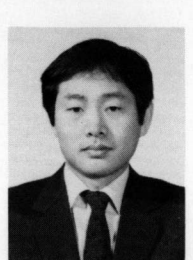
SIMULATION 44:3, 127-136.

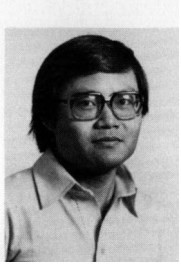
ISSN 0037-5497/85 $2.00+.10
Printed in the United States of America

Key words: robotics, robotic control, robot motion simulation, parameter identification, system identification

Adaptive perturbation control with feedforward compensation for robot manipulators*

C. S. George Lee
Department of Electrical Engineering
and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109





Myung Jin Chung
Korea Advanced Institute
of Science and Technology
Seoul, Korea

C. S. GEORGE LEE is an assistant professor in electrical engineering and computer science with the University of Michigan, Ann Arbor. His current interests include robotics, sensor-based robots, integrated computer manufacturing systems, and pattern analysis.

His previous positions include graduate research assistant in the Advanced Automation Research Laboratory at Purdue University, from 1974 to 1978; graduate teaching assistant in electrical engineering at Purdue from 1976 to 1978; and visiting assistant professor in electrical engineering, also at Purdue.

He received a BSEE (1973) and MSEE (1974) from Washington State University, and a PhD degree (1978) from Purdue. He is a member of Sigma Xi, Tau Beta Pi, the IEEE, the Society of Manufacturing Engineers, and a distinguished visitor in the IEEE Computer Society's Distinguished Visitor Program.

MYUNG JIN CHUNG received his BS degree (1973) in electrical engineering from Seoul National University, Seoul, Korea; his MS degree (1977) in electrical and computer engineering; and his PhD degree (1983) in computer information and control engineering from the University of Michigan, Ann Arbor.

From 1981 to 1983, he was a research assistant in the Center of Robotics and Integrated Manufacturing at the University of Michigan.

Currently, he is an assistant professor and senior research scientist in electrical engineering at the Korea Advanced Institute of Science and Technology, Korea. His current interests are in the development of design techniques for multivariable control systems, robotics and automation.

ABSTRACT

An adaptive perturbation control can track a time-based joint trajectory as closely as possible for all times over a wide range of manipulator motion and payloads. The adaptive control is based on the linearized perturbation equations in the vicinity of a nominal trajectory. The highly coupled nonlinear dynamic equations of a manipulator are expanded in the vicinity of a nominal trajectory to obtain the perturbation equations. The controlled system is characterized by feedforward and feedback components which can be computed separately and simultaneously. Given the joint trajectory set points, the feedforward component computes the corresponding nominal torques from the Newton-Euler equations of motion to compensate for all the interactions between joints. The feedback component, consisting of recursive least square identification and an optimal adaptive self-tuning control algorithm for the linearized system, computes the perturbation torques which reduce the position and velocity errors of the manipulator along the nominal trajectory. Because of the parallel structure, computations of the adaptive control may be implemented in low-cost microprocessors. This adaptive control strategy reduces the manipulator control problem from a nonlinear control to controlling a linear control system about a desired trajectory. Computer simulation results demonstrated its applicability to a three-joint PUMA robot arm.

INTRODUCTION

The purpose of manipulator control is to maintain a prescribed motion for the manipulator along a desired time-based trajectory by applying corrective compensation torques to the actuators to adjust for any deviations of the manipulator from the trajectory. Recently, various control techniques have been developed for this purpose and most of the control schemes emphasize nonlinear compensations of the interaction forces among the various joints and control the manipulator at the joint level[2,7,9,16,22] or the hand level.[14,20,21] These control algorithms are inadequate because they neglect the changes of the load in a task cycle. These changes in the payload of the controlled system are significant enough to render conventional feedback control strategies ineffective. The results are reduced servo response speed and damping of the manipulator, which limits the precision and speed of the end-effector. Any significant performance gain in robot arm control requires the consideration of adaptive control techniques.

*This work was supported in part by the National Science Foundation under Grant ECS-81-06954. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the funding agency.

Recently, various adaptive control algorithms[4,9,10,11] have been proposed. Dubowsky[4] proposed a model referenced adaptive control, which uses a linear second-order time invariant differential equation as the referenced model for each degree of freedom of the robot arm. The manipulator is controlled by adjusting the position and velocity feedback gains to follow the model. A steepest descent method is used to update the feedback gains. Koivo[8] proposed an adaptive self-tuning controller using an autoregressive model to fit the input-output data from the manipulator. Both control algorithms assume that the interaction forces among the joints are negligible.

An adaptive perturbation control strategy can track a desired time-based manipulator trajectory as closely as possible for all times over a wide range of manipulator motion and payloads in joint-variable coordinates. The proposed adaptive control differs from the above adaptive controls by taking all the interactions among the various joints into consideration. The adaptive control is based on the linearized perturbation equations in the vicinity of a nominal trajectory. The highly coupled nonlinear dynamic equations of a manipulator are linearized about the planned manipulator trajectory to obtain the linearized perturbation system. The desired trajectory is specified by an interpolated and smoothed joint trajectory whose angular position, angular velocity, and angular acceleration are known at every sampling instant. The controlled system is characterized by feedforward and feedback components which can be computed separately and simultaneously. Using the Newton-Euler equations of motion as an inverse dynamics of the manipulator, the feedforward component computes the nominal torques which compensate for all the interaction forces among the various joints along the nominal trajectory. The feedback component computes the perturbation torques which reduce the position and velocity errors of the manipulator to zero along the nominal trajectory. An efficient recursive real-time least square identification scheme is used to identify the system parameters in the perturbation equations. An optimal adaptive self-tuning control is then designed to control the linearized perturbation system about the nominal trajectory. The parameters and the feedback gains of the linearized system are updated and adjusted in each sampling period to obtain the necessary control effort. The total torques applied to the joint actuators then consist of the nominal torques computed from the Newton-Euler equations of motion and the perturbation torques computed from the optimal adaptive self-tuning control of the linearized system. A computer simulation study was conducted to evaluate the performance of the proposed adaptive control for a three-joint robot arm.

ROBOT ARM DYNAMICS

A priori information needed for developing the adaptive control strategy is an appropriate dynamic model describing the dynamic behavior of the manipulator. The dynamics of an n-joint manipulator can be derived either by Lagrange-Euler,[2,9,17] Newton-Euler,[1,13,15] recursive Lagrangian,[6] or generalized d'Alembert formulations.[12] The resulting equations of motion are highly nonlinear and consist of inertia loading, coupling reaction forces (Coriolis and centrifugal) among the various joints, and gravity loading effects. In general, the Lagrange-Euler equations of motion of an n-joint manipulator, excluding the actuator dynamics, gear friction and backlash, are a set of second-order coupled nonlinear differential equations and can be expressed in vector matrix notation as in Reference 9,

$$\mathbf{D}(\mathbf{q})\,\ddot{\mathbf{q}}(t) + \mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{c}(\mathbf{q}) = \tau(t) \tag{1}$$

where $\tau(t)$ is an $n\times 1$ applied torque vector to joint actuators, $\boldsymbol{q}$ is the angular positions, $\dot{\boldsymbol{q}}$ is the angular velocities, $\ddot{\boldsymbol{q}}(t)$ is an $n\times 1$ acceleration vector, $\boldsymbol{c}(\boldsymbol{q})$ is an $n\times 1$ gravitational force vector, $\boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is an $n\times 1$ Coriolis and centrifugal force vector and $\mathbf{D}(\mathbf{q})$ is an $n\times n$ acceleration-related matrix.

The above dynamic equations of motion for an n-joint manipulator are highly nonlinear and the interaction torques/forces depend on the manipulator's physical parameters, instantaneous joint configuration, and the load it is carrying. Because of its explicit matrix structure, this formulation is appealing from a control viewpoint in that it gives a set of state equations suitable for control analysis and design as in Eq. 11. This form allows one to design a control law that compensates all the nonlinear effects easily. Computationally, however, these equations of motion are extremely inefficient in computing the joint torques for real-time control as compared with other formulations.[19]

As an alternative to deriving more efficient equations of motion, various forms of Newton-Euler equations of motion have been proposed.[1,13,15] An approach which has the advantage of computing the applied joint torques efficiently per trajectory set point is developed by Luh, Walker, and Paul.[13] The derivation was based mainly on the "moving coordinate systems" and d'Alembert Principle. This formulation yields a set of forward and backward recursive equations which can be applied to the manipulator links sequentially. The forward recursion propagates kinematics information (such as angular velocities, angular accelerations, and linear accelerations of the center of mass) of each link from the base reference frame (inertial frame) to the end-effector. The backward recursion transforms the forces that exert on each link from the end-effector of the manipulator to the base reference frame; the applied joint torques are computed from these forces. Because of the nature of the Newton-Euler formulation and its method of systematically computing the applied joint torques, the computations of the joint torques are much simpler. The computation time of the applied torques is found linearly proportional to the number of joints of the robot arm and independent of the robot arm configurations. This enables the implementation of a simple real-time control algorithm for a robot arm in the joint-variable space. The Luh's Newton-Euler equations of motion for manipulators having all rotary joints are listed below for convenience.

Forward equations: for $i = 1, 2, \ldots, n$

$$\omega_i = \mathbf{R}_i^{i-1}(\omega_{i-1} + \mathbf{z}_o\dot{\theta}_i) \tag{2}$$

$$\dot{\omega}_i = \mathbf{R}_i^{i-1}(\dot{\omega}_{i-1} + \mathbf{z}_o\ddot{\theta}_i + \omega_{i-1}\times \mathbf{z}_o\dot{\theta}_i) \tag{3}$$

$$\dot{\mathbf{v}}_i = \dot{\omega}_i \times \mathbf{p}_{i-1}^i + \omega_i \times (\omega_i \times \mathbf{p}_{i-1}^i) + \mathbf{R}_i^{i-1}\dot{\mathbf{v}}_{i-1} \tag{4}$$

$$\bar{\mathbf{a}}_i = \dot{\omega}_i \times \bar{\mathbf{r}}_i + \omega_i \times (\omega_i \times \bar{\mathbf{r}}_i) + \dot{\mathbf{v}}_i \tag{5}$$

Backward equations: for $i = n, \ldots, 1$

$$\mathbf{f}_i = \mathbf{R}_i^{i+1}\mathbf{f}_{i+1} + m_i\bar{\mathbf{a}}_i \tag{6}$$

$$\mathbf{n}_i = \mathbf{R}_i^{i+1}\mathbf{n}_{i+1} + \mathbf{P}_{i-1}^i \times \mathbf{R}_i^{i+1}\mathbf{f}_{i+1} + (\mathbf{P}_{i-1}^i + \bar{\mathbf{r}}_i) \times m_i\bar{\mathbf{a}}_i + \mathbf{I}_i\dot{\omega}_i + \omega_i \times (\mathbf{I}_i\omega_i) \tag{7}$$

$$\tau_i = (\mathbf{R}_i^{i-1}\mathbf{z}_o)^T\mathbf{n}_i \tag{8}$$

where $\boldsymbol{z}_o = (0,0,1)^T$ and the "usual" initial conditions are $\boldsymbol{\omega}_o = 0, \dot{\mathbf{v}}_o = g\boldsymbol{z}_o$, $\dot{\boldsymbol{\omega}}_o = 0, g = 9.8062 m/s^2$, and $\boldsymbol{f}_{n+1}$ and $\boldsymbol{n}_{n+1}$ are external force and moment exerted on the hand, respectively.

The variables used in the Newton-Euler equations of motion are defined as:

ω_i = the angular velocity of link i with respect to the ith coordinate system

$\dot{\omega}_i$ = the angular acceleration of link i with respect to the ith coordinate system

$\boldsymbol{P}_{i-1}^{i}$ = the origin of the ith coordinate system from the $i-1$th coordinate system with respect to the ith coordinate system

$\bar{\boldsymbol{r}}_i$ = the center of mass of link i with respect to the ith coordinate system

$\dot{\boldsymbol{v}}_i$ = the linear acceleration of link i with respect to the ith coordinate system

$\bar{\boldsymbol{a}}_i$ = the linear acceleration of the center of mass of link i with respect to the ith coordinate system

$\boldsymbol{R}_{i-1}^{i}$ = the rotation matrix that maps position vectors from the ith coordinate system to the $i-1$th coordinate system

$\boldsymbol{I}_i$ = the inertia tensor about center of mass of link i with respect to the ith coordinate system

$\boldsymbol{f}_i$ = the force exerted on link i by link $i-1$ with respect to the ith coordinate system

$\boldsymbol{n}_i$ = the moment exerted on link i by link $i-1$ with respect to the ith coordinate system

τ_i = the torque exerted on link i.

Two robot arm dynamic models have been discussed. The Lagrange-Euler equations provide a set of system equations suitable for control analysis and design. They are very inefficient for computing the joint torques from a given trajectory set point $(\boldsymbol{q}^d(t), \dot{\boldsymbol{q}}^d(t), \ddot{\boldsymbol{q}}^d(t))$ (i.e., solving the inverse dynamics problem). The computation time of the joint torques from Eq. 1 is of order 0 (n^4). For a six-joint PUMA (PUMA is a trademark of Unimation Inc.) manipulator, this amounts to about 5,916 multiplications and 4,840 additions to compute the joint torques per trajectory set point.[19] In direct contrast is the recursive Newton-Euler equations of motion which are highly efficient in computing the joint torques per trajectory set point. The above Newton-Euler equations involve about 678 multiplications and 597 additions for a six-joint PUMA manipulator per trajectory set point.

We shall make use of the characteristics of these dynamic models in developing the proposed adaptive perturbation control strategy. The Newton-Euler equations of motion will be used as an inverse dynamics of the manipulator to generate the open-loop nominal joint torques along a planned trajectory and the Lagrange-Euler equations of motion will be used to derive the linearized perturbation equations of motion for the adaptive system about the planned trajectory.

ADAPTIVE PERTURBATION CONTROL FORMULATION

The proposed adaptive perturbation control is based on the linearized perturbation equations about the referenced trajectory. Defining a $2n$-dimensional state vector for the system as

$$\mathbf{x}^T(t) \triangleq (\mathbf{q}^T(t), \dot{\mathbf{q}}^T(t)) \triangleq (q_1, \ldots, q_n, \dot{q}_1, \ldots, \dot{q}_n) \triangleq (x_1 x_2, \ldots, x_{2n}) \tag{9}$$

and an n-dimensional input vector as

$$\mathbf{u}^T(t) \triangleq (\tau_1, \tau_2, \ldots, \tau_n) \triangleq (u_1, u_2, \ldots, u_n) \tag{10}$$

Equation 1 can be expressed in state space representation as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{11}$$

where $\boldsymbol{f}(\cdot) : R^{2n} \times R^n \rightarrow R^{2n}$ and continuously differentiable, and n is the number of degree of freedom of the manipulator.

Since $\boldsymbol{D}(\boldsymbol{q})$ is always nonsingular, the above equation can be expressed explicitly as

$$\begin{aligned} \dot{x}_i(t) &= f_i(\mathbf{x}) = x_{i+n}(t) \\ \dot{x}_{i+n}(t) &= f_{i+n}(\mathbf{x}) + b_{i+n}(\mathbf{x})\mathbf{u}(t) \; ; \; i = 1, \ldots, n \end{aligned} \tag{12}$$

where $f_{i+n}(\mathbf{x})$ is the ith component of $-\boldsymbol{D}^{-1}(\boldsymbol{q})[\boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{c}(\boldsymbol{q})]$ and $b_{i+n}(\mathbf{x})$ is the ith row of the matrix $\boldsymbol{D}^{-1}(\boldsymbol{q})$.

With this formulation, the control problem is to find a feedback control law $\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t))$ such that the closed loop control system $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{g}(\boldsymbol{x}(t)))$ is asymptotically stable and tracks a desired trajectory as closely as possible over a wide range of payloads for all times.

Perturbation equations of motion

Suppose that the nominal states $\mathbf{x}_n(t)$ of the system (Eq. 11) are known from the planned trajectory, and the corresponding nominal torques $\boldsymbol{u}_n(t)$ are also known from the computations of the joint torques using the Newton-Euler equations of motion. Then both $\mathbf{x}_n(t)$ and $\boldsymbol{u}_n(t)$ satisfy Eq. 11,

$$\dot{\mathbf{x}}_n(t) = \mathbf{f}(\mathbf{x}_n(t), \mathbf{u}_n(t)) \tag{13}$$

Using the Taylor series expansion on Eq. 11 about the nominal trajectory and subtracting Eq. 13 from it, the associated linearized perturbation model for this control system can be obtained:

$$\delta\dot{\mathbf{x}}(t) = \nabla_x \mathbf{f}|_n \delta\mathbf{x}(t) + \nabla_u \mathbf{f}|_n \delta\mathbf{u}(t) + \mathbf{d}(t) \tag{14}$$

$$= \mathbf{A}(t)\, \delta\mathbf{x}(t) + \mathbf{B}(t)\, \delta\mathbf{u}(t) + \mathbf{d}(t) \tag{15}$$

where $\nabla_x \boldsymbol{f}|_n$ and $\nabla_u \boldsymbol{f}|_n$ are the Jacobian matrices of $\mathbf{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ evaluated at $\mathbf{x}_n(t)$ and $\boldsymbol{u}_n(t)$, respectively, $\delta\boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_n(t)$, $\delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_n(t)$, and $\boldsymbol{d}(t)$ is included to account for the bias and modeling errors of the system.

The system parameters, $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$, of Eq. 15 depend on the instantaneous manipulator position and velocity along the nominal trajectory and are thus slowly varying in time. Because of the complexity of the manipulator equations of motion, it is extremely difficult to find the elements of $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ explicitly. However, the design of a feedback control law for the perturbation equations requires that the system parameters of Eq. 15 be known at all times. Thus parameter identification techniques must be used to identify the unknown elements in $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$.

As a result of this formulation, the manipulator control problem is reduced to determining $\delta\boldsymbol{u}(t)$ which drives $\delta\boldsymbol{x}(t)$ to zero at all times along the nominal trajectory. The overall controlled system is thus characterized by feedforward and feedback components. Given the planned trajectory set points $(\boldsymbol{q}^d(t), \dot{\boldsymbol{q}}^d(t), \ddot{\boldsymbol{q}}^d(t))$, the feedforward component computes the corresponding nominal torques $\boldsymbol{u}_n(t)$ from the Newton-Euler equations of motion. The feedback component computes the corresponding perturbation torques $\delta\boldsymbol{u}(t)$ which provide control efforts to compensate for small deviations from the nominal trajectory. The computation of the perturbation torques is based on the optimal control solution derived in the next section. The main advantages of this formulation are twofold. First, it reduces a nonlinear control problem to a linear control problem about a nominal trajectory, and second the computations of the nominal and perturbation torques can be performed separately and simultaneously. Because of the parallel computational structure, the proposed adaptive control can be easily implemented using present day low-cost microprocessors.

Parameter identification of the linearized perturbation system

For implementation on digital computers, Eq. 15 needs to be discretized to obtain appropriate discrete linear equations for parameter identification:

$$\mathbf{x}((k+1)T) = \mathbf{F}(kT)\,\mathbf{x}(kT) + \mathbf{G}(kT)\,\mathbf{u}(kT) + \mathbf{w}(kT)\ ;\ k = 0, 1, \ldots \tag{16}$$

where T is the sampling period, $\boldsymbol{u}(kT)$ is an n-dimensional piecewise constant control input vector of $\boldsymbol{u}(t)$ over the time interval between any two consecutive sampling instants for $kT \leqslant t < (k+1)T$, and $\boldsymbol{x}(kT)$ is a $2n$-dimensional perturbed state vector which is given by:

$$\mathbf{x}(kT) = \Gamma(kT,t_o)\,\mathbf{x}(t_o) + \int_{t_o}^{kT} \Gamma(kT,t)\mathbf{B}(t)\mathbf{u}(t)dt \tag{17}$$

and $\Gamma(kT,t_o)$ is the state-transition matrix of the system in Eq. 15, and $\boldsymbol{w}(kT)$ is a $2n$-dimensional bias vector to account for modeling error,

$$\mathbf{w}(kT) = \int_{kT}^{(k+1)T} \Gamma((k+1)T,t)\mathbf{d}(t)dt \tag{18}$$

$\boldsymbol{F}(kT)$ and $\boldsymbol{G}(kT)$ are, respectively, $2n \times 2n$ and $2n \times n$ matrices and are given by

$$\mathbf{F}(kT) = \Gamma((k+1)T,kT) \tag{19}$$

and

$$\mathbf{G}(kT)\mathbf{u}(kT) = \int_{kT}^{(k+1)T} \Gamma((k+1)T,t)\mathbf{B}(t)\mathbf{u}(t)dt \tag{20}$$

With this model, a total of $6n^2$ parameters in the $\boldsymbol{F}(kT)$ and $\boldsymbol{G}(kT)$ matrices need to be identified. Without confusion, we shall drop the sampling period T from the rest of the equations for clarity and simplicity.

Various identification algorithms, such as the methods of least squares, maximum-likelihood, instrumental variable, cross-correlation, and stochastic approximation, etc., have been applied successfully to the parameter identification problem.[5] Due to its simplicity and ease of applying, a recursive real-time least square parameter identification scheme is selected for identifying the system parameters in $\boldsymbol{F}(k)$ and $\boldsymbol{G}(k)$. In the parameter identification scheme, we make the following assumptions: (1) The parameters of the system are slowly time-varying, but the variation speed is slower than the adaptation speed; (2) Measurement noise is negligible; and (3) The state variables $\boldsymbol{x}(k)$ of Eq. 16 are measurable. The first assumption was justified from our simulation and the same findings were reported in Reference 8. The second assumption was made to simplify the identification scheme and it needs to be verified epxerimentally. The last assumption was based on the fact that the state variable $\boldsymbol{x}(k)$ can be measured from the optical encoder embedded in each joint motor.

In order to apply the recursive least square identification algorithm to Eq. 16, we need to rearrange the system equations in a form that is suitable for parameter identification. Defining and putting the ith row of the unknown parameters of the system at the kth instant of time in a $(3n+1)$-dimensional vector, we have

$$\boldsymbol{\theta}_i^T(k) = (f_{i1}(k), \ldots, f_{ip}(k), g_{i1}(k), \ldots, g_{in}(k), \ldots, w_i(k))\ ;\quad i = 1, 2, \ldots, p \tag{21}$$

or expressed in matrix form as

$$\Theta(k) = \begin{bmatrix} f_{11}(k) & \cdots & f_{p1}(k) \\ \vdots & \vdots & \vdots \\ f_{1p}(k) & \cdots & f_{pp}(k) \\ g_{11}(k) & \cdots & g_{p1}(k) \\ \vdots & \vdots & \vdots \\ g_{1n}(k) & \cdots & g_{pn}(k) \\ w_1(k) & \cdots & w_p(k) \end{bmatrix} = \left[\boldsymbol{\theta}_1(k), \boldsymbol{\theta}_2(k), \ldots, \boldsymbol{\theta}_p(k)\right] \tag{22}$$

where $p = 2n$. Similarly, defining the outputs and inputs of the perturbation system (Eq. 16) at the kth instant of time in a $(3n+1)$-dimensional vector as

$$\mathbf{z}^T(k) = (x_1(k), x_2(k), \ldots, x_p(k), u_1(k), u_2(k), \ldots, u_n(k), 1) \tag{23}$$

and the states at the kth instant of time in a $2n$-dimensional vector as

$$\mathbf{x}^T(k) = (x_1(k), x_2(k), \ldots, x_p(k)) \tag{24}$$

the corresponding system equation expressed in Eq. 16 can be written as

$$x_i(k+1) = \mathbf{z}^T(k)\boldsymbol{\theta}_i(k)\ ;\ i = 1, 2, \ldots, p \tag{25}$$

With this formulation, we would like to identify the parameters in each column of $\Theta(k)$ based on the measurement vector $\boldsymbol{z}(k)$. In order to examine the "goodness" of the least square estimation algorithm, a $2n$-dimensional error vector $\mathbf{e}$, often called residual, is included to account for the modeling error and noise in Eq. 16:

$$e_i(k) = x_i(k+1) - \mathbf{z}^T(k)\hat{\boldsymbol{\theta}}_i(k)\ ;\ i = 1, 2, \ldots, p \tag{26}$$

where hat is used to indicate the estimate of the parameters $\boldsymbol{\theta}_i(k)$.

The basic least square parameter estimation assumes that the unknown parameters are constant values and the solution is based on batch processing N sets of measurement data, which are weighted equally, to estimate the unknown paremeters. Unfortunately this algorithm cannot be applied to the time-varying parameters. Furthermore, the solution requires matrix inversion which is computational intensive. In order to reduce the number of numerical computations and to track the time-varying parameters $\Theta(k)$ at each sampling period, a sequential least square identification scheme which updates the unknown parameters at each sampling period based on the new set of measurements at each sampling interval provides an efficient algorithmic solution to the identification problem. Such a recursive real-time least squares parameter identification algorithm can be found by minimizing an exponentially weighted error criterion which has an effect of placing more weights on the squared errors of the more recent measurements,[5]

$$J_N = \sum_{j=1}^{N} \varrho^{N-j} e_i^2(j) \tag{27}$$

where the error vector is weighted as

$$\mathbf{e}_i^T(N) = (\sqrt{\varrho^{N-1}}e_i(1), \sqrt{\varrho^{N-2}}e_i(2), \ldots, e_i(N)) \tag{28}$$

and $N > 3n+1$ is the number of measurements used to estimate the parameters $\boldsymbol{\theta}_i(N)$. Minimizing the error criterion in Eq. 27 with respect to the unknown parameters vector $\boldsymbol{\theta}_i$ and utilizing the matrix inverse lemma,[5] a recursive real-time least square identification scheme can be obtained for $\boldsymbol{\theta}_i(k)$ after simple algebraic manipulations[5]:

$$\hat{\boldsymbol{\theta}}_i(k+1) = \hat{\boldsymbol{\theta}}_i(k) + \gamma(k)\mathbf{P}(k)\mathbf{z}(k)\,[\mathbf{x}_i(k+1) - \mathbf{z}^T(k)\hat{\boldsymbol{\theta}}_i(k)] \quad (29)$$

$$\mathbf{P}(k+1) = \mathbf{P}(k) - \gamma(k)\mathbf{P}(k)\mathbf{z}(k)\mathbf{z}^T(k)\mathbf{P}(k) \quad (30)$$

$$\gamma(k) = [\mathbf{z}^T(k)\mathbf{P}(k)\mathbf{z}(k) + \varrho]^{-1}\ ;\ 0 < \varrho < 1 \quad (31)$$

where $\boldsymbol{P}(k) = \varrho[\boldsymbol{Z}(k)\boldsymbol{Z}^T(k)]^{-1}$ is a $(3n+1)\times(3n+1)$ symmetric positive definite matrix, and $\boldsymbol{Z}(k) = [\mathbf{z}(1),\mathbf{z}(2), \ldots, \mathbf{z}(k)]$ is the measurement matrix up to the kth sampling instant. If the errors $e_i(k)$ are identically distributed and independent with zero mean and variance σ^2, then $\boldsymbol{P}(k)$ can be interpreted as the covariance matrix of the estimate if ϱ is chosen as σ^2.

The above recursive equations indicate that the estimate of the parameters $\hat{\boldsymbol{\theta}}_i(k+1)$ at the k+1th sampling period is equal to the previous estimate $\hat{\boldsymbol{\theta}}_i(k)$ corrected by the term proportional to $[x_i(k+1) - \boldsymbol{z}^T(k)\hat{\boldsymbol{\theta}}_i(k)]$. The $\boldsymbol{z}^T(k)\hat{\boldsymbol{\theta}}_i(k)$ is the prediction of the value $x_i(k+1)$ based on the estimate of the parameters $\boldsymbol{\theta}_i(k)$ and the measurement vector $\boldsymbol{z}(k)$. The components of the vector $\gamma(k)\boldsymbol{P}(k)\boldsymbol{z}(k)$ are weighting factors which indicate how the corrections and the previous estimate should be weighted to obtain the new estimate $\hat{\boldsymbol{\theta}}_i(k+1)$. The parameter ϱ is a weighting factor and is commonly used for tracking slowly time-varying parameters by exponentially forgetting the "aged" measurements. If $\varrho << 1$, a large weighting factor is placed on the more recent sampled data by rapidly weighing out previous samples. If $\varrho << 1$, accuracy in tracking the time-varying parameters will be lost due to the truncation of measured data sequence. We can compromise between fast adaptation capabilities and loss of accuracy in parameter identification by adjusting the weighting factor ϱ. In most applications for tracking slowly time-varying prameters, ϱ is usually chosen to be $0.93 \leqslant \varrho < 1.0$. The estimate of the parameters will be used to determine the optimal control solution for the perturbation system in the next section.

Finally, the above identification scheme (Eqs. 29-31) can be started by choosing the initial values of $\boldsymbol{P}(0)$ to be

$$\mathbf{P}(0) = \alpha\, \mathbf{I}_{3n+1} \quad (32)$$

where α is a very large positive scalar and $\boldsymbol{I}_{3n+1}$ is a $(3n+1)\times(3n+1)$ identity matrix. The initial estimate of the unknown parameters $\boldsymbol{F}(k)$ and $\boldsymbol{G}(k)$ can be approximated by the following equations:

$$\mathbf{F}(0) \approx \mathbf{I}_{2n} + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]T + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]^2\frac{T^2}{2} \quad (33a)$$

$$\mathbf{G}(0) \approx \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]T + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]T^2 + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]^2\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]\frac{T^3}{2} \quad (33b)$$

where T is the sampling period.

Control of the linearized perturbation system

With the determination of the parameters in $\boldsymbol{F}(k)$ and $\boldsymbol{G}(k)$, proper control laws can be designed to obtain the required correction torques to reduce the position and velocity errors of the manipulator along a nominal trajectory. This can be done by finding an optimal control, $\boldsymbol{u}^*(k)$, which minimizes the performance index, $J(k)$, while satisfying the constraints of Eq. 16:

$$J(k) = (1/2)[\mathbf{x}^T(k+1)\mathbf{Q}\mathbf{x}(k+1) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)] \quad (34)$$

where $\boldsymbol{Q}$ is a $p\times p$ semipositive definite weighting matrix and $\boldsymbol{R}$ is an $n\times n$ positive definite weighting matrix. The one-step performance index in Eq. 34 indicates that the objective of the optimal control is to drive the position and velocity errors of the manipulator to zero along the nominal trajectory in a coordinated position and rate control per interval step, while at the same time, a cost is attached to the use of control efforts. The optimal control solution which minimizes the functional in Eq. 34 subject to the constraints of Eq. 16 is well-known and is found to be[3,18]

$$\mathbf{u}^*(k) = -[\mathbf{R} + \hat{\mathbf{G}}^T(k)\mathbf{Q}\hat{\mathbf{G}}(k)]^{-1}\,\hat{\mathbf{G}}^T(k)\mathbf{Q}\hat{\mathbf{F}}(k)[\mathbf{x}(k) + \hat{\mathbf{w}}(k)] \quad (35)$$

where $\hat{\mathbf{F}}(k)$, $\hat{\mathbf{G}}(k)$, and $\hat{\mathbf{w}}(k)$ are the system parameters obtained from the identification algorithm (Eqs. 29-31) at the kth sampling instant.

The above identification and control algorithms in Eqs. 29-31 and Eq. 35 do not require complex computations. In Eq. 31, $(\boldsymbol{z}^T(k)\boldsymbol{P}(k)\boldsymbol{z}(k) + \varrho)$ gives a scalar value which simplifies its inversion. Although the weighting factor ϱ can be adjusted for each ith parameter vector $\boldsymbol{\theta}_i(k)$ as desired, this requires excessive computations in the $\boldsymbol{P}(k+1)$ matrix. For real-time robot arm control, such adjustment is not desirable. $\boldsymbol{P}(k+1)$ is computed only once at each sampling time using the same weighting factor ϱ. Moreover, since $\boldsymbol{P}(k)$ is a symmetric positive definite matrix, only the upper diagonal matrix of $\boldsymbol{P}(k)$ needs to be computed. The combined identification and control algorithm can be computed in $O(n^3)$ time. The computational requirements of the proposed adaptive control are tabulated in Table 1. The proposed adaptive control block diagram is shown in Figure 1.

Table 1. Computations of the adaptive controller.

Adaptive controller	Multiplications	Additions
Newton-Euler equations of motion	$117n - 24$	$103n - 21$
Least squares identification algorithm	$30n^2 + 5n + 1$	$30n^2 + 3n - 1$
Control algorithm	$8n^3 + 2n^2 + 39$	$8n^3 - n^2 - n + 18$
Total mathematical operations of adaptive controller	$8n^3 + 32n^2 + 5n + 40$	$8n^3 + 29n^2 + 2n + 17$

COMPUTER SIMULATION: A THREE-JOINT *PUMA* ROBOT ARM

A computer simulation study was conducted on a VAX-11/780 computer to evaluate and compare the performance of the proposed adaptive control strategy in the joint-variable space with the computed torque technique[16] for various loading conditions along a given trajectory for a three-joint PUMA manipulator. The computed torque technique is basically a simple proportional plus derivative control (PD controller) with constant feedback gains and the controller has a structure of

$$\tau(t) = \mathbf{D}(\mathbf{q})[\ddot{\mathbf{q}}^d(t) + \boldsymbol{K}_v(\dot{\mathbf{q}}^d(t) - \dot{\mathbf{q}}(t)) + \boldsymbol{K}_p(\mathbf{q}^d(t) - \mathbf{q}(t))] + \mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{c}(\mathbf{q}) \quad (36)$$

where $\boldsymbol{K}_v$ and $\boldsymbol{K}_p$ are, respectively, $n\times n$ velocity and position feedback gain matrices. They are set to $\{20[(n-m)/(\text{rad/s})]\}\mathbf{I}_6$ and $\{100[(n-m)/(\text{rad})]\}\mathbf{I}_6$, respectively, to achieve near critically-damped response of each joint. Although a better performance could be achieved by varying these gains, we kept them constant because the internal control of the PUMA robot arm has fixed constant feedback gains.

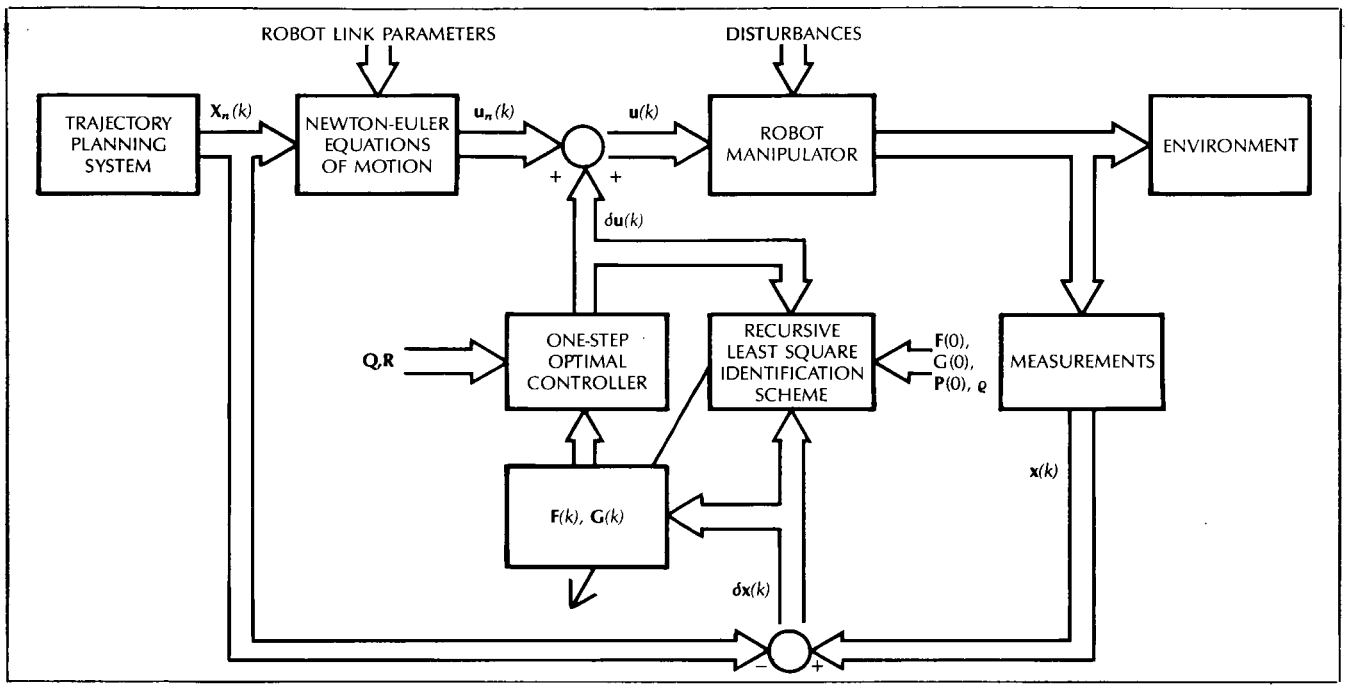


Figure 1. The proposed adaptive control system.

The state equations of the three-joint PUMA robot arm can be obtained by simple manipulations of Lagrange-Euler equations of motion. Defining the state variables, x_i, and the inputs, u_i, as

$$x_i(t) = \theta_i(t) \; ; \; x_{i+3}(t) = \dot{\theta}_i(t) \; ; \; u_i(t) = \tau_i(t) \; ; \; i = 1,2,3$$

the state equations for the three-joint PUMA robot can be obtained as in Eq. 12. Applying the Taylor series expansion to linearize the state equations about the nominal trajectory, the perturbation equations obtained can be transformed to discrete perturbation equations of motion as in Eq. 16.

The manipulator is simulated to move from an initial position $\boldsymbol{q}_{\text{initial}} = (0°, 45°, 45°)^T$ to a final position $\boldsymbol{q}_{\text{final}} = (90°, -45°, 135°)^T$. The required time for this motion is set to one second. A 4-3-4 joint-interpolated trajectory[15] has been preplanned for this motion. From this motion trajectory, it can be easily seen from the kinematics of the manipulator that link two and link three of the arm will be fully stretched at 0.5 seconds. At this position, $\boldsymbol{q}(0.5) = (45°, 0°, 90°)^T$, the joint torques due to gravity loading on the links and the absolute values of joint velocity of the arm have maximum values. The accelerations are changed sharply from maximum values to minimum values or vice versa.

The initial values of the unknown system parameters $\mathbf{F}(k)$ and $\mathbf{G}(k)$ are determined from Eq. 33 at $\boldsymbol{q}(0) = (0°, 45°, 45°)^T$:

$$\mathbf{F}(0) = \begin{bmatrix} 1.000 & 0.002 & 0.000 & 0.010 & 0.000 & 0.000 \\ 0.000 & 0.999 & 0.000 & 0.000 & 0.100 & 0.000 \\ 0.000 & 0.001 & 1.000 & 0.000 & 0.000 & 0.010 \\ 0.000 & 0.032 & -0.001 & 0.999 & 0.002 & 0.000 \\ 0.000 & -0.141 & -0.001 & -0.001 & 0.999 & 0.000 \\ 0.000 & 0.284 & 0.007 & 0.003 & -0.001 & 1.000 \end{bmatrix} \tag{37}$$

$$\mathbf{G}^T(0) = \begin{bmatrix} 0.000 & 0.000 & 0.000 & 0.003 & 0.001 & 0.001 \\ 0.000 & 0.000 & 0.000 & 0.001 & 0.004 & -0.008 \\ 0.000 & 0.000 & 0.000 & 0.001 & -0.008 & 0.047 \end{bmatrix}$$

The weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$, in Eq. 35 are selected, respectively, as

$$\mathbf{Q} = 0.1\, \mathbf{I}_6 \; ; \; \mathbf{R} = 0.000001\, \mathbf{I}_3$$

where $\mathbf{I}_n$ is an $n \times n$ identity matrix. The weighting factor, ϱ, is set to 0.95 to provide proper tracking of the slowly time-varying parameters.

With reference to the robot arm dynamic equations, the physical geometric parameters of the robot, such as the location of center of mass of each link and inertia tensor matrix of each link, are obtained from calculations based on the physical structure of the PUMA robot. We also assume that the robot arm does not know the weight of the object that it is carrying. The coefficients of the state equations are updated in every sampling period based on the actual joint values obtained from integrating the Lagrange-Euler equations of motion. The numerical integration routine is based on Runge-Kutta-Gill fourth-order method with variable step size. The sampling time and the integration step size were chosen to be 10 ms and 2.5 ms, respectively. When the robot is simulated to pick up an object, the inertia matrix of the last link in the Lagrange-Euler equations of motion is modified appropriately to reflect the loading effect; while the inertia matrix of each link in the Newton-Euler equations of motion remains unchanged. A flow chart of computer simulation of the proposed adaptive control strategy is shown in Figure 2.

The performances of the PD and adaptive controllers are compared and evaluated for three different loading conditions and the results are tabulated in Table 2: Case (a) no-load and 10% error in inertia tensor matrix, Case (b) one-half of the maximum load and 10% error in inertia tensor matrix, and Case (c) maximum load (5 lb.) and 10% error in inertia tensor matrix. In each case, a 10% error in inertia matrices means ±10% error about its measured inertial values. For all the above cases, the adaptive controller shows better performance than the PD controller

both in trajectory tracking and the final position errors. Plots of angular position errors for the above three cases for the adaptive control are shown in Figures 3-5.

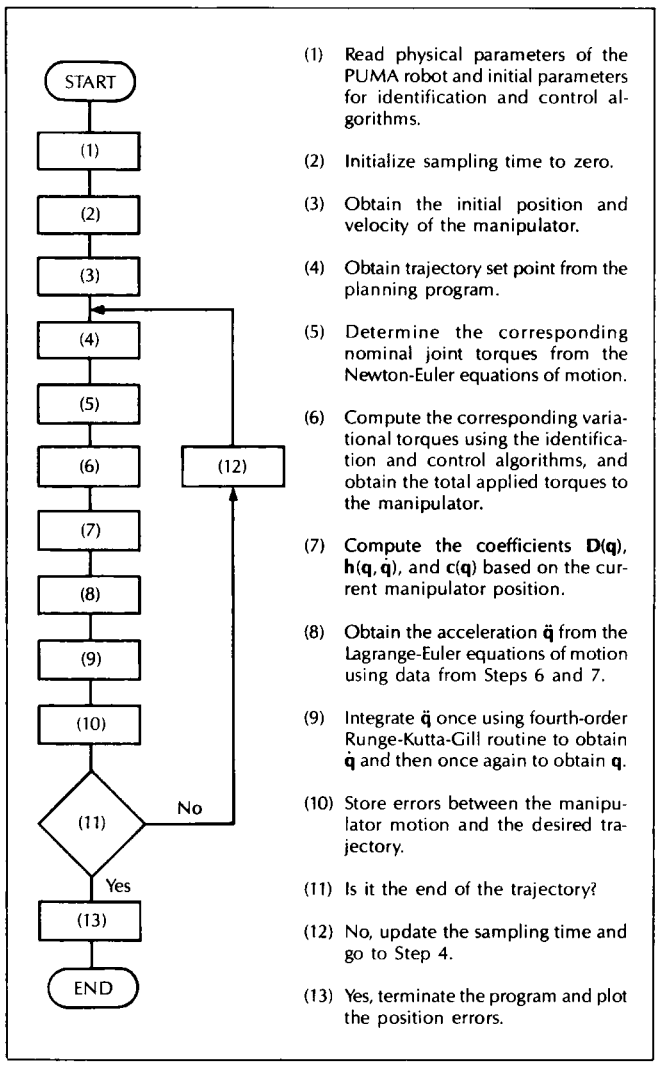


Figure 2. Flow chart showing simulation of the adaptive control strategy.

Given the $F(0)$ and $G(0)$ matrices in Eq. 37, the performance of the adaptive controller is quite sensitive to the initial values in the $P(0)$ matrix and less sensitive to the Q and R weighting matrices. This result is expected as the $P(k)$ matrix has the same effect as the error covariance matrix in stochastic identification with zero mean modeling error, if the weighting factor ϱ is chosen appropriately. In general, the convergence of the parameter identification and hence the performance of the adaptive controller depends on the proper selection of the $P(0)$ matrix. The initial large values of the $P(0)$ matrix usually give better convergence. The values of the Q matrix did not affect the performance of the controller very much; however, a small value of the R matrix did give a better tracking result. For the simulation, two typical initial values of the $P(0)$ matrix are chosen:

$$\mathbf{P}_1(0) = 1500\,\mathbf{I}_{10} \text{ and } \mathbf{P}_2(0) = 150000\,\mathbf{I}_{10}$$

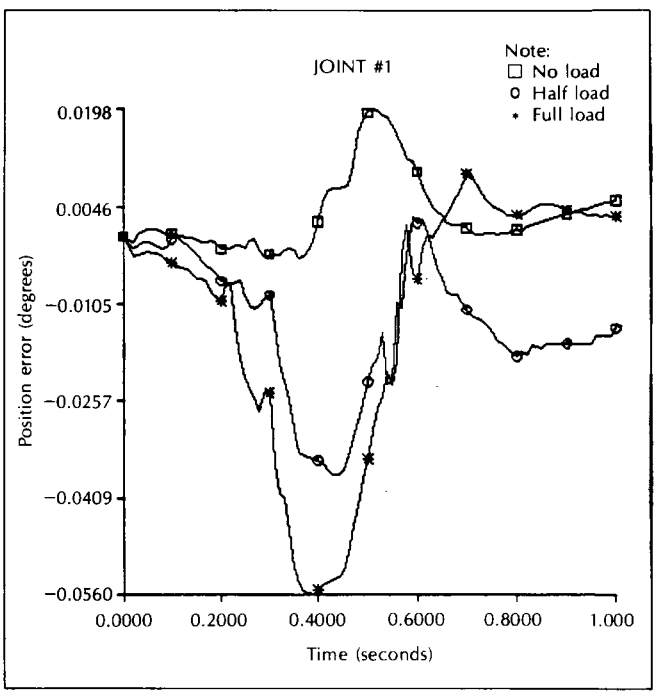


Figure 3. Joint one position error under various loads $\mathbf{P}(0) = \mathbf{P}_2(0)$.

Table 2. Comparisons of the PD and adaptive controllers.

	Various loading conditions	Joint	PD controller: Trajectory tracking: Maximum error (degree)	PD controller: Trajectory tracking: Maximum error (mm)	PD controller: Trajectory tracking: Final position error (degree)	Adaptive controller: Trajectory tracking: Maximum error (degree)	Adaptive controller: Trajectory tracking: Maximum error (mm)	Adaptive controller: Trajectory tracking: Final position error (degree)
(a)	No load and 10% error in inertia tensor	1	0.089	1.55	0.025	0.020	0.34	0.000
		2	0.098	1.71	0.039	0.020	0.36	0.004
		3	0.328	2.86	0.121	0.032	0.28	0.002
(b)	One-half maximum load and 10% error in inertia tensor	1	0.121	2.11	0.054	0.045	0.78	0.014
		2	0.147	2.57	0.078	0.065	1.14	0.050
		3	0.480	4.19	0.245	0.096	0.83	0.077
(c)	Maximum load and 10% error in inertia tensor	1	0.145	2.53	0.082	0.069	1.20	0.023
		2	0.185	3.23	0.113	0.069	1.22	0.041
		3	0.607	5.30	0.360	0.066	0.58	0.019

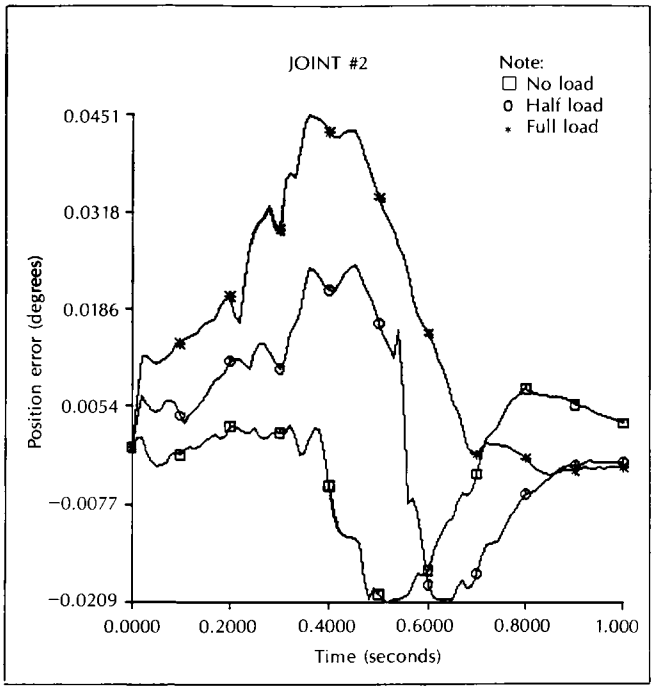


Figure 4. Joint two position error under various loads $\mathbf{P}(0) = \mathbf{P}_2(0)$.

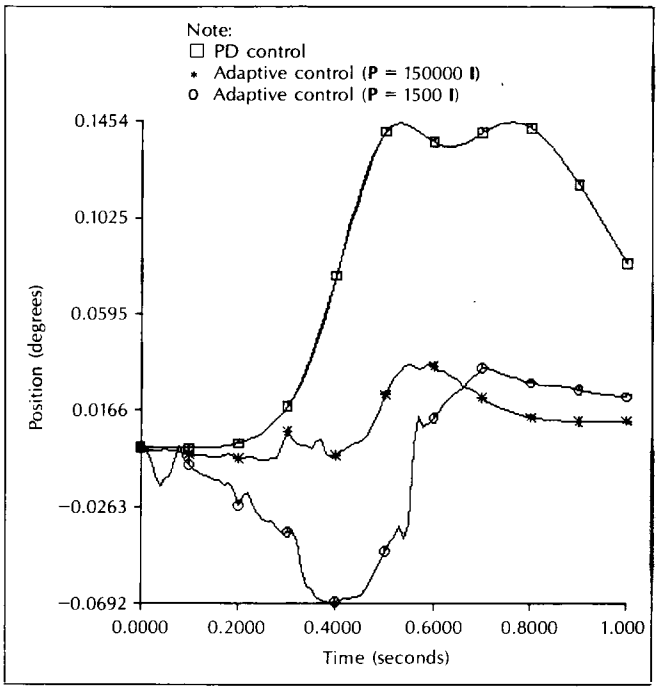


Figure 6. Case (c) (maximum load (5 lb) and a 10% error in inertia tensor): joint one position error using both controllers.

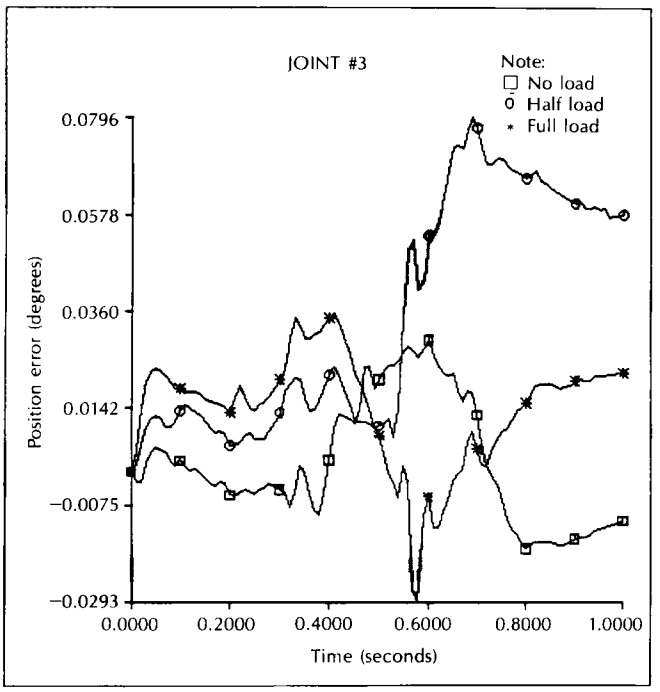


Figure 5. Joint three position error under various loads $\mathbf{P}(0) = \mathbf{P}_2(0)$.

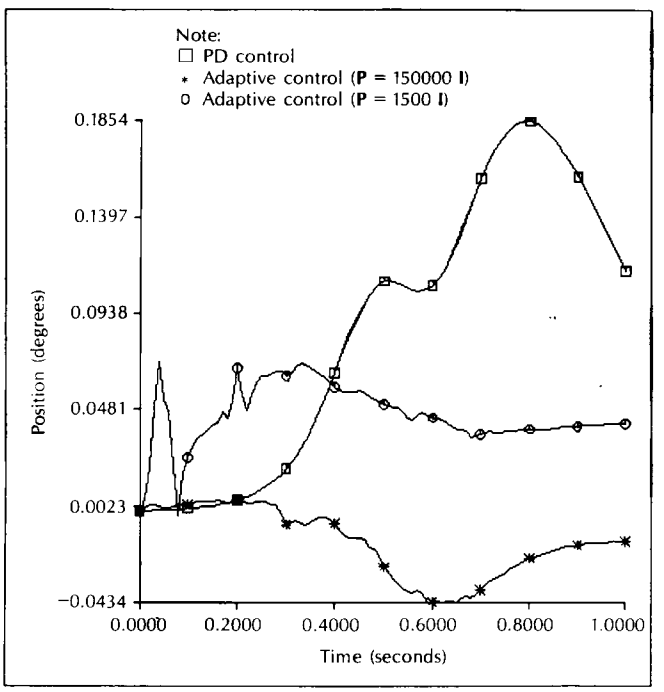


Figure 7. Case (c) (maximum load (5 lb) and a 10% error in inertia tensor): joint two position error using both controllers.

Plots showing the convergence of the angular positions of joint three in Case (c) due to the effect of $\boldsymbol{P}(0)$ can be seen in Figures 6-8. In these figures, the angular position errors using the $\boldsymbol{P}_2(0)$ matrix converged much faster than the $\boldsymbol{P}_1(0)$ matrix for Case (c). Using the same initial values in the $\boldsymbol{F}(0),\boldsymbol{G}(0),\boldsymbol{Q},\boldsymbol{R},\boldsymbol{P}_1(0)$ and $\boldsymbol{P}_2(0)$ matrices, a second trajectory with the same initial positions, but different final position, revealed the same findings. The applied torques computed from the PD and adaptive controllers for Case (c) are shown in Figure 9.

The proposed adaptive control in joint-variable coordinates can be implemented in a PDP-11/45 computer for real-time control of a three-joint PUMA robot arm. Based on the manufacturer's specification sheet of a PDP-11/45 computer, an ADDF (floating point addition) instruction requires 5.17 μs and a MULF (floating point multiply) instruction requires 7.17 μs. If we assume that for each ADDF and MULF instruction, we need to fetch data from the core memory twice and the memory cycle time is 450 ns, then the proposed adaptive control requires approx-

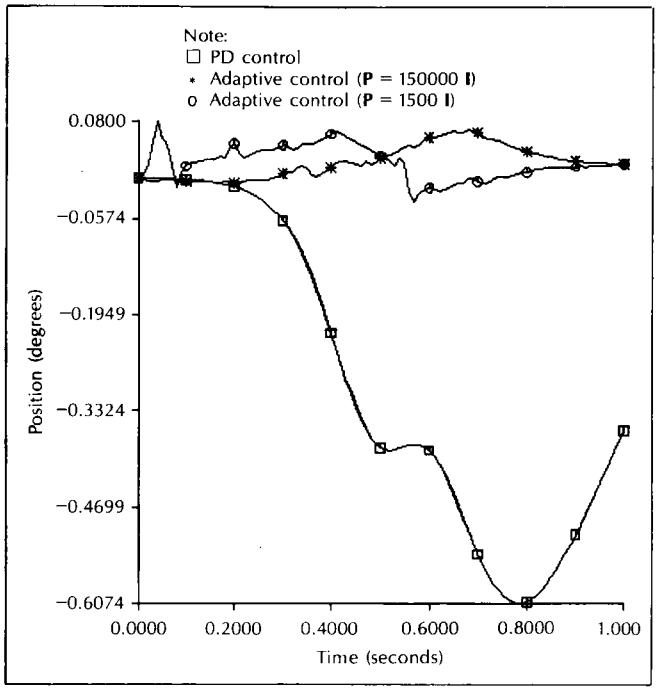


Figure 8. Case (c) (maximum load (5 lb) and a 10% error in inertia tensor): joint three position error using both controllers.

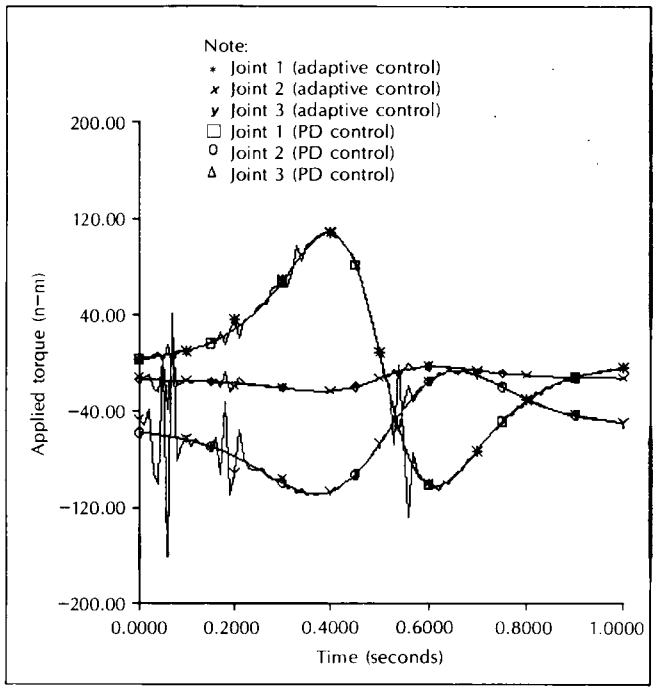


Figure 9. Case (c) (maximum load (5 lb) and a 10% error in inertia tensor): applied torques computed from the PD and adaptive controllers.

imately 7.5 ms to compute the necessary joint torques to servo the first three joints of a PUMA robot arm for a trajectory set point.

CONCLUSIONS

An adaptive perturbation control strategy which tracks a desired time-based trajectory as closely as possible for all times over a wide range of manipulator motion and payloads in joint-variable coordinates has been presented. The adaptive control is based on the perturbation equations of the manipulator system along a nominal trajectory. A computer simulation study was conducted to evaluate the performance of a three-joint PUMA robot arm. From the simulation study, the adaptive control was found to perform better for various loading conditions than the proportional plus derivative controller both in trajectory tracking and final position error.

In summary, the adaptive control system is characterized by feedforward and feedback components. The feedforward component computes the nominal torques $\boldsymbol{u}_n(t)$ from the Newton-Euler equations of motion using the joint information from the trajectory planning program. This computation can be completed in $O(n)$ time. The feedback component consisting of recursive least square identification and an optimal adaptive self-tuning control algorithm for the linearized system computes the perturbation torques in $O(n^3)$ time. Since the computations of the nominal and perturbation torques can be performed in parallel, the computations of the adaptive control may be implemented in low-cost microprocessors. The computations of the adaptive control for a three-joint PUMA robot arm can be completed within 10 ms using a PDP-11/45 computer. The analysis and computer simulation of the proposed adaptive control strategy presented an ideal system study; physical implementation of the proposed adaptive control may require further investigation on the effects of gear friction, backlash, control device dynamics, and flexible link structure to the controller.

REFERENCES

1 ARMSTRONG, W. M.
"Recursive Solution to the Equations of Motion of an N-link Manipulator." *Proceedings of the 5th World Congress, Theory of Machines, Mechanisms,* 2 (July 1979), 1343-1346.

2 BEJCZY, A. K.
"Robot Arm Dynamics and Control." Technical Memorandum 33-669. Jet Propulsion Laboratory (February 1974).

3 CLARKE, D. W. and GAWTHROP, P. J.
"Self-tuning Controller." *Proceedings of IEEE* 122:9 (1975), 929-934.

4 DUBOWSKY, S. and DESFORGES, D. T.
"The Application of Model Referenced Adaptive Control to Robotic Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control,* 101 (September 1979), 193-200.

5 EYKHOFF, P.
System Identification: Parameter and State Estimation. Wiley-Interscience (1974), 240-241.

6 HOLLERBACH, J. M.
"A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity." *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-10:11 (November 1980), 730-736.

7 KAHN, M. E. and ROTH, B.
"The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains." *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control,* 93 (September 1971), 164-172.

8 KOIVO, A. J. and GUO, T. H.
"Adaptive Linear Controller for Robotic Manipulators." *IEEE Transactions on Automatic Control,* AC-28:1 (February 1983), 162-171.

9 LEE, C. S. G.
"Robot Arm Kinematics, Dynamics, and Control." *Computer,* 15:12 (December 1982), 62-80.

10 LEE, C. S. G. and CHUNG, M. J.
"An Adaptive Control Strategy for Mechanical Manipulators." *IEEE Transactions on Automatic Control,* AC-29:9 (September 1984), 837-840.

11 LEE, C. S. G. and LEE, B. H.
"Resolved Motion Adaptive Control for Mechanical Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement and Control* 106 (June 1984), 134-142.

12 LEE, C. S. G.; LEE, B. H.; and NIGAM, R.
"Development of the Generalized d'Alembert Equations of Motion for Mechanical Manipulator." *Proceedings of the 22nd Conference on Decision and Control.* San Antonio, Texas (December 14-16, 1983), 1205-1210.

13 LUH, J. Y. S.; WALKER, M. W.; and PAUL, R. P.
"On-line Computational Scheme for Mechanical Manipulators." Transactions of the ASME, Journal of Dynamics Systems, Measurement and Control 102 (June 1980), 69-76.

14 LUH, J. Y. S.; WALKER, M. W.; and PAUL, R. P.
"Resolved-Acceleration Control of Mechanical Manipulators." *IEEE Transactions on Automatic Control*, AC-25:3 (June 1980), 468-474.

15 ORIN, D. E.; MCGHEE, R. B.; VUKOBRATOVIC, M.; and HARTOCH, G.
"Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods." *Math. Biosc.* 43 (1979), 107-130.

16 PAUL, R. P.
"Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm." Stanford Artificial Intelligence Laboratory, A.I. Memo 177 (September 1972).

17 PAUL, R. P.
Robot Manipulators: Mathematics, Programming, and Control. MIT Press (1981).

18 SARIDIS, G. N. and LOBBIA, R. N.
"Parameter Identification and Control of Linear Discrete-Time Systems." *IEEE Transactions on Automatic Control*, AC-17:1 (February 1972), 52-60.

19 TURNEY, J. L.; MUDGE, T. N.; and LEE, C. S. G.
"Equivalence of Two Formulations for Robot Arm Dynamics." CRIM Technical Report RSD-TR-3-82. The University of Michigan (December 1980).

20 WHITNEY, D. E.
"Resolved Motion Rate Control of Manipulators and Human Prostheses." *IEEE Transactions on Man-Machine System*, MMS-10:2 (June 1969), 47-53.

21 WHITNEY, D. E.
"The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 94 (December 1972), 303-309.

22 YOUNG, K. K. D.
"Controller Design for a Manipulator Using Theory of Variable Structure Systems." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8:2 (February 1978), 101-109.

MEETINGS AND CONFERENCES

Report on EAI Computer Users' Group Meeting – 1984

DFVLR (German Aerospace Research Establishment) organized the annual meeting of the International Chapter of the EAI Computer Users' Group; the meeting took place in Oberpfaffenhofen near Munich, Federal Republic of Germany, 17-19 October 1984.

A total of 45 people assembled together from Austria, France, Germany, Great Britain, Holland, Italy, Yugoslavia, the U.S.A. and the People's Republic of China to talk about their experiences and problems with EAI equipment and to meet EAI representatives and other users.

Flight Systems Dynamics of DFVLR began the meeting with a survey of the activities of the institute. Not only was a presentation given on the simulation and control of the DFVLR cryo-wind-tunnel, but also an excursion was made to GSOC (German Space Operation Center) and the hybrid computer facility, where the simulation and control of the cryo-wind-tunnel is running.

Continuing in Herrsching on the pleasant Ammersee, the program contained the following items:

(1) EAI presentations:

- Overview of 1984 EAI product range
- EAI SIMSTAR – product overview and status report 1984

(2) User presentations:

- Hybrid simulation at GASUNIE research
- Hardware-in-the-loop evaluation of turbine engine controls
- Microprocessor-based time delay generator
- Equipment for ATTAS System Simulation
- A 6-DOF simulation of the VFW 614 Aircraft (ATTAS)
- Simulation of computer control systems at the Faculty of Electrical Engineering in Ljublijana
- The use of a small hybrid computer in cardiovascular research
- High-speed data transfer between EAI 2000 and Gould 3277
- Some problems of HYSHARE 700 in simulation application
- Replacing an old PACER 600 by SIMSTAR

(3) Working groups:

- EAI 2000 systems
- Questions and comments related to the EAI SIMSTAR System

Next year's Users' Group Meeting is scheduled for 2-4 October 1985 in Santa Margherita near Genova, Italy, and will be organized by ANSALDO. For information, please contact D. Marras, ANSALDO, Via Pacinotti 20, I-15161 Genova, Italy.