N90-29832

DIRECT ADAPTIVE CONTROL OF A PUMA 560 INDUSTRIAL ROBOT

Homayoun Seraji, Thomas Lee
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109, USA

Michel Delpech
DRT/TVE/EA
Centre National d'Etudes Spatiales
Toulouse 31055
France

Abstract

The paper describes the implementation and experimental validation of a new direct adaptive control scheme on a PUMA 560 industrial robot. The testbed facility consists of a Unimation PUMA 560 six-jointed robot and controller, and a DEC MicroVAX II computer which hosts the RCCL (Robot Control "C" Library) software. The control algorithm is implemented on the MicroVAX which acts as a digital controller for the PUMA robot, and the Unimation controller is effectively bypassed and used merely as an I/O device to interface the MicroVAX to the joint motors. The control algorithm for each robot joint consists of an auxiliary signal generated by a constant-gain PID controller, and an adaptive position-velocity (PD) feedback controller with adjustable gains. The adaptive independent joint controllers compensate for the inter-joint couplings and achieve accurate trajectory tracking without the need for the complex dynamic model and parameter values of the robot. Extensive experimental results on PUMA joint control are presented to confirm the feasibility of the proposed scheme, in spite of strong interactions between joint motions. Experimental results validate the capabilities of the proposed control scheme. The control scheme is extremely simple and computationally very fast for concurrent processing with high sampling rates.

1. Introduction

During the past decade, the control of robot manipulators has been the focus of considerable research, and many different control schemes have been suggested in the literature. With a few exceptions, all existing schemes are tested on manipulators through computer simulations *only*, often using the popular two-link arm paradigm. Although the simulations are useful for illustration and proof-of-concept, practical issues such as effect of friction, limitation on controller gains, and sampling rate constraint are often neglected. Despite the large number of proposed manipulator control schemes, the number of schemes that have actually been experimentally evaluated on manipulators, and particularly on industrial robots, is very small today.

This paper describes the implementation and experimental validation of a newly developed direct adaptive control scheme [1,2] on a six-jointed PUMA 560 industrial robot. The control scheme has a decentralized structure and consists of a number of simple local feedback controllers. Each local controller consists of an auxiliary signal generated by a constant-gain PID controller, and an adaptive position-velocity (PD) feedback controller whose gains are updated on-line in real time. The control scheme is implemented on a MicroVAX II computer using the RCCL software, and generates the control signals that drive the PUMA joint motors directly. This simple control scheme is *not* based on the complex PUMA dynamic model and is therefore implemented at a high sampling rate and yields a good tracking performance.

The paper is structured as follows. In Section 2, an overview of the design theory for adaptive joint controllers is given. The descriptions of the testbed facility at JPL and the RCCL software are given in Section 3. Section 4 discusses the experimental results on simultaneous control of all six joint angles of the PUMA 560 industrial robot. The conclusions drawn from the paper are discussed in Section 5.

2. Theory Overview

In this section, the design theory for direct adaptive control of manipulators is outlined. The proposed control scheme has a decentralized structure, where each manipulator joint is controlled independently of the others by use of a local feedback controller. Therefore, the dynamics of each joint will be considered separately.

Consider a robot manipulator with the n joint angles $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$ and the corresponding n joint torques $T = [T_1, T_2, \ldots, T_n]$. The dynamic model of the i^{th} manipulator joint which relates θ_i to T_i can be represented by the second-order nonlinear differential equation

$$m_{ii}(\theta)\ddot{\theta}_i + \sum_{\substack{j=1 \\ \neq i}}^{n} m_{ij}(\theta)\ddot{\theta}_j + n_i(\theta,\dot{\theta}) + g_i(\theta) + h_i(\dot{\theta}_i) = T_i \tag{1}$$

where m_{ij} is the $(i,j)^{th}$ element of the inertia matrix, and n_i, g_i, h_i are the i^{th} elements of the Coriolis/centrifugal vector, gravity vector, and friction vector, respectively. The terms in equation (1) are highly complicated nonlinear functions of the manipulator configuration θ, the speed of motion $\dot{\theta}$, and the payload inertial parameters. The manipulator control problem is to generate the joint torques $T_i(t)$, for $i = 1, \ldots, n$, such that the joint angles $\theta_i(t)$ track some desired trajectories $\theta_{di}(t)$ as closely as possible.

In the proposed decentralized control scheme, the i^{th} joint is controlled by the local adaptive feedback control law [2]

$$T_i(t) = f_i(t) + k_{pi}(t)e_i(t) + k_{vi}(t)\dot{e}_i(t) \tag{2}$$

as shown in Figure 1, where $e_i(t) = \theta_{di}(t) - \theta_i(t)$ and $\dot{e}_i(t) = \dot{\theta}_{di}(t) - \dot{\theta}_i(t)$ are the position and velocity tracking-errors, and the controller terms are given by

Weighted tracking-error

$$r_i(t) = w_{pi}e_i(t) + w_{vi}\dot{e}_i(t) \tag{3}$$

Auxiliary signal

$$f_i(t) = f_i(0) + \delta_i \int_0^t r_i(t)dt + \rho_i r_i(t) \tag{4}$$

Position feedback gain

$$k_{pi}(t) = k_{pi}(0) + \alpha_i \int_0^t r_i(t)e_i(t)dt + \beta_i r_i(t)e_i(t) \tag{5}$$

Velocity feedback gain

$$k_{vi}(t) = k_{vi}(0) + \gamma_i \int_0^t r_i(t)\dot{e}_i(t)dt + \lambda_i r_i(t)\dot{e}_i(t) \tag{6}$$

In equations (3)-(6), $\{\delta_i, \alpha_i, \gamma_i\}$ are positive scalar integral adaptation gains, $\{\rho_i, \beta_i, \lambda_i\}$ are non-negative scalar proportional adaptation gains, and $\{w_{pi}, w_{vi}\}$ are scalar positive weighting factors of the position and velocity tracking-errors.

From the implementation viewpoint, the auxiliary signal $f_i(t)$ can be generated by a constant-gain PID feedback controller acting on the position tracking-error $e_i(t)$, since from equations (3) and (4), $f_i(t)$ can be expressed as

$$\begin{aligned} f_i(t) &= f_i(0) + \rho_i\left[w_{pi}e_i(t) + w_{vi}\dot{e}_i(t)\right] + \delta_i \int_0^t \left[w_{pi}e_i(t) + w_{vi}\dot{e}_i(t)\right]dt \\ &= f_i(0) + \left[\rho_i w_{pi} + \delta_i w_{vi}\right]e_i(t) + \left[\rho_i w_{vi}\right]\dot{e}_i(t) + \left[\delta_i w_{pi}\right]\int_0^t e_i(t)dt \end{aligned} \tag{7}$$

Hence, the joint torque (2) can be generated by the adaptive PID feedback controller

$$T_i(t) = T_i(0) + [\overline{k}_{pi}(t) + \overline{k}_{Ii}\int_0^t dt + \overline{k}_{vi}(t)\frac{d}{dt}]e_i(t) \tag{8}$$

where $\overline{k}_{pi}(t) = k_{pi}(t) + \rho_i w_{pi} + \delta_i w_{vi}$, $\overline{k}_{Ii} = \delta_i w_{pi}$, and $\overline{k}_{vi}(t) = k_{vi}(t) + \rho_i w_{vi}$ are the adjustable PID gains and $T_i(0) = f_i(0)$ is the initial joint torque.

The dynamics of manipulator joints are highly coupled, as can be seen from equation (1). The local control law (2) for each joint compensates, to a large extent, for the static and dynamic cross-couplings that exist between the manipulator joints. For fast simultaneous motion of all joints, the inter-joint coupling effects can be significant and may cause instability under the decentralized adaptive control scheme (2)-(6). In such cases, the controller adaptation laws are modified slightly in order to achieve robust stability in the presence of the unmodeled cross-coupling effects. A popular approach is the "σ-modification" method [3], which yields the adaptation laws

$$f_i(t) = f_i(0) + \delta_i \int_0^t r_i(t)dt + \rho_i r_i(t) - \sigma_i \int_0^t f_i(t)dt \tag{9}$$

$$k_{pi}(t) = k_{pi}(0) + \alpha_i \int_0^t r_i(t)e_i(t)dt + \beta_i r_i(t)e_i(t) - \sigma_i \int_0^t k_{pi}(t)dt \tag{10}$$

$$k_{vi}(t) = k_{vi}(0) + \gamma_i \int_0^t r_i(t)\dot{e}_i(t)dt + \lambda_i r_i(t)\dot{e}_i(t) - \sigma_i \int_0^t k_{vi}(t)dt \tag{11}$$

where σ_i is a positive scalar design parameter. The σ-modified adaptation laws produce a non-zero residual tracking-error of $0(\sqrt{\sigma_i})$ but guarantee stability in the presence of inter-joint couplings.

It is seen that the controller adaptation laws (9)-(11) and the control action (2) are based entirely on the observed manipulator performance through $\theta(t)$ and $\theta_d(t)$ rather than on the manipulator dynamic model (1). As a consequence, the knowledge of either the complex dynamic model and the parameter values of the manipulator or the inertial parameters of the payload is *not* required in the control law formulation. Thus, the adaptive controllers can cope with uncertainties or variations in the manipulator or the payload parameters. This is a highly desirable feature in practical applications, where some dynamic effects such as friction can not be modeled accurately and the payload mass can vary substantially. Furthermore, the proposed decentralized control scheme is extremely fast computationally, since the controller gains are generated on-line in real time by simple adaptation laws, and hence the control action can be evaluated very rapidly. Due to its simplicity and decentralized structure, the proposed scheme can be implemented on parallel processors for distributed concurrent computing with high sampling rates, yielding improved dynamic performance.

3. Description of Testbed Facility

In this section, we describe the robotic testbed facility at the Jet Propulsion Laboratory.

The testbed facility at the JPL Robotics Research Laboratory consists of a six-jointed Unimation PUMA 560 robot and controller, and a DEC MicroVAX II computer, as shown in the functional diagram of Figure 2. The major components of the Unimation controller are the LSI 11/73 microcomputer, six 6503 microprocessor boards (one per joint), and serial/parallel interface cards. The MicroVAX II hosts the RCCL (Robot Control "C" Library) software, which was developed by Hayward and Lloyd at Purdue and McGill Universities [4,5]. The original version runs on a DEC VAX 750 computer, and later the software is ported to a DEC MicroVAX II running UNIX 4.3 BSD operating system. The organization of the robot software is reflected in the control hierarchy diagram illustrated in Figure 2. The complete software resides on two different pieces of computing hardware. A MicroVAX computer, which plays the supervisory role, hosts a two-level software written in the "C" language. The lower level, called Robot Control Interface (RCI), provides the programmer a facility to write real-time control procedures. It serves as a substrate to the higher level (RCCL), which gets its collective name from the robot software. The higher level consists of routines to specify a robot trajectory in Cartesian coordinates. The second piece of hardware is the Unimation controller hosting an LSI 11/73 processor on which an I/O control program called "moper" executes to monitor communication between the 6503 joint microprocessors and the RCI control level.

At the lowest level of the hierarchy, robot servoing is achieved by the 6503 joint microprocessors. The processor has two different operational modes: position and current. In position mode, the 6503 processor accepts a position setpoint from the LSI 11/73 and servos to the desired position by executing a control code written in the assembly language that is stored in ROM (this servo code was developed by the Unimation Corporation). In this mode, the control task, triggered by a hardware clock, executes approximately at the sampling rate of 1 KHz. In current mode, each input is interpreted as current (or torque) by the 6503 processor and is simply converted into an analog value to be forwarded directly to the power drive electronics. This mode makes possible the implementation of any joint control law (e.g. force control and adaptive control) on a remote computer that can interface with the LSI 11/73. In order to implement the adaptive control algorithm, the joint PID servos provided by the resident Unimation code are in effect disconnected by selection of the 6503 current mode, and current inputs are supplied directly from the MicroVAX for driving the robot.

At the intermediate level, the LSI 11/73 executes the "moper" communication monitor program that transfers data and commands back and forth between the LSI 11/73 and the MicroVAX II host computer. The interface between the two processors is a DRV11 high-speed parallel link for control communication. The moper program synchronizes the operation of the entire system as described in the following. A hardware clock located in the Unimation controller constantly interrupts the LSI 11/73, and at each interrupt, the moper collects data relating to the robot state, such as joint positions, currents, and arm status, and transfers it to the MicroVAX via an interrupt. The MicroVAX receives this data and immediately sends position or current values that have been computed in the previous cycle by the control code to the LSI 11/73 for execution. Available system sampling time is in increments of 7 milliseconds, ranging from 7 to 56 milliseconds (18—143 Hz). The sampling time of 28 milliseconds is best suited for the 6503 position mode and is set as default. In our adaptive control implementation, the lowest sampling time of 7 milliseconds (143 Hz) is chosen to obtain the best control performance.

On the MicroVAX, two programs run concurrently: the foreground planning level (RCCL) and the background control level (RCI). The planning level executes in the foreground in the sense that it interacts with the user and performs high-level computations as well as communicating with the control level. It has access to standard I/O resources such as files, devices, and system calls. At this level, the programmer can specify the task in Cartesian coordinate system in terms of homogeneous transformations and define a robot end-effector trajectory with a series of via points. Essentially the planning level consists of task sequencing, motion planning and queueing, and modeling of the world in terms of homogeneous transformations.

RCI (Robot Control Interface) level executes a series of control routines during each sampling period to interface in real-time with the robot. In practice, the control level communicates with the planning level only through global external variables (i.e., shared memory). Both levels can communicate with the robot through predefined global variables: "how," that contains information describing the state of the arm, and "chg," that is used to control the arm. These variables are used for robot control and are usually accessed at the control level. To meet the constraints imposed by the sampling time in the range of milliseconds, the control level is executed in the UNIX kernel mode at the highest priority, which effectively locks out all hardware interrupts. Once initiated, it quickly loads the memory context of the robot control code, performs I/O with moper, and executes the control code and supporting RCI interface routines. Since the control level is in kernel mode, it cannot access the usual system calls or I/O facilities.

The RCCL software was originally designed to control the robot in the 6503 position mode (i.e., with the control action provided at the 6503 joint processor level at 1 KHz sampling rate). This way the user can concentrate mostly on the programming aspects of performing a task rather than the real-time control issues. For this purpose, by default, a set of standard real-time control routines is provided to perform functions such as trajectory generation, error checking, event synchronization, kinematic computations, and coordination with the planning level. In order to implement a different control scheme, however, the user selects the 6503 current mode to drive the robot, which in turn necessitates writing control procedures essentially to replace the nominal control functions. In addition, the user must consider meeting his newly imposed sampling time requirement; more often he wants to lower the sampling time to control the robot more effectively. As a consequence, smaller and more compact control code must be written to meet the real-time constraint. For example, to implement our adaptive controller in the sampling rate of 7 milliseconds (lowest rate available in RCCL), the need for a trajectory generator is met by writing a simple cycloidal generator that provides smooth series of setpoints without added features such as real-time trajectory modification. In addition, the adaptive control algorithm for computing the desired motor currents from the observed joint positions is coded in its most numerically efficient form.

4. Experimentation with a PUMA 560 Robot

In this section, the theoretical results outlined in Section 2 are applied to a six-jointed PUMA 560 industrial robot in the testbed facility described in Section 3.

To test and evaluate the control scheme of Section 2, the adaptive controllers are implemented on all six joints of the PUMA 560 robot. The dynamic model for a typical i^{th} joint of PUMA can be written as

$$m_{ii}(\theta)\ddot{\theta}_i + \sum_{\substack{j=1 \\ \neq i}}^{6} m_{ij}\ddot{\theta}_j + n_i(\theta,\dot{\theta}) + g_i(\theta) + h_i(\dot{\theta}_i) = T_i \tag{12}$$

where $\theta = [\theta_1, \ldots, \theta_6]$ and the terms in equation (12) are highly complicated nonlinear functions of θ and $\dot{\theta}$ as given in [6]. From equation (12), it is seen that the effective inertia $m_{ii}(\theta)$, the gravity loading $g_i(\theta)$, and the Coriolis/centrifugal torque $n_i(\theta,\dot{\theta})$ seen at each joint are nonlinear functions of *all* six joint angles; i.e., the robot configuration and speed. Furthermore, there are inertial couplings between joint motions, as indicated by $m_{ij}\ddot{\theta}_j$ terms, with the coupling factors dependent on the robot configuration. In the adaptive control implementation, the i^{th} joint is controlled independently by the local feedback law

$$T_i(t) = f_i(t) + k_{pi}(t)e_i(t) + k_{vi}(t)\dot{e}_i(t) \tag{13}$$

where $e_i(t) = \theta_{di}(t) - \theta_i(t)$ is the position tracking-error, $\theta_{di}(t)$ is the reference trajectory, and $[f_i, k_{pi}, k_{vi}]$ are the auxiliary signal, position and velocity feedback gains for the i^{th} joint, respectively. It is seen that although the joint dynamics (12) are coupled, the proposed control scheme (13) is decentralized, i.e., the control torque T_i does not depend on the joint angle θ_j for $j \neq i$.

In the experiment on adaptive control of PUMA, the sampling period is chosen as the smallest possible value $T_s = 7$ milliseconds (i.e., sampling frequency $f_s = 143$ Hz), since the on-line computations involved in the adaptive control law (13) are a few simple arithmetic operations. The adaptation gains in equations (3)-(6) are selected after a few trial-and-errors as *

$$\begin{gathered} w_{p1} = 15 \quad , \quad w_{v1} = 10 \quad , \quad w_{p2} = 40 \quad , \quad w_{v2} = 20 \quad , \quad w_{p3} = 12 \quad , \quad w_{v3} = 4 \\ w_{p4} = 3 \quad , \quad w_{v4} = 2 \quad , \quad w_{p5} = 3 \quad , \quad w_{v5} = 2 \quad , \quad w_{p6} = 3 \quad , \quad w_{v6} = 2 \\ \text{All joints} : \delta = 30, \alpha = 100, \gamma = 800, \rho = \beta = \lambda = \sigma = 0 \end{gathered} \tag{14}$$

The initial values of all controller gains are chosen as zero, i.e. $k_{pi}(0) = k_{vi}(0) = 0$ for $i = 1, \ldots, 6$. The initial values of the auxiliary signals are chosen as

$$\begin{aligned} f_2(0) &= 12\text{sgn}[\theta_{d2}(\tau) - \theta_2(0)] + 1.02 \sin\theta_2(0) \\ &\quad - 8.4 \sin[\theta_2(0) + \theta_3(0)] - 37.2 \cos\theta_2(0) && \text{Nt.meter} \\ f_3(0) &= 2\text{sgn}[\theta_{d3}(\tau) - \theta_3(0)] + 0.25 \cos[\theta_2(0) + \theta_3(0)] \\ &\quad - 8.4 \sin[\theta_2(0) + \theta_3(0)] && \text{Nt.meter} \\ f_1(0) &= f_4(0) = f_5(0) = f_6(0) = 0 \end{aligned} \tag{15}$$

In the above expressions, the first term is chosen empirically to overcome the large stiction (static friction) present in the joints, and the remaining terms are used to compensate for the initial gravity loading [6]. It is important to note that friction and gravity compensations are *not* used separately in addition to the adaptive controllers, and are used merely as the initial conditions of the auxiliary signals in order to improve the initial responses of the joint angles. Furthermore, no information about the PUMA dynamic model or parameter values is used for implementation of the control scheme.

In the experiment, the PUMA arm is initially at the "zero" position $\theta(0) = [0,0,0,0,0,0]$ with the upper arm horizontal and the forearm vertical, forming configuration. All the six joint angles are then commanded to change simultaneously from the zero positions to the goal positions $\theta_d(3) = [60°, -60°, 60°, 60°, -60°, -60°]$ in three seconds, and the desired trajectories $\theta_{di}(t)$ are synthesized by a cycloidal trajectory generator software in RCCL. While the robot is in motion, the readings of the joint encoders at each sampling instant are recorded directly from the robot, converted into degrees and stored in a data file. Figures 3(i)-(vi) show the desired and actual trajectories of all six PUMA joint angles. It is seen that each joint angle $\theta_i(t)$ tracks the desired trajectory $\theta_{di}(t)$ very closely despite inter-joint couplings. The experimental results demonstrate that adaptive independent joint control of the PUMA robot is feasible, in spite of the static and dynamic couplings between the joints.

* The unit of angle in the control program is "radian," and hence the numerical values of the adaptation gains are large.

5. Conclusions

A decentralized direct adaptive control scheme has been experimentally validated on a PUMA 560 robot, where each robot joint is controlled independently by a simple local feedback controller at a high sampling rate. This avoids the computational burden of a centralized controller, which results in a slower sampling rate and hence degrades the robot performance. The experimental results demonstrate that accurate trajectory tracking is achieved by a simple control algorithm, without any knowledge of the complex PUMA dynamic model.

Adaptive control is particularly useful in applications (such as in space) where the manipulator has long light-weight arms and handles payloads of unknown and heavy weights. In such cases, the dynamics of the manipulator is dominated by the inertial terms due to the payload. The controller adaptation can then compensate for such terms and provide a stable and consistent performance under gross payload variations.

Finally, it is important to note that the rate of sampling, f_s, has a central role in the performance of any digital control system. In general, the value of f_s is dictated largely by the amount of on-line computations that need to be performed during each sampling period in order to calculate the required control action. The simplicity of the control scheme proposed in this paper allows joint servo loops to be implemented with a high sampling rate, yielding improved tracking performance.

6. Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

7. References

1. H. Seraji: "A new approach to adaptive control of manipulators," ASME Journ. Dynamic Systems, Measurement and Control, 1987, 109(3), pp. 193-202.
2. H. Seraji: "Adaptive independent joint control of manipulators: Theory and experiment," Proc. IEEE Intern. Conf. on Robotics and Automation, Philadelphia, 1988, Vol. 2, pp. 854-861.
3. P.A. Ioannou: "Decentralized adaptive control of interconnected systems," IEEE Trans. Auto. Control, 1986, Vol. AC-31, No. 4, pp. 291-298.
4. V. Hayward and J. Lloyd: "RCCL User's Guide," CVaRL/McGill University, Canada, April 1984.
5. J. Lloyd: "Implementation of a Robot Control Development Environment," Master's Thesis, Electrical Engineering Department, McGill University, Canada, 1985.
6. B. Armstrong, O. Khatib, and J. Burdick: "The explicit dynamic model and inertial parameters of the PUMA 560 arm," Proc. IEEE Intern. Conf. on Robotics and Automation, San Francisco, 1986, pp. 510-518.

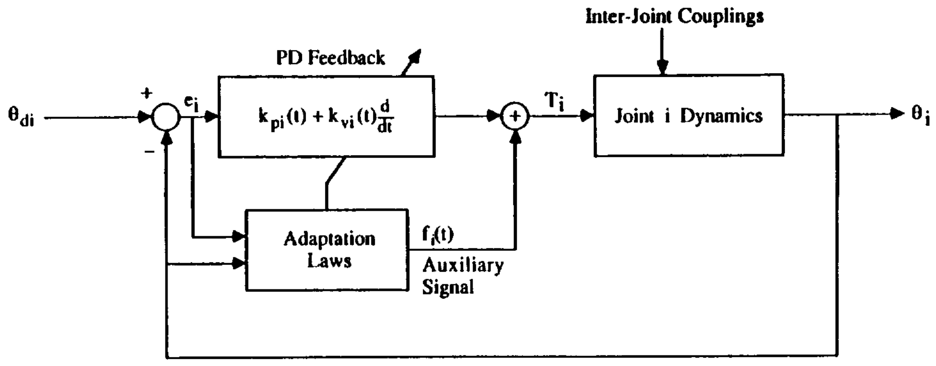

Figure 1. Decentralized Adaptive Control Scheme for Manipulator Joint i

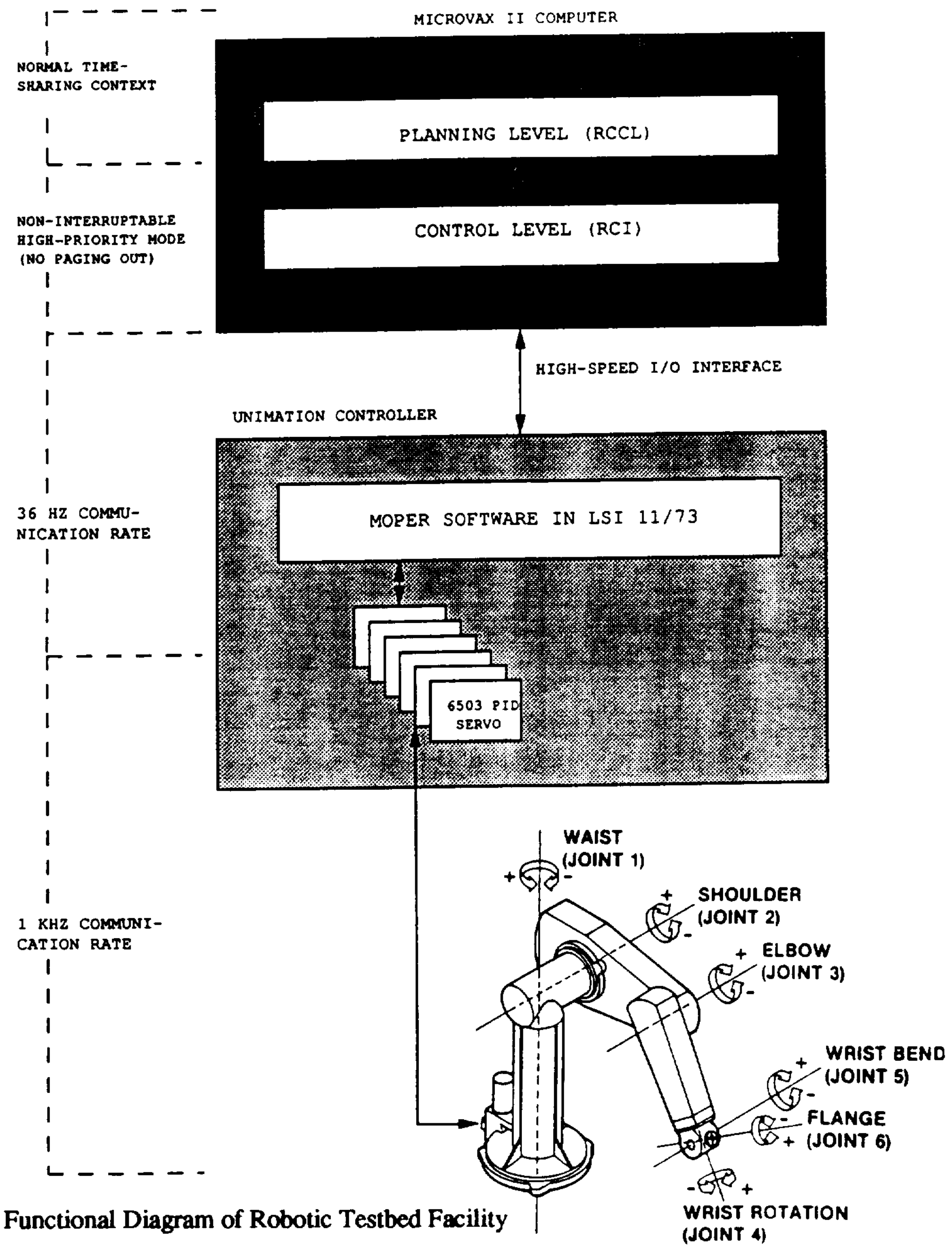

Figure 2. Functional Diagram of Robotic Testbed Facility

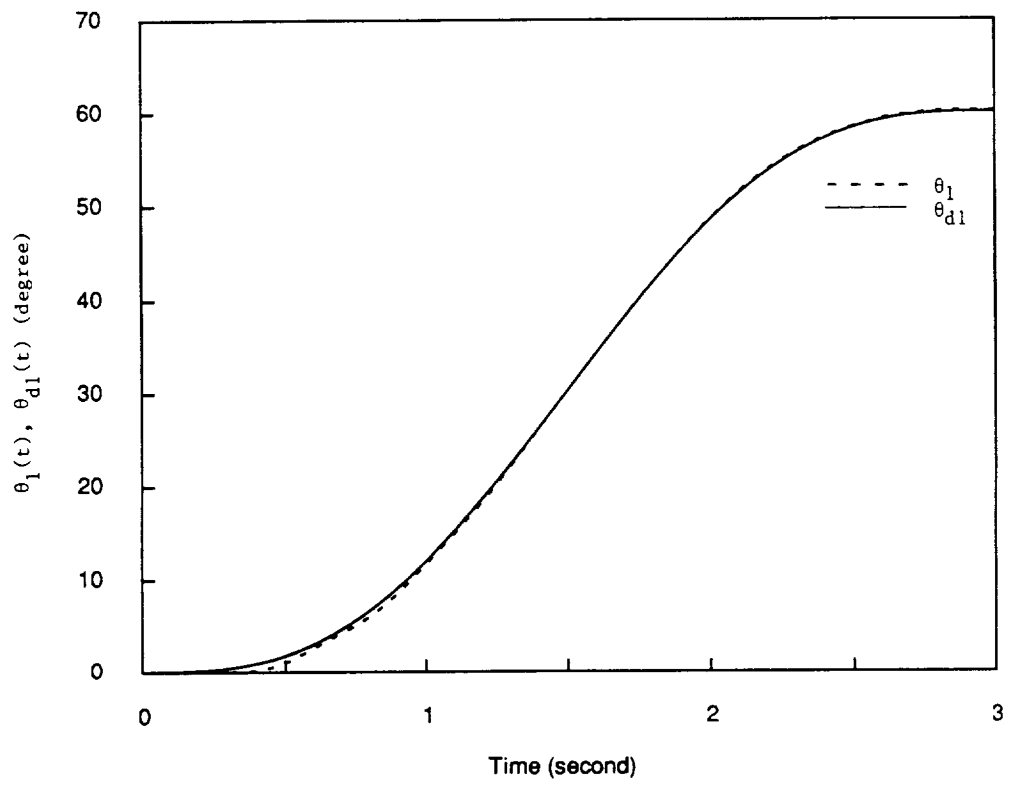

Figure 3(i). Response of Waist Angle θ_1 under Adaptive Controller

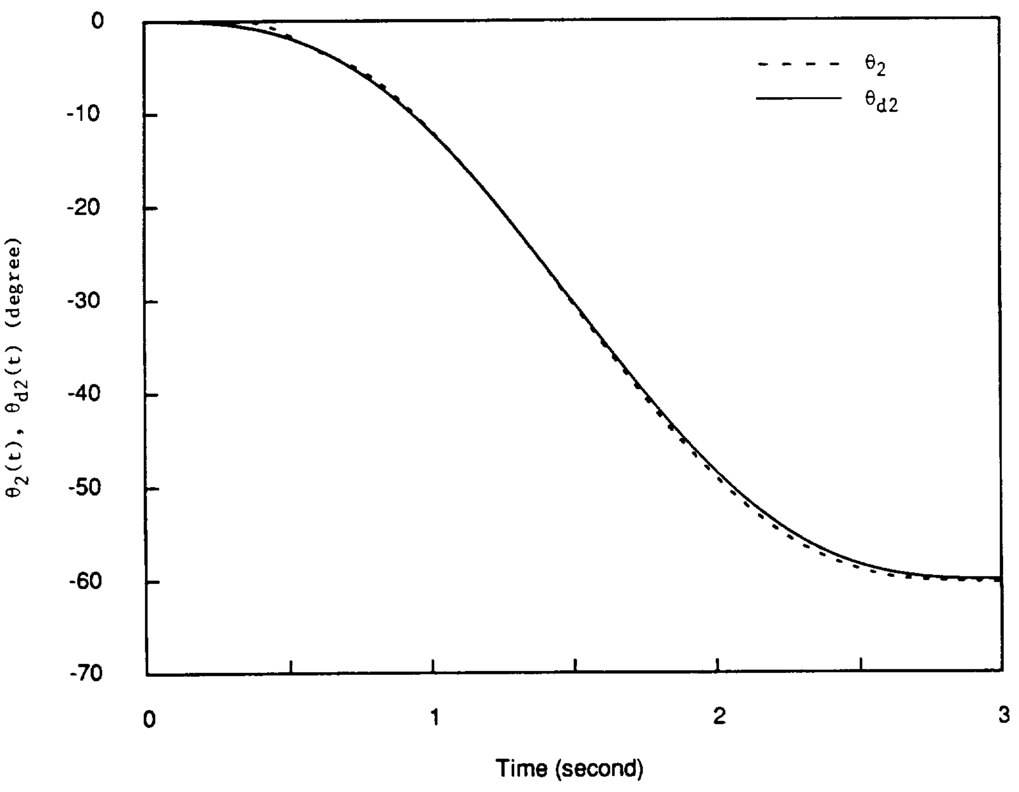

Figure 3(ii). Response of Shoulder Angle θ_2 under Adaptive Controller

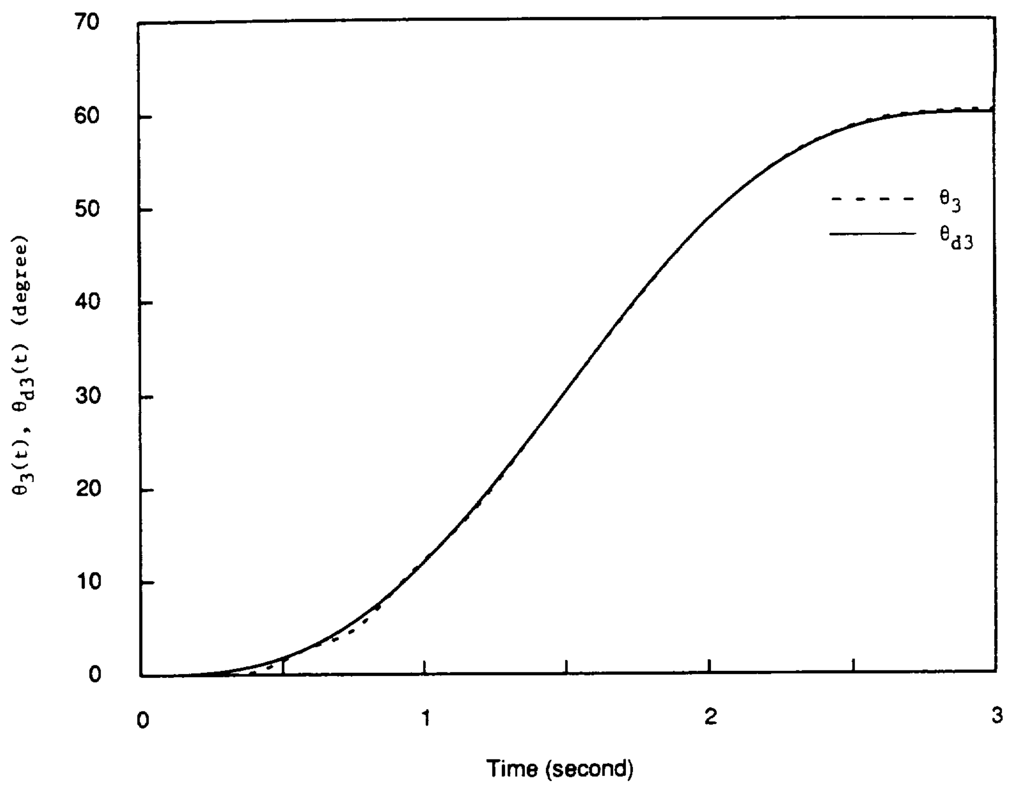

Figure 3(iii). Response of Elbow Angle θ_3 under Adaptive Controller

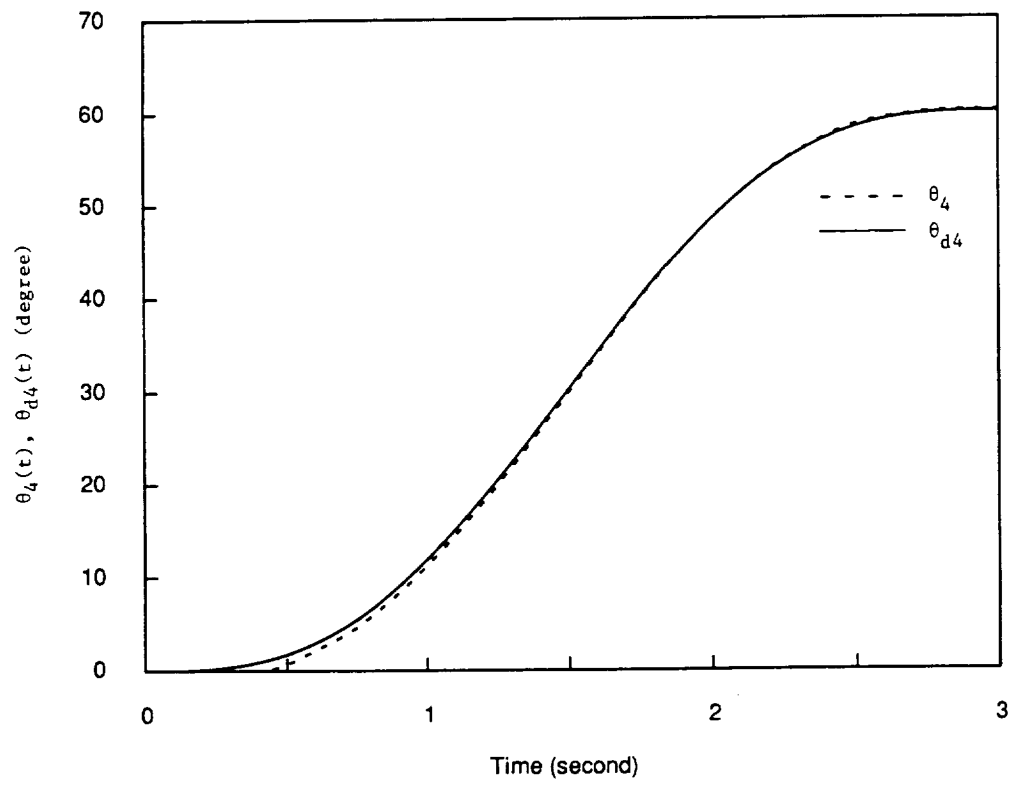

Figure 3(iv). Response of Wrist Angle θ_4 under Adaptive Controller

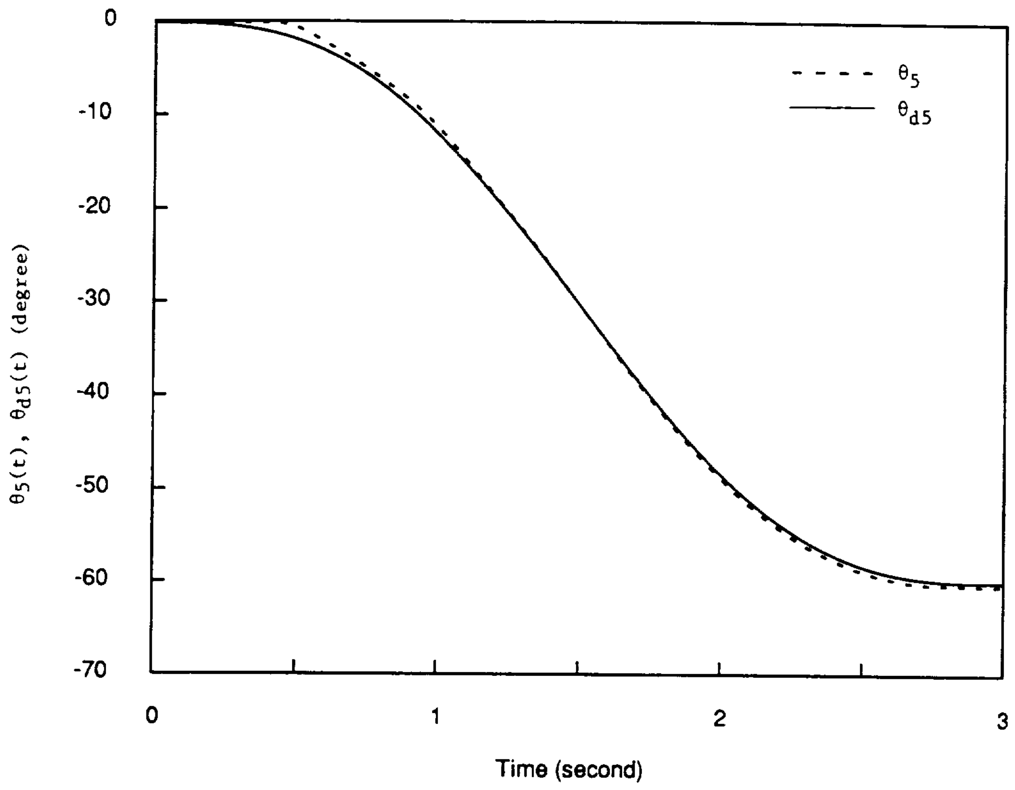

Figure 3(v). Response of Wrist Angle θ_5 under Adaptive Controller

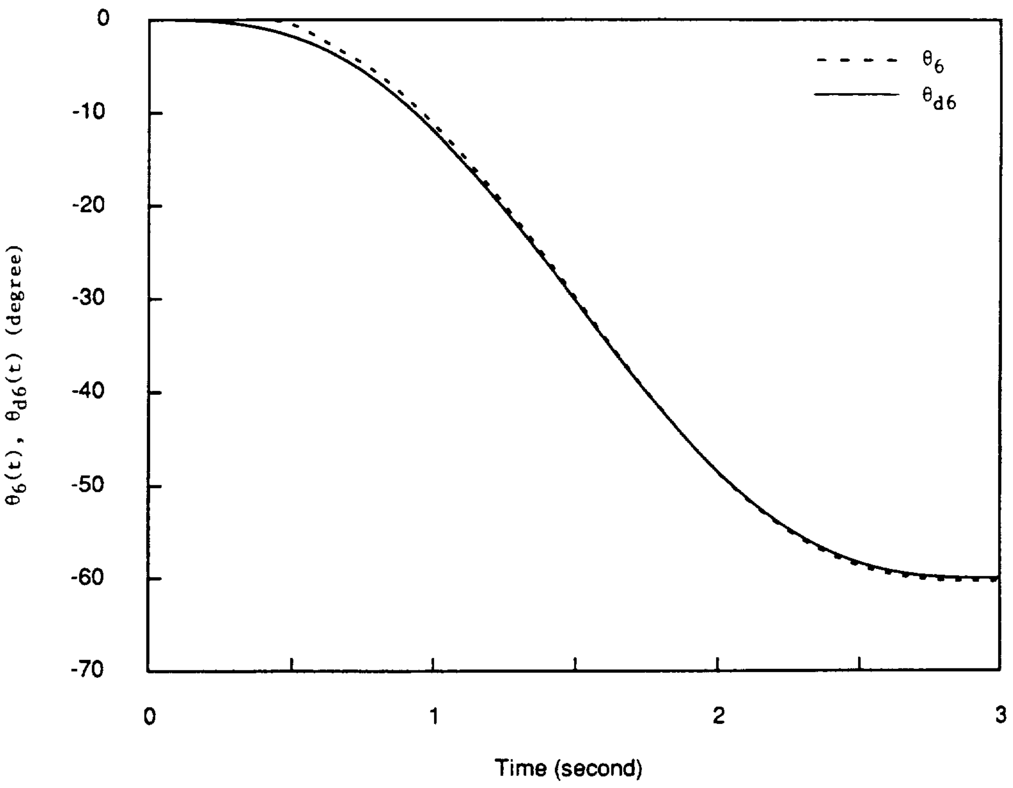

Figure 3(vi). Response of Wrist Angle θ_6 under Adaptive Controller