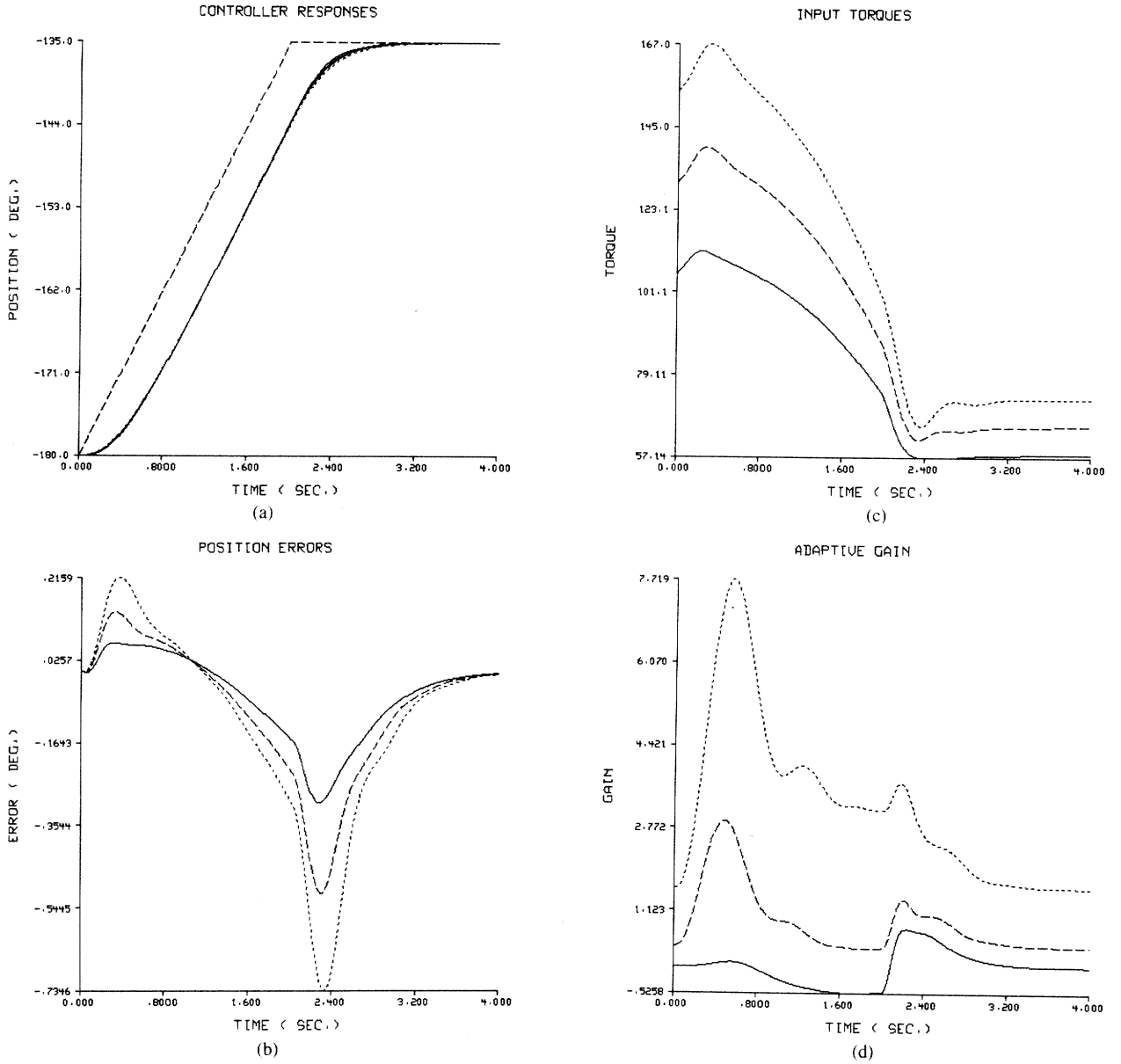


Fig. 3. Controller responses of joint 2 with saturated ramp command input. – – – – : command input; ------ : response of the reference model; ———— : response of the manipulator. (b) Position errors in magnified scale with saturated ramp command. ——— : no load; – – – – : load of 2.5 kg; ---- load of 5 kg. (c) Control inputs for joint 2 with saturated ramp command. ——— : no load; – – – – : load of 2.5 kg; ------ : load of 5 kg. (d) Adaptive feedforward gain of joint 3 with saturated ramp command. ———— : no load; – – – – : load of 2.5 kg; ------ : load of 5 kg.

other existing methods, changes in manipulator payload can be handled effectively without requiring the payload to be specified explicitly in the design of the manipulator control system. In other words, the control system is designed to provide any desired performance (e.g., accuracy in position and velocity) equally well for a wide range of payload. This capability is important to many manufacturing applications since any flexible automation of medium-volume batch manufacturing, a prime target of robotization, requires various parts to be handled, assembled, and inspected. Second, it is conceptually and computationally simpler than any other existing method without loss of performance (actually with better performance). Third, it guarantees the stability of the manipulator controller, leaving control designers free from the stability problem which is often very difficult to an analyze if not impossible. Assurance of the manipulator controller stability cannot be overemphasized, especially in view of the potential danger of human injury and loss of expensive equipment as a result of the controller instability.

Traditionally, robot manufacturers have been mainly concerned with more or less mechanical solutions to the manipulator control problem (e.g., counterbalancing with additional weight attachment). Controllers employed in most industrial robots are rather simple and primitive. For example, controllers in such robots assume constant inertia and zero Coriolis force. Therefore, these

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
Comparison of Computational Requirements

	Multiplications	Additions
Lagrangian	order of n^4	order of n^4
Recursive Lagrangian	$412n-277$	$320n-201$
Newton–Euler	$150n-48$	$131n-48$
Horowitz and Tomizuka	$n^3 + 2n^2 + 10n + N_m$	$n^3 + n^2 + 8n + N_a$
Dubowsky and DesForges	$33n$	$26n$
Kim and Shin	$14n$	$13n$

controllers provide acceptable performance only in a narrow operation range and are incapable of offering the high level of performance required for diversified, modern manufacturing applications. On the other hand, solutions based on mechanics (e.g., Lagrangian mechanics) and kinematics are often too complex to implement with inexpensive microprocessors available in today's market. Considering these facts, the manipulator control method proposed here has high potential use—due to its computational simplicity and high performance capability—in designing intelligent controllers for the growing number of sophisticated industrial manipulators.

REFERENCES

[1] Albus, J.S. (1975)
A new approach to manipulator control: The cerebellar model articulation controller.
ASME J. DSMC, 97 (Sept. 1975), 220–227.

[2] Dubowsky, S., and DesForges, D.T. (1979)
The application of model-references adaptive control to robot manipulators.
ASME J. DSMC, 101 (Sept. 1979), 193–200.

[3] Hollerbach, J.M. (1980)
A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity.
IEEE Trans. Syst., Man, Cybern., SMC-11, 11 (1980), 730–736.

[4] Koivo, A.J., and Guo, T.H. (1981)
Control of robotic manipulator with adaptive controller.
In *Proc. IEEE Conf. Decision and Control,* 1981, pp. 271–276.

[5] Landau, Y.D. (1979)
Adaptive Control: The Model Reference Approach.
New York: Marcel Dekker, Inc., 1979.

[6] Luh, J.Y.S., Walker, M.W., and Paul, R. (1980)
Resolved-acceleration control of mechanical manipulators.
IEEE Trans. Automat. Contr., AC-25, 3 (June 1980), 468–474.

[7] Luh, J.Y.S., Walker, M.W., and Paul, R.P.C. (1980)
On-line computational scheme for mechanical manipulators.
ASME J. DSMC, 102 (June 1980), 69–76.

[8] Paul, R. (1977)
The mathematics of computer controlled manipulator.
In *Proc. Joint Automat. Contr. Conf., 1* (1977), 124–131.

[9] Paul, R. (1981)
Robot Manipulators: Mathematics, Programming, and Control.
Cambridge, Mass.: M.I.T. Press, 1981.

[10] Raibert, M.H. (1977)
Analytical equations vs. table look-up for manipulator: A unifying concept.
In *Proc. IEEE Conf. Decision and Control,* 1977, 576–579.

[11] Whitney, D.E. (1969)
Resolved motion rate control of manipulators and human prosthesis.
IEEE Trans. Man–Machine Systems, MMS-10, 2 (June 1969), 47–53.

[12] Whitney, D.E. (1972)
The mathematics of coordinated control of prosthetic arms and manipulators.
ASME J. DSMC, 94 (Dec. 1972).

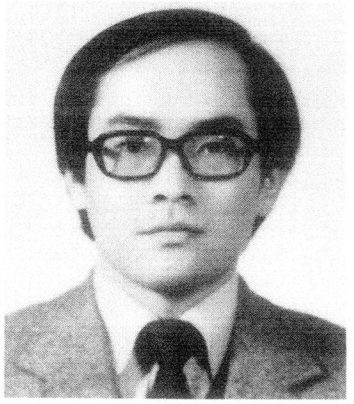
[13] Luh, J.Y.S. (1983)
An anatomy of industrial robot and their controls.
IEEE Trans. Automat. Contr., AC-23, 2 (Feb. 1983), 133–153.

[14] Horowitz, R., and Tomizuka, M. (1980)
An adaptive control scheme for mechanical manipulators—Compensation of nonlinearity and decoupling control.
In *Proc. Winter Annual Meeting of the Dynamics System and Control Division of the ASME* (Chicago, Ill., Nov. 16–21, 1980), pp. 1–12.

[15] Dubowsky, S. (1981)
On the adaptive control of robotic manipulators: The discrete-time case.
In *Proc. of Joint Automatic Control Conf.* (1981).

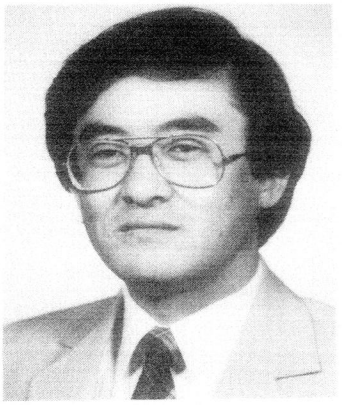
[16] Raibert, M.H. and Horn, B.K.P. (1978)
Manipulator control using the configuration space method.
Ind. Robot, 5 (June 1978), 69–73.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.



Byung Kook Kim (S'75—M'81) was born in Choongju, Korea, on October 5, 1952. He received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1975 and the M.S. and the Ph.D. degrees in electrical science from the Korea Advanced Institute of Science and Technology (KAIST), Seoul, in 1977 and 1981, respectively.

From 1981 to 1982, he was a Research Engineer with Woojin Instrument Co., Seoul. Since 1982 he has been a Visiting Scholar at Rensselaer Polytechnic Institute, Troy, N.Y., and then at the University of Michigan, Ann Arbor, conducting research in robotics. His fields of current interest include computer control systems, time-delay systems, and robotics.



Kang G. Shin (S'75, M'78, SM'82) was born in the province of Choongbuk, Korea, on October 20, 1946. He received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970 and both the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, N.Y., in 1976 and 1978, respectively.

From 1970 to 1972 he served in the Korean Army as an ROTC officer and from 1972 to 1974 he was on the research staff of the Korea Institute of Science and Technology, Seoul, working on the design of VHF/UHF communication systems. From 1974 to 1978 he was a Teaching/Research Assistant and then an Instructor in the School of Electrical Engineering, Cornell University. From 1978 to 1982 he was an Assistant Professor at Rensselaer Polytechnic Institute, Troy, N.Y. He was also a Visiting Scientist at the U.S. Airforce Flight Dynamics Laboratory in 1979 and at Bell Laboratories, Holmdel, N.J. in 1980, where his work was concerned with distributed airborne computing and cache memory architecture, respectively. He also taught short courses for the IBM Computer Science Series in the area of computer architecture. Since 1982, he has been an Assistant Professor in the Department of Electrical and Computer Engineering at The University of Michigan, Ann Arbor. His current teaching and research interests are in the areas of distributed and fault-tolerant computing, computer architecture, and robotics and automation.

Dr. Shin is a member of the Association for Computing Machinery (ACM), Sigma Xi, and Phi Kappa Phi.

Authorized licensed use limited to: University of Michigan Library. Downloaded on January 19,2023 at 17:53:12 UTC from IEEE Xplore. Restrictions apply.